Balancing Fairness: Learning with Conflicting Objectives and User Feedback

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

[1]Google Research.
[2]Tel-Aviv University.
[3]Courant Institute of Mathematical Sciences.

Contributing authors: pranjalawasthi@google.com; corinna@google.com; mansour@google.com; mohri@google.com;

Abstract

Large-scale learning systems are often evaluated across multiple, sometimes conflicting, objectives. How can we effectively learn and optimize such complex systems with potentially incompatible goals? Moreover, how do we improve these systems when user feedback, potentially highlighting previously unaddressed issues, becomes available? We propose a novel theoretical model for learning and optimizing such systems. Instead of relying on a static or predefined trade-off among objectives, our model dynamically adapts based on received feedback, updating its internal state accordingly. This approach accommodates multiple objectives in a general form, accounting for their potential incompatibilities. We explore both stochastic and adversarial settings. In the stochastic setting, we demonstrate that our framework can be naturally formulated as a Markov Decision Process with stochastic losses, for which we provide efficient algorithms that achieve vanishing regret. In the adversarial setting, we design algorithms with competitive ratio guarantees. Finally, we report a series of experimental results validating the effectiveness of our stochastic algorithms.

Keywords: Learning with competing criteria; group fairness; individual fairness.

1 Introduction

Learning algorithms trained on vast datasets are increasingly used across various applications, driving complex large-scale systems such as e-commerce platforms, online advertising auctions, and recommender systems. Designers of these systems must consider multiple metrics to optimize their performance (Kaminskas and Bridge, 2016; Masthoff, 2011; Lin et al., 2019). For instance, in a recommendation system for recipes, videos, or fashion, there is no single metric that defines a successful recommendation engine. Instead, it is essential to assess metrics that evaluate the quality of recommendations for end-users, their relevance and utility, the long-term growth of content creators, and the overall revenue generated for the hosting platform.

Moreover, it is critical to address the potential risk of bias in these systems (Speicher et al., 2018; Xiao et al., 2017; Holstein et al., 2019). This requires incorporating additional metrics that measure performance across different demographic groups, geographical locations, and other identity factors. Consequently, designers may face the challenge of optimizing hundreds of metrics simultaneously to enhance user satisfaction.

Further complicating this scenario is the fact that the multiple metrics considered are often incompatible and inherently conflict with one another (Kleinberg et al., 2017; Sener and Koltun, 2018; Jin, 2006). For example, in the context of a recommendation system, there is a tension between maximizing revenue through ad placements and enhancing end-user satisfaction. Similarly, designers may face conflicts between prioritizing the quality of recommendations and ensuring their diversity.

Resolving these conflicts frequently necessitates difficult trade-offs among metrics that may seem reasonable in isolation, requiring careful consideration of current usage patterns and the associated benefits and drawbacks. A compelling illustration of this issue is the analysis of the COMPAS tool for predicting recidivism by Angwin et al. (2016). The authors revealed that, among Black defendants who did not recidivate, the tool made incorrect predictions at twice the rate compared to White defendants in the same category, indicating unfairness based on the *false positive rate* metric. In response, the tool's creator, Northpointe, argued that the system was fair according to other common metrics, such as the Area Under the ROC Curve (AUC), where both groups exhibited similar values. Subsequent research demonstrated that such tensions are inherent in the design of these systems and that it is often impossible to satisfy multiple seemingly reasonable criteria simultaneously (Kleinberg et al., 2017) (see also Feller et al. (2016)).

The preceding discussion raises the question of how to define the optimal trade-off among multiple conflicting metrics to enhance user satisfaction. One common approach is to establish these trade-offs statically, relying on domain knowledge, human expertise, or analysis of historical data. Another avenue of research focuses on optimizing in the presence of multiple objectives by designing algorithms that can compete with any linear combination of these objectives (Mohri et al., 2019; Cortes et al., 2020) or by developing Pareto-optimal solutions (Sener and Koltun, 2018; Shah and Ghahramani, 2016). However, these static solutions may be suboptimal in dynamic environments where user feedback is available.

While algorithms designed for a specific metric or a combination of metrics may perform well initially, experience shows that they can become irrelevant over time. Once deployed, systems interact with users, revealing inefficiencies in the design through their feedback, which may necessitate a reevaluation of the metrics initially considered (Liu et al., 2018). In the context of the COMPAS tool discussed earlier, this reflects a situation where user complaints prompt system designers to adjust the loss function to ensure equal false positive rates. Importantly, the underlying data distribution on which the tool was trained remains unchanged; rather, new user feedback highlights shortcomings in the system.

Motivated by these insights, we propose a theoretical data-driven model for optimizing multiple conflicting metrics that incorporates user feedback. Our framework enables the design of algorithmic solutions with strong theoretical guarantees.

To further illustrate the complexity of this problem, consider a deployed system. As end-users interact with it, hidden inefficiencies and biases within the algorithm may surface, leading to feedback or complaints from users or external auditors. For example, in a recommender system like YouTube, user complaints may indicate that certain videos are perceived as "low-quality", "violent", or simply receive a "dislike". Transitioning from these complaints to actionable solutions involves a sequence of steps that require careful consideration.

In a recommender system, user-initiated feedback may take the form of a "thumbs down," "too spicy", or "age inappropriate" label (Chen and Pu, 2012), but feedback can also be inferred indirectly through metrics like high abandonment rates or low click-through rates. Turning complaints into actionable solutions involves several steps. First, human specialists typically analyze the feedback and attribute it to predefined criteria, such as low classifier accuracy, high false positive rates, or suboptimal AUC scores. A single complaint might trigger multiple criteria, which specialists monitor to assess aggregate performance.

Because these criteria often conflict, decisions must be made regarding which subset to prioritize for improvement. This resource allocation process is informed by the analysis of complaints and their impact on key metrics, and it repeats iteratively as new feedback is received (Yu et al., 2020). While human input remains essential in this workflow, many parts of the process could be automated and made algorithmic.

Our model assumes predefined costs associated with user complaints across multiple metrics. The challenge in optimizing for user satisfaction lies in the fact that the nature and volume of complaints depend on the current state of the model. Ideally, if no complaints are received, the model has reached an optimal state. However, in most cases, complaints will persist. Addressing one set of complaints is likely to trigger a different set, leading to an ongoing cycle of adjustments. Fully optimizing the model would require exploring all incompatible states and observing the corresponding complaint sets—an approach that is impractical in terms of both time and development resources. This paper introduces a model that efficiently navigates toward a favorable state while providing performance guarantees.

The rest of the paper is organized as follows. In Section 3, we define our model. In the stochastic setting (Section 4), we show how our framework can be formulated as a Markov Decision Process with stochastic losses, for which we provide efficient

algorithms with vanishing regret. We also discuss our modeling assumptions and potential extensions. The adversarial setting is covered in Appendix 5, where we present algorithms with competitive ratio guarantees. In Section 6, we demonstrate the practical implementation of our framework and report experimental results in the stochastic setting, highlighting the effectiveness and applicability of our model.

A discussion of related work along with several proofs and an in-depth analysis of the COMPAS example is deferred to the appendix.

2 Related Work

There is a vast body of literature focused on optimizing multiple metrics or objectives under specific criteria. Recent works by Mohri et al. (2019) and Cortes et al. (2020) explore optimization in the presence of multiple base objectives. Given objectives $L_1, \ldots, L_i$, these works aim to design "agnostic" algorithms that can simultaneously compete with any linear or convex combination of the objectives. Another line of research focuses on designing algorithms that achieve Pareto-optimal solutions (Jin and Sendhoff, 2008; Sener and Koltun, 2018; Shah and Ghahramani, 2016; Marler and Arora, 2004).

Another line of research focuses on optimizing multiple constraints, particularly those inspired by group fairness metrics, using constrained non-convex optimization (Agarwal et al., 2018; Cotter et al., 2019; Thomas et al., 2019). These works typically approach the problem by either reducing it to cost-sensitive classification (Agarwal et al., 2018; Dwork et al., 2018) or by replacing non-convex constraints with convex proxies, which are then optimized using external or swap regret minimization algorithms (Cotter et al., 2019).

While classification has been the primary focus of algorithmic fairness, significant work has also addressed fairness in ranking (Celis et al., 2018; Beutel et al., 2019; Narasimhan et al., 2020), clustering (Chierichetti et al., 2017; Schmidt et al., 2019; Backurs et al., 2019), and online learning settings such as multi-armed bandits (Joseph et al., 2016; Gillen et al., 2018; Liu et al., 2017). Unlike these works, which typically tailor algorithms to specific fairness metrics within those domains, our framework addresses the dynamic optimization of arbitrary, potentially conflicting criteria via user feedback.

There have also been studies of the inherent tension between satisfying multiple metrics. Kleinberg et al. (2017) and Feller et al. (2016) demonstrate that it is impossible to satisfy equal opportunity and calibration at the same time. Inspired from fairness applications the work of Menon and Williamson (2018) studies the tradeoff between accuracy and other metrics of interest such as false positive and false negative rates.

Recent works have also studied the long-term impact of optimizing multiple conflicting criteria in settings with feedback mechanisms (Liu et al., 2018; Hashimoto et al., 2018; Mouzannar et al., 2019; Kannan et al., 2019). Liu et al. (2018) show that, in certain situations, constrained loss minimization to equalize certain criteria could lead to further disparate impact on the end users in the long run. Hashimoto et al. (2018) proposed algorithms for minimizing such disparate impact in settings involving repeated loss minimization. More recently, Jabbari et al. (2017); Wen et al. (2021)

study the problem of satisfying multiple constraints in reinforcement learning settings involving a Markov Decision Process. The authors in Jabbari et al. (2017) consider learning in an MDP where the criteria to be optimized require that the algorithm never takes an action a over action a' if the long-term reward is higher. It is clear to see that the optimal policy for the MDP indeed satisfies this property. Hence, there does exist a policy that satisfies the required criterion. However, the authors show that finding a near optimal policy while satisfying the criterion requires time exponential in the size of the state space.

Wen et al. (2021) consider other metrics such as demographic parity in the context of learning in MDPs. Doroudi et al. (2017) show that existing importance sampling methods for off-policy policy selection in reinforcement learning can lead to bad outcomes according to other natural criteria and present algorithms to mitigate this effect.

While our work also involves learning in a Markov Decision Process (MDP) and optimizing multiple criteria in the long term, the setup and the motivation are different. Unlike all the previous work mentioned, we do not commit to a fixed definition of quality or a metric, and allow for arbitrary criteria. Hence, states in our MDP correspond to the current configurations of different criteria. Rather than studying each metric in isolation, the objective of our work is to propose a data-driven model that can learn from feedback, a near-optimal configuration of the metrics to impose on the system. To the best of our knowledge, ours is the first work to incorporate optimizing metrics of arbitrary types in an online setting. In this context, inspired by fairness applications, the recent work of Kearns et al. (2019) studies a specific combination of group and individual fairness metrics. The authors consider a setting where there is a distribution over individuals as well as a distribution over classification tasks. They consider algorithms for achieving *average* individual fairness, that is in expectation over classification tasks, the performance of the algorithm on a group fairness metric such as demographic parity should be the same for each individual.

An important aspect of our stochastic MDP-based model requires the ability to observe the losses associated with different criteria at each time. This relates to the problem of evaluating and monitoring the performance of the system according to different metrics from data. There has been work in recent years on developing auditing and monitoring approaches (Bastani et al., 2019; Ghosh et al., 2021; Bellamy et al., 2018). Furthermore, many metrics require access to both labeled data and to certain sensitive attribute information such as race or gender, for accurate evaluation. A recent line of work has studied this estimation problem when one has limited and/or noisy access to sensitive attribute information (Gupta et al., 2018; Coston et al., 2019; Lamy et al., 2019; Wang et al., 2020). Finally, we note that our model learns from feedback received as a form of complaints. These complaints are a result of a (potentially incorrect) decision made by an ML system. There has been recent work in developing counterfactual based explanations (Tsirtsis and Gomez-Rodriguez, 2020) for such decisions and exploring recourse strategies (Gupta et al., 2019).

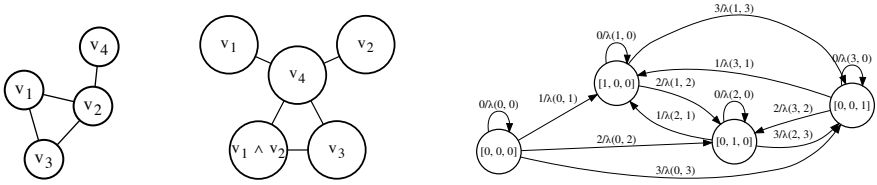

Figure 1: **(a)** Illustration of the incompatibility graph $\mathcal{G}$ (also called the constraint graph). Vertices v_1, v_2, v_3, v_4 represent 4 different criteria. An edge between vertices v_i and v_j indicates that criteria v_i and v_j cannot be simultaneously satisfied. **(b)** More generally, 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.

3 General Problem Formulation

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 $\mathcal{G} = (V, E)$, which we refer to as the *incompatibility graph* (or equivalently, the *constraint graph*). Each vertex represents a criterion and 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.

We consider a 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.

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 at end of Section 4 and Appendix B, 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 et al., 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 at end of Section 4). The notion of complaints and the associated loss may seem abstract at this point. In Section 6, 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 6 for a more concrete example. Our model also provides the flexibility of accounting for incompatibilities among criteria such as those discussed by Kleinberg et al. (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 cannot be all simultaneously satisfied.

