A Theory of Learning with Competing Objectives and User Feedback

Pranjal Awasthi[1], **Corinna Cortes**[1], **Yishay Mansour**[1, 2], **Mehryar Mohri**[1, 3],

[1]Google Research; [2]Tel-Aviv University; [3]Courant Institute of Mathematical Sciences.
pranjalawasthi@google.com, corinna@google.com, mansour@google.com, mohri@google.com.

Abstract

Large-scale deployed learning systems are often evaluated along multiple objectives or criteria. But, how can we learn or optimize such complex systems, with potentially conflicting or even incompatible objectives? How can we improve the system when user feedback becomes available, feedback possibly alerting to issues not previously optimized for by the system? We present a new theoretical model for learning and optimizing such complex systems. Rather than committing to a static or pre-defined tradeoff for the multiple objectives, our model is guided by the feedback received, which is used to update its internal state. Our model supports multiple objectives that can be of very general form and takes into account their potential incompatibilities. We consider both a stochastic and an adversarial setting. In the stochastic setting, we show that our framework can be naturally cast as a Markov Decision Process with stochastic losses, for which we give efficient vanishing regret algorithmic solutions. In the adversarial setting, we design efficient algorithms with competitive ratio guarantees. We also report the results of experiments with our stochastic algorithms validating their effectiveness.

1 Introduction

Learning algorithms trained on large amounts of data are increasingly adopted in a variety of applications and form the engine driving complex large-scale systems such as e-commerce platforms, online advertising auctions and recommender systems. Their system designer must take into account multiple metrics when optimizing them (Kaminskas and Bridge 2016; Masthoff 2011; Lin et al. 2019). As an example, consider the case of a recommendation system for recipes, videos or fashion. There is no single metric that defines what a good recommendation engine should do. One needs to carefully take into consideration metrics measuring the quality of recommendations provided to end-users, their relevance and utility, the long-term growth of the content creators, and the overall revenue generated for the hosting platform. Furthermore, it is crucial to consider the risk of bias in these systems (Speicher et al. 2018; Xiao et al. 2017; Holstein et al. 2019). Hence, additional metrics may need to be incorporated, such as performance across demographic groups, geographical locations or other identity terms. This can easily lead to hundreds of metrics that need to be simultaneously optimized for user satisfaction.

Further complicating the above scenario is the fact that often the multiple metrics considered are incompatible and inherently in conflict with each other (Kleinberg, Mullainathan, and Raghavan 2017; Sener and Koltun 2018; Jin 2006). For instance, in the context of a recommendation system, there is a tension between maximizing revenue via ad placements and maximizing end-user "happiness". Another tension may be between maximizing quality versus diversity of recommendations. In many cases, resolving such conflicts may force the designer to make hard choices among notions that seem perfectly reasonable in isolation, weighing in current use-patterns, wins and losses. An illuminating example is the analysis of the COMPAS tool for predicting recidivism by (Angwin et al. 2019). The authors showed that, among black defendants who do not recidivate, the tool predicted incorrectly at twice the rate than it did for white defendants who did not recidivate, i.e., the tool was unfair according to the *false positive rate* metric. The creator of the tool, Northpointe, responded by demonstrating that the tool was fair according to other natural measures such as AUC (Area Under the ROC Curve), for which each group had similar values. Later work showed that this tension is inherent and that it is often impossible to simultaneously satisfy multiple seemingly natural criteria (Kleinberg, Mullainathan, and Raghavan 2017) (see also (Feller et al. 2016)).

The above discussion raises the question of how one should define the optimal trade-off among multiple conflicting metrics to optimize for user satisfaction. A natural approach is to define the trade-offs in a static manner, either by using domain knowledge and human expertise, or by analyzing past historical data. Another line of work studies optimization in the presence of multiple objectives by designing algorithms that compete with *any* linear combination of the objectives (Mohri, Sivek, and Suresh 2019; Cortes et al. 2020) or by designing pareto-optimal solutions (Sener and Koltun 2018; Shah and Ghahramani 2016). However, these solutions may be sub-optimal for the richer situation where user feedback is available. While algorithms tailored to a specific metric or a combination of metrics would be effective at first, experience shows that they become non pertinent over time: once a system is deployed and it interacts with its end-users, inefficiencies in the system design emerge, as evident via the user feedback, which in turn could lead one to prefer metrics originally not accounted for (Liu et al. 2018). In the context of the

COMPAS tool discussed above, this would correspond to the situation where user complaints make the system designers change loss function to ensure equal false-positive rates. The important aspect is that the underlying data distribution on which the tool has been trained doesn't change, new user feedback simply alerts the designers to short-comings of the system. Motivated by the above, in this work, we present a theoretical data-driven model for optimizing multiple conflicting metrics by taking into account the user feedback. Our proposed framework allows for the design of algorithmic solutions with strong theoretical guarantees.

In the context of a recommender system, user initiated feedback may be a "dislike", "too spicy", or "age inappropriate" (Chen and Pu 2012), but feedback may also be indirectly observed by, e.g., high abandonment rates or low click-through rates. Going from complaints to actionable solutions involves many steps. First, the complaints are analyzed, typically by human specialists, and attributed to a set of predefined criteria, such as low accuracy of classifiers, false positive rates or AUC scores. Each complaint could trigger several criteria and a human specialist can monitor the aggregate performance on each criterion. Since criteria are often incompatible, based on the analysis of the complaints and their effect on the criteria, a decision is made to allocate resources to improve a subset of them and this process repeats (Yu et al. 2020). While human involvement is crucial in the above process, a large portion of the above process could be made algorithmic and automated.

