Corporate Social Responsibility via Multi-Armed Bandits

Tom Ron[*]
Technion - Israel Institute of Technology
ront@campus.technion.ac.il

Omer Ben-Porat[*]
Technion - Israel Institute of Technology
omerbp@campus.technion.ac.il

Uri Shalit
Technion - Israel Institute of Technology
urishalit@technion.ac.il

Abstract

We propose a multi-armed bandit setting where each arm corresponds to a subpopulation, and pulling an arm is equivalent to granting an opportunity to this subpopulation. In this setting the decision-maker's fairness policy governs the number of opportunities each subpopulation should receive, which typically depends on the (unknown) reward from granting an opportunity to this subpopulation. The decision-maker can decide whether to provide these opportunities or pay a pre-defined monetary value for every withheld opportunity. The decision-maker's objective is to maximize her utility, which is the sum of rewards minus the cost of withheld opportunities. We provide a no-regret algorithm that maximizes the decision-maker's utility and complement our analysis with an almost-tight lower bound. Finally, we discuss the fairness policy and demonstrate its downstream implications on the utility and opportunities via simulations.

1 Introduction

Algorithmic decision making plays a fundamental role in many facets of our lives; criminal justice [10, 11, 28], banking [3, 17, 31, 38], online-advertisement [27, 29], hiring [1, 2, 4, 7] , and college admission [5, 25, 34] are just a few examples. With the abundance of applications in which algorithms operate, concerns about their ethics, fairness, and privacy have emerged. For instance, classification algorithms that were deemed to be unfair and discriminate based on factors like gender, race, and more [15, 19, 36, 39]. Algorithmic fairness is a framework that, among other means, is aimed at ensuring the long-term welfare of such subpopulations when subject to algorithmic decision making.

Consider the following online advertisement use-case. A company wants to publish a job ad online and optimizes its campaign based on the cost-per-click. As witnessed by Lambrecht and Tucker [23], women are less likely to see job ads for STEM positions since they have higher cost-per-click than men. If women are not exposed to information about STEM career opportunities, they may never apply to such jobs [14]. In order to act fairly and display ads to all the subpopulations the company will need to sacrifice part of its short-term utility and pay a higher cost-per-click. This is an example of a cost of fairness. Our goal in this paper is to better understand the trade-off that companies who wish to ensure their algorithms are more equitable face.

We focus on exploring the cost of fairness versus the cost of alternatives such as Corporate Social Responsibility [12] (CSR hereinafter). CSR is an approach towards the goal of long-term welfare which is becoming increasingly popular among tech-giants these days. CSR is a self-regulation act of

[*]Equal contribution

Preprint. Under review.

philanthropic responsibility in response to the rising concerns on ethical issues in businesses. For example, in 2019, Microsoft spent more than three billion dollars with minority, disabled, veteran, LGBTQ, and woman-owned businesses[2].

In this paper, we suggest an algorithmic approach to CSR in the setting of sequential decision making. Sequential decision making is often modeled as Multi-armed bandit problems (hereinafter MAB; see Auer et al. [8] for a brief introduction). MABs enjoy massive commercial success and have myriad real-world applications [13, 16, 35, 37]. It is therefore unsurprising that fair aspects of MAB are examined. In this work we treat arms as subpopulations, and require that subpopulations would not starve from lack of *opportunities*. Opportunities can be granted to a subpopulations explicitly, i.e., by pulling the subpopulation's arm, or implicitly via CSR channels. Given the example above, companies have the choice whether to display ads to subpopulations with higher cost-per-click or to invest money in organizations that promote the long term well-being of those subpopulations.