States. We will adopt the following simplifying assumptions and will later discuss their extensions or relaxation in Section 4. 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.[1] An action corresponds to fixing a particular criterion, or set of criteria, and moving to a different state. *Fixing* the criterion 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 Section 4 and illustrate it in Section 6.

[1] Note, that this implies that the number of states is exponential, hence, any algorithm which depends polynomially on the number of states would have an exponential overhead.

4 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.

4.1 Description

We model the generation of user feedback as a stochastic process dependent on the system's configuration. Formally, we associate with each state $s \in \mathcal{S}$ a joint probability distribution $\mathcal{D}_s$ over the loss domain $[0, B]^k$. Accordingly, we formulate our problem as a Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{L})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the deterministic transition function, and $\mathcal{L}$ is the loss function (analogous to the negative reward).

Specifically, the state space is $\mathcal{S} \subseteq \{0, 1\}^k$ 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 $\mathcal{G}$, a state s is valid if and only if the set of fixed criteria $\{v_i \mid s(i) = 1\}$ forms an *independent set* of $\mathcal{G}$; that is, no two fixed criteria are connected by an edge in $\mathcal{G}$.

When in state $s \in \mathcal{S}$, the system incurs a loss vector $\boldsymbol{\ell}^s = (\ell_1^s, \ldots, \ell_k^s)$ drawn independently from $\mathcal{D}_s$. The component ℓ_i^s corresponds to the complaints related to criterion $i \in [k]$ and is a random variable taking values in $[0, B]$ with mean $\mu_i^s = \mathbb{E}_{\boldsymbol{\ell} \sim \mathcal{D}_s}[\ell_i^s]$. 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.

Bounded Losses. Our theoretical analysis assumes losses are bounded in $[0, B]$ to utilize standard concentration inequalities. In practice, this assumption holds for most relevant metrics: rates and probabilities lie in $[0, 1]$, and user feedback often arrives on fixed scales (e.g., 1-5 stars). For metrics that are theoretically unbounded (e.g., response latency), it is standard practice to apply a clipping function (e.g., $\min(\ell, B_{\max})$) or a normalization step to map values to a bounded range, preventing single outliers from destabilizing the learning process.

Correlation Sets. A major challenge in navigating the state space $\mathcal{S}$ is its size: with k criteria, there are 2^k possible states. If the loss distribution for a criterion v_i depended on the global configuration of all k criteria, learning the optimal state would require visiting every state, which is exponentially expensive.

However, in practice, user complaints regarding a specific criterion usually stem from *local* interactions with a small subset of other criteria, rather than the entire system. To capture this structure and make learning efficient, we introduce *correlation sets*.

Intuitively, this is analogous to diagnosing a house's electrical system: if the lights go out in the kitchen (a local loss), it is likely due to the kitchen breaker or a specific appliance (the local correlation set). The status of the bedroom windows (independent criteria) provides no information about the kitchen lights, and thus the parameters governing the kitchen's failure remain invariant regardless of the bedroom's state.

Formally, we assume a collection $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ is given, where each $\mathcal{C}_j$ is a subset of the k criteria with size at most m. The expected loss μ_i^s for criterion i in state s is modeled as the sum of contributions from the correlation sets that i belongs to:

$$\mu_i^s = \sum_{j=1}^{n} \theta_j^s \, \mathbb{I}(i \in \mathcal{C}_j). \tag{2}$$

Here, θ_j^s represents the "interaction effect" or loss contribution generated by the specific configuration of the criteria in $\mathcal{C}_j$ within state s.

Parameter Sharing Assumption. The core of this model is the following assumption: if two global states s and s' have the exact same configuration for the criteria in a specific set $\mathcal{C}_j$ (i.e., $s(v) = s'(v)$ for all $v \in \mathcal{C}_j$), then the loss parameter is shared: $\theta_j^s = \theta_j^{s'}$.

This assumption implies conditional independence: the loss contribution from an interaction group $\mathcal{C}_j$ is independent of the status of criteria outside that group. This drastically reduces the number of parameters we must learn. We quantify this efficiency gain in Table 1, which contrasts the exponential cost of naive approaches with the linear scaling of our correlation set model.

Example: News Recommender. To illustrate why this assumption holds, consider a system with three criteria: Political Diversity (v_1), Topic Relevance (v_2), and App Latency (v_3).

- We define a correlation set $\mathcal{C}_1 = \{v_1, v_2\}$ because enforcing diversity often conflicts with relevance, triggering specific user complaints.
- We define $\mathcal{C}_2 = \{v_3\}$ because latency is a system metric independent of content.

Consider two global states:

- State s: Diversity Fixed, Relevance Unfixed, **Latency Fixed**.

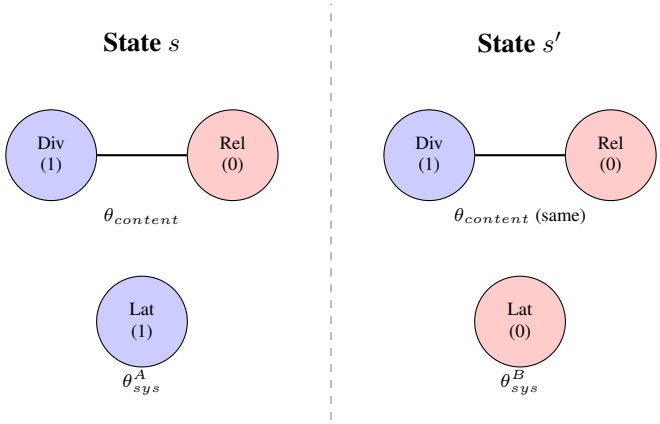

Figure 2: Visualizing Conditional Independence: The loss contribution $\theta_{content}$ depends only on the local configuration of Diversity and Relevance. It remains constant even when the independent Latency criterion changes state.

Table 1: Comparison of Parameter Complexity. Without correlation sets, the number of parameters to learn scales exponentially with the total number of criteria (k). With correlation sets of size m, it scales linearly with the number of sets (n).

Model Assumption	**Number of Parameters**	**Learning Feasibility**
Full Dependency (Naive)	2^k	Impossible for large k
Pairwise ($m = 2$)	$\approx 4 \times \lvert E \rvert$	Efficient (Polynomial)
Correlation Sets (m)	$\approx \mathbf{n \times 2^m}$	**Efficient (if** $m \ll k$**)**

- State s': Diversity Fixed, Relevance Unfixed, **Latency Unfixed**.

Even though s and s' are different global states, the configuration of $\mathcal{C}_1$ is identical ($[1, 0]$) (see Figure 2). Users complaining about "irrelevant articles" (the loss contribution from $\mathcal{C}_1$) are reacting to the tension between diversity and relevance. They are typically not reacting to the app's loading speed (v_3). Therefore, it is reasonable to assume $\theta_1^s = \theta_1^{s'}$. This allows us to learn the diversity-relevance trade-off without needing to explore every permutation of latency settings.

Connection to Graphical Models. The reader may question why we model dependencies via correlation sets rather than other forms. Our formulation can be viewed as a generalized *Factor Graph* (Kschischang et al., 2001; Loeliger, 2004; Cortes et al., 2016) or a higher-order *Ising Model*, where the global energy is the sum of local potentials. The additive form is natural for user feedback: distinct sources of dissatisfaction (interaction sets) independently generate complaint events, which accumulate additively in the system's aggregate loss logs.

- If $m = 1$, our model corresponds to an independent field (linear loss).
- If $m = 2$, it captures pairwise interactions, similar to edges in a standard graph.

- By allowing arbitrary sets $\mathcal{C}_j$ of size m, we capture hyper-graph dependencies (e.g., a user only complains if criteria A, B, *and* C fail simultaneously), which simpler pairwise models cannot represent.

This structure offers the necessary expressiveness to model complex user behaviors while maintaining a manageable sample complexity.

Let $\boldsymbol{\theta}$ denote the vector of all distinct parameters. Our model is denoted $\mathrm{MDP}(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$.

4.2 Algorithm

We consider an online algorithm $\mathfrak{A}$ that, at time t, takes action $a_t \in \mathcal{A}$ from state s_t and reaches state s_{t+1}, starting from the initial state $\mathbf{0} = (0, \ldots, 0)$.

The objective of the algorithm is to minimize the cumulative loss over T interactions. We evaluate performance using the T-step pseudo-regret, denoted $\mathsf{Reg}_T(\mathfrak{A})$. Instead of comparing against an abstract policy, we compare the algorithm's total loss against the loss of the *optimal state* s^*:

$$\mathsf{Reg}_T(\mathfrak{A}) = \sum_{t=1}^{T} \mathbb{E}\left[\lambda_t(s_t, a_t)\right] - \sum_{t=1}^{T} g(s^*). \tag{3}$$

Here, the expectation is over the random generation of complaint losses. The term $g(s)$ represents the expected per-step loss of staying in state s, defined as $g(s) \coloneqq \sum_{i=1}^{k} \mu_i^s$. Let $S^* = \operatorname{argmin}_{s \in \mathcal{S}} g(s)$ be the set of loss-minimizing states. We define the reference optimal state s^* as an arbitrary element of this set:

$$s^* \in S^*. \tag{4}$$

This definition compares the algorithm against an oracle strategy that moves to s^* at $t = 0$ and remains there. We strictly define the regret against the *stationary* performance of s^*, omitting the one-time transition cost required for the oracle to reach s^* from the initial state $\mathbf{0}$.

This omission is theoretically justified because the transition cost is bounded by $k \cdot \max_i c_i$ (the maximum path length in the graph times the maximum fixing cost). This value is a constant independent of T. Therefore, ignoring it introduces only a negligible additive constant to the regret, which does not affect the asymptotic bounds (e.g., $\tilde{O}(\sqrt{T})$) or the required time horizon for learning.

Since our problem can be viewed as learning in a deterministic MDP with stochastic losses, one could ostensibly adopt existing algorithms such as UCRL2 (Jaksch et al., 2010). However, the regret guarantees and running time of such algorithms depend polynomially on the size of the state space $|\mathcal{S}|$, which here is 2^k. To avoid this exponential dependence on k, we exploit the structure of the correlation sets to design algorithms with complexity polynomial in k and the number of parameters.

We assume access to an oracle that, given $\boldsymbol{\theta}$, can optimize (4). In Appendix A, we show how to approximately solve (4) for the case of singleton correlation sets, where the true parameters $\boldsymbol{\theta}$ can also be estimated efficiently (see Theorem 7).

Finally, we note an important distinction between our model and standard online learning settings. Here, the state evolves as a result of stochastic losses, which in turn depend on the distribution of complaints. This distribution should not be confused with the distribution of data over which classifiers are trained (which is assumed constant).

Optimization. We envision that the algorithm at every step has access to and is solving a constraint optimization problem with the criteria as constraints.

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.

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 captures the important case where fixing a particular criterion affects the rate of complaints of its neighbors.

Our algorithm consists of an exploration phase where it observes the losses for a specific subset of at most $4n$ states. We will show that after the exploration phase, the algorithm can accurately estimate the expected loss for any other state $s \in \mathcal{S}$. Notice that the number of states in $\mathcal{S}$ is generally exponential in k. Thus, the subset of states to observe must be carefully chosen to exploit the structure of the graph $\mathcal{G}$.

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. (4) to find a near-optimal state $\hat{s}$, and selects to stay at state $\hat{s}$ for the remaining time steps. Let c denote the maximum fixing cost: $c = \max_{i\in[k]} c_i$. We will show that the pseudo-regret of our algorithm is bounded by $O(k \log k(c + B)^{1/3} T^{2/3} \log(kT))$.

We first describe how we select the subset of states to observe. 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.

Example 1 (Dichotomy as a Controlled Experiment) Suppose we want to measure how fixing *Latency* (v_j) affects the loss of *Relevance* (v_i), while Relevance remains unfixed ($b = 0$). We look for a dichotomy $(i, j, 0)$, which requires two states:

- **State s (Baseline):** Relevance = 0, Latency = 0.
- **State s' (Treatment):** Relevance = 0, Latency = 1.

By comparing the loss of Relevance (μ_i) between s and s', we isolate the specific interaction effect (or "friction") that fixing Latency imposes on Relevance.

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.

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 (5), run the oracle for the optimization (4) 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.

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:*

$$\mu_i^s = \mu_i^{s'} + \sum_{j=1}^{k} X_b^{i,j}\left[\mathbb{I}(s(j) = 1)\,\mathbb{I}(s'(j) = 0)\right] - \sum_{j=1}^{k} X_b^{i,j}\left[\mathbb{I}(s(j) = 0)\,\mathbb{I}(s'(j) = 1)\right], \quad (5)$$

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

Using this theorem, we can derive 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 (4), 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 14, Appendix A). 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.

4.3 Beyond $T^{\frac{2}{3}}$ Regret

In this section, we present algorithms for our problem that achieve $\tilde{O}(\sqrt{T})$ regret. Let us first point out that our problem can be cast as an instance of the stochastic multi-armed bandit problem with switching costs, where each state s is viewed as an arm and the cost of transitions corresponds to the switching cost. While algorithms for *structured stochastic bandits* (Combes et al., 2017) could theoretically exploit the correlations between arms (states), applying generic solvers to our setting presents a computational challenge: the number of arms (states) is exponential (2^k). Generic

structured bandit algorithms typically require solving an optimization problem over the set of arms at every step (or periodically), which would be intractable here.

