OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning

Hantian Zhang
Georgia Institute of Technology
hantian.zhang@cc.gatech.edu

Xu Chu
Georgia Institute of Technology
xu.chu@cc.gatech.edu

Abolfazl Asudeh
University of Illinois at Chicago
asudeh@uic.edu

Shamkant B. Navathe
Georgia Institute of Technology
shamkant.navathe@cc.gatech.edu

ABSTRACT

Machine learning (ML) is increasingly being used to make decisions in our society. ML models, however, can be unfair to certain demographic groups (e.g., African Americans or females) according to various fairness metrics. Existing techniques for producing fair ML models either are limited to the type of fairness constraints they can handle (e.g., preprocessing) or require nontrivial modifications to downstream ML training algorithms (e.g., in-processing).

We propose a declarative system OmniFair for supporting group fairness in ML. OmniFair features a declarative interface for users to specify desired group fairness constraints and supports all commonly used group fairness notions, including statistical parity, equalized odds, and predictive parity. OmniFair is also model-agnostic in the sense that it does not require modifications to a chosen ML algorithm. OmniFair also supports enforcing multiple user declared fairness constraints simultaneously while most previous techniques cannot. The algorithms in OmniFair maximize model accuracy while meeting the specified fairness constraints, and their efficiency is optimized based on the theoretically provable monotonicity property regarding the trade-off between accuracy and fairness that is unique to our system.

We conduct experiments on commonly used datasets that exhibit bias against minority groups in the fairness literature. We show that OmniFair is more versatile than existing algorithmic fairness approaches in terms of both supported fairness constraints and downstream ML models. OmniFair reduces the accuracy loss by up to 94.8% compared with the second best method. OmniFair also achieves similar running time to preprocessing methods, and is up to 270× faster than in-processing methods.

KEYWORDS

Algorithmic Bias; Group Fairness; Declarative Systems

ACM Reference Format:
Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B. Navathe. 2021. OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3448016.3452787

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3452787

1 INTRODUCTION

Machine learning (ML) algorithms, in particular classification algorithms, are increasingly being used to aid decision making in every corner of society. There are growing concerns that these ML algorithms may exhibit various biases against certain groups of individuals. For example, some ML algorithms are shown to have bias against African Americans in predicting recidivism [17, 23], in NYPD stop-and-frisk decisions [26], and in granting loans [22]. Similarly, some are shown to have bias against women in job screening [15] and in online advertising [42]. ML algorithms can be biased primarily because the training data these algorithms rely on may be biased, often due to the way the training data are collected [5, 35].

Due to the severe societal impacts of biased ML algorithms, various research communities are investing significant efforts in the general area of fairness — two out of five best papers in the premier ML conference ICML 2018 are on algorithmic fairness, the best paper in the premier database conference SIGMOD 2019 is also on fairness [40], and even a new conference ACM FAccT (previously FAT*) dedicated to the topic has been started since 2017. One commonly cited reason for such an explosion of efforts is the lack of an agreed mathematical definition of a fair classifier [7, 34, 44]. As such, many different fairness metrics have been proposed to determine how fair a classifier is with respect to a "protected group" of individuals (e.g., `African-American` or `female`) compared with other groups (e.g., `Caucasian` or `male`), including statistical parity [19], equalized odds [27], and predictive parity [16].

Example 1. *A popular model called COMPAS developed by Northpointe, Inc is used frequently in various stages in the criminal justice system, which predicts a dependent's risk of re-offending. Various studies have attempted to judge how fair the model h is for two groups of individuals, namely, g_1 =* `African Americans` *and g_2 =* `Caucasians`.

The famous Propublica study [4] concluded that the COMPAS model h is unfair because the probability of an individual being classified as "high-risk" depends on their race. Specifically, `African-American` *individuals are more likely to be classified as high-risk, i.e., $Pr(h(x) = 1|g_1) > Pr(h(x) = 1|g_2)$, a violation of statistical parity [19].*

Northpointe Inc. later published a response [2] to the Propublica study. They claim that the COMPAS model is indeed fair by demonstrating that the model achieves approximately equal accuracy across different groups, namely, $Pr(h(x) = y|g_1) \simeq Pr(h(x) = y|g_2)$. Similarly, researchers also find that the model approximately equal false omission rate and false discovery rate (also known as predictive parity) across different race groups, namely, $Pr(y = 1|g_i, h(x) = 0) \simeq Pr(y = 1|g_j, h(x) = 0)$ and $Pr(y = 0|g_i, h(x) = 1) \simeq Pr(y = 0|g_j, h(x) = 1)$.

An investigation by US Court [1] also deemed the COMPAS model h to be fair because it has both approximately equal false positive rate

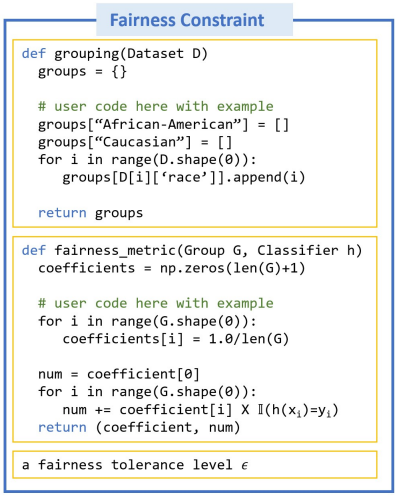


Figure 1: The declarative interface.

Table 1: Comparison with existing algorithmic fairness methods.

Method	Stage	One Fairness Constraint (c.f. § 3.2)	Multiple Constraints	Constraint Customization	Model Agnostic
Kamiran et al. [28]	Preprocessing	SP	No	No	Yes
Calmon et al. [11]	Preprocessing	SP	No	No	Yes
Zafar et al. [47]	In-Processing	MR,SP FPR,FNR	No	No	No
Celis et al. [12]	In-Processing	MR,SP FPR,FNR FDR,FOR	Yes*	No	No
Agarwal et al. [3]	In-Processing**	MR,SP FPR,FNR	Yes*	No	Yes
OmniFair	In-Processing**	MR,SP FPR,FNR FOR,FDR	Yes	Yes	Yes

Yes* means theoretically yes, but practically difficult to do.
In-Processing** means that these approaches are considered in-processing systems as they still need access to the ML algorithms to decide on weights. However, they do not need to modify the ML algorithms, like preprocessing techniques.

and equal false negative rate across different groups (also known as equalized odds). In other words, it found that the chance of mistakenly classifying an individual as high-risk does not depend on his/her race, *namely,* $Pr(h(x) = 1|g_i, y = 0) \simeq Pr(h(x) = 1|g_j, y = 0)$ *and* $Pr(h(x) = 0|g_i, y = 1) \simeq Pr(h(x) = 0|g_j, y = 1)$.

All aforementioned "definitions" of fairness seem reasonable and what to consider as fair depends on a particular application scenario and to a beholder, and new definitions keep coming up [34, 44].

Existing Approaches. Many techniques have been developed to produce fair models in two categories: *pre-processing* approaches that modify the input training data [11, 21, 28, 49] to remove the correlations between the *sensitive attribute* (e.g., race) and the training labels y before ML algorithms are applied; and *in-processing* approaches that enforce fairness constraints in the training process [3, 12, 47]. The main advantage of preprocessing approaches is that they are model-agnostic, and thus users can use ML algorithms as-is; however, they can only handle statistical parity [11, 21, 49], as other constraints require access to both the prediction $h(x)$ and the ground-truth y at the same time (e.g., equalized odds and predictive parity). In contrast, in-processing techniques can handle a much wider range of fairness constraints and can often achieve a better accuracy-fairness trade-off; however, they usually require non-trivial modifications to the ML training procedures [12, 47], and hence are usually model-specific (except for [3] as far as we know).

Key Components of Our Proposal. In light of the many current and constantly increasing types of fairness constraints and the drawbacks of existing approaches, we develop OmniFair. A comparison between OmniFair and existing approaches is shown in Table 1, as we shall elaborate next.

(1) Declarative Group Fairness. Current algorithmic fairness techniques are mostly designed for particular types of group fairness constraint. In particular, preprocessing techniques often only handle statistical parity [11, 28]. While in-processing techniques generally support more types of constraints, they often require significant changes to the model training process. For example, as shown in Table 1, while Celis et al. [12] supports all constraints shown in § 3.2, it is not model-agnostic. In addition, all current techniques cannot easily be adapted for customized constraints.

OmniFair is able to support all the commonly used group fairness constraints. In addition, OmniFair features a declarative interface that allows users to supply future customized fairness metrics. As shown in Figure 1, a *fairness specification* in OmniFair is a triplet (g, f, ε) with three components: (1) a grouping function g to specify demographic groups; (2) a fairness_metric function f to specify the fairness metric to compare between different groups; and (3) a value ε to specify the maximum disparity allowance between groups.

Given a dataset D, a chosen ML algorithm $\mathcal{A}$ (e.g., logistic regression), and a fairness specification (g, f, ε), OmniFair will return a trained classifier h that maximizes accuracy on D and, at the same time, ensures that, for any two groups g_i and g_j in D according to the grouping function g, the absolute difference between their fairness metric numbers according to the fairness_metric function f is within the disparity allowance ε. Figure 1 shows an example of constraint specification using our interface, which we will explain further in § 4. We will show that our interface can support not only all common group fairness constraints but also customized ones, including customized grouping functions such as intersectional groups [20, 24] and customized fairness metrics.

(2) Example Weighting for Model-Agnostic Property. The main advantage of preprocessing techniques is that they can be used for any ML algorithm $\mathcal{A}$. The model-agnostic property of preprocessing techniques is only possible when they limit the supported fairness constraints to those that do not involve both the prediction $h(x)$ and the ground-truth label y (i.e., statistical parity).

Our system OmniFair not only supports all constraints current in-processing techniques support, but also does so in a model-agnostic way. Our key innovation to achieve the model-agnostic property is to translate the constrained optimization problem (i.e., maximizing for accuracy subject to fairness constraints) into a weighted unconstrained optimization problem (i.e., maximizing for weighted accuracy). Specifically, in order to incorporate the fairness constraints, the original objective function of an ML algorithm $\mathcal{A}$ that maximizes for accuracy is turned into a version that maximizes for weighted accuracy (see Equation (1) to Equation (2) below),

where λ is a hyperparameter controlling the trade-off between accuracy and disparity of fairness metrics between two groups.