We highlight the tension between the decision-maker that wants to maximize her reward and the cost of CSR. We consider the bandit reward to be the benefit derived from granting the opportunity to the subpopulation represented by the arm. For simplicity we use the term expected reward from here on. The amount of opportunities depends on how fairness is perceived by the decision-maker and the expected rewards. Unfortunately, information about the expected rewards is not known in advance and has to be explored by the decision-maker. We take a utilitarian approach: The utility of the decision-maker is composed of the rewards, clicks on displayed ads, and a transfer cost. The transfer cost is the amount the decision-maker invests in CSR for every deferred opportunity. Knowing the transfer cost in advance, the decision-maker can make an informed decision on how to allocate its resources. Our model casts light on the trade-off between the cost of opportunity and the cost of transferring the opportunity requirement to an external source.

1.1 Our Contribution

Our contribution is two-fold: technical and conceptual. Technically, we consider the typical MAB setting with K Bernoulli arms with horizon T and expectation vector $\boldsymbol{\mu}$, which is unknown. In addition, we introduce a *fairness function* f, $f : [0,1]^K \to [0,1]^K$, which determines the minimal number of pulls for each arm given the expected reward vector $\boldsymbol{\mu}$. The term $T \cdot f(\boldsymbol{\mu})_i$ quantifies the amount of *opportunities* subpopulation i deserves, which is a function of its own expected reward and the expected rewards of the other subpopulations. The decision-maker gains rewards, but pays a transfer cost of λ for every round of unmet opportunity; namely, $\lambda \sum_{i=1}^{K} \max\{0, T \cdot f(\boldsymbol{\mu})_i - N_i\}$ where N_i is the number of pulls of arm i. We assume that both f and λ are known in advance. We characterize the optimal algorithm that achieves a sub-linear regret of $\tilde{O}(T^{2/3})$, and show a matching lower bound. We augment our theoretical analysis with experimental one, examining the implications of different fairness functions f and values of λ.

On the conceptual side, our framework reflects the trade-off between monetary rewards and subpopulation opportunities, which can be viewed as a means of providing long-term welfare. This perspective follows, e.g., self-regulation in revenue-driven commercial companies (as decision-makers) contributing to societal goals, or a policy maker that ensures that the decision-maker is fairness aware. In the former, sufficient opportunities are a CSR [18] that is integrated in the company's objective by design. In the latter, the decision-maker provides opportunities explicitly by arm pulls, or implicitly by payments that are invested in that subpopulation by the policy maker (for, e.g., better computer labs in public schools). Crucially, the number of required opportunities depends on the expected rewards, known only in hindsight.

1.2 Related Work

Multi-armed bandits have been the subject of many fairness-related research [20, 21, 24, 30]. Joseph et al. [20, 21] study fairness in MABs from the eyes of the decision-maker. In their work, a learning process is considered unfair if an arm with a lower expected reward is favored over an arm with a higher expected reward. We study fairness from the perspective of the arms and view arm pulling as granting an opportunity. This view was also adopted by Liu et al. [24]. The authors define a *calibrated fair policy* to be a policy that selects action i with probability equal to the probability that

[2]https://aka.ms/2019CSRReport

the reward realization of arm i is the highest. By measuring only the number of rounds the algorithm is miscalibrated, Liu et al. [24] neglect the decrease in reward incurred by the fairness requirement. In contrast to these works, we explicitly suggest a utilitarian approach that penalizes for lack of pulls (in our terminology, opportunities). A penalty of similar flavor was suggested recently in other work on fair ML [9, 22], but in different settings.

The work most related to ours is Patil et al. [30]. The authors define fairness as pulling each arm at least a minimal number of times according to a predefined vector (where each entry corresponds to a subpopulation). The predefined vector is given by the policy maker and is independent of the subpopulation properties. However, our work differs from Patil et al. [30] in two crucial aspects. First, while Patil et al. [30] model fairness as a hard constraint, we better address real-world applications and treat it as a soft one. Our utilitarian approach, which is well-studied in economic contexts [26, 33], accounts for trading rewards with opportunities. If providing opportunities explicitly by pulling the arms is financially unbearable, the decision-maker can do that implicitly by monetary transfers. Second, Patil et al. [30] construct the fairness constraint by a predefined vector, while in our work the opportunity requirements depend on the expected rewards, which is only known in hindsight. This uncertainty exacerbate the problem even further . These differences and others lead to a lower bound of $\tilde{O}(T^{2/3})$ compared to a $\tilde{O}(\sqrt{T})$ in theirs.