Similarly, for bandits with switching costs, Cesa-Bianchi et al. (2013)[Appendix A] provided an algorithm achieving $\tilde{O}(\sqrt{T})$ regret. However, the complexity of that algorithm also scales with the number of arms. To overcome this, we leverage the specific additive structure of our correlation sets to design an algorithm that achieves $\tilde{O}(\sqrt{T})$ regret with computational complexity polynomial in k.

We first consider the case where the correlations sets in $\mathcal{C}$ are of size one ($m = 1$). In this case, 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. In this case, the cover $\mathcal{K}$ is of size $k + 1$ and includes the all-zero state, as well as k states corresponding to the indicator vectors of the k vertices. Our algorithm is similar to the UCB algorithm for multi-armed bandits Auer et al. (2002) and maintains optimistic estimates of the parameters. For every vertex i, we denote by $\tau_{i,t}^0$ the total number of time steps up to t (including t) during which the vertex v_i is in an unfixed position and by $\tau_{i,t}^1$ the total number of times steps up to t during which vertex v_i is in a fixed position. Fix $\delta \in (0, 1)$ and let $\hat{\gamma}_{i,t}^b$ be the empirical expected loss observed when vertex v_i is in state b, for $b \in \{0, 1\}$. Our algorithm maintains the following optimistic estimates at each time step t, projecting onto the valid loss range $[0, \infty)$:

$$\tilde{\gamma}_{i,t}^b = \max\left(0,\ \hat{\gamma}_{i,t}^b - 10B\sqrt{\frac{\log(kT/\delta)}{\tau_{i,t}^b}}\right). \tag{6}$$

To minimize the fixing cost incurred when transitioning from one state to another, our algorithm works in episodes. In each episode h, the algorithm first uses the current optimistic estimates to query the optimization oracle and determine the current best state s. Next, it remains at state s for $t(h)$ time steps before querying the oracle again. The number of time steps $t(h)$ will be chosen carefully to avoid incurring the fixing costs too often. The algorithm is described in Figure 4. We will prove that it benefits from the following regret guarantee.

Theorem 3 *Consider an MDP$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$ with losses bounded in $[0, B]$ and maximum cost of fixing a vertex being 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 (4), the algorithm runs in time polynomial in k.*

Proof. We first bound the total number of different states visited by the algorithm. Initially the algorithm visits $k + 1$ states in the cover. After that, each time the optimization oracle returns a new state s, by the definition of $t(h)$, the number of time steps where some vertex is in a 0 or 1 position is doubled. Hence, at most $O(k \log T)$ calls are made to the optimization oracle. Noticing that one can move from one state to another in at most k time steps, the total loss incurred during the switching of the states is bounded by $O(k^2(c + B)\log T)$.

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 (6).
3. For episodes $h = 1, 2, \ldots$ do:
   - Run the optimization oracle for solving Eq. (4) with the optimistic estimates as in (6) to get a state s.
   - Move from current state to state s. Stay in state s for $t(h)$ time steps and update the corresponding estimates using (6). 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.

For $\epsilon > 0$ to be chosen later, we consider the episodes where the algorithm plays a state s with expected loss at most ϵ more than that of the best state s^*. The total expected regret accumulated in these *good* episodes is at most ϵT. We next bound the expected regret accumulated during the bad episodes.

From Hoeffding's inequality we have that for any time t, with probability at least $1 - \frac{\delta}{T^3}$, for all $i \in [k], b \in \{0, 1\}$,

$$\tilde{\gamma}_{i,t}^b + 20B\sqrt{\frac{\log(kT/\delta)}{\tau_{i,t}^b}} \geq \gamma_i^b \geq \tilde{\gamma}_{i,t}^b. \tag{7}$$

Let G be the good event that (7) holds for all $t \in [1, T]$. Conditioned on G we also have that for any state s and vertex i

$$\mu_i^s \geq \tilde{\mu}_i^s, \tag{8}$$

where $\tilde{\mu}_i^s$ is the estimated loss using the optimistic estimates. We will bound the expected regret accumulated in the bad episodes conditioned on the event G above.

In order to do this we define certain key quantities. Consider a particular trajectory $\mathcal{T}$ of T time steps executed by the algorithm. Furthermore, let $\mathcal{T}$ be such that the good event in (7) holds during the T time steps. We associate the following random variables with the trajectory. Let N_ϵ be the total number of time steps spent in bad episodes. Furthermore, let Reg_ϵ be the total accumulated regret during these time steps. Then it is easy to see that $\mathbb{E}[\mathsf{Reg}_\epsilon | G] > \epsilon N_\epsilon$. For each vertex v_i and $b \in \{0, 1\}$ we define $\tau_\epsilon(i, b)$ to be the total number of time steps that vertex v_i spends in bad episodes in position b and $\tau_\epsilon(i, b, t)$ to be the total number of time steps spent in bad episodes up to time step t. Notice that

$$\sum_b \sum_i \tau_\epsilon(i, b) \leq 2kN_\epsilon. \tag{9}$$

Consider a particular bad episode h and let s be the state returned by the optimization oracle during that episode. Then, conditioned on the good event G, the total regret Reg_h accumulated during episode h satisfies

$$\begin{aligned}
\mathbb{E}[\mathsf{Reg}_h|\mathcal{T}] &= \sum_i \left(\mu_i^s - \mu_i^{s^*}\right) t(h) \\
&\leq \sum_i \left(\mu_i^s - \tilde{\mu}_i^{s^*}\right) t(h) \qquad \text{(from(8))} \\
&\leq \sum_i \left(\mu_i^s - \tilde{\mu}_i^s\right) t(h) \\
&\quad \text{(since } s \text{ is best state according to the optimistic losses)} \\
&\leq \sum_i \left(\gamma_i^{s(i)} - \tilde{\gamma}_{i,t_h}^{s(i)}\right) t(h) \\
&\leq \sum_i 20B \sqrt{\frac{\log(kT/\delta)}{\tau_{i,t_h}^b}} t(h). \qquad \text{(from (6))}
\end{aligned}$$

In the above inequality, the expectation is taken over the loss distribution for each vertex during states visited in the trajectory $\mathcal{T}$. we have that

$$\mathbb{E}[\mathsf{Reg}_h|\mathcal{T}] \leq \sum_i 20B \sqrt{\frac{\log(kT/\delta)}{\tau_\epsilon(i,b,t_h)}} t(h).$$

Summing over bad episodes, the total expected regret in bad episodes can be bounded by

$$\mathbb{E}[\mathsf{Reg}_\epsilon|\mathcal{T}] \leq \sum_i \sum_b \sum_{h:h \text{ is bad}} 20B \sqrt{\frac{\log(kT/\delta)}{\tau_\epsilon(i,b,t_h)}} t(h). \tag{10}$$

Notice that $\tau_\epsilon(i,b,t_h) = \sum_{h'<h:h' \text{ is bad}} t(h')$. Furthermore, we know that (Jaksch et al. (2010)) for any sequence $z_1, z_2, \ldots, z_h$ of non-negative numbers such that $z_i \geq 1$,

$$\sum_{i=1}^h \frac{z_i}{\sqrt{\sum_{j=1}^{i-1} z_j}} \leq (1+\sqrt{2}) \sqrt{\sum_{i=1}^h z_i}. \tag{11}$$

From (11) we get:

$$\sum_{h:h \text{ is bad}} \frac{t(h)}{\sqrt{\tau_\epsilon(i,b,t_h)}} \leq \sqrt{\tau_\epsilon(i,b)}.$$

Substituting into (10) yields

$$\mathbb{E}[\mathsf{Reg}_\epsilon|\mathcal{T}] \leq \sum_i \sum_b 20B\sqrt{\log(kT/\delta)}\sqrt{\tau_\epsilon(i,b)}.$$

Using (9) we have that the above expected regret is maximized when $\tau_\epsilon(i,b)$ are equal, thereby implying

$$\mathbb{E}[\mathsf{Reg}_\epsilon|\mathcal{T}] \leq 20Bk\sqrt{\log(kT/\delta)}\sqrt{N_\epsilon}.$$

Using the fact that $\mathbb{E}[\mathsf{Reg}_\epsilon|G] > \epsilon N_\epsilon$ we get that conditioned on G,

$$N_\epsilon \leq \frac{400B^2k^2\log(kT/\delta)}{\epsilon^2}.$$

Combining trajectories $\mathcal{T}$ where the good event G holds, we get that the total expected regret accumulated in the bad episodes satisfies

$$\mathbb{E}[\mathsf{Reg}_\epsilon|G] \leq 20Bk\sqrt{\log(kT/\delta)}\sqrt{N_\epsilon} \leq 400B^2k^2\frac{\log(kT/\delta)}{\epsilon}.$$

Combining the above with the total expected regret accumulated in the good episodes, the loss of moving to different states, and the probability of good event G not holding, we get

$$\mathsf{Reg}(\mathcal{A}) \leq 400B^2k^2\frac{\log(kT/\delta)}{\epsilon} + \epsilon T + \frac{k(c+B)}{T^3} + O(k^2(c+B)\log T).$$

Setting $\epsilon = \frac{1}{\sqrt{T}}$ and $\delta = \frac{1}{T^4}$ yields $\mathsf{Reg}(\mathcal{A}) \leq O\big((c+B)^2k^2\sqrt{T}\log(T)\big)$. □

The above result extends to higher m values, assuming that each vertex does not participate in too many correlation sets. If a vertex v_i appears in at most $O(\log k)$ correlation sets, then the total loss incurred by vertex v_i in any state depends on the position of v_i and every other vertex that it is correlated with. Hence the total loss incurred by vertex v_i depends on an $O(m\log k)$-dimensional vector. For every configuration $\mathbf{b}$ of this vector, we associate with each vertex v_i, parameters $\gamma_i^{\mathbf{b}}$. Notice that there are at most $O(k^m)$ such parameters. Each parameter is in turn a sum of a subset of the parameters in $\boldsymbol{\theta}$. Notice that in this case the size of the cover $\mathcal{K}$ is upper bounded by $O(k^{m+1})$. Our algorithm for higher m values is similar to the one for $m = 1$, but instead maintains optimistic estimates of the parameters $\gamma_i^{\mathbf{b}}$ via

$$\tilde{\gamma}_{i,t}^{\mathbf{b}} = \max\left(0,\ \hat{\gamma}_{i,t}^{\mathbf{b}} - 10B\sqrt{m\frac{\log(kT/\delta)}{\tau_{i,t}^{\mathbf{b}}}}\right). \tag{12}$$

Here $\tau_{i,t}^{\mathbf{b}}$ is the total time spent up to and including t where the vertex i and the vertices that it is correlated with are in configuration $\mathbf{b}$. Similarly, for a given state

s, we will denote by $\mathbf{s}(i)$, the configuration of the vertex i and the vertices that it is correlated with. The algorithm is sketched below

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

1. Let $\mathcal{K}$ be the cover of size $O(k^{m+1})$.
2. Move to each state in the cover once and update the optimistic estimates according to (12).
3. For episodes $h = 1, 2, \ldots$ do:
   - Run the optimization oracle (4) with the optimistic estimates as in (12) to get a state s.
   - Move from current state to state s. Stay in state s for $t(h)$ time steps and update the corresponding estimates using (12). Here $t(h) = \min_i \tau_{i,t_h}^{\mathbf{s}(i)}$ and t_h is the total number of time steps before episode h starts.

Figure 5: Online algorithm for higher m.

For $m \geq 1$, we obtain the following guarantee.

Theorem 4 *Consider an MDP$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$ with losses bounded in $[0, B]$ and maximum cost of fixing a vertex being c. Given correlations sets $\mathcal{C}$ of size at most m such that each vertex participates in at most $O(\log k)$ sets, the algorithm in Figure 5 achieves a pseudo-regret bounded by $O(mk^{2m+2}(c+B)^2\sqrt{T}\log T)$. Furthermore, given access to an oracle for (4), the algorithm runs in time polynomial in $O(k^{m+1})$.*

Proof. The proof is very similar to that of Theorem 3. Since each time the optimization oracle is called the time spent in some configuration $\mathbf{s}(i)$ is doubled, we get that the total number of calls to the optimization oracle are bounded by $O(k^m \log T)$. Hence the total loss incurred during the exploration phase can be bounded by $O(k^m(c + B)\log T)$. Let G be the good event that (12) holds for all $t \in [1, T]$.