Our model assumes predetermined costs for user complaints along the multiple metrics. The difficulty in optimizing for user happiness arises from the fact that the nature and volume of the complaints depend on the state of the model. Of course if no complaints is received, an optimal state has been reached, but most often complaints will arise. Fixing the model to optimize for this set of complaints will most likely spur a different set of complaints, etc. Only by visiting all incompatible states of the model and observing the associated complaint set would one be able to fully optimize the model. Such an exhaustive search is prohibitive from both a time and development perspective. This paper presents a model that effectively reaches a beneficial state and provides performance guarantees.

The rest of the paper is organized as follows. In Section 2, we define our model. In the stochastic setting (Section 3), we show that our framework can be cast as a Markov Decision Process with stochastic losses, for which we give efficient vanishing regret algorithmic solutions. We also further discuss our modeling assumptions and extensions. The adversarial setting is discussed in Appendix D. Here, we give algorithms with competitive ratio guarantees. Section 4 demonstrates how our framework can be realized in practice and reports the results of experiments with our algorithms in the stochastic setting that demonstrate their effectiveness and the applicability of our model. We also defer the related work discussion as well as many of the proofs to the appendix.

2 Conflict Resolution Model

We consider optimization in the presence of multiple criteria, where not all criteria can be satisfied simultaneously. The constraints are specified by an undirected graph $\mathcal{G} = (V, E)$, where each vertex represents a criterion and where an edge between vertices v_i and v_j indicates that criteria v_i and v_j cannot be simultaneously satisfied. We denote by $V = \{v_1, \ldots, v_k\}$ the set of k criteria considered. Figure 1 illustrates these definitions. Note that vertices may represent joint criteria as in Figure 1(b).

We consider a machine learning system that evolves over a sequence of time steps in the presence of the criteria represented by the graph $\mathcal{G}$. At each time step t, the system is in some state s_t characterized by its performance on all criteria in V. Note, a state is distinct from a vertex of $\mathcal{G}$. The system then receives a new batch of feedback that depend on its current state and incurs a loss. The objective of the algorithm is to minimize the total cost incurred over a period of time, which includes the total loss accrued, as well as the total cost of fixing various criteria over that period. We envision that the algorithm is solving a constraint optimization problem with the criteria as constraints.

The assignment of a complaint to criteria can be achieved by human analysis or via a multi-class multi-label classifier trained on past data and making use of known classifiers for specific criteria. Even when a complaint is related to a single criterion, we do not simply advocate taking that raw feedback as the ground truth. We discuss the risks associated with doing so in Section 3 and Appendix F, in the context of the COMPAS example. To further improve and maintain the accuracy of this multi-class multi-label classifier, in practice, there may be ongoing data labeling and assistance by expert auditors analyzing complaints. Note that not all complaints received by the system are relevant and the classifier, or a human in the loop, may decide to not assign a complaint to any criterion. This also helps protect the system against potential attacks by coordinated users. Recent work on interactive models for ML fairness has studied this for specific metrics and auditor behavior (Bechavod, Jung, and Wu 2020).

Loss. As a result of the complaints, the system incurs a loss and responds by changing its state. The definition of the loss, which depends on the criteria affected by the complaints is critical, a poor choice can yield a so-called *loudest voice* effect (see discussion in Section 3). The notion of complaints and the associated loss may seem abstract at this point. In Section 4, we demonstrate how our model can be applied in practice.

Graph and criteria. The assumption that the graph $\mathcal{G}$ is known a priori may seem restrictive. However, in many settings, graph $\mathcal{G}$ can be derived from analyzing past complaints and by measuring how fixing one criterion affects the performance on others. For instance, in the recommendation system example discussed above, where each metric corresponds to the false positive rate on a different slice of the data, one can easily use past data to see how optimizing the false positive rate on one slice affects the other and get the graph of incompatibilities. See the experiments in Section 4 for a more concrete example. Our model also provides the flexibility of accounting for incompatibilities among criteria such as those discussed by (Kleinberg, Mullainathan, and Raghavan 2017) and (Feller et al. 2016). This can be achieved by augmenting the graph with vertices representing joint criteria as in Figure 1(b). The graph stipulates in particular that v_1, v_2 and v_3