2 Model

We consider a stochastic bandit problem; a decision-maker is given K arms, and pulls one at each time step $t = 1, 2, \ldots, T$. We denote by i_t the arm pulled at time t. When arm i is pulled at time t, the decision-maker receives a random reward, $r_t \sim \mathcal{D}_i$. We assume that for every $i \in [K]$, the reward distribution $\mathcal{D}_i$ is a Bernoulli distribution with expected value μ_i. This is without loss of generality, since we can reduce any instance with general $[0, 1]$-supported distribution to an instance with Bernoulli arms using the technique of Agrawal and Goyal [6]. We use $\boldsymbol{\mu}$ to denote the vector of expected rewards, i.e., $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$. We denote by $N_{i,t}$ the number of times arm i is pulled by the end of round t, and let $\Delta_i = \mu^* - \mu_i$ be the gap between the expected reward of the optimal arm and the expected reward of arm i.

We now present the *Reward-Opportunity MAB* (R-O MAB) model. An instance of R-O MAB is represented by a tuple $\langle K, T, \boldsymbol{\mu}, f, \lambda \rangle$. The tuple $\langle K, T, \boldsymbol{\mu} \rangle$ is an instance of standard stochastic bandit as described above. The combination of f and λ creates what we call the "fairness policy".

The fairness requirements are expressed by a function f, $f : [0, 1]^K \to [0, 1]^K$. f receives as input a vector of expected rewards and outputs a vector of minimal fraction of times each arm has to be pulled in order not to be penalized. We let $f(\boldsymbol{\mu})_i$ denote the i'th entry of $f(\boldsymbol{\mu})$. We assume that $\sum_{i=1}^{K} f(\boldsymbol{\mu})_i \leq 1$ and that f is Lipschitz continuous with a Lipschitz constant L with respect to the l_1 norm. That is, for all $\boldsymbol{\mu}, \boldsymbol{\mu}' \in [0, 1]^k$ it holds that $\|f(\boldsymbol{\mu}) - f(\boldsymbol{\mu}')\|_1 \leq L \|\boldsymbol{\mu} - \boldsymbol{\mu}'\|_1$. Intuitively, satisfying the Lipschitz condition implies that two similar expected reward vectors get similar fairness requirements. For simplicity, from here on we call f the *fairness function*.

The difference between the fairness requirement and the number of times an arm was pulled, $Tf(\boldsymbol{\mu})_i - N_{i,T}$, represents the deviation from the fairness constraint. If the deviation is positive, it means the arm was not pulled enough times, i.e. the subpopulation did not receive enough opportunities according to the fairness function. In such a case, the decision-maker pays a cost. The paid cost for a single arm pull's deviation from the fairness requirement is given by λ_i, the *transfer cost* for arm i. If arm i was pulled less than $Tf(\boldsymbol{\mu})_i$ times, the reward will be deducted by $\lambda_i(Tf(\boldsymbol{\mu})_i - N_{i,T})$. The transfer cost is known to the decision-maker in advance. To account for all cases, the possible cost which stems from the deviation is $\lambda_i \max\{Tf(\boldsymbol{\mu})_i - N_{i,T}, 0\}$. For simplicity, we use $\lambda_i = \lambda$ for all $i \in [K]$, but stress that our results hold with minor modifications in the general case as well.