As before, the loss incurred during good episodes is bounded by ϵT. Define $\tau_\epsilon(i, \mathbf{b})$ to be the total time that vertex i and vertices that it is correlated with spend in configuration $\mathbf{b}$ during bad episodes. Then analogous to (9) we have

$$\sum_{\mathbf{b}} \sum_{i} \tau_\epsilon(i, \mathbf{b}) \leq O(k^m) N_\epsilon.$$

For a trajectory $\mathcal{T}$ where the good event G holds, the total expected regret in bad episodes can be bounded as

$$\mathbb{E}[\mathsf{Reg}_\epsilon|\mathcal{T}] \leq \sum_i \sum_b \sum_{h:h \text{ is bad}} 20B\sqrt{m\frac{\log(kT/\delta)}{\tau_\epsilon(i, \mathbf{b}, t_h)}}t(h) \tag{13}$$

$$\leq \sum_i \sum_{\mathbf{b}} 20B\sqrt{m\log(kT/\delta)}\sqrt{\tau_\epsilon(i, \mathbf{b})} \tag{14}$$

$$\leq O(Bk^{m+1})\sqrt{m\log(kT/\delta)}\sqrt{N_\epsilon}. \tag{15}$$

Using the fact that $\mathbb{E}[\mathsf{Reg}_\epsilon|\mathcal{T}] > \epsilon N_\epsilon$ we get that for a trajectory where the event G holds,

$$N_\epsilon \leq \frac{O(R^2k^{2m+2}m\log(kT/\delta))}{\epsilon^2}.$$

Hence we get that conditioned on the good event G, the total expected regret accumulated in the bad episodes is at most

$$\mathbb{E}[\mathsf{Reg}_\epsilon|G] \leq O\big(R^2mk^{2m+2}\frac{\log(kT/\delta)}{\epsilon}\big).$$

Combining the above with the total expected regret accumulated in the good episodes, the loss of moving to different states, and the probability of the event G not holding we get

$$\mathsf{Reg}(\mathcal{A}) \leq O\big(B^2mk^{2m+2}\frac{\log(kT/\delta)}{\epsilon}\big) + \epsilon T + \frac{k(c+B)}{T^3} + O(k^m\log T).$$

Setting $\epsilon = \frac{1}{\sqrt{T}}$ and $\delta = \frac{1}{T^4}$ yields $\mathsf{Reg}(\mathcal{A}) \leq O\big((c+B)^2mk^{2m+2}\sqrt{T}\log(T)\big)$. □

The following is an important corollary.

Corollary 5 *If $\mathcal{G}$ is a constant degree graph with correlation sets consisting of subsets of edges in $\mathcal{G}$, then there is a polynomial time algorithm that achieves a pseudo-regret bounded by $O(k^6(c+B)^2\sqrt{T}\log T)$.*

Temporal Dependencies and Change Aversion. The reader may note that real-world user feedback often contains delayed effects or contextual dependencies (e.g., users complaining specifically because the system is changing too frequently). While our stochastic model is Markovian (losses depend on the current state), it captures the "cost of change" via the fixing costs c_i. A high c_i explicitly models the penalty associated with altering a criterion, effectively discouraging rapid oscillations that might fatigue users. For deeper temporal dependencies where complaints depend on the history of states (non-Markovian dynamics), our *Adversarial Setting* (Section 5) provides a robust alternative, as it guarantees performance against arbitrary complaint sequences without relying on state-based distributional assumptions.

Remark 1 (On the Magnitude of m) The regret bounds for our general algorithms (e.g., Theorem 11 in Appendix B) depend exponentially on the correlation set size m. We acknowledge that if $m \approx k$, these bounds become vacuous, reflecting the fundamental hardness of learning an arbitrary function over 2^k states. However, in the context of user feedback systems, m is naturally small (typically $m \leq 3$) for two reasons:

1. **Cognitive Limits of Feedback:** User complaints are generated by human perception. Users typically react to single failures (e.g., "The app is slow") or pairwise tensions (e.g., "These recommendations are diverse but irrelevant"). It is cognitively implausible for a user to generate a complaint that is strictly conditional on a specific joint configuration of a large number of disparate metrics.
2. **System Sparsity:** Large-scale systems are modular. A metric related to UI accessibility is unlikely to be correlated with a metric related to database backend consistency. While the graph $\mathcal{G}$ may be large, the *interaction hyper-edges* (correlation sets) remain local.

Thus, while the theoretical worst-case is hard, the *practical* instances motivated by this work lie comfortably in the regime where $m \ll k$.

5 Adversarial Setting

5.1 Problem Setup

Motivated by the discussion on adversarial manipulation in Section 4, we consider a scenario where complaints arrive in an adversarial manner without any distributional assumptions. We consider an adversarial model where, at each time step, multiple complaints are submitted for the vertices of the incompatibility graph $\mathcal{G}$. As for the stochastic setting, initially, each vertex in $\mathcal{G}$ is in an *unfixed state* and has an associated fixing cost c_i. Each time, the algorithm can decide to fix a particular vertex and, as a result, its neighbors become unfixed. At time step t, if criterion v_i is unfixed, then the algorithm incurs a loss of $\ell_{i(t)}$ (which depends on the current state of the system), otherwise the algorithm incurs no loss. For an algorithm $\mathcal{A}$, during T time steps, the total loss is

$$\text{Loss}(\mathcal{A}) = \sum_{i=1}^{k} \sum_{t=1}^{T} \ell_{i(t)} \, \mathbb{I}(s_t(i) = 0) + \sum_{i=1}^{k} \sum_{t=2}^{T} c_i \, \mathbb{I}(s_{t-1}(i) = 0, s_t(i) = 1). \quad (16)$$

Let OPT be the algorithm that, given the entire loss sequence in advance, makes the decisions to fix vertices. We define the *competitive ratio* (Borodin and El-Yaniv, 1998) of $\mathcal{A}$ to be the maximum of $\text{Loss}(\mathcal{A})/\text{Loss}(\text{OPT})$ over all possible complaint sequences. We seek to devise efficient online algorithms for processing complaints that achieve a constant competitive ratio. In this setting, for the competitive ratio to remain finite, we must impose bounds on both the losses and the fixing costs of the vertices. Specifically, we assume that the cost of fixing each vertex is at least one, and as previously mentioned, the losses are bounded within the range $[0, B]$. For simplicity, we will assume that at each time step, a complaint is filed against a single vertex in $\mathcal{G}$. A straightforward reduction shows that an algorithm competitive with OPT in this simplified setting remains competitive in the more general case, with the same competitive ratio. This reduction will be discussed at the end of the section. As a result, we can represent the loss sequence in the form $((i_1, \ell_{i_1}), \ldots, (i_T, \ell_{i_T}))$, where i_t is the index of the vertex receiving the tth complaint, and ℓ_{i_t} is the corresponding loss.

To better understand the adversarial setting, consider the case where the graph $\mathcal{G}$ representing the criteria has no edges, that is, there are no conflicts between vertices. In this case, given a sequence of complaints, each with a unit loss value, the

Input: The graph $\mathcal{G}$, fixing costs c_i, loss sequence $(i_1, \ell_{i_1}), \ldots, (i_T, \ell_{i_T})$.

1. For each $i \in [k]$, initialize τ_i, κ_i to 0.
2. Process the complaints in sequence and for each complaint (i, ℓ_i) such that v_i is unfixed do:
   (a) $\tau_i = \tau_i + \ell_i$.
   (b) While $\ell_i > 0$ and exists $j \in N(i)$ with $\kappa_j > 0$ do:
      (i) Set $\Delta = \min(\ell_i, \kappa_i)$ and reduce both κ_i and ℓ_i by Δ.
   (c) If $\tau_i \geq \max\left(c_i, \sum_{j \in N(i)} \kappa_j\right)$ fix v_i. Set τ_i to 0 and κ_i to c_i. Set $\tau_j = 0$ for all $j \in N(i)$.

Figure 6: Online algorithm for the adversarial setting.

optimal offline algorithm, which has access to the entire loss sequence in advance, can independently decide for each vertex. Specifically, if the total loss from complaints at vertex v_i exceeds its fixing cost c_i, the optimal decision is to fix vertex v_i; otherwise, the algorithm incurs the loss from the complaints.

In this simplified scenario, the online algorithm can also treat each vertex independently. At each vertex, the algorithm faces the classical *ski-rental* problem, for which there exists a deterministic algorithm that is 2-competitive with the optimal solution (Karlin et al., 1988). The online algorithm waits until the total loss at vertex i reaches or exceeds c_i before fixing it. It is straightforward to see that the total cost incurred by this strategy is at most twice the cost incurred by OPT.

However, the algorithm described above performs poorly in the presence of conflicts within the graph $\mathcal{G}$. Consider a simple example with two vertices, v_i and v_j, connected by an edge. Let the fixing cost of v_i be 1, and the fixing cost of v_j be $C \gg 1$. Now, imagine a sequence of complaints, each causing a unit loss, consisting of C complaints for v_j, followed by one complaint for v_i. If this sequence is repeated over T cycles, the optimal offline algorithm (OPT) would incur a total cost of $C+T$, fixing v_j and absorbing the losses from v_i. In contrast, the previously discussed algorithm would incur a cost of $(2C + 2)T$, leading to an unbounded competitive ratio.

To achieve a competitive ratio that remains constant, decisions must be made not just based on the losses at an individual vertex v_i, but also considering the state of its neighboring vertices. Our main result in this section is an algorithm (detailed in Figure 6) that achieves a constant-factor competitive ratio.

5.2 The Barrier Algorithm

Our main result in this section is an algorithm (detailed in Figure 6) that achieves a constant-factor competitive ratio.

The algorithm in Figure 6 makes decisions based on the local neighborhood of each vertex. Intuitively, if the optimal algorithm fixes a vertex only once or a few times, the goal is to avoid fixing it excessively. To achieve this, whenever a vertex v_i is fixed, it creates a 'barrier' for its neighbors: specifically, a threshold $\kappa_i = c_i$ is added to the loss that any neighboring vertex must incur before it can be fixed. As a result, if a vertex is connected to many fixed vertices, it faces a higher threshold before it

can be fixed. Throughout the algorithm, each unfixed vertex operates in one of two phases. In the first phase, the vertex accumulates losses in order to surpass the barriers introduced by its neighbors (see step 2(b) of the algorithm).

In phase two, once the barrier has been crossed, the vertex follows the standard ski-rental strategy, making independent decisions about whether to fix itself or not. Notably, through step 2(b) of the algorithm, multiple neighbors of a vertex v_i can contribute to reducing the barrier c_i that was introduced when v_i was fixed. This coordination is crucial to prevent the online algorithm from incurring excessive losses by delaying the fixing of a vertex for too long.

Consider a graph $\mathcal{G}$ with k vertices and $k-1$ edges, where vertex v_0 is the central vertex connected to all other vertices. Let the fixing cost of v_0 be a large value C, while the fixing cost of all other vertices is 1. Now, suppose we have a sequence of C complaints, each with a unit loss, first submitted for vertex v_0, followed by C complaints for v_1, and so on. The optimal offline solution incurs a total loss of $C+k$ by fixing every vertex except v_0.

In contrast, the online algorithm, after processing C complaints for v_0, will fix v_0 and incur a loss of $2C$. As it processes C complaints for v_1, it will fix v_1, incurring an additional loss of $C+1$. Crucially, due to step 2(b) of the algorithm, the barrier κ_0 introduced by v_0 has been reduced to zero, so for the remaining vertices, the algorithm incurs only a loss of 2 per vertex, resulting in a total loss of $3C+2k-1$. Without step 2(b), each vertex would incur a loss of C, leading to a significantly larger competitive ratio.

Note that the algorithm in Figure 6 is designed for scenarios where, at each time step, complaints are submitted for a single vertex of $\mathcal{G}$. If complaints accumulate for multiple vertices in the same time step, we can simply process them sequentially by choosing an arbitrary order and running the algorithm on the resulting sequence. Let OPT represent the optimal offline algorithm based on the chosen order of complaints, and let OPT' represent the optimal offline algorithm when handling multiple complaints simultaneously per time step. Since, at any time step, the loss incurred by OPT cannot exceed that of OPT', as every decision available to OPT' is also available to OPT, it is sufficient to design an algorithm competitive with OPT. This leads to the following theorem, which is the main result of this section.