Max for accuracy without fairness constraints:

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(h_\theta(x_i) = y_i) \tag{1}$$

Max for weighted accuracy with fairness constraints:

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} w_i(\lambda, h_\theta)\mathbb{1}(h_\theta(x_i) = y_i) \tag{2}$$

This translation is only possible due to the way the fairness metric function is defined, as we shall see later. Solving the weighted objective function, if the example weights $w_i(\lambda, h_\theta)$ are constant values, does not require any modification to $\mathcal{A}$, since ML algorithms implemented in most ML packages (e.g., scikit-learn) already include an optional parameter to specify training example weights. Even when some ML algorithm implementations do not have the optional parameter, we can simulate weighting by replicating training examples — for example, a training dataset with two examples with weights 0.4 and 0.6, respectively, can be simulated by replicating the first example two times and the second example three times.

In § 5, we first show in detail how the translation is performed and what the example weights are for the commonly used fairness metrics. We then theoretically prove a monotonicity property for both fairness disparity and accuracy with respect to the hyperparameter λ. This unique property allows us to design efficient and effective hyperparameter tuning algorithm to maximize for accuracy while meeting the specified fairness constraint, which involves solving Equation (2) multiple times for different λ settings.

(3) Supporting Multiple Fairness Constraints. As discussed before, existing fairness ML techniques already support fewer types of single group fairness constraints than OmniFair. In practice, users may wish to enforce multiple fairness constraints simultaneously. While some in-processing techniques theoretically can support these cases, it is practically extremely difficult to do so, as each fairness constraint is hard-coded as part of the constrained optimization training process (c.f. Table 1).

Our system OmniFair can easily support multiple fairness constraints without any additional coding. Instead of having one hyperparameter λ, we will have a vector of fairness hyperparameters Λ, where each element in Λ is used to control one fairness constraint. Before tuning Λ to maximize for accuracy, we need to consider the *feasibility* question under the multi-constraint setting, namely, does there even exist a model that can satisfy all constraints at the same time. Indeed, prior research has shown theoretically that no model, regardless of which ML algorithms are used, can achieve perfect ($\varepsilon = 0$) statistical parity, perfect equalized odds, and perfect predictive parity for any dataset [30].

Given a particular dataset D, a particular ML algorithm $\mathcal{A}$, and a particular set of fairness constraints, in § 6, we first prove that, under some mild conditions, if a given set of fairness constraints can be satisfied simultaneously, there must exist some Λ value such that training $\mathcal{A}$ on the weighted dataset produces the feasible model. Since we cannot exhaustively search all Λ values (they are real numbers), we devise a practical hill-climbing algorithm for tuning Λ, which not only is much more efficient than a grid search, but also empirically shows a higher chance of reaching a feasible solution if one exists due to the finer grained search steps we can take.

Key features of OmniFair. In summary, we propose OmniFair for enforcing group fairness in ML with the following salient features:

(1) Versatile. OmniFair is versatile and easy to use — users only need to specify the desired fairness constraints and the ML algorithm, OmniFair is then able to produce a model that maximizes accuracy while meeting the constraints without changing the ML algorithm. OmniFair also allows for customized fairness constraints, and can also enforce multiple fairness constraints simultaneously.

(2) High Quality. OmniFair consistently outperforms existing methods w.r.t. accuracy-fairness trade-off. For example, when enforcing a statistical parity constraint with $\varepsilon = 0.03$, OmniFair reduces the accuracy loss by up to 94.8% compared with the second best method, when evaluated on the unseen test set. However, we caution that we can only explore the accuracy-fairness trade-off using an input dataset D (which is split into a training and a validation), and the obtained model may not satisfy declared fairness constraints on arbitrary test sets.

(3) Efficient. OmniFair includes efficient algorithms for hyperparameter tuning, which are designed based on theoretical properties unique in our design. We ensure that achieving high quality models of any ML algorithm under declarative group fairness constraints does not come at the expense of high computational overhead. OmniFair achieves comparable running time to existing preprocessing methods, and is up to 270× faster than in-processing methods.

2 RELATED WORK

Systems Support for ML. With the widespread use of ML analytics, we have seen a surge of systems work from the database community to manage parts of or the whole ML lifecycle with the general goal of democratizing ML [6, 9, 36]. Systems such as Snorkel [37] and Goggles [14] provide declarative programmatic interfaces, in the form of labeling functions and affinity functions, respectively, to allow users to easily express their domain knowledge that is useful for data labeling; systems such as Helix [46] and MLFlow [48] aim at managing and optimizing the iterative trial-and-error process that is the nature of ML workflow development; and systems such as Vista [33] provide declarative feature transfer to allow for jointly analyzing image data and structured data at scale; and systems such as Data Civilizer [38] and ActiveClean [31] help users deal with data errors in building ML models. OmniFair is our systems effort to allow ML users to easily incorporate fairness constraints into ML applications by providing a declarative programming interface while remaining model-agnostic.

Algorithmic Fairness Work. Table 1 shows the major works in algorithmic fairness literature that enforces various group fairness constraints in classification tasks. They can be grouped into two categories: Preprocessing and In-Processing.Preprocessing methods [11, 28] usually remove the dependency between the sensitive attribute and the target label by transforming the training data via example weighting or feature trasnformation. Preprocessing methods are typically very efficient, as they only need to prepare the training data and do not need to involve the ML model training process, but can only handle statistical parity.

Zafar et al. [47], Agarwal et al. [3], and Celis et al. [12] are three state-of-the-art in-processing methods, which modify ML algorithms to incorporate fairness constraints. Hence, they usually can handle more constraint types but are usually less efficient, depending on how they solve the constrained optimization problem. Zafar et al. [47] only works for decision boundary based classifiers (e.g., Logistic Regression, SVMs). It uses the covariance of the sensitive attribute and the signed distance between the feature vectors of misclassified data points and the decision boundary to represent fairness constraints and then solve the constrained optimization problem. It does not support predictive parity. Celis et al. [12] is the only in-processing work that supports predictive parity. It transforms the optimization problem with predictive parity constraint, which is non-convex, to a specific family of fair classification problem with convex constraints, and hence is not model-agnostic.

To the best of our knowledge, Agarwal et al. [3] is the only in-processing work that is also model-agnostic. OmniFair is different from [4] in three major aspects. First, OmniFair supports more constraint types than [4]. In particular, OmniFair supports predictive parity and customized constraints whose example weights are parameterized by h_θ while [4] does not. Second, for constraints both OmniFair and [4] support, OmniFair is much more user-friendly in terms of constraint specification due to our declarative interface. In contrast, users of [4] would need to understand the codebase and implement new constraint classes that connect with the rest of the code, a highly non-trivial task. Third, while both OmniFair and [4] leverage the similar idea of Lagrangian multiplier to translate constraint optimization problems to unconstrained optimization problems, our optimization algorithms are much faster because we leveraged the monotonicity property, while [4] used a saddle point finding algorithm, which is known to be difficult to solve.

Another recent work [43] shares the same overall objective as OmniFair, i.e. , making it dramatically easier for users to specify and regulate bias in ML without having to think about how to modify ML training algorithms. Thomas et al. [43] proposes a principled framework for ML researchers or engineers to design new ML algorithms that can take fairness constraints as input. In contrast, OmniFair can be used with any existing black-box ML algorithms.

3 PRELIMINARY

3.1 Machine Learning Background

In this paper, we limit our scope to *binary classification*, as almost all current fairness literature does. That is, we consider each tuple to include a set of features $x \in \mathbb{R}^d$ and to be associated with a binary label $y \in \{0, 1\}$. Our goal is to learn a classifier $h_\theta(x)$, where $h : \mathbb{R}^d \to \{0, 1\}$ is the classifier function that is parameterized by θ. We assume a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ is used for training the classifier. h_θ predicts the class label of a query point x as $h_\theta(x)$.

An ML training algorithm $\mathcal{A}$ takes D as input and produces a classifier $h_\theta(x)$ by maximizing for empirical accuracy (c.f. Equation (1)). Different ML algorithms optimize for the empirical accuracy differently. Many algorithms use some *loss function* $L(x_i, y_i; \theta)$ as a proxy for the identity function $\mathbb{1}(h_\theta(x_i) = y_i)$, and minimize the empirical loss. For example, for logistic regression, $L(x_i, y_i; \theta)$ is the logistic loss, while for support vector machines (SVM), it is the hinge loss. On the other hand, some ML algorithms (e.g., decision trees) may not have an explicit loss function, but still optimize for accuracy (e.g., by maximizing for purity of nodes in the tree).

3.2 Group Fairness Constraints

While there exist other fairness notions (e.g., individual fairness [19] and causal fairness [39, 40]), group fairness remains the most popular in practice [4, 11, 12, 21, 27, 28]. An ML model $h(\cdot)$ satisfies some group fairness constraints if the model has equal or similar performance (according to some fairness metrics) on different groups $\mathcal{G}$. We list major existing group fairness constraints as follows:

- **Statistical Parity (SP)** is the most studied group fairness constraint, which makes the independence assumption of the model from demographic groups ($h \perp \mathcal{G}$). Under SP [19], the probability of a model outcome is equal or similar across different groups:
$$\forall g_i, g_j \in \mathcal{G}, Pr(h(x) = 1|g_i) \simeq Pr(h(x) = 1|g_j) \quad (3)$$
- **False Positive Rate Parity (FPR)** makes the independence assumption of model from demographic groups, conditional to the true labels being 0 ($h \perp \mathcal{G} \mid y = 0$).
$$\forall g_i, g_j \in \mathcal{G}, Pr(h(x) = 1|g_i, y = 0) \simeq Pr(h(x) = 1|g_j, y = 0) \quad (4)$$
- **False Negative Rate Parity (FNR)** is similar to FPR. The only difference is that FNR is conditioned on $y = 1$. If both FPR and FNR are satisfied, then **Equalized Odds** [27] is satisfied.
- **False Omission Rate Parity (FOR)** makes the independence assumption of true label from groups, conditional on the negative model prediction ($y \perp \mathcal{G} | \ h = 0$).
$$\forall g_i, g_j \in \mathcal{G}, Pr(y = 1|g_i, h(x) = 0) \simeq Pr(y = 1|g_j, h(x) = 0) \quad (5)$$
- **False Discovery Rate Parity (FDR)** is similar to FOR. The only difference is that FDR is conditioned on $h(x) = 1$. If both equal or similar FOR and FDR are satisfied, then **Predictive Parity** [16] is satisfied.
- **Misclassification Rate Parity (MR)** equalizes the misclassification rate across different groups.
$$\forall g_i, g_j \in \mathcal{G}, Pr(h(x) = y|g_i) \simeq Pr(h(x) = y|g_j) \quad (6)$$

4 THE DECLARATIVE SPECIFICATION

We introduce the declarative interface of OmniFair. Our interface not only supports all group fairness constraints presented in § 3.2, but also allows users to supply customized constraints.

Definition 1. ***Fairness Specification and Fairness Constraint*** *A* fairness specification *is given by a triplet* $(\mathsf{g}, \mathsf{f}, \varepsilon)$. *One fairness specification on* D *induces* $\binom{|\mathsf{g}(\mathrm{D})|}{2}$ fairness constraints, *each defined over a pair of groups in* $\mathsf{g}(\mathrm{D})$.