The utility of the decision-maker is denoted by $\mathcal{U}_{\lambda,f}$. It is an additive utility of the reward minus the total deviation from the fairness requirement. Notice that i_t and consequently $N_{i,T}$ depend on the algorithm playing the arms. Formally, given an algorithm ALG,

$$\mathcal{U}_{\lambda,f}(ALG; T) \overset{\text{def}}{=} \sum_{t=1}^{T} r_{i_t} - \lambda \sum_{i=1}^{k} \max\{Tf(\boldsymbol{\mu})_i - N_{i,T}, 0\}. \tag{1}$$

As is customary in the MAB literature, we focus on the *regret* of the decision-maker, which we denote $\mathcal{R}_{\lambda,f}(ALG;T)$. Let OPT be an algorithm maximizing the utility $\mathcal{U}_{\lambda,f}(OPT)$ (we discuss OPT in Subsection 2.1). The regret is the gap between the expected utility of OPT and ALG:

$$\mathcal{R}_{\lambda,f}(ALG;T) = \mathbb{E}(\mathcal{U}_{\lambda,f}(OPT;T)) - \mathbb{E}(\mathcal{U}_{\lambda,f}(ALG;T)). \tag{2}$$

When λ and f are arbitrary or clear from the context, we omit the subscript and simply denote $\mathcal{U}$ and $\mathcal{R}$. Full proofs appear in Section A.

2.1 Optimal Algorithm

The structure of the optimal algorithm in classic MABs is straightforward: In every round, pick the arm with the highest expectation. However, in our case, the transfer cost makes the optimal algorithm a bit more complex, as we now elucidate. Let i denote an arbitrary index of a sub-optimal arm, i.e., an arm such that $\mu_i < \max_{i' \in [K]} \mu_{i'}$. The decision-maker has to decide whether to support the subpopulation associated with that arm explicitly (by pulling it $Tf(\boldsymbol{\mu})_i$ times) or implicitly (by paying $\lambda T f(\boldsymbol{\mu})_i$). Note that $Tf(\boldsymbol{\mu})_i$ can be non-integer, in this case, we assume $Tf(\boldsymbol{\mu})_i$ is rounded to the previous integer. In each one of those $Tf(\boldsymbol{\mu})_i$ rounds, the decision-maker loses Δ_i if she pulls arm i (as she could pick the optimal arm) but saves λ (as she does not need the pay the transfer cost). Therefore, if the reward gap of arm i is greater than the transfer cost, $\Delta_i > \lambda$, the decision-maker does not pull arm i at all and pays the transfer cost. Otherwise, if $\Delta_i < \lambda$, the decision-maker would have greater utility by pulling arm i exactly $Tf(\boldsymbol{\mu})_i$ times and not incurring the transfer cost. If $\Delta_i = \lambda$, the decision-maker is indifferent between the two options. More formally,

Lemma 1. *Fix an arbitrary instance $\langle K, T, \boldsymbol{\mu}, f, \lambda \rangle$ and let OPT be an optimal algorithm for that instance. For every sub-optimal arm i, if $\Delta_i < \lambda$ then OPT pulls i exactly $Tf(\boldsymbol{\mu})_i$ times; if $\Delta_i > \lambda$, OPT does not pull i at all. If $\Delta_i = \lambda$, OPT pulls arm i between zero and $Tf(\boldsymbol{\mu})_i$ times.*

Proof sketch. We argue that every algorithm that is different from the algorithm described above, i.e., disagrees with it for at least two arms with different expected rewards, has a lower expected utility. In such algorithms, at least one arm is pulled too much compared with the described algorithm and at least one arm is not pulled sufficient number of times compared with the described algorithm. Moving pulls from one arm to another, ensures that at least one of those arms now agrees with the algorithm described above and yields a higher expected utility. By repeating this process at most K times we reach the described algorithm and higher expected utility. □

Notice that the utility (Equation 1) is order insensitive; hence, OPT is not unique. The regret analysis is only concerned with the utility of an optimal algorithm so our characterization suffices.