5.3 Competitive Analysis

Theorem 6 *Let $\mathcal{G}$ be a graph with fixing costs at least one. Then, the algorithm of Figure 6 achieves a competitive ratio of at most $2B+4$ on any sequence of complaints with loss values in $[0, B]$.*

Proof. Recall that $\ell_{i(t)}$ denotes the loss incurred by vertex v_i at time t. We divide this loss into the amount that was used to reduce the κ_j value of one of its neighbors and the rest. Formally, for every edge (i,j) we define $\delta^t_{i\to j}$ as follows. If in time step t, the complaint is submitted for vertex i and step 2(b) was executed to reduce κ_j by Δ,

then we define $\delta^t_{i\to j} = \Delta$. Otherwise, we define $\delta^t_{i\to j}$ to be zero. We also define

$$\delta^t_{i\to i} = \ell_{i(t)} - \sum_{j\in N(i)} \delta^t_{i\to j}. \tag{17}$$

If vertex v_i is fixed f_i times during the course of the algorithm then we have that the total loss incurred by the algorithm can be written as

$$\text{Loss}(\mathcal{A}) = \sum_{i=1}^{k} f_i c_i + \sum_{i=1}^{k}\sum_{t=1}^{T} \big(\delta^t_{i\to i} + \sum_{j\in N(i)} \delta^t_{i\to j}\big). \tag{18}$$

Next, we observe that each time a vertex v_i is fixed, it accumulates a value of $\kappa_i = c_i$. Moreover, the total loss incurred by the vertices due to the execution of step 2(b) is upper-bounded by the total accumulated κ values. Thus, we have

$$\sum_{t=1}^{T}\sum_{i=1}^{k}\sum_{j\in N(i)} \delta^t_{i\to j} \le \sum_{i=1}^{k} f_i c_i. \tag{19}$$

Substituting into (18) we have

$$\text{Loss}(\mathcal{A}) \le \sum_{i=1}^{k} 2 f_i c_i + \sum_{i=1}^{k}\sum_{t=1}^{T} \delta^t_{i\to i}. \tag{20}$$

Next, we analyze the loss for each vertex separately. For a given vertex v_i that is fixed f_i times by the algorithm, we can partition the time steps into $f_i + 1$ intervals. The first interval, denoted as I_0, spans from $t = 0$ to (and including) the first time v_i is fixed. The subsequent f_i intervals correspond to the periods between successive fixes of v_i. Denoting these intervals as $I_0, I_1, \ldots$, we have

$$2 f_i c_i + \sum_{i=1}^{k}\sum_{t=1}^{T} \delta^t_{i\to i} = \sum_{t\in I_0} \delta^t_{i\to i} + \sum_{t\in I_r} (2c_i + \delta^t_{i\to i}). \tag{21}$$

Next, we compare the above to the loss incurred by OPT for vertex v_i. Let $\ell^*_{i(t)}$ be the loss incurred by OPT for vertex v_i at time t. We will denote by s^*_t the state of the vertices at time t according to OPT. We instead redefine the loss incurred by OPT for vertex v_i at time t to be

$$\tilde{\ell}_{i(t)} = \ell^*_{i(t)} + \sum_{j\in N(i)} \delta^t_{j\to i}\, \mathbb{I}(s^*_t(j) = 0). \tag{22}$$

Note that we have $\sum_{i\in N(j)} \delta^t_{j\to i}\,\mathbb{I}(s^*_t(j)=0) \le \ell^*_{j(t)}$. Thus, we can write

$$\sum_{i=1}^{k}\sum_{t=1}^{T}\tilde{\ell}_{i(t)} \le \sum_{i=1}^{k}\Big(\sum_{t=1}^{T}\ell^*_{i(t)} + \sum_{j\in N(i)}\ell^*_{j(t)}\Big) \le 2\text{Loss}(\text{OPT}). \tag{23}$$

Next we consider each interval in (20) separately. For any interval I_r we have that

$$\sum_{t\in I_r}\delta^t_{i\to i} \le Bc_i. \tag{24}$$

This is because after incurring a loss of more than c_i, any additional loss incurred by v_i is due to step 2(b), since otherwise step 2(c) will be executed and v_i will be fixed.

Next consider interval I_0. The loss incurred by the algorithm on vertex v_i equals $\sum_{t\in I_0}\delta^t_{i\to i} \le Bc_i$. Either OPT fixes v_i at least once during this interval or incurs the total loss. Either way we have that the loss incurred by OPT is at least

$$\min\big(c_i, \sum_{t\in I_0}\delta^t_{i\to i}\big) \ge \frac{\sum_{t\in I_0}\delta^t_{i\to i}}{B}. \tag{25}$$

Next, consider an interval I_r between two successive fixes. The loss incurred by the algorithm for vertex v_i during this interval is at most

$$\sum_{t\in I_r}\delta^t_{i\to i} + 2c_i \le (B+2)c_i.$$

If OPT fixes vertex v_i at least once during this interval, it incurs a cost of c_i. Conversely, if v_i remains unfixed in OPT throughout the interval, it incurs a loss of at least c_i. This is due to the fact that v_i transitions from being unfixed to fixed during the latter half of the interval, which means that a total loss of at least c_i must have been incurred for vertex v_i during this timeframe.

Now, let us consider the scenario where vertex v_i is fixed in OPT before the start of the interval and remains fixed throughout. Since v_i changes from being fixed to unfixed in the first half of the interval, it follows that

$$\sum_{t\in I_r}\sum_{j\in N(i)}\delta^t_{j\to i} \ge c_i.$$

Moreover, because v_i is fixed by OPT during this interval, OPT incurs a loss on all neighbors of j. Specifically, from equation (22), we have:

$$\sum_{t\in I_r}\tilde{\ell}_{i(t)} \ge \sum_{t\in I_r}\sum_{j\in N(i)}\delta^t_{j\to i}\,\mathbb{I}(s^*_t(j)=0) \ge c_i. \tag{26}$$

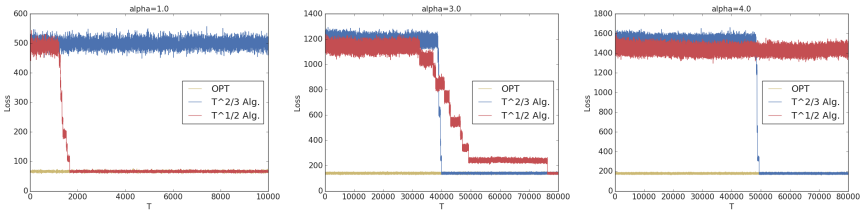

Figure 7: Cumulative loss of the Algorithms of Figure 3 and Figure 5 on a graph with $k = 100$ criteria. $\alpha = 1, 3, 4$ determines the number of random pairs of vertices, αk, added into correlation sets.

In anyone of the three cases, the loss $\sum_{t \in I_r} \tilde{\ell}_{i(t)}$ incurred by OPT is at least a $1/(B+2)$ fraction of the loss incurred by the algorithm. Summing over all the vertices and the corresponding intervals, we can bound the total loss incurred by the algorithm by

$$\text{Loss}(\mathcal{A}) \leq (B+2) \sum_{t=1}^{T} \sum_{i=1}^{k} \tilde{\ell}_{i(t)} \leq 2(B+2)\text{Loss}(\text{OPT}),$$

which completes the proof. □

6 Experiments

In this section, we report the resuts of a series of experiments with our algorithms given for the stochastic setting.

6.1 Experiments with Simulated Data

We first evaluated the performance of our stochastic setting algorithms (Section 4) on simulated data. We generated the graph $\mathcal{G}$ from the Erdős-Rényi model: $G(k, p)$ and set $p = 2\frac{\log k}{k}$, to ensure connectivity.

We generated correlation sets ($\mathcal{C}$) of size two by picking αk pairs of vertices at random and adding them to $\mathcal{C}$, where α is a parameter. Hence, on average, each vertex is in α correlation sets. We also added to $\mathcal{C}$ singleton sets for each vertex in $\mathcal{G}$. The fixing cost of a vertex was sampled uniformly in $[1, 5]$. Parameters governing the loss distribution were drawn from a Beta distribution. We approximated the oracle for the optimization in (4) via a linear programming relaxation. Our algorithms of Figure 3 and Figure 5 admit complementary guarantees. The former admits a higher regret as a function of T, but only a polynomial dependence on α, i.e., the average number of correlation sets a vertex participates in. The latter incurs a smaller regret of $\tilde{O}(\sqrt{T})$ as a function of T at the expense of an exponential dependence on α. Figure 7 shows an empirical illustration of this: for smaller values of α, the $\tilde{O}(\sqrt{T})$ regret algorithm performs significantly better, while for larger values of α the $\tilde{O}(T^{2/3})$ regret algorithm is more desirable. We choose to compare the performance of our two proposed stochastic algorithms as we are not aware of any existing baselines for simultaneously optimizing multiple diverse metrics. Additionally, we focus on

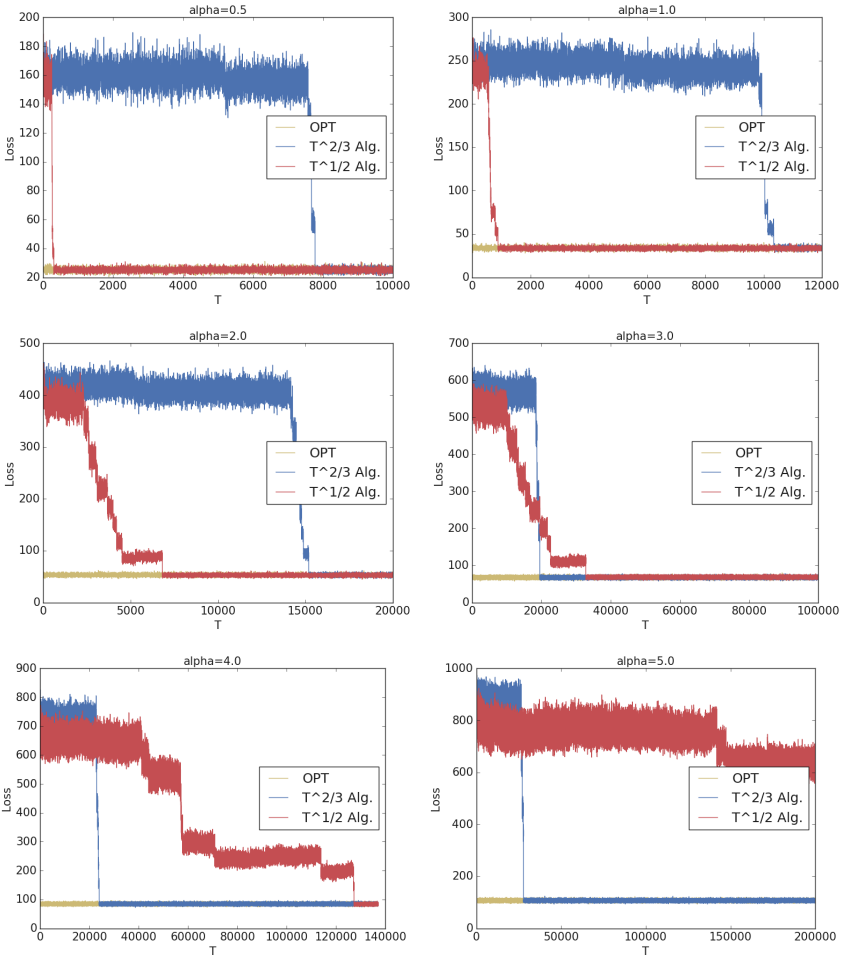

Figure 8: The figure shows the total accumulated loss incurred by the Algorithms in Figure 3 and Figure 5 on a graph with $k = 50$ criteria. The parameter α controls the total number of correlation sets. For each value of α, we add αk random pairs of vertices into correlation sets.

experiments with the stochastic model as it is hard to approximate the best offline algorithm in the adversarial setting of Appendix 5.

We consider a simulated environment where the incompatibility graph $\mathcal{G}$ is generated from the Erdős-Renyi model: $G(k, p)$ where we set $p = 2\frac{\log k}{k}$. This ensures that with high probability $\mathcal{G}$ is connected. Next we generate correlation sets $\mathcal{C}$ consisting of pairs of vertices in $\mathcal{G}$ sampled uniformly at random. For a parameter $\alpha > 0$ that we vary, we choose αk pairs of vertices at random and add them as correlation sets in $\mathcal{C}$. Hence on average, each vertex participates in α correlation sets. We also add to $\mathcal{C}$

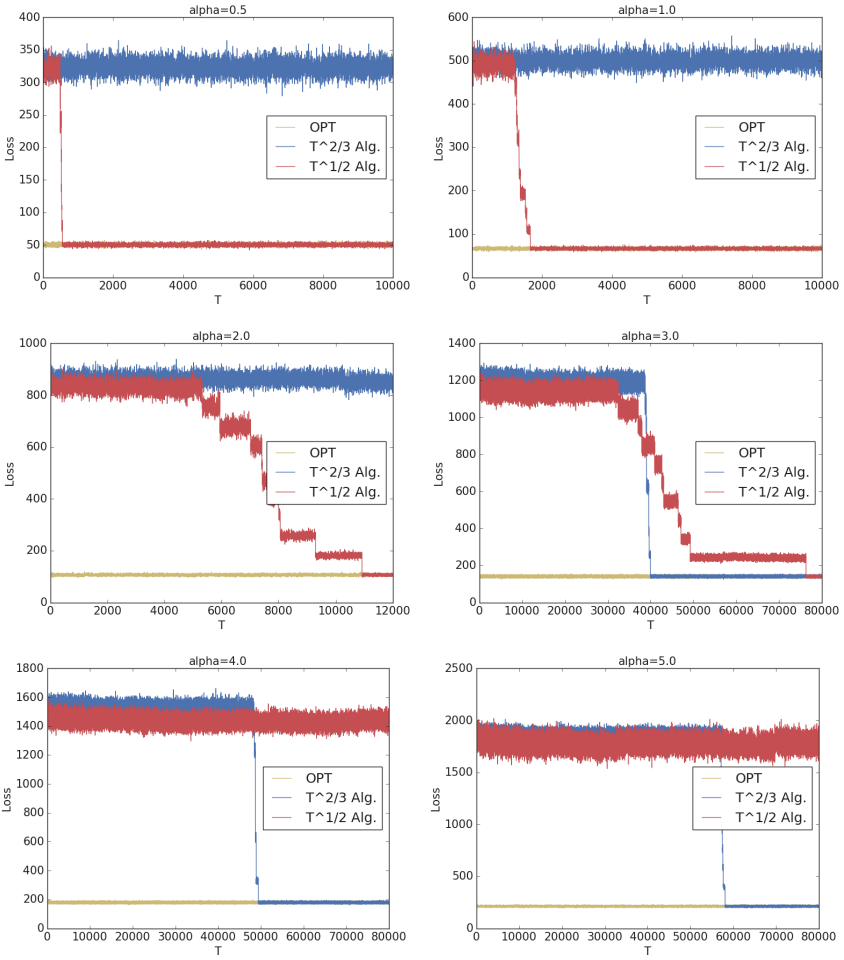

Figure 9: The figure shows the total accumulated loss incurred by the Algorithms in Figure 3 and Figure 5 on a graph with $k = 100$ criteria. The parameter α controls the total number of correlation sets. For each value of α, we add αk random pairs of vertices into correlation sets.

singleton sets for each vertex in $\mathcal{G}$. The fixing cost of each vertex is samples uniformly at random in the range $[1, 5]$.