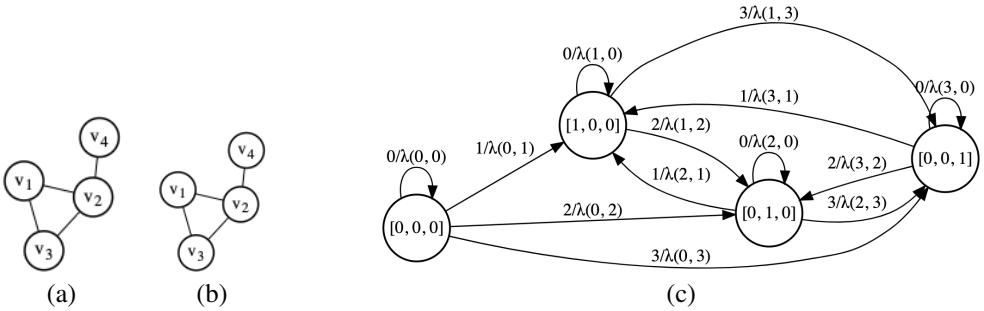

Figure 1: **(a)** Illustration of constraints graph $\mathcal{G}$. Vertices v_1, v_2, v_3, v_4 represent 4 different criteria. **(b)** More generally, each vertex can represent a joint criterion, for example $v_1 \wedge v_2$. This helps specify joint constraints such as the following: v_1, v_2, and v_3 cannot be simultaneously satisfied. **(c)** Illustration of the MDP for a fully connected incompatibility graph $\mathcal{G}$ over three criteria. The state set is $\mathcal{S} = \{\mathbf{0} = [0,0,0], \mathbf{1} = [1,0,0], \mathbf{2} = [0,1,0], \mathbf{3} = [0,0,1]\}$, the action set $\mathcal{A} = \{0,1,2,3\}$. Each transition is labeled with $a/\lambda(s,a)$, where a is the action taken from s and where $\lambda(s,a)$ is the total loss incurred as a result.

cannot be all simultaneously satisfied.

States. We will adopt the following simplifying assumptions and will later discuss their extensions or relaxation in Section 3. We assume that each criterion can only be in one of two states: *fixed*, meaning that criterion v_i is met or is not violated, or *unfixed*, meaning the opposite. Hence, the overall state of the system can be described by a k-dimensional Boolean vector. An action corresponds to fixing a particular criterion, or set of criteria, and moving to a different state. *Fixing* the criteria associated to v_i entails an algorithmic and resource allocation cost that we denote by c_i. Initially, all criteria are unfixed. At each time step, a conflict resolution system or algorithm selects some action, which may be to fix an unfixed vertex v_i, thereby incurring the cost c_i and *unfixing* any vertex adjacent to v_i, or the algorithm may select the null action, not to fix or unfix any vertex and wait to collect more data. Note that the incompatibilities in our model defined via edges in the graph are data agnostic. In practice, it is possible that two generally incompatible criteria can be simultaneously satisfied for a given dataset, say via incorporating a slack. This is a direction for future work.

Fixing costs. This can be estimated from past experience. In the absence of any prior information, one could assume a unit fixing cost. We deliberately avoid making specific choices. This gives us flexibility in dealing with different types of metrics in a unified manner. While the focus of our study is theoretical, let us emphasize that our model is easily applicable and implementable in practice. We further discuss this in the end of Section 3 and illustrate it in Section 4.

3 Stochastic Setting

We first detail a stochastic setting of our model that can be described in terms of a Markov Decision Process (MDP). Next, we present algorithms with strong regret guarantees.

Description. The distribution of complaints received by the system is a function of its current *state*, i.e., the current set of fixed or unfixed criteria v_i. Thus, we consider an MDP with a state space $\mathcal{S} \subseteq \{0,1\}^k$ representing the set of bit vectors for criteria: a state $s \in \{0,1\}^k$ is defined by $s(i) = 0$ when criterion v_i is unfixed and $s(i) = 1$ when it is fixed. By definition of the incompatibility graph $\mathcal{G}$, s is a valid state iff the set of fixed criteria at s is an independent set of $\mathcal{G}$.

When in state $s \in \mathcal{S}$, the system incurs a loss ℓ_i^s due to complaints related to criterion $i \in [k]$. Loss ℓ_i^s is a random variable assumed to take values in $[0, B]$ with mean μ_i^s. We do not assume independence across criteria, i.e., ℓ_i^s and ℓ_j^s may be dependent for a given state s. The action set is $\mathcal{A} = \{0, 1, \ldots, k\}$. A non-zero action i corresponds to fixing criterion i. Action 0 is the null action, i.e., no criterion is fixed. Transitions are deterministic: given state s and action $i \in \mathcal{A}$, the next state is s if $i = 0$, otherwise, for $i \neq 0$ the next state is the state s' that only differs from s by $s'(i) = 1$ and (possibly) $s'(j) = 0$ for all $j \in N(i)$, where $N(i)$ is the neighbors of v_i in $\mathcal{G}$, since neighbors of i must be unfixed once i is fixed.

Each action $a = i$ admits a fixing cost c_i. The cost for unfixing, as well as the null action, is zero. The loss incurred when taking action a at state s is the sum of the fixing cost c_a and the complaint losses at the (possibly) next state s': $\lambda(s,a) = c_a + \sum_{i=1}^k \ell_i^{s'}$. The expected loss of transition (s, a, s') is:

$$\mathbb{E}\left[c_a + \sum_{i=1}^k \ell_i^{s'}\right] = c_a + \sum_{i=1}^k \mu_i^{s'}. \tag{1}$$

Note, c_a and the losses $\ell_i^{s'}$ are observed by the algorithm, but the mean values $\mu_i^{s'}$ are unknown. To keep the formalism simple we assume that the cost c_a of taking an action a is independent of the current state s.

Figure 1(c) illustrates our stochastic model for three mutually incompatible criteria. The notion of each metric in a binary state is a simplifying modeling assumption for our theory. We discuss this more at the end of this section.

Correlation sets. In practice, the distribution of complaints related to a criterion v_i at two different states may be related. To capture these correlations in a general way, we assume that a collection $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ of *correlation sets* is given, where each $\mathcal{C}_j$ is a subset of the k criteria and has size at most m. By allowing correlation sets of varying sizes, we can capture a range of dependencies that may exist between different criteria. These dependencies affect the loss observed by the algorithm at each time.