2.2 About the Fairness Policy

The fairness policy is comprised of the fairness function f and the transfer cost λ. The function f represents the decision-maker's view on how opportunities should be distributed. For example, the zero function $f^0(\boldsymbol{\mu})_i \overset{\text{def}}{=} 0$ corresponds to standard Multi-Armed bandit problem without any constraints. Generalizing this case for any constant function, e.g., $f^{\text{uni}}(\boldsymbol{\mu})_i \overset{\text{def}}{=} \frac{1}{K}$, alludes that the decision-maker believes that all subpopulations are entitled to the same share of opportunities irrespective of their expected rewards. The fairness function can also grow linearly with each expected reward, for instance $f^{\text{lin}}(\boldsymbol{\mu})_i \overset{\text{def}}{=} \frac{\mu_i}{K}$. In the most general case, the number of required opportunities to a subpopulation can also depend on its expected reward relative to the expected rewards of other subpopulations. For example, $f^{\text{sft}}(\boldsymbol{\mu};c)_i \overset{\text{def}}{=} \frac{\exp^{c\mu_i}}{\sum_{j=1}^{K} \exp^{c\mu_j}}$. Our modelling and results support these special cases and many other natural candidates for the fairness function. Selecting λ complements the decision-maker's view on revenue and opportunities. As described in Section 2.1, if the transfer cost is high the decision-maker will tend to grant the opportunities explicitly, and would not grant opportunities explicitly only when the subpopulations' expected rewards have big differences. If the transfer cost is low the decision-maker would derive a higher utility by supporting subpopulations via CSR and not by directly granting opportunities. As pointed out earlier, λ can vary between different subpopulations but for simplicity is assumed equal.

Algorithm 1: Fairness-Aware-ETC

```
Input: $N$ - number of exploration rounds
1 for $i = 1, \dots K$ do
2     pull arm $i$ for $N$ rounds
3 for $i = 1, \dots K$ do
4     if $\hat{\Delta}_i < \lambda$ then
5         pull arm $i$ for $\max\{Tf(\hat{\boldsymbol{\mu}})_i - N, 0\}$ rounds
6 pull an arbitrary arm from $\arg\max_{i \in [K]} \hat{\mu}_i$ until the execution ends
```

3 No-Regret Algorithms

In this section, we present our main algorithmic contribution. We devise *Self-regulated Utility Maximization*, which incurs a regret of $\tilde{O}(T^{2/3})$. Before we discuss it, we first demonstrate that classical MAB algorithms fail miserably on our setting. This is expected given that such algorithms were not devised for a setting like ours, but it will serve us later on. Classical MAB algorithms are tuned to pull sub-optimal arms as little as possible. As shown in Subsection 2.1, it is not always optimal for R-O MAB. If the cost of opportunity (Δ_i) is lower than the transfer cost (λ), the optimal algorithm pulls arm i according to the fairness function.

To better illustrate, consider the famous Explore-Then-Commit (ETC) algorithm. ETC explores all arms for a predetermined number of rounds (N), and then follows the best preforming arm for the remaining rounds. We focus on a R-O MAB instance with 2 arms, $\boldsymbol{\mu} = (1, \frac{1}{2})$, transfer cost $\lambda = 0.6$ and constant fairness function $f(\boldsymbol{\mu})_i = \frac{1}{2}$. The optimal algorithm from Subsection 2.1 pulls each arm $\frac{T}{2}$ times and obtains an expected utility of $0.75T$. ETC with optimized exploration parameter will discover that arm 1 is the better one relatively fast, and will pick that arm forever; hence, its utility is $T - \lambda\frac{T}{2} - o(1) \approx 0.7T$. The regret is therefore $0.05T$, which is linear in the number of rounds T.

In R-O MAB, we face a unique challenge comparing to the classic MAB problem. Classical MAB algorithms are aimed at identifying the optimal arm but do not estimate the expected rewards $\boldsymbol{\mu}$. As discussed in Section 2.1, the optimal algorithm depends on the relation between the reward gaps and the transfer cost; hence, unlike classic MAB, accurate approximation of the reward gaps $(\Delta_i)_{i\in[K]}$ is crucial for our problem. Additionally, $f(\boldsymbol{\mu})$ should also be approximated correctly for arms i with $\Delta_i < \lambda$, to align with the optimal algorithm. These two challenges are singular to our settings and are reflected in the lower bound.