Next we describe the choice of parameters governing the loss distribution of the different states in the MDP. For a correlation set $\{i\}$ of size one corresponding to vertex v_i, we sample a parameter γ_i^1 from the beta distribution $\text{Beta}(0.5, 0.5)$. For a given state s with $s(i) = 1$, the loss generated due to $\{i\}$ is drawn from an exponential distribution with mean γ_i^1. For a given state s with $s(i) = 0$, the loss generated due to $\{i\}$ is drawn from an exponential distribution with mean $\lambda\gamma_i^1$, where $\lambda > 1$ is a parameter that we vary. For a correlation set $\{i, j\}$ of size two, we

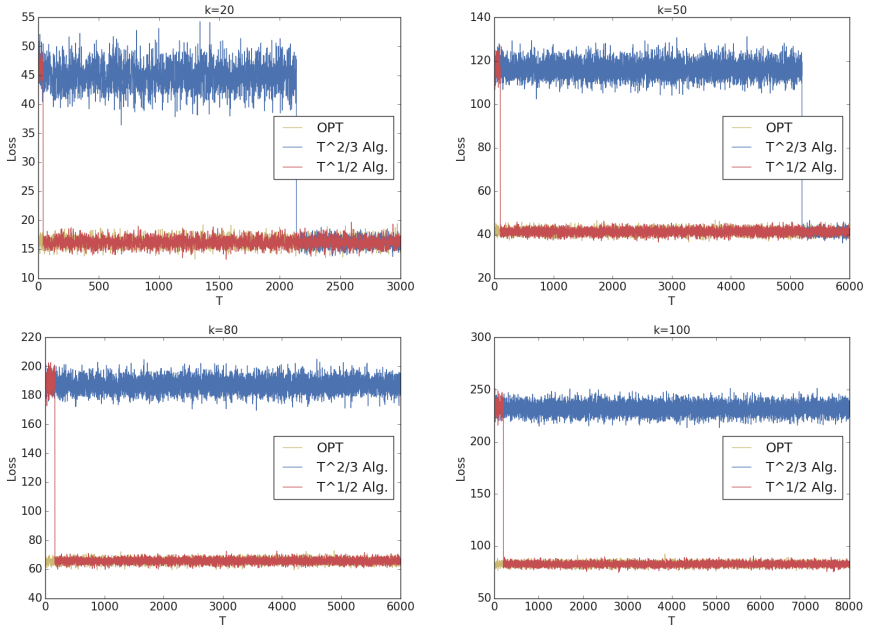

Figure 10: The figure shows the total accumulated loss incurred by the Algorithms in Figure 3 and Figure 4 for the case of $m = 1$ and varying graph sizes.

generate two parameters $\gamma_{i,j}^{1,1}$ and $\gamma_{i,j}^{1,0}$ from the beta distribution Beta$(0.5, 0.5)$ such that $\gamma_{i,j}^{1,0} > \gamma_{i,j}^{1,1}$. For a given state s with $s(i) = 1$ and $s(j) = 1$, the loss generated due to $\{i, j\}$ is drawn from an exponential distribution with mean $\gamma_{i,j}^{1,1}$. For states where $s(i) = 0$ and $s(j) = 1$ or vice-versa, the loss is generated from an exponential distribution with mean $\gamma_{i,j}^{1,0}$. Finally, for states where both $s(i) = 0$ and $s(j) = 0$, the loss is generated from an exponential distribution with mean $\lambda\gamma_{i,j}^{1,0}$.

In general, computation of the optimal state in (4) requires time exponential in k. In our experiments we approximate the optimal state by a linear programming relaxation of the optimization in (4) and use the appropriately rounded linear programming relaxation solution as a proxy for the optimal state.

For general m, our proposed algorithms in Figure 3 and Figure 5 have complementary strengths. While the algorithm in Figure 3 incurs a higher regret as a function of the number of time steps T, its running time has a polynomial dependence on the parameter α, i.e., the number of correlation sets that a vertex participates in, on average. The algorithm in Figure 5 incurs a smaller regret of $\tilde{O}(\sqrt{T})$ as a function of T at the expense of an exponential dependence on α. In Figures 8 and 9 we empirically demonstrate this behavior where for small values of α, the $\tilde{O}(\sqrt{T})$-regret algorithm is much better, whereas for higher values of α the $\tilde{O}(T^{2/3})$-regret algorithm is more desirable.

For the case of $m = 1$ however, i.e., singleton correlation sets, the algorithm in Figure 5 achieves a smaller regret and runs in polynomial time and hence is expected to outperform the explore-exploit based algorithm from Figure 3. As can be seen

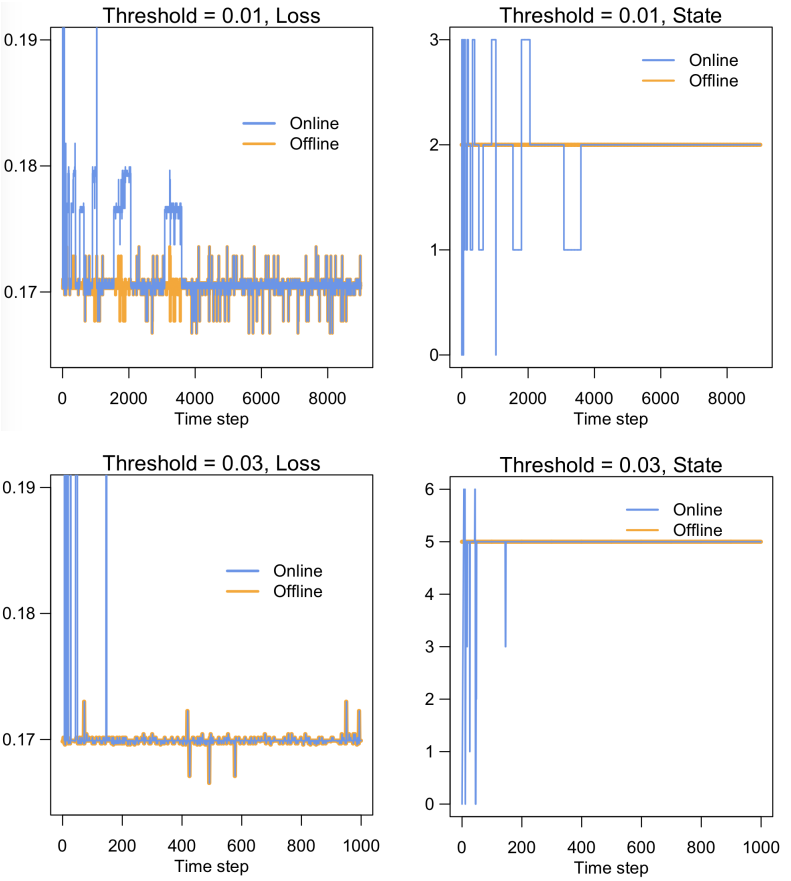

Figure 11: The figure shows the performance of the Algorithm in Figure 4 on the UCI Adult dataset. We present results for two threshold values, and in each case plot the loss of the offline solution and the online algorithm as well as the states chosen by the online algorithm, as a function of the time steps.

from Figure 10 this is indeed the case and the $\tilde{O}(\sqrt{T})$ regret algorithm significantly outperforms the $\tilde{O}(T^{2/3})$ regret algorithm.

6.2 Experiments with a Real-World Dataset

In this section, we demonstrate through experiments how our framework and algorithms can be applied to real-world data. To achieve this, we study the UCI Adult dataset (Kohavi, 1996), which consists of $48{,}852$ examples, each represented by 124 features after binarizing the categorical attributes. Each data point corresponds to an individual, with a label indicating whether their income exceeds $50{,}000$ (represented as a binary value of 0 or 1). The dataset includes sensitive attributes such as race and gender.

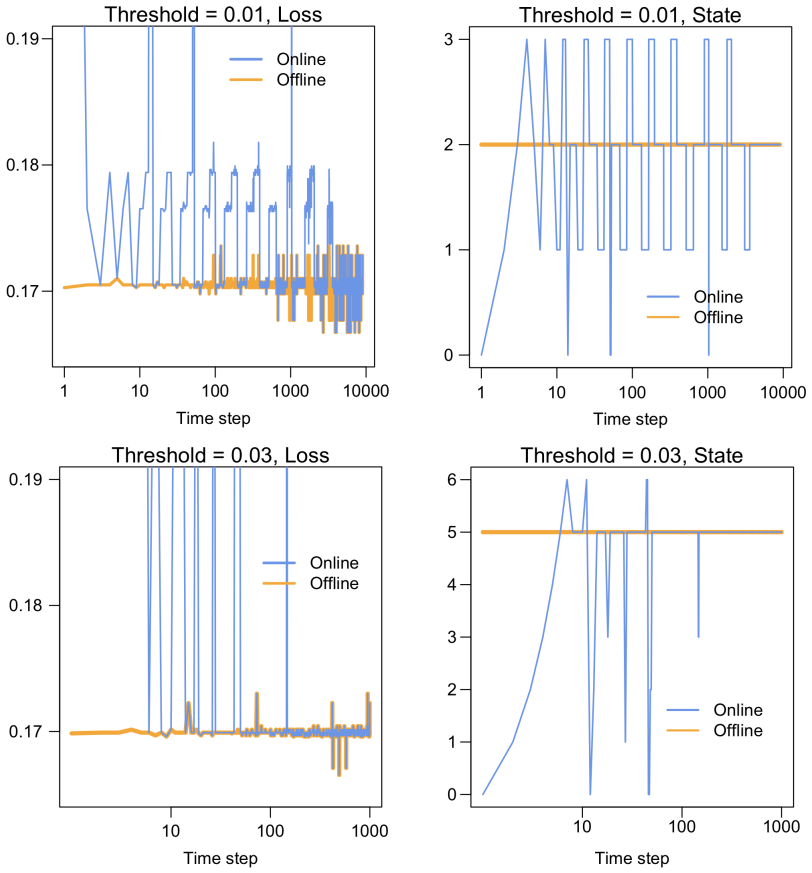

Figure 12: The figure shows the performance (x-axis on a log scale) of the Algorithm in Figure 4 on the UCI Adult dataset. We present results for two threshold values, and in each case plot the loss of the offline solution and the online algorithm as well as the states chosen by the online algorithm, as a function of the time steps.

We simulate an online scenario in which a classifier predicts individuals' incomes. At each time step, a batch of complaints is received, resulting in an incurred loss. The system then responds by transitioning to a different state and updating the classifier accordingly. Next, we describe how we instantiate various components of our stochastic model as outlined in Section 4.

Graph $\mathcal{G}$: We consider race as a sensitive attribute with values in $\{\text{black}, \text{white}\}$, which defines two sub-populations. We focus on two key performance metrics: the true positive rate (TPR) and the area under the curve (AUC) score. Additionally, we include overall classifier accuracy as a fundamental performance requirement. This results in a graph with $k = 5$ vertices: $\{tpr_w, tpr_b, auc_w, auc_b, \text{accuracy}\}$. Consequently, the state space is $\mathcal{S} \subseteq \{0, 1\}^5$.

Losses and Correlation Sets: We consider correlation sets of size one, meaning that the total loss incurred at any state is the sum of the losses associated with each criterion. For the accuracy criterion, we define the loss simply as the error of the classifier.

Next, we explain how we define the loss for the tpr_w criterion. We first compute the overall true positive rate (TPR) of the classifier and also the true positive rate (TPR) on the white population. If the two deviate by more than a threshold τ, we impose a linear penalty proportional to the extent of the violation. Thus, the loss for tpr_w is defined as: $\max(0, |tpr_{overall} - tpr_w| - \tau)$. We apply the same loss definition for all other criteria. In our experiments, we set $\tau = 0.005$.

Incompatibilities and State Transitions: To generate incompatibilities among criteria we compute a set of valid and invalid states as follows. For each state $s \in \{0, 1\}^5$, we solve a constrained optimization problem on a training set to compute a classifier. We then evaluate the classifier on the test set to compute the loss of each criterion. If the loss of any criterion is more than a specific threshold then we consider the state as an invalid state, otherwise the state is valid. In our experiments we set a threshold of 0.4 for the accuracy criterion.

We present results for two thresholds: 2τ and 6τ. The first one resulting in 4 valid states and other second one resulting in 7 valid states. To solve the constrained optimization problem, we use the tensorflow constrained optimization toolkit (Cotter et al., 2019,?). We use the default parameter settings provided by the toolkit. The toolkit is released under Apache license 2.0.

If a state s has accuracy criterion set to 1, then we optimize for model accuracy subject to constraints for the other criteria that are set to 1 in s. If the accuracy criterion is set to 0 then we optimize for a constant loss function subject to constraints. Recall that our proposed algorithms function by fixing a criterion and as a result moving to another state. We obtain these state transitions as follows. If the algorithm asks to fix criterion v_i in state s, we set $s(i) = 1$ to go to the next state s'. While s' is invalid, we unfix the criterion (not including v_i) with the highest loss in the state s' to reach another state.

Fixing Cost: We simply take the fixing cost of each criterion to be 1.