A fairness specification is said to be satisfied by a classifier h *on* D *if and only if all induced fairness constraints are satisfied, i.e.,* $\forall g_i, g_j \in \mathsf{g}(\mathrm{D}),\ |\mathsf{f}(h, g_i) - \mathsf{f}(h, g_j)| \leq \varepsilon$

We note that the above definition defines two concepts: fairness specification and fairness constraint. A specification is a triplet $(\mathsf{g}, \mathsf{f}, \varepsilon)$ that users specify. A given specification can induce one or multiple constraints, one for each pair of groups given by g. To specify constraints that require different grouping or metrics, users need to provide multiple specifications. Section 5 handles a single specification that gives one constraint only, i.e., the grouping function generates only two groups. Section 6 handles multiple constraints, which may be induced by a single specification that generates multiple groups or multiple specifications.

Problem Formulation. Given a dataset D, an ML algorithm $\mathcal{A}$, one or multiple group fairness constraints given by one or multiple fairness specifications, our goal is to obtain an ML model h_θ that maximizes for accuracy, while satisfying given constraint(s).

Discussion: In practical deployment, we obtain h_θ using an input dataset D, which we will split into D_{train} and D_{val} for better generalizability when tuning hyperparameters. We make sure that the model h_θ satisfies the declared constraints on D_{val}. However, h_θ may still not satisfy the constraints on arbitrary unseen test sets.

4.1 Declarative Grouping Function

Naturally, any declarative group fairness specification must allow users to specify the interested demographic groups.

Definition 2. *(Declarative Grouping Function) A declarative grouping function* g *is a user-defined function that takes a dataset* D *as input, and* partitions *the tuples to different groups $\mathcal{G}$. We implement $\mathcal{G}$ as a dictionary, in which the keys are group ids and the values are the set of tuples in each group.*

Example 2. *As a toy example, consider a dataset $D = \{t_1, t_2, \cdots t_{10}\}$, where t_4, t_5, t_7, and t_9 are African American and others are Caucasian. A user-specified grouping function is shown in Figure 1: The above function partitions D as:* $\mathrm{g}(D) = \{$`Caucasian` : $[1, 2, 3, 6, 8, 10]$, `AfricanAmerican` : $[4, 5, 7, 9]\}$.

Note that almost all current group fairness constraints assume that the groups are implicitly induced by a given sensitive attribute (e.g., race). In fact, some preprocessing techniques are applicable only when groups are induced by a sensitive attribute, since they enforce group fairness by removing the correlations between the sensitive attribute and the training data labels [11, 21]. Our grouping function removes such implicit assumption and allows greater flexibility in declaring groups as we further explain in § 4.3.

4.2 Declarative Fairness Metric Function

Designing the interface for specifying fairness metric requires greater consideration. In particular, the fairness metric interface needs to strike a balance between its ability to express various group fairness constraints and the hardness of designing model-agnostic techniques to enforce the declared fairness constraints. We propose to express the group fairness metric as a weighted linear summation of the identity function $\mathbb{1}(h(x_i) = y_i)$. Our intuition is that in order to design a model-agnostic approach for enforcing fairness constraints, we cannot assume any knowledge about how the ML algorithm $\mathcal{A}$ works internally. The only behavior we do know about $\mathcal{A}$ is that it optimizes for accuracy in the absence of any constraints, and the calculation of accuracy is based on whether each individual prediction is correct, i.e., $\mathbb{1}(h(x_i) = y_i)$. If we are able to express fairness constraints based on $\mathbb{1}(h(x_i) = y_i)$, we may be able to optimize for accuracy and fairness simultaneously.

Definition 3. *(Declarative Fairness Metrics) A fairness metric* f *function takes as input a classifier h and a group g, and returns $(1 + |g|)$ coefficients that specify how the metric is computed:*

$$\mathrm{f}(h, g) = \sum_{i \in g} c_i \mathbb{1}(h(x_i) = y_i) + c_0 \tag{7}$$

Different fairness metrics are specified by different coefficients c_i on each data point in group g. Our interface supports all existing group fairness constraints (the coefficients for group fairness constraints are summarized in Table 2), and we can design efficient model-agnostic techniques for enforcing constraints declared this way (c.f. § 5.1). We show the derivation of SP as an example, and refer readers to the full report [50] for additional examples.

Table 2: Coefficients for different popular group fairness metrics. The coefficients c_i for the i^{th} example in a group g are split into two groups based on the label y_i. Note that for the first four metrics, the coefficients are not parameterized by $h(x)$; and for the last two metrics, the coefficients are parameterized by $h(x)$.

	$c_i \mid y_i = 0$	$c_i \mid y_i = 1$	c_0
MR	$1/\lvert g\rvert$	$1/\lvert g\rvert$	0
SP	$-1/\lvert g\rvert$	$1/\lvert g\rvert$	$\lvert\{i : i \in g, y_i = 0\}\rvert/\lvert g\rvert$
FPR	$1/\lvert\{i : i \in g, y_i = 0\}\rvert$	0	0
FNR	0	$1/\lvert\{i : i \in g, y_i = 1\}\rvert$	0
FOR	$1/\lvert\{i : i \in g, h(x_i) = 0\}\rvert$	0	0
FDR	0	$1/\lvert\{i : i \in g, h(x_i) = 1\}\rvert$	0

Example 3. **Expressing SP.** *As shown in Equation* (3)*, under SP, the fairness metric is $f(h, g) = Pr(h(x) = 1)$, which can be decomposed into two parts, $y_i = 0$ and $y_i = 1$ as follows:*

$$\begin{aligned} & f(h, g) = Pr(h(x) = 1) \\ & = Pr(y = 1) Pr(h(x) = 1 \mid y = 1) + Pr(y = 0) Pr(h(x) = 1 \mid y = 0) \\ & = Pr(y = 1) Pr(h(x) = 1 \mid y = 1) + Pr(y = 0)(1 - Pr(h(x) = 0 \mid y = 0)) \\ & = \frac{1}{|g|} \sum_{\{i : i \in g, y_i = 1\}} \mathbb{1}(h(x_i) = y_i) + \frac{-1}{|g|} \sum_{\{i : i \in g, y_i = 0\}} \mathbb{1}(h(x_i) = y_i) \\ & \quad + \frac{|\{i : i \in g, y_i = 0\}|}{|g|} \end{aligned} \tag{8}$$

Therefore, for data points in g with label $y_i = 1$ (resp. $y_i = 0$), the coefficient is $c_i = \frac{1}{|g|}$ (resp. $c_i = -\frac{1}{|g|}$). We also have $c_0 = \frac{|\{i : i \in g, y_i = 0\}|}{|g|}$.

4.3 Fairness Constraint Customization

In addition to supporting existing group fairness constraints, our declarative interface allows for declaring customized constraints.

Customization of Grouping Function. The grouping function (§ 4.1) can easily support intersectional fairness [20, 24], where fairness metrics are defined for subgroups defined over the intersection of multiple sensitive attributes, and some richer forms of subgroup fairness [29], where groups are defined based on some classification models. Whatever ways one may desire to form groups can be encoded in the grouping function. For example, if both race and gender are considered sensitive attributes in a dataset, then one can write a grouping function that outputs the `African American female` group, the `African American male` group, and so on.

Customization of Fairness Metrics. The abstraction to declarative fairness metric enables a wide range of user-customized metrics defined based on societal norms, as long as the fairness metric can be expressed as a weighted linear combination of the identify function (c.f. Definition 3). We provide an example of metric customization.

Example 4. *A model for binary classification tasks can make two types of errors: false positives and false negatives. In statistics, they are called Type I error and Type II error. In different applications, false positives and false negatives may incur different costs [8].*

Users can easily specify a particular cost for the two error types and define a customized fairness metric that calculates the average cost per group. We refer to the full report [50] for concrete definition.

Scope and Limitations. OmniFair currently supports group fairness constraints defined in the context of binary classification problems. In terms of the grouping function, users can write any logic for forming groups as long as the function returns at least two groups of tuples (e.g., intersectional groups defined over multiple sensitive attributes). In addition, the returned groups do not necessarily need to be disjoint. In terms of the fairness metric function, OmniFair can only support those metrics that can be expressed as a linear combination of the identify function (c.f. Definition 3).

We further note that OmniFair can only constrain the difference of a given metric between a pair of groups to be within a threshold, i.e., $|\mathsf{f}(h, g_i) - \mathsf{f}(h, g_j)| \leq \varepsilon$. For example, we do not support the constraining of the division of a given metric between two groups to be within a threshold, and we do not support constraints expressed between more than two groups.

5 SINGLE FAIRNESS CONSTRAINT

In this section, we consider a single fairness constraint defined over two groups, and defer the general case of handling multiple constraints to § 6. Specifically, given a dataset D, an ML algorithm $\mathcal{A}$, and a fairness specification (g, f, ε), where applying $g(D)$ yields two groups g_1 and g_2, we aim to obtain a classifier that maximizes for accuracy while satisfying the fairness constraint.

Without loss of generality, we assume the goal of $\mathcal{A}$ is to learn some model parameters θ, and we use h_θ to denote the classifier. To simplify notations, we use $AP(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(h_\theta(x_i) = y_i)$ to denote the *accuracy part* and $FP(\theta) = \mathsf{f}(h_\theta, g_1) - \mathsf{f}(h_\theta, g_2)$ to denote the *fairness part* of the constrained optimization problem. We can thus now formally write the constrained optimization problem as:

$$\begin{aligned} \max_{\theta} \quad & AP(\theta) \\ \text{s.t.} \quad & |FP(\theta)| \leq \varepsilon \end{aligned} \tag{9}$$

5.1 Problem Translation

Solving the constrained problem directly in Equation (9) would require modifications to $\mathcal{A}$ (as current in-processing techniques do). Instead, we translate Equation (9) into an unconstrained optimization problem by using the *Lagrangian dual function* [10] $h(\lambda_1, \lambda_2)$ (the last two terms in $h(\lambda_1, \lambda_2)$ come from expanding the $|FP(\theta)| \leq \varepsilon$ constraint in Equation (9) into two constraints):

$$\begin{aligned} h(\lambda_1, \lambda_2) &= \max_{\theta} AP(\theta) + \lambda_1(\varepsilon - FP(\theta)) + \lambda_2(\varepsilon + FP(\theta)) \\ &= \max_{\theta} AP(\theta) + (\lambda_2 - \lambda_1)FP(\theta) + (\lambda_1 + \lambda_2)\varepsilon \end{aligned} \tag{10}$$

Following directly from Lagrangian duality (as we will show in the following lemma), $h(\lambda_1, \lambda_2)$ provides an upper bound for Equation (9) for any $\lambda_1 > 0, \lambda_2 > 0$ and the bound is proven to be tight under the so-called "strong duality" assumption [10].