Algorithm 1, which we term Fairness-Aware-ETC, is a modified version of ETC, which is aware of the fairness function and the transfer cost. Fairness-Aware-ETC pulls each arm N times, where N is received as an input. After the KN exploration rounds, it constructs estimates for $\boldsymbol{\mu}$ and $f(\boldsymbol{\mu})$, which we denote using the hat notation, i.e., $\hat{\boldsymbol{\mu}}$ and $f(\hat{\boldsymbol{\mu}})$. It then continues optimally with respect to these estimates (similarly to the optimal algorithm for the estimated quantities). The number of exploration rounds per arm N balances the tension between exploration and exploitation. Setting N very high ensures that the estimates $\hat{\boldsymbol{\mu}}$ and $f(\hat{\boldsymbol{\mu}})$ are close to their actual counterparts with high probability, but can allow little exploitation and hence high regret. Picking N too low can result in wrong estimation of $\boldsymbol{\mu}$ and as a consequence also wrongly evaluate f.

Theorem 1. *Fix any arbitrary instance of R-O MAB, and let $N = 8L^{2/3}T^{2/3}\log^{1/3} T$. Algorithm 1 has a regret of $O(KL^{2/3}T^{2/3}\log^{1/3} T)$.*

Proof sketch. Given the specified value for N, we show that with high probability the expected rewards of all the arms are approximated correctly within a confidence interval of size $2\sqrt{\frac{2\log T}{N}}$. We address this case as the *clean event*. Assuming the clean event occurs, the regret for each arm is bounded by N and the approximation error of f. After pulling each arm N times, the approximation error of f is bounded by $T\left|f(\hat{\boldsymbol{\mu}}) - f(\boldsymbol{\mu})\right| \leq TL\left|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\right| \leq KL^{2/3}T^{2/3}\log^{1/3} T$. Summing the regret on all the arms results a regret bounded by $O(KL^{2/3}T^{2/3}\log^{1/3} T)$. □

Notice that Algorithm 1 is almost data independent. The exploration phase in Lines 1-2 continues even if the estimates of $\boldsymbol{\mu}$ and $f(\boldsymbol{\mu})$ are accurate. The predefined exploration length N prevents the algorithm from stopping the exploration early. Such an early stopping is important after identifying arms with high opportunity cost or arms that already satisfy the fairness requirements.

3.1 Fairness Aware Black-Box algorithm

In this section, we present a data dependent algorithm addressing the problems of Algorithm 1 above. If the algorithm is certain (w.h.p.) that the cost of opportunity is higher than the transfer cost, the decision-maker can stop pulling this arm. If the fairness function admits low values, the algorithm can satisfy the fairness requirement within less than $\tilde{O}\left(T^{2/3}\right)$ pulls.

We now explain the course of Algorithm 2. Full version of the algorithm appears in Section C. The algorithm takes α and β, which we describe shortly, and ALG, a black-box no-regret MAB algorithm as input, where ALG is no-regret with respect to the classical, rewards-only MAB objective (e.g. UCB1 [32]).

In Lines 1-3 the main variables are initialized: confidence bounds representing the probable estimates of the reward gaps, i.e., $LCB(\Delta_i)$, $UCB(\Delta_i)$, and C_t (Line 3) which is the hyper-cube of probable estimates of $\boldsymbol{\mu}$. Lines 4-10 consist of four different phases. In the first phase (Lines 4-5), the reward gaps are approximated up to a factor of β. After this phase, the algorithm knows w.h.p. for each arm whether its reward gap is higher or lower than the transfer cost by more than β. To be precise, we care for accurate approximation of Δ_i, only if it is close to the transfer cost, λ.