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

Benchmark and Results: We compare our Algorithm from Figure 4 with an offline optimal solution that has been computed to find the state with the minimum average loss over the arrival sequence of complaints. The results are averaged over 10 independent runs.

Discussion of Results on Real-World Dataset and Additional Figures: The results are shown in Figure 11 and Figure 12. We show results for two values of the threshold parameters and in each case 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 number of time steps. As can be seen from Figure 11 our algorithm quickly converges to the

offline optimal solution after an initial exploration phase. To get a better understanding of the performance of the algorithm in the initial phases, in Figure 12 we plot the same setting as in the case of Figure 11, but with x-axis on a log-scale. For the case of threshold being 0.01, one can see that the state 0 results in much higher loss and, during exploration, the algorithm alternates in a periodic pattern between states 1 and 3 that have similar loss. A similar pattern holds for the case of the threshold being 0.03. It is important to 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.

Compute Resources. All our experiments were performed on a machine containing a Tesla P100 GPU with 80 GB of RAM and four CPUs.

Hyperparameters. For the case of simulated data the hyperparameters have been mentioned in Section 6.1. For the case of real data, apart from the hyperparameters mentioned in Section 6.2, we used the default learning rates and optimizers provided by the tensorflow constrained optimization toolkit (Cotter et al., 2019,?). We performed a random train/test split as detailed in Section 6.2.

Assets. We used publicly available code from the tensorflow constrained optimization toolkit[2] and the publicly available UCI Adult Dataset[3].

7 Conclusion and Future Work

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 and open questions remain.

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 game-theoretic scenario is a direction for future work.

Continuous States and Spectrums. While our model formally defines states as binary (fixed/unfixed), it can effectively approximate continuous criteria or "spectrums" of incompatibility via discretization (e.g., multiple vertices representing "strict" vs "loose" constraints). Furthermore, subjective ambiguity in criteria is naturally handled by the stochastic loss function: if user opinions are divided, the algorithm observes the aggregate complaint volume as a signal and optimizes the trade-off accordingly. Future work could explore extending the state space $\mathcal{S}$ directly to continuous domains to avoid the discretization overhead.

Dynamic vs. Static Fairness. Current fairness approaches often rely on static definitions based on predefined protected attributes (e.g., race or gender). However,

[2]License at: https://github.com/google-research/tensorflow_constrained_optimization/blob/master/README.md.
[3]https://archive.ics.uci.edu/ml/datasets/adult.

real-world systems often impact "emergent" social groups defined by complex behavioral traits (e.g., users with specific usage patterns or niche interests) that static attributes fail to capture. Our model advocates for a *dynamic* definition of fairness driven by user reactions. By optimizing for aggregate complaint losses, the system can adapt to protect these evolving groups and address hidden unfairness effects that are only revealed post-deployment, offering a more flexible alternative to rigid constraint-based approaches.

Finally, future research could explore time-dependent fixing costs to capture varying algorithmic prices or human effort costs, as well as non-stationary loss distributions to express the growing urgency of unaddressed criteria.

Acknowledgements

YM has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation, the Yandex Initiative for Machine Learning at Tel Aviv University and a grant from the Tel Aviv University Center for AI and Data Science (TAD).

References

Agarwal, A., A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*, pp. 60–69. PMLR.

Angwin, J., J. Larson, S. Mattu, and L. Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. ProPublica.

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

Backurs, A., P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner 2019. Scalable fair clustering. In *Advances in Neural Information Processing Systems*, Volume 32, pp. 12892–12903.

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

Bechavod, Y., C. Jung, and Z.S. Wu 2020. Metric-free individual fairness in online learning. In *Advances in Neural Information Processing Systems*, Volume 33, pp. 12333–12344.

Bellamy, R.K., K. Dey, M. Hind, S.C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, et al. 2018. AI Fairness 360: An extensible

toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. arXiv preprint arXiv:1810.01943.

Beutel, A., J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E.H. Chi, et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD*.

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

Celis, L.E., D. Straszak, and N.K. Vishnoi 2018. Ranking with fairness constraints. In *International Colloquium on Automata, Languages and Programming (ICALP)*.

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

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

Chierichetti, F., R. Kumar, S. Lattanzi, and S. Vassilvitskii 2017. Fair clustering through fairlets. In *Neural Information Processing Systems (NIPS)*.

Combes, R., S. Magureanu, and A. Proutiere 2017. Minimal exploration in structured stochastic bandits. In *Advances in Neural Information Processing Systems*, Volume 30, pp. 3349–3357.

Cortes, C., V. Kuznetsov, M. Mohri, and S. Yang 2016. Structured prediction theory based on factor graph complexity. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. Curran Associates, Inc. Also available on arXiv: arXiv:1605.06443 :contentReference[oaicite:0]index=0.

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

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

Cotter, A., M. Gupta, H. Jiang, N. Srebro, K. Sridharan, S. Wang, B. Woodworth, and S. You. 2019. Training well-generalizing classifiers for fairness metrics and other data-dependent constraints. *Journal of Machine Learning Research 20*(26): 1–50 .

Cotter, A., H. Jiang, and K. Sridharan 2019. Two-player games for efficient non-convex constrained optimization. In A. Garivier and S. Kale (Eds.), *Algorithmic*

Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA, Volume 98 of *Proceedings of Machine Learning Research*, pp. 300–332. PMLR.

Doroudi, S., P.S. Thomas, and E. Brunskill 2017. Importance sampling for fair policy selection. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

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

Feller, A., E. Pierson, S. Corbett-Davies, and S. Goel. 2016, October. 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., D. Basu, and K.S. Meel 2021. Justicia: A stochastic SAT approach to formally verify fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 35, pp. 3755–3763.

Gillen, S., C. Jung, M. Kearns, and A. Roth 2018. Online learning with an unknown fairness metric. In *Advances in Neural Information Processing Systems*.

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

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

Hashimoto, T.B., M. Srivastava, H. Namkoong, and P. Liang 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pp. 1929–1938. PMLR.

Holstein, K., J. Wortman Vaughan, H. Daumé III, M. Dudik, and H. Wallach 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*, pp. 1–16.

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

Jaksch, T., R. Ortner, and P. Auer. 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 B. Sendhoff. 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 .

Joseph, M., M. Kearns, J.H. Morgenstern, and A. Roth 2016. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems*, pp. 325–333.

Kaminskas, M. and D. Bridge. 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., A. Roth, and J. Ziani 2019. Downstream effects of affirmative action. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 240–248.

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

Kearns, M., A. Roth, and S. Sharifi-Malvajerdi 2019. Average individual fairness: Algorithms, generalization and experiments. In *Advances in Neural Information Processing Systems*, Volume 32, pp. 8240–8249.

Kleinberg, J., S. Mullainathan, and M. Raghavan 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, pp. 202–207.

Kschischang, F.R., B.J. Frey, and H.A. Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory 47*(2): 498–519. https://doi.org/10.1109/18.910572 .

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

Lin, X., H. Chen, C. Pei, F. Sun, X. Xiao, H. Sun, Y. Zhang, W. Ou, and P. Jiang 2019. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender systems*, pp. 20–28.

Liu, L.T., S. Dean, E. Rolf, M. Simchowitz, and M. Hardt 2018. Delayed impact of fair machine learning. In *International Conference on Machine Learning*, pp. 3150–3158. PMLR.

Liu, Y., G. Radanovic, C. Dimitrakakis, D. Mandal, and D.C. Parkes. 2017. Calibrated fairness in bandits. arXiv preprint arXiv:1707.01875.

Loeliger, H.A. 2004. An introduction to factor graphs. *IEEE Signal Processing Magazine 21*(1): 28–41. https://doi.org/10.1109/msp.2004.1267047 .

Marler, R.T. and J.S. Arora. 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, *Recommender systems handbook*, 677–702. Springer.

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

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

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

Narasimhan, H., A. Cotter, M. Gupta, and S. Wang 2020. Pairwise fairness for ranking and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34-04, pp. 5248–5255.

Puterman, M.L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Schmidt, M., C. Schwiegelshohn, and C. Sohler 2019. Fair coresets and streaming algorithms for fair k-means clustering. In *International Workshop on Approximation and Online Algorithms*, pp. 232–251. Springer.

Sener, O. and V. Koltun 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, Volume 31, pp. 2901–2911.

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

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

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

Tsirtsis, S. and M. Gomez-Rodriguez 2020. Decisions, counterfactual explanations and strategic behavior. In *Advances in Neural Information Processing Systems*,

Volume 33, pp. 1671–1682.

Wang, S., W. Guo, H. Narasimhan, A. Cotter, M. Gupta, and M.I. Jordan 2020. Robust optimization for fairness with noisy protected groups. In *Advances in Neural Information Processing Systems*, Volume 33, pp. 5330–5343.

Wen, M., O. Bastani, and U. Topcu 2021. Fairness with dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 35, pp. 10350–10358.

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

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

A Stochastic Setting

We first show that in the stochastic model, if correlation sets are of size one then one can efficiently approximate the cost of the optimal state up to a factor of two.

Theorem 7 *If correlations sets are of size one ($m = 1$), then, for any $\epsilon, \delta > 0$, the true parameter vector for MDP$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$ can be approximated to ϵ-accuracy in ℓ_∞-norm with probability at least $1 - \delta$, in at most $O(\frac{B^2 k}{\epsilon^2} \log(\frac{k}{\delta}))$ time steps and exploring at most $k + 1$ specific states in $\mathcal{S}$. Furthermore, given a parameter vector $\boldsymbol{\theta}$, there is an algorithm that runs in time polynomial in k and finds an approximately optimal state s' such that $g(s') \leq 2 \min_{s \in \mathcal{S}} g(s)$.*

Proof. Notice that when correlation sets are of size one, the expected loss incurred for criterion v_i at any given state s solely depends on whether $s(i) = 0$ or $s(i) = 1$. Hence in this case the MDP consists of $2k$ parameters where we use γ_i^1 and γ_i^0 to denote the expected losses incurred by vertex i when it is in fixed and unfixed position respectively. For any $\delta > 0$, by Hoeffding's inequality, we have that if we stay in state $s = (0, 0, \ldots, 0)$ for $N = \frac{B^2}{\epsilon^2} \log(2k/\delta)$ time steps then with probability at least $1 - \frac{\delta}{2}$, we have each γ_i^0 estimated up to ϵ accuracy. Let $e_i \in \{0, 1\}^k$ denote the indicator vector for i. If we stay in state $s = e_i$ for $\frac{B^2}{\epsilon^2} \log(2k/\delta)$ time steps, then with probability at least $1 - \frac{\delta}{2}$ we have γ_i^1 estimated up to ϵ accuracy. Hence, overall after $O(\frac{B^2 k}{\epsilon^2} \log(\frac{k}{\delta}))$ time steps, we have each parameter estimated up to ϵ accuracy. Notice that in total we observe at most $k + 1$ states.

Next, we show how to efficiently approximate the loss of the best state. Given the parameters of the MDP each vertex has two costs $\Lambda_i^{(1)} = \gamma_i^0$, denoting the cost incurred if the vertex is unfixed and $\Lambda_i^{(2)} = c_i + \gamma_i^1$, denoting the cost incurred if the vertex is fixed. Without loss of generality assume that $\Lambda_i^{(1)} > \Lambda_i^{(2)}$ (any vertex that does not satisfy this can be safely left unfixed). For each i, define $y_i = 1$ if vertex i is unfixed otherwise define $y_i = 0$. Then the offline problem of finding the best state can be written as

$$\begin{aligned} \min \sum_{i=1}^{k} (1 - y_i)\Lambda_i^2 + y_i \Lambda_i^1 &= \sum_{i=1}^{k} y_i \gamma_i + \sum_{i=1}^{k} \Lambda_i^{(2)} \\ \text{s.t. } y_i &\in \{0, 1\} \\ y_i + y_j &\geq 1, \ \forall (v_i, v_j) \in E. \end{aligned}$$

Here $\gamma_i = \Lambda_i^{(1)} - \Lambda_i^{(2)} > 0$. By relaxing y_i to be in $[0, 1]$ and solving the corresponding linear programming relaxation, we get a solution $y_1^*, y_2^*, \ldots, y_k^*$. Let LPval denote the linear programming objective value achieved by $y_1^*, y_2^*, \ldots, y_k^*$. Since the linear programming formulation is a valid relaxation of the problem of finding the best state, we have LPval $\leq \min_{s \in \mathcal{S}} g(s)$.

We output the state s' in which a vertex i if and only if $y_i^* < 1/2$. Let S be the set of fixed vertices. We have

$$\begin{aligned} g(s') &= \sum_{i\in S} \Lambda_i^{(2)} + \sum_{i\notin S} \Lambda_i^{(1)} \\ &= \sum_{i=1}^{k} \Lambda_i^{(2)} + \sum_{i\notin S} (\Lambda_i^{(1)} - \Lambda_i^{(2)}) \\ &= \sum_{i=1}^{k} \Lambda_i^{(2)} + \sum_{i\notin S} \gamma_i \\ &< \sum_{i=1}^{k} \Lambda_i^{(2)} + 2\sum_{i\notin S} y_i^* \gamma_i \\ &< 2\Big(\sum_{i=1}^{k} \Lambda_i^{(2)} + \sum_{i=1}^{k} y_i^* \gamma_i\Big) \\ &< 2 \cdot \text{LPval} \\ &\leq \min_{s\in\mathcal{S}} 2g_{\mathbf{p}}(s). \end{aligned}$$

□

A.1 Case $m = 2$

In this section, we provide the proofs for the setting where correlation sets are defined on subsets of size at most two. Recall from Section 4 that $\mathcal{K}$ denotes a cover of $\mathcal{C}$, and a dichotomy is defined as a specific pair of states used to isolate interaction effects.