The above direct application of Lagrangian duality would require us to tune two hyperparameters (λ_1 and λ_2). We further simplify Equation (10) into Equation (11) with only one hyperparameter λ. We note that this is only possible in our setting because the two constraints are acting on the same term involving θ, i.e., $FP(\theta)$.

$$\max_{\theta} AP(\theta) + \lambda FP(\theta) \tag{11}$$

This simplification is justified since we can show that for any $\lambda_1 > 0, \lambda_2 > 0$, there always exists a $\lambda = \lambda_2 - \lambda_1$, such that the optimal solution to Equation (10) will also be the optimal solution to Equation (11). We summarize the direct application of Lagrangian duality and our simplification as two parts of the following lemma.

Lemma 1. *Assume Equation* (9) *is feasible and let* θ^* *be an optimal solution to Equation* (9)*, then*

(1) for any $\lambda_1, \lambda_2 > 0$, $h(\lambda_1, \lambda_2) \geq AP(\theta^*)$*; and under strong duality assumption,* $\min_{\lambda_1>0,\lambda_2>0} h(\lambda_1, \lambda_2) = AP(\theta^*)$.

(2) for any $\lambda_1, \lambda_2 > 0$*, let* $\tilde{\theta}$ *be an optimal solution to Equation* (10)*, then there exists* $\lambda \in \mathbb{R}$ *(i.e.,* $\lambda = \lambda_2 - \lambda_1$*) such that* $\tilde{\theta}$ *also optimizes Equation* (11)*; and under strong duality assumption, there exists* λ *such that* θ^* *optimizes Equation* (11).

We refer readers to the full report [50] for the proof. We note that there are multiple different conditions under which strong duality assumption holds (e.g., when the primal problem Equation (9) is a linear optimization problem; or when the primal problem is a convex optimization problem and the Slater's condition holds [10]). Even without the strong duality assumption, the Lagrangian dual is still commonly used in practice to give a good approximate solution for the primal problem (by optimizing for the upper/lower bound). In OmniFair's setting, while we cannot solve the primal problem in a model-agnostic way, we are able to solve the dual problem in a model-agnostic way. We also observe that the strong duality assumption has little implication on empirical results — we used four ML models, three of which have a non-convex loss, i.e., $AP(\theta)$, and OmniFair performs well under all four tested models.

Given Lemma 1, we can essentially solve Equation (9) naively in three steps: (i) enumerating all possible λ values; (ii) finding the optimal solution to Equation (11) for every λ value; and (iii) out of all the optimal solutions for different λ values, pick the one that has the maximal $AP(\theta)$ while $|FP(\theta))| \leq \varepsilon$. In Sec 5.3, instead of enumerating all λ values, we design a smart algorithm to search for λ efficiently.

5.2 Solving Equation (11) for a Given λ

Deriving Example Weights. The key to solving Equation (11) for any constraints $FP(\theta)$ is to transform it into a weighted regular ML optimization problem. Intuitively, this is possible because both $AP(\theta)$ and $FP(\theta)$ are expressed as linear combinations of $\mathbb{1}(h_\theta(x_i) = y_i)$, and hence the overall $AP(\theta) + \lambda FP(\theta)$ can be converted into a weighted regular ML optimization problem.

Let us now expand Equation (11) by plugging in $AP(\theta)$ and $FP(\theta)$. We abbreviate $\mathbb{1}(h_\theta(x_i) = y_i)$ as $\mathbb{1}_i$, and we use c_i^g to denote the coefficient of the i^{th} tuple in group g in the calculation of $f(h, g)$.

$$\begin{aligned} &\max_{\theta} AP(\theta) + \lambda FP(\theta) \\ &= \max_{\theta} \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_i + \lambda\Big(\sum_{i\in g_1}(c_i^{g_1}\mathbb{1}_i + c_0^{g_1}) - \sum_{i\in g_2}(c_i^{g_2}\mathbb{1}_i + c_0^{g_2})\Big) \\ &= \max_{\theta} \frac{1}{N}\sum_{i=1}^{N} w_i(\lambda, h_\theta)\mathbb{1}(h_\theta(x_i) = y_i) \end{aligned} \tag{12}$$

The weight w_i for each data point in D can be computed directly from the terms that they are participating in. For the points in D that belong to g_1 only, their weights are $w_i = 1 + N\lambda c_i^{g_1}$; and for data points in g_2 only, their weights are $w_i = 1 - N\lambda c_i^{g_2}$. For the points that belong to both g_1 and g_2 (as two groups may be overlapping),

Table 3: Weights for popular group fairness metrics. Here, we assume g_1 and g_2 are disjoint for simplicity. The weights w_i are split into four groups based on their labels y_i and their group g. The weights that do not belong to any of those four groups are 1. In addition, weights may or may not be parameterized by θ.

weight	metric	$w_i \mid y_i = 0, g_1$	$w_i \mid y_i = 1, g_1$	$w_i \mid y_i = 0, g_2$	$w_i \mid y_i = 1, g_2$
$w_i(\lambda)$	MR	$1 + \lambda N/\lvert g_1\rvert$	$1 + \lambda N/\lvert g_1\rvert$	$1 - \lambda N/\lvert g_2\rvert$	$1 - \lambda N/\lvert g_2\rvert$
	SP	$1 - \lambda N/\lvert g_1\rvert$	$1 + \lambda N/\lvert g_1\rvert$	$1 + \lambda N/\lvert g_2\rvert$	$1 - \lambda N/\lvert g_2\rvert$
	FPR	$1 - \lambda N/\lvert\{i : i \in g_1, y_i = 0\}\rvert$	1	$1 + \lambda N/\lvert\{i : i \in g_2, y_i = 0\}\rvert$	1
	FNR	1	$1 - \lambda N/\lvert\{i : i \in g_1, y_i = 1\}\rvert$	1	$1 + \lambda N/\lvert\{i : i \in g_2, y_i = 1\}\rvert$
$w_i(\lambda, h_\theta)$	FOR	$1 - \lambda N/\lvert\{i : i \in g_1, h(x_i) = 0\}\rvert$	1	$1 + \lambda N/\lvert\{i : i \in g_2, h(x_i) = 0\}\rvert$	1
	FDR	1	$1 - \lambda N/\lvert\{i : i \in g_1, h(x_i) = 1\}\rvert$	1	$1 + \lambda N/\lvert\{i : i \in g_2, h(x_i) = 1\}\rvert$

their weights are $w_i = 1 + N\lambda c_i^{g_1} - N\lambda c_i^{g_2}$. For points that belong to neither group, their weights are $w_i = 1$.

We list the weights for popular group fairness metrics in Table 3 (by essentially plugging in c_i from Table 2). As we can see, the example weights derived for different fairness metrics are divided into two categories: those that are not parameterized by $h_\theta(x)$ (e.g., SP) and those that are parameterized by $h_\theta(x)$ (e.g., FOR), depending on whether the coefficients c_i in the fairness metric are parameterized by $h_\theta(x)$ (c.f. Table 2).

Solving Equation (12). For the fairness metrics whose induced example weights are not parameterized by $h_\theta(x)$, the weights $w_i(\lambda)$ are constant values given λ, hence Equation (12) can be solved using any black-box ML algorithm $\mathcal{A}$ by providing weighted examples.

For the fairness metrics whose induced example weights are indeed parameterized by $h_\theta(x)$, i.e., $w_i(\lambda, h_\theta)$, we cannot solve Equation (12) exactly any more, assuming $\mathcal{A}$ is a black-box. We thus resort to find an approximate solution. Our intuition is that *given λ_1 and λ_2 that are extremely close to each other, e.g., $\lambda_2 - \lambda_1 \le \delta = 0.0001$, the predictions from the optimal models θ_1 (given λ_1) and θ_2 (given λ_2) should almost be identical.* Therefore, we can approximate the weights associated with λ_2 using a close enough model θ_1 derived from a close-enough λ_1, namely, $w_i(\lambda_2, h_{\theta_2}) \simeq w_i(\lambda_2, h_{\theta_1})$. When $\lambda = 0$, we can derive the optimal solution θ_0 by setting all the weights $w_i(0, h_\theta)$ as 1. Therefore, starting from $\lambda = 0$ and taking small incremental steps δ, we can obtain accurate weight estimates, which allows us to solve Equation (11) using black-box $\mathcal{A}$.

The additional complexity for dealing with FOR and FDR (i.e., predictive parity), or in general any group fairness constraint that conditions on the prediction $h_\theta(x)$, is not surprising, because those fairness constraints are non-differentiable. Indeed, even in the fairness literature of in-processing techniques that modify ML algorithms, supporting predictive parity is only a recent development that offers only approximate solutions [12]. As we shall see, OmniFair greatly outperforms [12] in this case.

As we can see, by rewriting Equation (11) into Equation (12), we can solve it in a model-agnostic way. We emphasize that the above translation is possible due to how we abstracted our fairness metric function f, which allows us to merge the objective of maximizing for accuracy and the objective of satisfying fairness constraints into a weighted unconstrained optimization problem. This was a major motivation for our fairness metric interface in § 4.2.

5.3 Tuning the Hyperparameter λ

Intuitively, the hyperparameter λ in Equation (11) controls the trade-off between accuracy and fairness — each λ value will produce an ML model (by solving Equation (12)) with a particular accuracy and bias. We thus need to find the optimal λ that leads to a model h_θ with the maximum possible $AP(\theta)$ while satisfying $|FP(\theta)| \le \varepsilon$.

Canonical algorithms for hyperparameter search include grid search and random search that randomly tries some values of λ. Grid search usually produces models with better performance at the cost of efficiency. In our case, the hyperparameter directly controls the trade-off between accuracy and fairness. In fact, we can theoretically prove a *monotonicity* property for $AP(\theta)$ and $FP(\theta)$ with respect to λ, which allows us to design efficient hyperparameter tuning procedures that cover the entire space.