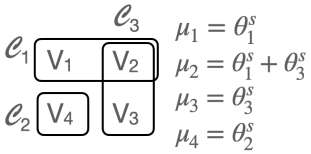

Figure 2: Example of correlation sets and associated losses for a graph with four criteria.

We assume that at a given state s, each set $\mathcal{C}_j$ generates losses with mean value θ_j^s per vertex, and that if two states s and s' admit the same configuration for the vertices in $\mathcal{C}_j$, then they share the same parameter $\theta_j^s = \theta_j^{s'}$. Given a criterion i and a state s, we assume that the loss incurred by criterion i equals the sum of the individual losses due to each correlation set $\mathcal{C}_j$ that contains i. Thus, μ_i^s can be expressed as follows: $\mu_i^s = \sum_{j=1}^n \theta_j^s \mathbb{I}(i \in \mathcal{C}_j)$. If a criteria is not correlated with any other vertex, we add to $\mathcal{C}$ a correlation set of size one for that criterion. See Figure 2 for an illustration. For each $j \in [n]$, there are at most 2^m configurations for the vertices of $\mathcal{C}_j$ in a state s, hence there are at most $2^m n$ distinct parameters θ_j^s. Let $\boldsymbol{\theta}$ denote the vector of all distinct parameters θ_j^s. Our MDP model can then be denoted MDP$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$.

Algorithm. We consider an online algorithm that at time t takes action a_t from state s_t and reaches state s_{t+1}, starting from the initial state $(0, \ldots, 0)$. settings, The objective of an algorithm can be formulated as that of learning a policy, that is a mapping $\pi: \mathcal{S} \to \mathcal{A}$, with a value close to that of the optimal. We are mainly interested in the cumulative loss of the algorithm over the course of T interactions with the environment. The goal is to minimize the pseudo-regret:

$$\mathsf{Reg}(\mathcal{A}) = \sum_{t=1}^{T} \mathbb{E}\Big[\lambda_t(s_t, a_t)\Big] - \sum_{t=1}^{T} \mathbb{E}\Big[\lambda_t(s_t^{\pi^*}, \pi^*(s_t^{\pi^*}))\Big],$$

where $\lambda_t(s, a)$ is the total loss incurred by taking action a at state s at time t, $s_1 = (0, \ldots, 0)$ and π^* is the optimal policy. Note, λ_t is only a function of the current state and the action taken. The expectation is over the random generation of the complaint losses. Given the correlation sets and the parameter $\boldsymbol{\theta}$, the optimal policy π^* corresponds to moving from the initial state $(0, \ldots, 0)$ to the state $s^* \in \mathcal{S}$ with the most favorable distribution and remaining at s^* forever. We define by $g(s)$ the expected (per time step) loss incurred by staying in state s, that is, $g(s) \coloneqq \sum_{i=1}^k \mu_i^s$. The optimal state s^* is then defined as follows:

$$s^* = \operatorname*{argmin}_{s \in \mathcal{S}} g(s). \tag{2}$$

Note, in this definition we disregard the one-time cost of moving to a state from the initial state, since in the long run the expected cost incurred by staying at a given state governs the choice of the optimal state. Since our problem can be seen as that of learning with a deterministic MDP with stochastic losses, we could adopt an existing algorithm for that problem (Jaksch, Ortner, and Auer 2010). However, the running-time of such algorithms would directly depend on the size of the state space $\mathcal{S}$, which here is exponential in k, and that of the action set $\mathcal{A}$. Furthermore, the regret guarantees of these algorithms would also depend on $|\mathcal{S}||\mathcal{A}|$. Instead, by exploiting the structure of the MDP, we can design vanishing regret algorithms with a computational complexity that is only polynomial in k and the number of parameters. We will assume access to an oracle that, given $\boldsymbol{\theta}$, can optimize (2). In Appendix B, we show how to approximately solve (2) for the case of singleton correlation sets, where the true parameters $\boldsymbol{\theta}$ can also be estimated efficiently (see Theorem 4).

There are important distinctions between our proposed model and the standard online learning setting. We have a notion of a state that evolves as a result of the stochastic losses suffered. These losses in turn depend on the distribution of complaints received. This distribution should not be confused with the distribution of data over which classifiers may be trained to fix a criterion. The latter is assumed constant in time. Via correlation sets we can model complex correlations among the criteria when defining the stochastic losses.

Case $m = 2$. To illustrate the ideas behind our general algorithm, we first consider a simpler setting where correlation sets are defined on subsets of size at most two. This setting also captures an important case where fixing a particular criterion affects the complaints of its neighbors. The algorithmic challenge we face here is to avoid exploring the exponentially many states in the MDP. Instead, we will design an algorithm that spends an initial exploration phase by visiting a specific subset of states of size at most $4n$. This subset denoted by $\mathcal{K}$, that we call a *cover* of $\mathcal{C}$ will help the algorithm estimate the expected loss of any state in the MDP given the estimates of losses for states in the cover. After the exploration phase, the algorithm creates an estimate $\hat{\boldsymbol{\theta}}$ of the true parameter vector $\boldsymbol{\theta}$, uses the optimization oracle for solving Eq. (2) to find a near optimal state $\hat{s}$ and selects to stay at state $\hat{s}$ for the remaining time steps. We next define the cover.