Theorem 8 (Theorem 1) *Let $\mathcal{K}$ be a cover for $\mathcal{C}$. Then, for any state $s \in \mathcal{S}$ and any $i \in [k]$ with $s(i) = b$, we have:*

$$\mu_i^s = \mu_i^{s''} + \sum_{j=1}^{k} X_b^{i,j} \left[\mathbb{I}(s(j) = 1)\,\mathbb{I}(s''(j) = 0) - \mathbb{I}(s(j) = 0)\,\mathbb{I}(s''(j) = 1)\right], \tag{27}$$

where s'' is any state in $\mathcal{K}$ with $s''(i) = b$.

Proof. Consider a correlation set $\{i, j\}$. The expected loss incurred by vertex v_i or v_j due to this set in any given state depends solely on the configuration of v_i and v_j in that state. Hence there are four parameters in the $\boldsymbol{\theta}$ vector corresponding to the correlation set $\{i, j\}$ and we denote them using $\gamma_{i,j}^{a,b}$, where $a, b \in \{0, 1\}$. Let s, s' be an (i, j, b) pair. When we switch from s to s' the only difference in the expected losses for vertex i comes from the pair (i, j). Hence we have

$$\mu_i^{s'} - \mu_i^s = \gamma_{i,j}^{b,1} - \gamma_{i,j}^{b,0} := X_b^{i,j}.$$

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 (27), run the oracle for the optimization (4) 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 13: Online algorithm for $m = 2$ achieving $\tilde{O}(T^{2/3})$ pseudo-regret.

Hence, given the loss estimates for states in $\mathcal{K}$ we can estimate $X_b^{i,j}$ for each $i, j \in [k]$ and $b \in \{0, 1\}$. Next, given an arbitrary state s with $s(i) = b$ let $s'' \in \mathcal{K}$ such that $s''(i) = b$. We have

$$\begin{aligned}
\mu_i^s &= \mu_i^{s''} + \sum_{\substack{j:s(j)=0 \\ s''(j)=1}} (\gamma_{i,j}^{b,0} - \gamma_{i,j}^{b,1}) + \sum_{\substack{j:s(j)=1 \\ s''(j)=0}} (\gamma_{i,j}^{b,1} - \gamma_{i,j}^{b,0}) \\
&= \mu_i^{s''} + \sum_{\substack{j:s(j)=1, \\ s''(j)=0}} X_b^{i,j} - \sum_{\substack{j:s(j)=0, \\ s''(j)=1}} X_b^{i,j} \\
&= \mu_i^{s''} + \sum_{j=1}^{k} X_b^{i,j} \left[\mathbb{I}(s(j) = 1)\,\mathbb{I}(s''(j) = 0) - \mathbb{I}(s(j) = 0)\,\mathbb{I}(s''(j) = 1)\right].
\end{aligned}$$

This completes the proof. □

From the above theorem we have the following guarantee.

Theorem 9 (Theorem 2) *Consider an MDP*$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$ *with losses in* $[0, B]$*, a 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 13 (same as 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 the optimization oracle for Eq.* (4)*, the algorithm runs in time polynomial in* k *and* $n = |\mathcal{C}|$*.*

Proof. In each time step the maximum loss incurred by any criterion is bounded by $c + B$. Let $\{s_1, s_2, \ldots, s_r\}$ be the states in $\mathcal{K}$. During the exploration phase the algorithm stays in each state for N time steps and incurs a total loss bounded by $kNr(c + B)$. During the exploration phase the algorithm moves from one state to another in at most k steps and incurs a total additional loss of at most $rk^2(c + B)$. At any given state $s \in \mathcal{K}$ and vertex v_i, after N time steps we will, with probability at least

$1-\delta$, an estimate of μ_i^s up to an accuracy of $2B\sqrt{\frac{\log 1/\delta}{N}}$. Setting $\delta = 1/(rkT^4)$ and using union bound, we have that at the end of the exploration phase, with probability at least $1 - \frac{1}{T^4}$, the algorithm will have estimate $\hat{\mu}_i^s$ for all $s \in \mathcal{K}$ and $i \in [k]$ such that

$$\hat{\mu}_i^s - \mu_i^s \leq 4B\sqrt{\frac{\log rkT}{N}}. \tag{28}$$

Hence during the exploitation phase, with high probability, the algorithm will have the estimate for the expected loss of each state in $\mathcal{S}$, i.e., $\sum_i \mu_i^s$ up to an error of $4kB\sqrt{\frac{\log rkT}{N}}$. Combining the above we obtain that the total pseudo-regret of the algorithm is bounded as follows:

$$\begin{aligned}&\mathsf{Reg}(\mathcal{A})\\ &\leq kNr(c+B) + rk^2(c+B) + \Big(1 - \frac{1}{T^4}\Big)4kBT\sqrt{\frac{\log rkT}{N}} + \frac{1}{T^4}k(c+B)T.\end{aligned}$$

Note that the regret bound depends linearly on the magnitude of the losses $(c+B)$, rather than quadratically. This is characteristic of Explore-Then-Commit strategies, where the dominant term is often the cost incurred during the purely exploratory phase (roughly $N \cdot (c+B)$), in contrast to UCB-style algorithms where the bound is often derived from variance terms that may scale differently.

Setting $N = 10\frac{T^{2/3}(\log rkT)^{1/3}}{r^{2/3}}$ yields

$$\mathsf{Reg}(\mathcal{A}) \leq O(kr^{1/3}(c+B)(\log rkT)^{1/3}T^{2/3}),$$

which completes the proof. □

A.2 General case

The algorithm for the case of $m = 2$ naturally extends to arbitrary correlation set sizes. Overall the structure of the algorithm remains the same where we pick a cover of $\mathcal{C}$ and estimate the losses incurred in states that belong to the cover. Using the estimated losses we are able to approximately estimate the loss of any vertex at any other state. In order to do this we extend the definition of the cover as follows. Given correlation sets of arbitrary size in $\mathcal{C}$, a vertex v_i may participate in many of them. We say that vertices v_i and v_j share a correlation set, if they appear together in a set in $\mathcal{C}$. Consider the set of indices of all the vertices that v_i shares a correlation set with. We partition this set into disjoint subsets such that no two vertices in different subsets share a correlation set. For a given vertex v_i, we denote this collection of disjoint subsets by I_i. For example, if $\mathcal{C}$ contains sets $\{1,2\}$, $\{1,3\}$, and $\{1,4\}$, then, I_1 consists of the set $\{2,3,4\}$. On the other hand if $\mathcal{C}$ contains sets $\{1,2,3\}$, $\{1,3,4\}$, and $\{1,6,7\}$ then, I_1 consists of sets $\{2,3,4\}$ and $\{6,7\}$. For a given state s and $J \in I_i$ we denote by $s(J)$ the vector s restricted to indices in J. Notice that, in the worst case, I_i will consist of a single set of size at most $\min(k-1, nm)$. However, for more structured cases (e.g, $m = 2$) we expect I_i to consist of subsets of small sizes.

Given $i \in [k]$, $J \in I_i$, $b \in \{0,1\}$ and vectors u_1, u_2, we say that (i, b, J, u_1, u_2) is a dichotomy, if there exist two states $s, s' \in \mathcal{S}$ such that: (1) $s(J) = u_1, s'(J) = u_2$, (2) $s(i) = b = s'(i)$, and (3) s, s' agree in all other criteria. We call such a pair of states s, s' an (i, b, J, u_1, u_2) pair. We next extend the definition of a cover as follows. A subset $\mathcal{K} \subseteq S$ is called a cover of $\mathcal{C}$ if for any valid dichotomy (i, b, J, u_1, u_2), there exists an (i, b, J, u_1, u_2) pair $s, s' \in \mathcal{K}$. In general, we will always have a cover of size at most $n2^{mn}$. Similar to (**??**), for a valid dichotomy (i, b, J, u_1, u_2), we define $X_{b,J}^{i,u_1,u_2}$ as

$$X_{b,J}^{i,u_1,u_2} := \mu_i^s - \mu_i^{s'}, \tag{29}$$

where $s, s' \in \mathcal{K}$ is an (i, b, J, u_1, u_2) pair. Given the loss values in the states present in $\mathcal{K}$, we can estimate the loss of any other state using Theorem 10 stated below.

Theorem 10 *Let $\mathcal{K}$ be a cover for $\mathcal{C}$. Then, for any state $s \in \mathcal{S}$ and any $i \in [k]$ with $s(i) = b$, we have:*

$$\mu_i^s = \mu_i^{s''} + \sum_{J \in I_i} X_{b,J}^{i,s(J),s''(J)} \tag{30}$$

Here s'' is any state in $\mathcal{K}$ with $s''(i) = b$.

Proof. Let $s, s' \in \mathcal{K}$ be an (i, b, J, u_1, u_2) pair. When we move from state s to s', the only difference between the expected losses incurred by vertex v_i comes from the configuration of the vertices in J. Hence there at at most $2^{|J|+1}$ distinct parameters governing the expected loss incurred by vertex i in a given state s due to the configuration of the vertices in J. Denoting these parameters by $\gamma_{i,J}^{b,s(J)}$ we have

$$\mu_i^{s'} - \mu_i^s = \gamma_{i,J}^{b,s'(J)} - \gamma_{i,J}^{b,s(J)} := X_{b,J}^{i,s'(J),s(J)}.$$

Given the loss values for the states in the cover $\mathcal{K}$, we can estimate the quantities $X_{b,J}^{i,s(J),s''(J)}$.

Next, for an arbitrary state s such that $s(i) = b$, let $s'' \in \mathcal{K}$ be such that $s''(i) = b$. We have

$$\mu_i^s = \mu_i^{s''} + \sum_{J \in I_i} \gamma_{i,J}^{b,s(J)} - \gamma_{i,J}^{b,s''(J)} = \sum_{J \in I_i} X_{b,J}^{i,s(J),s''(J)}.$$

This completes the proof. □

For general correlation sets with each vertex participating in at most n sets, we use (30) instead of (27) to estimate losses in step 4 of the algorithm in Figure 13. The algorithm for general m is described in Figure 14 and has the following associated regret guarantee. The proof is identical to the proof of Theorem 2.

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 (30), run the oracle for the optimization (4) 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 14: Online algorithm for general m achieving $\tilde{O}(T^{2/3})$ pseudo-regret.

Theorem 11 *Consider an MDP*$(\mathcal{S}, \mathcal{A}, \mathcal{C}, \boldsymbol{\theta})$ *with losses bounded in* $[0, B]$ *and maximum cost of fixing a vertex being* c*. Given correlations sets* $\mathcal{C}$ *of size at most* m*, and a cover* $\mathcal{K}$ *of* $\mathcal{C}$ *of size* $r \leq n2^{mn}$*, the algorithm in Figure 14 achieves a pseudo-regret bounded by* $O(kr^{1/3}(c + B)(\log rkT)^{1/3}T^{2/3})$*. Furthermore, given access to the optimization oracle for Eq.* (4) *the algorithm runs in time polynomial in* k*,* $n = |\mathcal{C}|$ *and* $r = |\mathcal{K}|$*.*

B Further Discussion of COMPAS Example

Throughout the main sections, we have mentioned that the choice of the loss function is important in the effectiveness of our model. We briefly discussed this in Section 4. Below, we present a more detailed discussion of the effect of the loss function on our model, by using the COMPAS scenario from Section 1 as an example.

Loss function – COMPAS illustration. Consider the COMPAS example with a graph $\mathcal{G}$ with four criteria namely, false positive rate on population A, false positive rate on population B, AUC score for population A and AUC score for population B. We want to understand what kinds of loss functions will result in an overall suboptimal system when our model and algorithms from Section 4 are used. Suppose our algorithm take an action to fix a criterion and reach a state where the true positive rates and the AUC scores associated with the four criteria are: $[0.1, 0.8, 0.5, 0.5]$. Then a poor choice of the loss function would be $f_1 \cdot 0.1 + f_2 \cdot 0.8 + f_3 \cdot 0.5 + f_4 \cdot 0.5$, where f_i represents the fraction of complaints that trigger criterion i. Such a choice of the loss function will make our system vulnerable to the loudest voices in the system and as a result might not lead to a good solution at all. A more reasonable choice of the loss is $0.1 + 0.8 + 0.5 + 0.5$, that weighs the criteria equally and does not take into account the underlying size of the population. Another alternative is $\lambda_1|0.1 - 0.8| + \lambda_2(|0.5 - 0.5|)$, that aims at keeping both the discrepancy in the false positive rate and the AUC scores small. Finally, the choice we make in our experiments of penalizing each criterion for the deviation from the value over the entire population, i.e., $\max(0, |tpr_{overall} - tpr_w| - \tau)$, also leads to good solutions empirically.

Another case where additive losses are a poor choice is if the criteria in $\mathcal{G}$ is not chosen carefully. For instance, consider a scenario in the COMPAS example where all but one of the criteria correspond to the performance of the system on population A. An additive loss would then naturally force the system to disproportionately favor population A over a period of time.