5.3.1 The monotonicity property. We show the *monotonicity* of both optimization metrics AP and FP with respect to λ:

Lemma 2. ***[Monotonicity for Single Fairness Constraint]*** *Consider two values λ_1 and λ_2, where $\lambda_1 < \lambda_2$, and let θ_1 and θ_2 denote two optimal solutions to Equation* (11) *given λ_1 and λ_2, assuming they exist when trained on the same training set D, respectively:*

$$\theta_1 = \arg\max_\theta AP(\theta) + \lambda_1 FP(\theta) \tag{13}$$

$$\theta_2 = \arg\max_\theta AP(\theta) + \lambda_2 FP(\theta) \tag{14}$$

Then, the following properties hold, where $AP(\theta_1)$, $AP(\theta_2)$, $FP(\theta_1)$, and $FP(\theta_2)$ are evaluated on the same input training set D:

$$FP(\theta_1) \le FP(\theta_2) \tag{15}$$

$$AP(\theta_1) \ge AP(\theta_2) \text{ when } \lambda_2 > \lambda_1 \ge 0 \tag{16}$$

$$AP(\theta_1) \le AP(\theta_2) \text{ when } 0 \ge \lambda_2 > \lambda_1 \tag{17}$$

We defer to the full paper [50] for proof. We note that the fairness part $FP(\theta)$ is monotonically increasing in the full range of λ, while the accuracy part $AP(\theta)$ is monotonically increasing in the positive range of λ, and is monotonically decreasing in the negative range of λ. Combining Equation (16) and Equation (17), we can deduce that the ML model has the maximum accuracy when $\lambda = 0$, which is exactly the original objective function for maximizing accuracy and the model may or may not satisfy the fairness constraint.

We note that while Lemma 2 is only theoretically true on the training set, it nevertheless provides a practically efficient way to search over all possible λ values. This is based on the assumption that the training set, the validation set, and the test set are from the same distribution, and hence a model with a higher or lower $AP(\theta)$ or $FP(\theta)$ on the training set would also most likely have a higher or lower $AP(\theta)$ or $FP(\theta)$ on the validation set or the test set.

5.3.2 Algorithm for Tuning λ. Algorithm 1 describes the procedure for tuning λ, which consists of three stages.

(1) Training a Model with $\lambda = 0$ (Lines 1 to 3). We first train an ML model θ_0 without any fairness constraint, i.e., $\lambda = 0$. If θ_0 already

Algorithm 1 Tuning Single λ

Input: Dataset D, a fairness constraint (g, f, ε), an ML Algorithm $\mathcal{A}$
Output: A fair ML model h_θ

1: $\theta_0 \leftarrow$ apply $\mathcal{A}$ with $w_i(0)$
2: $flag \leftarrow false$ **if** (weights are parameterized by θ) **else** $true$
3: **if** $|FP(\theta_0)| \leq \varepsilon$ **then return** h_{θ_0}
4: **if** $FP(\theta_0) > 0$ **then**
5: change the order of g_1 and g_2 in FP
6: $\lambda_l \leftarrow 0$ and $\lambda_u \leftarrow 1$
7: **if** $flag = true$ **then**
8: $\lambda_l, \lambda_u \leftarrow$ ExponentialSearch(λ_l, λ_u)
9: **else**
10: $\lambda_l, \lambda_u \leftarrow$ LinearSearch(λ_l, δ) // e.g. $\delta = 0.001$
11: **while** $\lambda_u - \lambda_l \geq \tau$ **do** // $\tau \rightarrow 0$; e.g. $\tau = 0.0001$
12: $\lambda_m \leftarrow (\lambda_u + \lambda_u)/2$
13: **if** $flag = true$ **then**
14: $\theta_m \leftarrow$ apply $\mathcal{A}$ with $w_i(\lambda_m)$
15: **else**
16: $\theta_m \leftarrow$ apply $\mathcal{A}$ with $w_i(\lambda_m, h_{\theta_l})$
17: **if** $FP(\theta_m) < -\varepsilon$ **then** $\lambda_l \leftarrow \lambda_m$
18: **else** $\lambda_u \leftarrow \lambda_m$
19: **return** h_{θ_m}
20:
21: **function** ExponentialSearch(λ_l, λ_u)
22: $\theta_u \leftarrow$ apply $\mathcal{A}$ with $w_i(\lambda_u)$
23: **while** $FP(\theta_u) < -\varepsilon$ **do**
24: $\lambda_l \leftarrow \lambda_u$
25: $\lambda_u \leftarrow 2 \times \lambda_u$
26: $\theta_u \leftarrow$ apply $\mathcal{A}$ with $w_i(\lambda_u)$
27: **return** λ_l, λ_u
28:
29: **function** LinearSearch(λ_l, δ)
30: $\lambda_u \leftarrow \lambda_l + \delta$
31: $\theta_u \leftarrow$ apply $\mathcal{A}$ given λ_u
32: **while** $FP(\theta_u) < -\varepsilon$ **do**
33: $\lambda_l \leftarrow \lambda_u$
34: $\theta_l \leftarrow \theta_u$
35: $\lambda_u \leftarrow \lambda_l + \delta$
36: $\theta_u \leftarrow$ apply $\mathcal{A}$ with $w_i(\lambda_u, \theta_l)$
37: **return** λ_l, λ_u

satisfies the fairness constraint, i.e., $|FP(\theta_0)| \leq \varepsilon$, we can simply return h_{θ_0}, which has the maximum $AP(\theta_0)$ according to Lemma 2. If $FP(\theta_0) > 0$, we switch the order between the two groups g_1 and g_2 to make sure $FP(\theta_0) < -\varepsilon$. This means that we need to search the positive values for λ, which we describe next.

(2) Bounding λ (Lines 21 to 27). While we know the optimal λ must be a positive value, i.e., $\lambda_{lower} = 0$, we do not know its upper bound.

(2.1) Exponential Search (Lines 21 to 27). When the weights are not parameterized by θ, then $w_i(\lambda)$ will be constant values for a given λ value. We thus employ an exponential search procedure to quickly locate the upper bound λ_{upper} by initially setting $\lambda_{upper} = 1$ and gradually doubling it until $FP(\theta_u) \geq -\varepsilon$.

(2.2) Linear Search (Lines 29 to 37). When the example weights are parameterized by θ (e.g., FOR and FDR), the example weights are no longer constant values for a given λ, and hence we cannot use exponential search to quickly identify the bounds for λ. As discussed in § 5.2, we have to resort to a linear search procedure for obtaining the approximate weights. Specifically, we start from $\lambda = 0$ and as we take small incremental steps δ, we can obtain accurate weight estimates, which will allow us to solve Equation (12) using the black-box ML algorithm $\mathcal{A}$.

(3) Binary Search for the Optimal λ (Lines 11 to 19). Because of the monotonicity property of $FP(\theta)$ in Equation (15), we know that there must exist a λ in $[\lambda_{lower}, \lambda_{upper}]$ that satisfies the fairness constraint. We can thus do a binary search to reach the smallest λ that satisfies the fairness constraint, which must have the maximum accuracy due to the property of $AP(\theta)$ in Equation (16).

Use of Validation Set for Generalizability. For a given $\mathcal{A}$, we treat λ as a hyperparameter that controls accuracy-fairness tradeoff. To avoid overfitting λ on the training set, we follow the standard practice in tuning hyperparameters in ML and use a separate validation set to evaluate the goodness of a λ value. In particular, we randomly split D into a D_{train} and a small D_{val}. We use D_{train} for training ML models for different λ values (i.e., when applying $\mathcal{A}$), and use D_{val} for evaluating ML models (i.e., when calculating $FP(\theta)$ and $AP(\theta)$). Our goal is to pick a value for λ such that the model trained with that λ value would have the maximum validation accuracy while satisfying the constraint on the validation set.

While the use of a separate validation set when tuning λ helps with the generalizability of the trained model, there is still no guarantee that the trained model will satisfy the declared fairness constraints on arbitrary unseen test sets. However, a larger validation set is better at ensuring that the accuracy and fairness numbers obtained on the validation set are close to those evaluated on a unseen test set, as shown empirically in Figure 3.

6 MULTIPLE FAIRNESS CONSTRAINTS

We consider the multiple fairness constraints case in this section. Given k constraints, the constrained optimization problem becomes:

$$\begin{aligned} \max_{\theta} \quad & AP(\theta) \\ \text{s.t.} \quad & |FP_i(\theta)| \leq \varepsilon \quad \forall i \in \{1..k\} \end{aligned} \tag{18}$$

The Lagrangian dual function [10] of Equation (18) is:

$$\begin{aligned} h(\Lambda_1, \Lambda_2) &= \max_{\theta} AP(\theta) + \sum_{i=1}^{k} \lambda_{1i}(\varepsilon - FP_i(\theta)) + \sum_{i=1}^{k} \lambda_{2i}(\varepsilon + FP_i(\theta)) \\ &= \max_{\theta} AP(\theta) + \sum_{i=1}^{k} (\lambda_{1i} - \lambda_{2i}) FP(\theta) + \sum_{i=1}^{k} (\lambda_{1i} + \lambda_{2i})\varepsilon \end{aligned} \tag{19}$$

,where $\Lambda_1 = \langle \lambda_{11}, \lambda_{12} ..., \lambda_{1k} \rangle$ and $\Lambda_2 = \langle \lambda_{21}, \lambda_{22} ..., \lambda_{2k} \rangle$.

$h(\Lambda_1, \Lambda_2)$ provides an upper bound for Equation (18) for any $\Lambda_1 > 0, \Lambda_2 > 0$ and the bound is tight under strong duality [10]. We can further simplify Equation (19) into Equation (20) by using half of the hyperparameters, namely, $\Lambda = \langle \lambda_1, \lambda_2 ..., \lambda_k \rangle$, similar to the single-constraint setting.

$$\max_{\theta} AP(\theta) + \sum_{i=1}^{k} \lambda_i \, FP_i(\theta) \tag{20}$$

$$= \max_{\theta} \frac{1}{N} \sum_{i=1}^{N} w_i(\Lambda, h_\theta) \mathbb{1}(h_\theta(x_i) = y_i) \tag{21}$$

The transformation from Equation (20) to Equation (21) is similar to what we did in § 5, which allows us to use any black-box ML algorithm $\mathcal{A}$ by adjusting example weights. The difference is that the weight for every example is now parameterized by Λ and θ.

Discussion on Feasibility. Given multiple fairness constraints, there may not exist an ML model that satisfies all constraints. Indeed, a well-known prior research has shown theoretically that no model, regardless of which ML algorithms are used, can achieve perfect ($\varepsilon = 0$) SP, FNR, and FOR, for any dataset [25]. Fortunately, we can show that our approximation of using Equation (20) still works with multiple constraints, if the origianl problem is feasible.

Lemma 3. *Assume Equation* (20) *is feasible, let* θ^* *be an optimal solution to Equation* (20)*, then*

(1) for any $\Lambda_1, \Lambda_2 > 0$, $h(\Lambda_1, \Lambda_2) \geq AP(\theta^*)$*; and under strong duality assumption* $\min_{\Lambda_1>0,\Lambda_2>0} h(\Lambda_1, \Lambda_2) = AP(\theta^*)$.