For two criteria i, j and $b \in \{0, 1\}$, we say that (i, j, b) is a *dichotomy* if there exist two states $s, s' \in \mathcal{S}$ such that: (1) $s(j) = 0$ and $s'(j) = 1$, and (2) $s(i) = s'(i) = b$. We call the two states s, s' an (i, j, b)-pair. Note that if an edge (v_i, v_j) is present in $\mathcal{G}$, then $(i, j, 1)$ cannot be a dichotomy, since criteria i and j cannot be fixed simultaneously. A cover $\mathcal{K}$ of $\mathcal{C}$ is simply a subset of the states in the MDP that contains an (i, j, b)-pair for every $\{i, j\} \in \mathcal{C}$ and valid dichotomy (i, j, b).

Furthermore, for every singleton set $\{i\}$ in $\mathcal{C}$, $\mathcal{K}$ contains states s, s' such that $s(i) = 0, s'(i) = 1$ and $s(j) = s'(j)$ for all $j \neq i$. Note that we only need the cover to contain an (i, j, b)-pair if $\{i, j\}$ is a correlation set. Hence, it is easy to see that when $m = 2$, there is always a cover of size at most $4n$. Next, we state our key result that estimating the loss values for the states in a cover is sufficient.

Theorem 1. *Let $\mathcal{K}$ be a cover for $\mathcal{C}$. For any state $s \in \mathcal{S}$ and any $i \in [k]$ with $s(i) = b$, we have:*

$$\mu_i^s = \mu_i^{s'} + \sum_{j=1}^{k} X_b^{i,j} \left[\mathbb{I}(s(j) = 1)\, \mathbb{I}(s'(j) = 0)\right] - \sum_{j=1}^{k} X_b^{i,j} \left[\mathbb{I}(s(j) = 0)\, \mathbb{I}(s'(j) = 1)\right], \tag{3}$$

where s' is any state in $\mathcal{K}$ with $s'(i) = b$, and for $\{i, j\} \in \mathcal{C}$, $X_b^{i,j} \coloneqq \mu_i^{s_1} - \mu_i^{s_2}$ where (s_1, s_2) is some (i, j, b) pair. If

$\{i,j\} \notin \mathcal{C}$, *we define* $X_b^{i,j}$ *to be zero.*

From the above theorem we have the following guarantee.

Theorem 2. *Consider an MDP*$(\mathcal{S},\mathcal{A},\mathcal{C},\boldsymbol{\theta})$ *with losses in* $[0,B]$*, maximum fixing cost* c*, and correlations sets of size at most* $m = 2$*. Let* $\mathcal{K}$ *be a cover of* $\mathcal{C}$ *of size* $r \leq 4n$*, then, the algorithm of Figure 3 achieves a pseudo-regret bounded by* $O(kr^{1/3}(c+B)(\log rkT)^{1/3}T^{2/3})$*. Furthermore, given access to an oracle for* (2)*, the algorithm runs in time polynomial in* k *and* $n = |\mathcal{C}|$*.*

There is a natural extension to arbitrary correlation sets via extending the notion of a dichotomy and a cover (Algorithm in Figure 7, Appendix B). Our algorithms are also scalable. During step 1 they only explore the states in the cover $\mathcal{K}$ that could be much smaller than the full state space.

Beyond $T^{\frac{2}{3}}$ regret. Next, we present algorithms that achieve $\tilde{O}(\sqrt{T})$ regret, first in the case $m = 1$, next for any m, under the assumption that each criterion does not participate in too many correlations sets. Although our problem can be cast as an instance of the stochastic multi-armed bandit problem with switching costs, and arms corresponding to the states in the MDP, existing algorithms achieving $\tilde{O}(\sqrt{T})$ have time complexity that depends on the number of arms which in our case is exponential (2^k) (Cesa-Bianchi, Dekel, and Shamir 2013; Simchi-Levi and Xu 2019). We will show here that, in most realistic instances of our model, we can achieve $\tilde{O}(\sqrt{T})$ regret efficiently. When correlation sets are of size one, the parameter vector $\boldsymbol{\theta}$ can be described using the following $2k$ parameters: for each $i \in [k]$, let γ_i^0 denote the expected loss incurred by criterion i when it is unfixed and γ_i^1 its expected loss when it is fixed. Our proposed algorithm is similar to the UCB algorithm (Auer, Cesa-Bianchi, and Fischer 2002). For every vertex i, let $\tau_{i,t}^0$ be the total number of time steps up to t (including t) during which v_i is in an unfixed position and let $\tau_{i,t}^1$ be the number of times steps up to t during which v_i is in a fixed position. Fix $\delta \in (0,1)$ and let $\hat{\gamma}_{i,t}^b$ be the empirical average loss observed when vertex v_i is in state b, for $b \in \{0,1\}$. Our algorithm maintains optimistic estimates

$$\tilde{\gamma}_{i,t}^b = \hat{\gamma}_{i,t}^b - 10B\sqrt{\frac{\log(kT/\delta)}{\tau_{i,t}^b}}. \tag{4}$$

The algorithm divides the T time steps into consecutive intervals that we call as *episodes*. In episode h, the algorithm moves to and stays at a fixed state for $t(h)$ time steps. in a fixed state. At the end of the episode it makes a query to the optimization oracle (using the current optimistic estimates) to decide on the state to go to for the next episode. The algorithm carefully chooses $t(h)$ to maintain low regret. The algorithm is described in Figure 4. We will prove that it benefits from the following regret guarantee.