(2) for any $\Lambda_1, \Lambda_2 > 0$, *let* $\tilde{\theta}$ *be an optimal solution to Equation* (19)*, then there exists* $\Lambda \in \mathbb{R}^k$ *(i.e.,* $\Lambda = \Lambda_2 - \Lambda_1$*) such that* $\tilde{\theta}$ *also optimizes Equation* (20)*; and under strong duality assumption, there exists* Λ *such that* θ^* *optimizes Equation* (20).

Lemma 3 generalizes Lemma 1 from single constraint to multiple constraints, and the proof is in the full report [50]. Therefore, similar to the case of tuning λ for solving Equation (9), we need to tune Λ in Equation (21) for solving Equation (18).

6.1 Monotonicity for Multiple Constraints

For a single constraint, we used a monotonicity property in Lemma 2. We now introduce a similar property for multiple constraints.

Lemma 4. ***[Marginal Monotonicity for Multiple Fairness Constraints]*** *Consider two settings* Λ_1 *and* Λ_2 *that differ only in the* j^{th} *dimension, namely,* $\Lambda_1[j] < \Lambda_2[j]$ *and* $\Lambda_1[i] = \Lambda_2[i]$ *for all* $i \neq j$. *Let* θ_1 *and* θ_2 *denote the optimal solution to Equation* (20) *given* Λ_1 *and* Λ_2*, when trained on the same training set* D *respectively, namely,*

$$\theta_1 = \arg\max_{\theta} AP(\theta) + \sum_{i=1}^{k} \Lambda_1[i] FP_i(\theta) \tag{22}$$

$$\theta_2 = \arg\max_{\theta} AP(\theta) + \sum_{i=1}^{k} \Lambda_2[i] FP_i(\theta) \tag{23}$$

Then, we have the following monotonicity property, where $FP_j(\theta_1)$ *and* $FP_j(\theta_2)$ *are evaluated on the same training set* D,

$$FP_j(\theta_1) \leq FP_j(\theta_2) \tag{24}$$

The proof of Lemma 4 is similar to that of Lemma 2 by treating $AP(\theta) + \sum_{i=1, i\neq j}^{k} \Lambda_1[i] FP_i(\theta)$ and $FP_j(\theta)$ in Lemma 4 as the old $AP(\theta)$ and $FP(\theta)$ in Lemma 2, respectively.

Note that we have a "weaker" version of the monotonicity property in this case. We can only show that the $FP_j(\theta)$ is monotonically increasing with respect to j^{th} dimension in Λ given every other dimension stays fixed, and we can provide no guarantees for the $AP(\theta)$ or other $FP_i(\theta), i \neq j$.

Satisfactory regions. Consider a hyperparameter setting Λ and the j^{th} fairness constraint to be $|FP_j(\theta)| = 0$, the derived solution θ (by solving Equation (21)) may or may not satisfy $|FP_j(\theta)| = 0$. Suppose $|FP_j(\theta)| = 0$ is not satisfied, according to the marginal monotonicity property, we can always fix λ_j^- (which denotes all but the j^{th} dimension of Λ) and update λ_j such that $|FP_j(\theta)| = 0$. In other words, for any setting of λ_j^-, we can always find a corresponding setting for λ_j such that $FP_j(\theta) = 0$. We call all such Λ settings the *zero-satisfactory region* for $|FP_j(\theta)| = 0$.

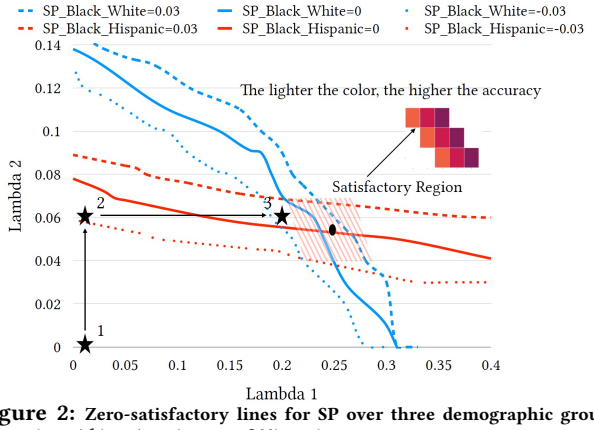


Figure 2: Zero-satisfactory lines for SP over three demographic groups Caucasian , AfricanAmerican , **and** Hispanic.

Now let us consider the general j^{th} constraint being $|FP_j(\theta)| \leq \varepsilon$. Similarly, for any setting of λ_j^-, we can find *a continuous range of possible settings* for λ_j, such that $|FP_j(\theta)| \leq \varepsilon$. We call all such Λ settings the *satisfactory region* for $|FP_j(\theta)| \leq \varepsilon$.

Geometrically, one can observe from marginal monotonicity property that every zero-satisfactory region for $FP_j(\theta) = 0$ is a $(k-1)$ dimensional curved hyperplane that intersects each axis-parallel line in exactly one point, and every satisfactory region for $FP_j(\theta) \leq \varepsilon$ is the intersection of two "parallel" half-spaces each identified by a $(k-1)$ dimensional curved hyperplane.

Example 5. *Figures 2 shows the satisfactory regions for an experiment on COMPAS dataset, with two fairness constraints (k = 2). Both constraints enforce statistical parity, one between African American and Caucasian groups and the other between African American and Hispanic groups; both have a disparity allowance of* $\varepsilon = 0.03$.

The solid blue line depicts the zero-satisfactory region for the first constraint, and the region between two blue dashed lines depict the satisfactory region for it. Similarly, the zero-satisfactory region and the satisfactory region for the second constraint is shown in orange.

All Λ *settings in the intersection between the blue band the the red band are are thus feasible solutions that satisfy both constraints.*

6.2 Algorithm for Tuning Λ

The (hypothetical) baseline solution for finding proper Λ is to search the entire space of possible values for Λ. We can do a grid search to do that. However, doing a grid search could be extremely slow especially when there are many constraints.

Algorithm 2 Hill-Climbing

Input: Dataset D, a set of k fairness constraint, and an ML Algorithm $\mathcal{A}$
Output: A fair ML model h_θ

1: $\Lambda \leftarrow [0, \cdots, 0]$
2: $\theta \leftarrow$ apply $\mathcal{A}$ with $w_i(0, -)$
3: **while** $\exists\ i$, s.t. $|FP_i(\theta)| > \varepsilon_i$ **do**
4: $\quad i \leftarrow \arg\max_k ||FP_k(\theta)| - \varepsilon_k|$
5: $\quad \theta \leftarrow$ call Algorithm 1 to tune for the i-th fairness constraint, while fixing $\Lambda[j], \forall j \neq i$ to their current values
6: **return** Λ **if** ($\Lambda \in$ intersection of all satisfactory regions) **else** "Not found after $5k$ iterations"

Following the marginal monotonicity property and our observations about the shape of satisfactory region, we design a marginal

hill-climbing algorithm for parameter tuning when there are multiple fairness constraints. At a high-level, the marginal hill-climbing algorithm (Algorithm 2) uses the satisfactory regions as the guidance to move toward their intersection. That is, at every step, instead of tuning all dimensions, it picks a marginal λ_j^- and tunes λ_j such that the j^{th} constraint is satisfied. While one can naively iterate over all dimensions in a round-robin fashion, at every step we pick the dimension whose fairness constraint is violated the most for faster convergence (Line 4). For a marginal λ_j^- at every iteration, the algorithm invokes Algorithm 1 for tuning λ_j so that the j^{th} constraint is satisfied. This process continues until all constraints are satisfied or a predefined number of iterations are reached (we use 5k in our experiments, where k is the number of constraints).

Note that the above procedure is implicitly maximizing the impact of loss of model accuracy. This is because, at every iteration, the algorithm satisfies the j^{th} constraint to its minimum (by invoking Algorithm 1). We empirically do observe that the smaller the update we do on λ_j, the less impact it has on the accuracy.

Example 6. *Figure 2 shows an instance of running Algorithm 2. Initially, we have* $\Lambda_1 = [0, 0]$ *(the star labeled as 1). The algorithm picks the second dimension (fixing the first dimension) and arrives at* $\Lambda_2 = [0, 0.06]$ *(the star labeled as 2). It next picks the first dimension (fixing the second dimension) and arrives at* $\Lambda_3 = [0.18, 0.06]$ *(the star labeled as 3), which satisfies both constraints. Notice that for all feasible solutions, i.e., in the intersection of both satisfactory regions, the closer* Λ *is to the origin point, the higher the model accuracy.*

While tuning the j^{th} dimension can certainly satisfy the j^{th} constraint, it may also impact other dimensions, in particular, it may cause other constraints that had previously been satisfied to be violated again. Indeed, we cannot prove that this greedy hill-climbing algorithm will always converge, and hence we set a maximum iteration threshold ($5k$). However, the experiments in § 7.3 show that empirically, Algorithm 2 can actually recover feasible solutions with a higher probability than grid search with a reasonable step size. We attribute this surprisingly good performance to two factors: (1) at every iteration, though we cannot guarantee tuning j^{th} constraint will not impact other constraints, we indeed are minimizing the potential impact by satisfying the j^{th} constraint to the minimum degree; and (2) Algorithm 2 invokes Algorithm 1 for tuning j^{th} dimension, which uses a binary search procedure that can search for extremely fine-grained values ($\tau = 0.0001$ in Algorithm 1).

7 EXPERIMENTS

We run experiments to evaluate the effectiveness and efficiency of OmniFair in comparison to existing approaches.

Table 4: Datasets used

Dataset	# Rows	# Attrs	Sens. Attr	Task
Adult [18]	48842	18	sex	To predict if Income > 50k
Compas [4]	11001	10	race	To predict recidivism
LSAC [45]	27477	12	race	To predict if bar exam is passed
Bank [32]	30488	20	age	To predict if marketing works

7.1 Experiment Setup

Datasets. We use four popular datasets in the algorithmic fairness literature that are known to have biases against minority groups. The detail of the datasets are shown in Table 4.

ML Algorithms. We include four ML algorithms, Logistic Regression (LR), Random Forest (RF), XGBoost (XGB), and Neural Networks (NN), to show that our system is model-agnostic. These are commonly used ML algorithms on structured data with completely different training processes — LR and NN has an explicit loss function and is trained via gradient descent; RF and XGB [13] has no explicit loss function and is trained by building decision trees.

Baselines. We compared with five baselines: two preprocessing approaches and three in-processsing approaches, shown in Table 1. We also compare with the fairness approach [43] that relies on a specific new ML algorithm CMA-ES developed within.