Theorem 3. *Consider MDP*$(\mathcal{S},\mathcal{A},\mathcal{C},\boldsymbol{\theta})$ *with losses in* $[0,B]$ *and maximum fixing cost* c*. Given correlations sets* $\mathcal{C}$ *of size one, the algorithm of Figure 4 achieves a pseudo-regret bounded by* $O(k^2(c+B)^2\sqrt{T}\log T)$*. Furthermore, given access to an oracle for* (2)*, the algorithm runs in time polynomial in* k*.*

The algorithm of Figure 4 can be extended to higher m values (see Figure 9 in Appendix C).

Modeling assumptions and extensions. Here we briefly discuss assumptions and extensions.

Scalability. The running time of our algorithms depends linearly on the size of the cover $\mathcal{K}$. While in the worst case the size of the cover could be exponential in n, m, in practice, we expect it to be rather small.

Loss function. The choice of the loss function is critical. We made the simplifying assumption that the loss at each time step is additive in the losses incurred by correlation sets. A careless choice of what the additive losses correspond to may result in a sub-optimal overall. For example, a poor choice is one that uses the volume of complaints, i.e., how many complaints have triggered a criterion. This will make us vulnerable to the loudest voices in the system. In Section 4, we discuss how our framework can be implemented in practice and present reasonable choices for the loss function. We further discuss the choice of the loss function in the case of the COMPAS example in Appendix F.

Adversarial manipulation. Our model may be vulnerable to strategic coordination. A malicious group of users can form a sub-community generating a large number of complaints to press the system to include a new criterion in the graph. The presence of such poor criteria may result in an overall suboptimal system. Modeling this scenario is beyond the scope of the current work.

Continuous states. While this is a direction for future work, our method offers a simple way to achieve this by adding, for each criterion i, new criteria to the graph with different levels or thresholds $\tau_1, \tau_2, \ldots$ for satisfying i.

4 Experiments

Real-world dataset. We next illustrate how our framework can be applied to real-world data. Due to space constraints, we provide a brief description here and refer the reader to Appendix E for details and more results. We studied the UCI Adult dataset (Kohavi 1996) which includes $48{,}852$ examples, each represented by 124 features, after processing. Each data point corresponds to a person and the label is a $0/1$-value representing whether the income of the person is more or less than $\$50{,}000$. The dataset contains information about sensitive attributes such as race and gender. We simulated an online scenario where a classifier is making predictions on the income of individuals. At each time step, a batch of complaints arrive, the system incurs a loss and responds by transitioning to a different state (and updating the classifier). We now describe the various components.

Graph $\mathcal{G}$*:* We used race in $\{\text{black}, \text{white}\}$ as an attribute to obtain two sub-populations and considered two natural criteria, namely the true positive rate and the AUC score, equivalent to the criteria of the COMPAS tool. This leads to four vertices $tpr_w, tpr_b, auc_w, auc_b$. Furthermore, we added the overall classifier accuracy as a fifth vertex. We consider a unit fixing cost for all criteria.

Losses and Correlation Sets: We consider correlation sets of size one, and the total loss of a state is the sum of the losses of each criterion. For the accuracy vertex, we define the loss to

Input: The graph $\mathcal{G}$, correlation sets $\mathcal{C}$, fixing costs c_i.

1. Pick a cover $\mathcal{K} = \{s_1, s_2, \ldots, s_r\}$ of $\mathcal{C}$.
2. Let $N = 10\frac{T^{2/3}(\log rkT)^{1/3}}{r^{2/3}}$.
3. For each state $s \in \mathcal{K}$ do:
   - Move from current state to s in at most k time steps.
   - Play action $a = 0$ in state s for the next N time steps to obtain an estimate $\widehat{\mu}_i^s$ for all $i \in [k]$.
4. Using the estimated losses for the states in $\mathcal{K}$ and Equation (3), run the oracle for the optimization (2) to obtain an approximately optimal state $\hat{s}$.
5. Move from current state to $\hat{s}$ and play action $a = 0$ from $\hat{s}$ for the remaining time steps.

Figure 3: Algorithm for $m = 2$ achieving $\tilde{O}(T^{2/3})$ pseudo-regret.

Input: graph $\mathcal{G}$, correlation sets $\mathcal{C}$, fixing costs c_i.

1. Let $\mathcal{K}$ be the cover of size $k+1$ that includes the all zeros state and the states corresponding to indicator vectors of the k vertices.
2. Move to each state in the cover once and update the optimistic estimates according to (4).
3. For episodes $h = 1, 2, \ldots$ do:
   - Run the optimization oracle for solving Eq. (2) with the optimistic estimates as in (4) to get a state s.
   - Move to state s. Stay in state s for $t(h)$ time steps and update corresponding estimates using (4). Here $t(h) = \min_i \tau_{i,t_h}^{s(i)}$ and t_h is the total number of time steps before episode h starts.

Figure 4: Online algorithm for $m = 1$ with $\tilde{O}(\sqrt{T})$ regret.

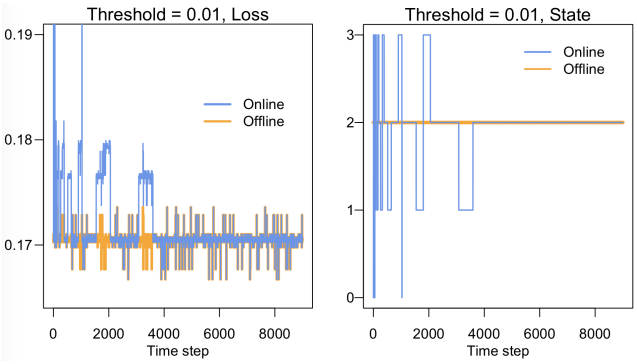

Figure 5: Performance of the Algorithm of Figure 4. Loss of the offline and online algorithms and states chosen by the online algorithm, as a function of the time steps.

be the error of the classifier. For a vertex, say tpr_w, if the overall tpr of the classifier and the tpr on the white population deviated by more than τ, we penalized the classifier linearly: the loss for tpr_w was defined as: $\max(0, |tpr_{overall} - tpr_w| - \tau)$. Other losses are defined similarly. We set $\tau = 0.005$.

Incompatibilities and State Transitions: We solved, for each state $s \in \{0,1\}^5$, an optimization via the tensorflow constrained optimization toolkit (Cotter et al. 2018; Cotter, Jiang, and Sridharan 2018) to get a classifier. We evaluated the classifier on a test set and if the loss of any criterion was more than a specific threshold (0.01), we considered the state invalid. As an example, in the instance corresponding to Figure 5, we obtained four valid states. We obtained the state transitions as follows. If the algorithm asked to fix v_i in state s, we set $s(i) = 1$ to go to the next state s'. While s' is invalid, we unfixed the criterion (not including v_i) with the highest loss in the state s' to reach another state.

Simulating Complaints: We divided the dataset into $16{,}000$ examples that we used to update our classifier at each time step and the remaining *test* set to simulate the arrival of complaints. At each step, we randomly selected a batch of examples from the test set to generate complaints. This batch was used to compute the loss at the given time step.

Benchmark and Results: We compared our Algorithm of Figure 4 with an offline optimal solution computed by finding the state with the minimum average loss over the sequence of complaints. The results are in Figure 5. We plot the loss of the algorithm as compared to the benchmark, as well as the states chosen by the algorithm, as a function of the time steps. Our algorithm quickly converges to the offline optimal solution after an initial exploration phase. Note that the choice of the loss functions was important in this case and that we did not weight each criterion by the volume of the complaints. This demonstrates that our algorithms, when combined with a good choice of the loss function, can be useful in practice. See Appendix E for details and additional experiments.

5 Conclusion

We presented a new data-driven model of online optimization from user feedback in the presence of multiple criteria, with algorithms benefiting from theoretical guarantees both in the stochastic and the adversarial setting. We provided empirical evidence that our model can be effectively realized in practice. Several extensions are worth exploring in future work. These include fixing costs that can vary with time to capture varying algorithmic price and human effort cost. Similarly, the losses in our stochastic model could be time-dependent to express the growing cost of a criterion not being addressed.

References

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

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2019. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. 2016. *URL https://www. propublica. org/article/machine-bias-risk-assessments-in-criminal-sentencing*.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3): 235–256.

Bastani, O.; Zhang, X.; and Solar-Lezama, A. 2019. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA): 1–27.

Bechavod, Y.; Jung, C.; and Wu, Z. S. 2020. Metric-Free Individual Fairness in Online Learning. *arXiv preprint arXiv:2002.05474*.

Bellamy, R. K.; Dey, K.; Hind, M.; Hoffman, S. C.; Houde, S.; Kannan, K.; Lohia, P.; Martino, J.; Mehta, S.; Mojsilovic, A.; et al. 2018. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*.

Borodin, A.; and El-Yaniv, R. 1998. *Online computation and competitive analysis.* Cambridge University Press.

Cesa-Bianchi, N.; Dekel, O.; and Shamir, O. 2013. Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, 1160–1168.

Chen, L.; and Pu, P. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1): 125–150.

Cortes, C.; Mohri, M.; Gonzalvo, J.; and Storcheus, D. 2020. Agnostic learning with multiple objectives. *Advances in Neural Information Processing Systems*, 33: 20485–20495.

Coston, A.; Ramamurthy, K. N.; Wei, D.; Varshney, K. R.; Speakman, S.; Mustahsan, Z.; and Chakraborty, S. 2019. Fair transfer learning with missing protected attributes. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 91–98.

Cotter, A.; Gupta, M.; Jiang, H.; Srebro, N.; Sridharan, K.; Wang, S.; Woodworth, B.; and You, S. 2018. Training well-generalizing classifiers for fairness metrics and other data-dependent constraints. *arXiv preprint arXiv:1807.00028*.

Cotter, A.; Jiang, H.; and Sridharan, K. 2018. Two-player games for efficient non-convex constrained optimization. *arXiv preprint arXiv:1804.06500*.

Doroudi, S.; Thomas, P. S.; and Brunskill, E. 2017. Importance Sampling for Fair Policy Selection. *Grantee Submission*.

Dwork, C.; Immorlica, N.; Kalai, A. T.; and Leiserson, M. 2018. Decoupled classifiers for group-fair and efficient machine learning. In *Conference on Fairness, Accountability and Transparency*, 119–133.