Hyperparameter Tuning. ML algorithm performances are known to be influenced by hyperparameters (e.g., learning rate, tree-depth, etc). For algorithmic fairness work, there is usually an additional hyperparameter that controls the accuracy-fairness trade-off (e.g., λ in OmniFair). A prior work FairPrep [41] showed that many existing algorithmic fairness work only focused on tuning the hyperparameter that controls the accuracy-fairness trade-off, and neglected to tune the traditional ML algorithm hyperparameters, which can strongly impact the reported results. We randomly split each dataset to 60% training, 20% validation, and 20% test. All hyperparameters (including both traditional ML hyperparameters that are specific to an ML algorithm and hyperparameters that are specific to fairness enforcement algorithms) are tuned using the validation set, and all reported results are on the unseen test set. All results reported are the average performance of 10 different random splits.

7.2 Single Fairness Constraint

In this section, we validate that our algorithm produces high quality results compared to other baselines using one fairness constraint given by a fairness specification (g, f, ε). g returns the groups that are defined on the sensitive attribute of each dataset. For the fairness metric f, we experiment with statistical parity (SP) and false discovery rate (FDR). We choose these two metrics because SP is representative for the fairness metrics whose induced example weights are NOT parameterized by the model θ, and FDR is representative for the fairness metrics whose induced example weights are parameterized by the model θ (c.f. Table 3). We will also include a customized fairness metric, as described in Example 4.

7.2.1 *Statistical parity as the fairness metric.*

Test accuracy comparison when $\varepsilon = 0.03$. Instead of reporting absolute accuracy numbers, we report the accuracy drop of different methods compared with no fairness constraints in Table 5. For example, the first number -1.2% in Table 5 means that, on the COMPAS dataset, the accuracy of the LR classifier obtained by OmniFair when enforcing the SP constraint with $\varepsilon = 0.03$ is -1.2% less than the accuracy of the LR classifier obtained without any fairness constraint. We highlight several observations: (1) OmniFair achieves the smallest accuracy drop in 8 out of 20 cases (4 datasets $\times$ 5 ML algorithms), and reduces the accuracy drop by up to $1 - 0.3/5.8 = 94.8\%$ compared with the second best method using the RF classifier on the LSAC dataset; (2) in the remaining 12/20 cases, OmniFair is always a close second best method apart from CMA-ES; (3) CMA-ES is an evolution strategy for optimization used by Thomas et al. [43] and it is not supported by all algorithms but Thomas et al. [43]; and (4) many existing methods either do not support some ML algorithms such as RF and XGBoost (i.e., NA(2))

Table 5: Accuracy drop compared with no fairness constraints when $\varepsilon = 0.03$ under SP. NA(1) means that no hyperparameter setting can be found to satisfy the SP constraint when $\varepsilon = 0.03$. NA(2) means that the classifier is not supported. NA(2)* means that theorectically it could be supported, but CMA-ES does not provide a clean API for invoking the ML algorithm.

	COMPAS					Adult					LSAC					Bank				
Algorithm	LR	RF	XGB	NN	CMA-ES	LR	RF	XGB	NN	CMA-ES	LR	RF	XGB	NN	CMA-ES	LR	RF	XGB	NN	CMA-ES
OmniFair	-1.2%	-0.8%	-0.7%	-1.2%	NA(2)*	-2.1%	-1.9%	-1.7%	-1.7%	NA(2)*	-0.3%	-0.3%	-0.4%	-0.1%	NA(2)*	-0.1%	-0.3%	-0.2%	-0.1%	NA(2)*
Kamiran et al. [28]	-2.5%	-1.3%	-1.2%	-1.5%	NA(2)*	-2.7%	-2.3%	-1.8%	-1.9%	NA(2)*	-0.4%	-5.6%	-2.2%	-0.4%	NA(2)*	+0.1%	-1.2%	-0.2%	-0.3%	NA(2)*
Calmon et al. [11]	-1.8%	-0.5%	-0.3%	-0.9%	NA(2)*	-3.7%	-3.1%	-3.0%	-2.4%	NA(2)	NA(1)	NA(1)	NA(1)	NA(1)	NA(2)*	NA(1)	NA(1)	NA(1)	NA(1)	NA(2)*
Zafar et al. [47]	-0.9%	NA(2)	NA(2)	NA(2)	NA(2)	-2.2%	NA(2)	NA(2)	NA(2)	NA(2)	-0.2%	NA(2)	NA(2)	NA(2)	NA(2)	-0.1%	NA(2)	NA(2)	NA(2)	NA(2)
Celis et al. [12]	NA(1)	NA(2)	NA(2)	NA(2)	NA(2)	NA(1)	NA(2)	NA(2)	NA(2)	NA(2)	NA(1)	NA(2)	NA(2)	NA(2)	NA(2)	NA(1)	NA(2)	NA(2)	NA(2)	NA(2)
Agarwal et at. [3]	-2.4%	-1.2%	-2.0%	-1.8%	NA(2)*	-2.8%	-2.2%	-2.0%	-2.0%	NA(2)*	-0.6%	-5.8%	-0.2%	-0.5%	NA(2)*	-0.1%	-0.3%	-0.0%	-0.1%	NA(2)*
Thomas et at. [43]	NA(2)	NA(2)	NA(2)	NA(2)	-1.1%	NA(2)	NA(2)	NA(2)	NA(2)	-1.7%	NA(2)	NA(2)	NA(2)	NA(2)	-0.4%	NA(2)	NA(2)	NA(2)	NA(2)	-0.1%

Figure 3: Accuracy and bias on the test set of the COMPAS dataset when varying the validation set size.

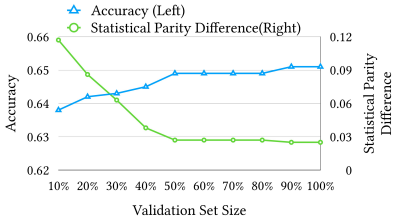
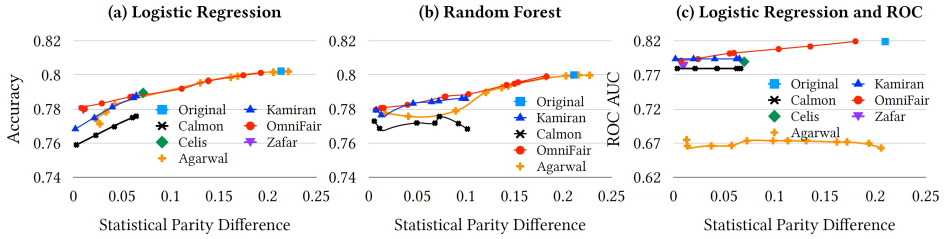


Figure 4: The trade-off between fairness metric and accuracy with (a)LR, (b)RF, and (c)LR (ROC AUC) on Adult dataset.



or fail to satisfy the fairness constraint (i.e., NA(1)). In particular, Celis et al. [12] is not able to generate a valid result at $\varepsilon = 0.03$.

Varying validation set size. While our default validation set size is 20% of the whole dataset, we conduct an ablation experiment to empirically observe the impact of validation set size on test accuracy and test bias. We vary the validation set size by using a subset of 20% validation split, and use the same training and test set. As shown in Figure 3, where the input fairness constraint is to enforce the SP difference between two groups to be within $\varepsilon = 0.03$, we can see that too small a validation set fails to generalize well (the test bias is bigger than 0.03); as we increase the validation set, the test set behavior stabilizes and its bias is close to 0.03.

Accuracy-Fairness Trade-off Comparison by Varying ε. To further understand the accuracy-fairness trade-off, we compare all methods by varying ε. We only show the results for Adult dataset for space reasons, and refer readers to the full report [50] for other datasets, which reveal similar findings. Figure 4 shows that OmniFair offers a much more flexible trade-off than existing methods in that it covers the entire span of different bias levels (i.e., the x-axis), while almost all baselines, except Agarwal et al [3], do not cover the full x-axis. This is because all baselines, is spite of providing various knobs that can influence the final model, do not offer guarantees for how those knobs affect the trade-off. For example, for Zafar et al, while we tuned the provided hyperparameter extensively, we discovered that the best model is always the same for different ε (hence, only one point in Figure 4(a)). OmniFair, on the other hand, has a hyperparameter λ that monotonically controls the accuracy-fairness trade-off. Compared with Agarwal et al [3], which offers a comparably flexible trade-off, OmniFair achieves better accuracy, especially when ε is small, as shown in Figure 4. Given that the adult dataset has imbalanced labels (76% negative), we also report ROC AUC in Figure 4(c): while Agarwal et al has a low ROC AUC on adult dataset, OmniFair is able to achieve both high accuracy and high ROC AUC while satisfying fairness constraints.

Efficiency Comparison. Figure 5 shows the comparison between the running time of different algorithms on Adult, COMPAS and LSAC datasets for statistical parity with LR. We only show the result for logistic regression here because it is supported by all baselines. The results for RF and XGB are similar, and could be found in our full report [50]. Compared with existing pre-processing methods, OmniFair achieves comparable running time. However, OmniFair supports a much wider set of constraints, while pre-processing methods only support SP. Compared with existing in-processing methods, OmniFair is not only faster (from 1.05× to 270×), but also achieves better accuracy-fairness trade-off (c.f. Figure 4). This is because in-processing methods such as Zafar et al. and Celis et al. require nontrivial modifications to ML training procedures. Compared with Agarwal et al., which also assumes black-box ML algorithms, OmniFair is about 10X faster due to our efficient hyperparameter tuning using the monotonocity property while Agarwal et al. needs to solve a difficult saddle point finding problem.

We also implement a warm-start optimization for the LR algorithm (which is also applicable to NN) by using parameter values learned in a previous invocation of $\mathcal{A}$ as the initialization of parameter values in the next invocation of $\mathcal{A}$. As shown in Table 6, warm-start leads to an 1.2× to 3.4× speed up on different datasets.

7.2.2 *False Discovery Rate and Customized Metric*

Accuracy-Fairness Trade-off Comparison by Varying ε. Similar to SP, we do the analysis of the trade-off between accuracy and fairness by varying ε. We show that even for metrics like false discovery rate where the weight is a function of θ, we are able to use our algorithm to achieve fairness. Among the five baselines, only Celis is able to support this. As shown in Figure 7, our algorithm is able to reduce the false discovery rate difference while having little accuracy drop, and significantly outperform Celis. We show that for a customized metric error cost (AEC) as described in § 4.3,

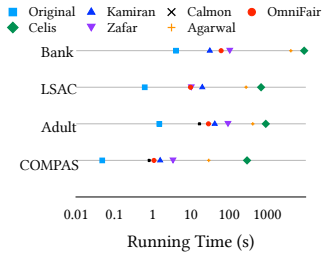


Figure 5: Running time comparison under SP constraint and LR classifier.

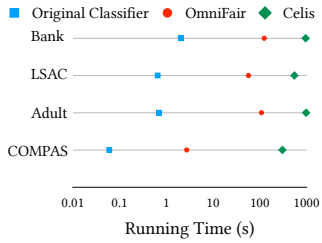


Figure 6: Running time comparison under FDR constraint and LR classifier.

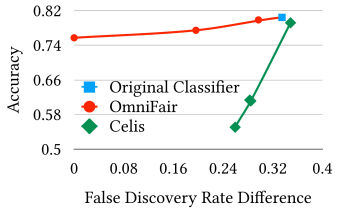


Figure 7: The accuracy-fairness trade-off under FDR constraint and LR classifier on Adult dataset.

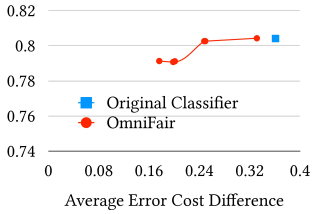


Figure 8: The accuracy-fairness trade-off under AEC constraint and LR classifier on Adult dataset.

Table 6: The speed up for Warm Start under LR. Time is shown in seconds

Dataset	No Warm Start (s)	Warm Start (s)	SpeedUp
Compas	1.1	0.8	1.4×
Adult	28.7	15.9	1.8×
LSAC	9.8	8.2	1.2×
Bank	25.6	7.5	3.4×

Figure 9: Enforcing SP for all three groups

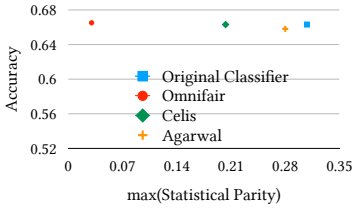


Table 7: Enforcing SP and FNR

ε	Accuracy	SP	FNR
Baseline	63.9%	0.233	0.180
0.01	N/A	N/A	N/A
0.02	N/A	N/A	N/A
0.03	63.6%	0.03	0.044
0.04	63.4%	0.016	0.035
0.05	63.3%	0.028	0.007
0.06	62.7%	0.057	0.032

Table 8: Grid vs hill-climbing (HC).

ε	Grid	HC	Grid Time	HC Time
0.01	No	No	96s	9s
0.02	No	No	96s	8s
0.03	No	Yes	96s	7s
0.04	Yes	Yes	96s	7s
0.05	Yes	Yes	96s	7s
0.06	Yes	Yes	96s	7s

we are able to use our algorithm to reduce the bias while keeping similar accuracy to the baseline, as shown in Figure 8.

Efficiency Comparison. Figure 6 shows the comparison between running time of different algorithms on Adult, COMPAS and LSAC datasets for false discovery rate with LR. We can see that for false discovery rate, we are 9× to 150× faster than Celis et. al., the only algorithm we found that could support false discovery rate.

7.3 Multiple Fairness Constraints

We validate our algorithm when there are multiple fairness constraints. As discussed before, one fairness specification (g, f, ε) can already generate multiple constraints if there are more than two groups in $g(D)$. Users could also use multiple fairness specifications to specify multiple constraints under different fairness metrics. We evaluate both cases. We also compare our algorithm with the naive grid search for hyperparameter tuning.

Multiple groups under one fairness metric. We train LR classifier on the COMPAS dataset. However, instead of considering only two groups: Black and White, we consider three groups: Black (B), White (W) and Hispanic (H). We want to minimize the statistical parity difference between any two groups.

All baseline algorithms do not support this by default. We went through the code for all algorithms, and managed to adapt Celis et al. and Agarwal et al.'s code to support this.

The comparison is shown in Figure 9. The x-axis is SP_{max}, the maximum value of SP difference among three groups, i.e., $SP_{max} = max(SP_{BW}, SP_{BH}, SP_{HW})$ and y-axis is accuracy. We can see that, although theoretically Celis and Agarwal can support multiple groups, they fail to reduce the SP difference among all three groups (the SP_{max} is still greater than 0.20). On the other hand, OmniFair can reduce SP_{max} to 0.03, while keeping high accuracy.

Multiple fairness metrics. We run the experiments on COMPAS dataset considering two fairness metrics: SP and FNR.

Theoretically, this could be supported by Celis and Agarwal; however, we failed to modify their code to support multiple metrics after spending two days on it as constraints are hard-coded into the optimization procedure. In contrast, OmniFair supports multiple metrics without any additional coding efforts.

As we can see in Table 7, for $\varepsilon = 0.01$ and $\varepsilon = 0.02$, our algorithm is not able to find a result. This is because we cannot find a set of weights such that ε is satisfied for both constraints in the validation set. In these cases, neither could grid search return a feasible answer. However, for $\varepsilon >= 0.03$, we can see that the fairness difference for both fairness constraints drop by one order of magnitude, and the accuracy loss is minimum ($< 1\%$).

Efficiency Optimization. We compare the hill-climbing algorithm with grid search in terms of efficiency and the ability to discover feasible solutions. We run the experiments on the COMPAS dataset and consider two fairness constraints: statistical parity and false negative rate with different degrees of disparity allowance ε. As we can see in Table 8, when Grid Search finds a feasible solution, the hill-climbing algorithm can always find one as well. Also, the hill-climbing algorithm runs around 10× than grid search.

8 CONCLUSION AND FUTURE WORK

In this paper, we propose OmniFair that allows users to declaratively specify group fairness constraints in ML. OmniFair is versatile in that it supports all major group fairness constraints, including customized ones and multiple constraints simultaneously, and it is model-agnostic. The algorithms in OmniFair maximize for accuracy while satisfying given fairness constraints. The algorithm designs are optimized based on monotonocity properties for accuracy and fairness unique in our system. Experimental results demonstrate that OmniFair supports more fairness metrics than existing methods, and can achieve better accuracy-fairness trade-off without sacrificing efficiency. Future work includes (1) improving the scalability of OmniFair, which could potentially be addressed by parallel model training and using a smaller sample training set to quickly prune certain λ values.; and (2) providing theoretical guarantees regarding constraint satisfiability on unseen test sets.

REFERENCES

[1] False positives, false negatives, and false analyses: A rejoinder to "machine bias: There's software used across the country to predict future criminals. and it's biased against blacks.". *US Court*, 2016.

[2] Response to propublica: Demonstrating accuracy equity and predictive parity. *Equivant*, 2018.

[3] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach. A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*, 2018.

[4] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias: Risk assessments in criminal sentencing. *ProPublica*, 2016.

[5] A. Asudeh, Z. Jin, and H. Jagadish. Assessing and remedying coverage for a given dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 554–565. IEEE, 2019.

[6] P. Bailis, K. Olukotun, C. Ré, and M. Zaharia. Infrastructure for usable machine learning: The stanford dawn project. *arXiv preprint arXiv:1705.07538*, 2017.

[7] S. Barocas, M. Hardt, and A. Narayanan. Fairness and machine learning: Limitations and opportunities. fairmlbook.org, 2019.

[8] R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, page 0049124118782533, 2018.

[9] M. Boehm, A. Kumar, and J. Yang. Data management in machine learning systems. *Synthesis Lectures on Data Management*, 11(1):1–173, 2019.

[10] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, pages 3992–4001, 2017.

[12] L. E. Celis, L. Huang, V. Keswani, and N. K. Vishnoi. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 319–328, 2019.

[13] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[14] N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu. Goggles: Automatic image labeling with affinity coding. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 1717–1732, New York, NY, USA, 2020. Association for Computing Machinery.

[15] J. Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. reuters (2018), 2018.

[16] W. Dieterich, C. Mendoza, and T. Brennan. Compas risk scales: Demonstrating accuracy equity and predictive parity. *Northpointe Inc*, 2016.

[17] J. Dressel and H. Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.

[18] D. Dua and C. Graff. UCI machine learning repository, 2017.

[19] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[20] C. Dwork and C. Ilvento. Group fairness under composition. *FATML*, 2018.

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

[22] M. Fernandez. Study finds disparities in mortgages by race. *New York Times*, 15, 2007.

[23] A. W. Flores, K. Bechtel, and C. T. Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *Fed. Probation*, 80:38, 2016.

[24] J. Foulds and S. Pan. An intersectional definition of fairness. *CoRR:1807.08362*, 2018.

[25] S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian. On the (im) possibility of fairness. *arXiv preprint arXiv:1609.07236*, 2016.

[26] S. Goel, J. M. Rao, R. Shroff, et al. Precinct or prejudice? understanding racial disparities in new york city's stop-and-frisk policy. *The Annals of Applied Statistics*, 10(1):365–394, 2016.

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

[28] F. Kamiran and T. Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

[29] M. Kearns, S. Neel, A. Roth, and Z. S. Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 100–109, 2019.

[30] J. Kleinberg, S. Mullainathan, and M. Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

[31] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12):948–959, 2016.

[32] S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[33] S. Nakandala and A. Kumar. Vista: Optimized system for declarative feature transfer from deep cnns at scale. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1685–1700, 2020.

[34] A. Narayanan. Tutorial: 21 fairness definitions and their politics, 2018. *URL https://www. youtube. com/watch*, 2018.

[35] A. Olteanu, C. Castillo, F. Diaz, and E. Kiciman. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data*, 2:13, 2019.

[36] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1723–1726, 2017.

[37] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[38] E. K. Rezig, L. Cao, M. Stonebraker, G. Simonini, W. Tao, S. Madden, M. Ouzzani, N. Tang, and A. K. Elmagarmid. Data civilizer 2.0: A holistic framework for data preparation and analytics. *Proceedings of the VLDB Endowment*, 12(12):1954–1957, 2019.

[39] C. Russell, M. J. Kusner, J. Loftus, and R. Silva. When worlds collide: integrating different counterfactual assumptions in fairness. In *Advances in Neural Information Processing Systems*, pages 6414–6423, 2017.

[40] B. Salimi, L. Rodriguez, B. Howe, and D. Suciu. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*, pages 793–810, 2019.

[41] S. Schelter, Y. He, J. Khilnani, and J. Stoyanovich. Fairprep: Promoting data to a first-class citizen in studies on fairness-enhancing interventions. In *Proceedings of the 23rd International Conference on Extending Database Technology (EDBT)*, 2020.

[42] L. Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10–29, 2013.

[43] P. S. Thomas, B. C. da Silva, A. G. Barto, S. Giguere, Y. Brun, and E. Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

[44] S. Verma and J. Rubin. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 1–7. IEEE, 2018.

[45] L. F. Wightman and H. Ramsey. *LSAC national longitudinal bar passage study*. Law School Admission Council, 1998.

[46] D. Xin, S. Macke, L. Ma, J. Liu, S. Song, and A. Parameswaran. Helix: holistic optimization for accelerating iterative machine learning. *Proceedings of the VLDB Endowment*, 12(4):446–460, 2018.

[47] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pages 1171–1180, 2017.

[48] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

[49] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

[50] H. Zhang, X. Chu, A. Asudeh, and S. B. Navathe. Omnifair: A declarative system for model-agnostic group fairness in machine learning. *arXiv preprint*, 2021.