Feller, A.; Pierson, E.; Corbett-Davies, S.; and Goel, S. 2016. A computer program used for bail and sentencing decisions was labeled biased against blacks. It's actually not that clear. *The Washington Post*.

Ghosh, B.; Basu, D.; and Meel, K. S. 2020. Justicia: A Stochastic SAT Approach to Formally Verify Fairness. *arXiv preprint arXiv:2009.06516*.

Gupta, M.; Cotter, A.; Fard, M. M.; and Wang, S. 2018. Proxy fairness. *arXiv preprint arXiv:1806.11212*.

Gupta, V.; Nokhiz, P.; Roy, C. D.; and Venkatasubramanian, S. 2019. Equalizing recourse across groups. *arXiv preprint arXiv:1909.03166*.

Hashimoto, T. B.; Srivastava, M.; Namkoong, H.; and Liang, P. 2018. Fairness without demographics in repeated loss minimization. *arXiv preprint arXiv:1806.08010*.

Holstein, K.; Wortman Vaughan, J.; Daumé III, H.; Dudik, M.; and Wallach, H. 2019. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, 1–16.

Jabbari, S.; Joseph, M.; Kearns, M.; Morgenstern, J.; and Roth, A. 2017. Fairness in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1617–1626. JMLR. org.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr): 1563–1600.

Jin, Y. 2006. *Multi-objective machine learning*, volume 16. Springer Science & Business Media.

Jin, Y.; and Sendhoff, B. 2008. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(3): 397–415.

Kaminskas, M.; and Bridge, D. 2016. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(1): 1–42.

Kannan, S.; Roth, A.; and Ziani, J. 2019. Downstream effects of affirmative action. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 240–248.

Karlin, A. R.; Manasse, M. S.; Rudolph, L.; and Sleator, D. D. 1988. Competitive snoopy caching. *Algorithmica*, 3(1-4): 79–119.

Kearns, M.; Roth, A.; and Sharifi-Malvajerdi, S. 2019. Average Individual Fairness: Algorithms, Generalization and Experiments. *arXiv preprint arXiv:1905.10607*.

Kleinberg, J.; Mullainathan, S.; and Raghavan, M. 2017. Inherent Trade-Offs in the Fair Determination of Risk Scores. In *Innovations in Theoretical Computer Science Conference (ITCS)*.

Kohavi, R. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, 202–207.

Lamy, A.; Zhong, Z.; Menon, A. K.; and Verma, N. 2019. Noise-tolerant fair classification. In *Advances in Neural Information Processing Systems*, 294–305.

Lin, X.; Chen, H.; Pei, C.; Sun, F.; Xiao, X.; Sun, H.; Zhang, Y.; Ou, W.; and Jiang, P. 2019. A pareto-efficient algorithm

for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender systems*, 20–28.

Liu, L. T.; Dean, S.; Rolf, E.; Simchowitz, M.; and Hardt, M. 2018. Delayed impact of fair machine learning. *arXiv preprint arXiv:1803.04383*.

Marler, R. T.; and Arora, J. S. 2004. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6): 369–395.

Masthoff, J. 2011. Group recommender systems: Combining individual models. In *Recommender systems handbook*, 677–702. Springer.

Menon, A. K.; and Williamson, R. C. 2018. The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency*, 107–118.

Mohri, M.; Sivek, G.; and Suresh, A. T. 2019. Agnostic federated learning. In *International Conference on Machine Learning*, 4615–4625. PMLR.

Mouzannar, H.; Ohannessian, M. I.; and Srebro, N. 2019. From fair decision making to social equality. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 359–368.

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*.

Shah, A.; and Ghahramani, Z. 2016. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, 1919–1927. PMLR.

Simchi-Levi, D.; and Xu, Y. 2019. Phase transitions and cyclic phenomena in bandits with switching constraints. In *Advances in Neural Information Processing Systems*, 7521–7530.

Speicher, T.; Ali, M.; Venkatadri, G.; Ribeiro, F. N.; Arvanitakis, G.; Benevenuto, F.; Gummadi, K. P.; Loiseau, P.; and Mislove, A. 2018. Potential for discrimination in online targeted advertising. In *Conference on Fairness, Accountability and Transparency*, 5–19. PMLR.

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

Tsirtsis, S.; and Gomez-Rodriguez, M. 2020. Decisions, Counterfactual Explanations and Strategic Behavior. *arXiv preprint arXiv:2002.04333*.

Wang, S.; Guo, W.; Narasimhan, H.; Cotter, A.; Gupta, M.; and Jordan, M. I. 2020. Robust Optimization for Fairness with Noisy Protected Groups. *arXiv preprint arXiv:2002.09343*.

Wen, M.; Bastani, O.; and Topcu, U. 2019. Fairness with Dynamics. *arXiv preprint arXiv:1901.08568*.

Xiao, L.; Min, Z.; Yongfeng, Z.; Zhaoquan, G.; Yiqun, L.; and Shaoping, M. 2017. Fairness-aware group recommendation with pareto-efficiency. In *Proceedings of the eleventh ACM conference on recommender systems*, 107–115.

Yu, B.; Yuan, Y.; Terveen, L.; Wu, Z. S.; Forlizzi, J.; and Zhu, H. 2020. Keeping designers in the loop: Communicating inherent algorithmic trade-offs across multiple objectives. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, 1245–1257.