The second phase (Lines 6-7), approximates f for arms with low opportunity cost up to a factor of α. If there is an arm with low opportunity cost for which the approximation of f is not accurate enough, all the arms are pulled. The term $\max_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_i - \min_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_i$ upper bounds the highest change of $f(\boldsymbol{\mu}')_i$ inside the hyper-cube C_t, and hence also upper bounds our estimation error of $f(\hat{\boldsymbol{\mu}})_i$. To clarify why we pull all arms in Line 7, recall that $f(\boldsymbol{\mu})_i$ also depends on μ_j, $j \neq i$, in the general case; if our estimate $\hat{\mu}_j$ is not accurate, the estimation of $f(\boldsymbol{\mu})_i$ might not be accurate as well. Pulling all arms ensures that all the estimates improve for the subsequent round, namely, C_t shrinks in all of its dimensions. In the third phase (Lines 8-9), we ensure that we pull all arms with low opportunity cost according to the estimate of $f(\hat{\boldsymbol{\mu}})_i$. Lastly, in the fourth step (Line 10), we invoke ALG until the end of the execution.

Next, we discuss the input hyper-parameters, α, β and ALG. α is the confidence interval hyper-parameter for the approximation of f. Setting α too small values implies that arms should be pulled many times and this can inflict a regret due to over pulling arms. The approximation error of f can be as big as $T\alpha$ is. The hyper-parameter β is the confidence interval for the approximation of the reward gaps. If the reward gap is not close to the transfer cost λ, it would be identified almost immediately. Otherwise, Algorithm 2 uses the black-box MAB algorithm ALG. This allows the decision-maker to devote the fourth and final phase to identifying the best arm and exploiting its reward.

The only computationally non-trivial step in Algorithm 2 appears in Line 6: Computing $\max_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_i - \min_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_i$. Finding the global maximum of a Lipschitz function inside a hyper-cube is a computationally challenging task. However, due to role f plays in our setting, we argue that it should have a natural structure. Indeed, f quantifies a societal requirement and as such should be easy to grasp: Providing opportunities according to a cumbersome, hard-to-optimize and unexplainable criteria is likely to be unfair in and of itself. Consequentially, we shall assume that there is an oracle that computes the minimal and maximal values f at entry i can obtain in a given hyper-cube. In Subsection A.5, we show that for the softmax function f^{sft}, implementing such an oracle boils down to solving a convex optimization problem over a single variable. As an additional example, the family of monotonically non-decreasing Cartesian product functions obtain minimum and maximum on the extreme points of the confidence intervals and hence are easy to handle. Examples of such functions are linear functions, exponential functions (e.g., $f(\boldsymbol{\mu})_i = \mu_i^c / K$ for $c \geq 1$) and constant functions. We are ready to state the guarantees of Algorithm 2.

Theorem 2. *Fix any arbitrary instance of R-O MAB, and let* $\alpha = K^{2/3} L^{2/3} T^{-1/3} \log^{1/3} T$, $\beta = T^{-1/3} \log^{1/3} T$. *Then, Algorithm 2 has a regret of* $O(K^{5/3} L^{2/3} T^{2/3} \log^{1/3} T)$.

Proof sketch. We show that the loop in Line 5 pulls each arm at most $O(T^{2/3} \log^{1/3} T)$ and the loop in Line 7 pull each arm at most $O(K^{2/3} L^{2/3} T^{2/3} \log^{1/3} T)$. Combined, these first two phases can lead to a regret of up to $O(K^{5/3} L^{2/3} T^{2/3} \log^{1/3} T)$. In the next phases of the algorithm, we enjoy the fact that

Algorithm 2: Self-regulated Utility Maximization

```
Input: Black-box bandit algorithm $ALG$, allowed approximation error parameters $\alpha$ and $\beta$
1  $t = 1$
2  Initialize arms' data - $N_i = 0, LCB(\Delta_i) = 0, UCB(\Delta_i) = 1$ for all $i \in [K]$
3  $C_1 = [0,1]^K$ // Hyper-cube of $\boldsymbol{\mu}$ values' in the clean event
4  while $\exists i \in [K]$ s.t $UCB(\Delta_i) > \lambda + \beta$ and $LCB(\Delta_i) < \lambda - \beta$ do // Phase 1
5      Pull all arms once, update $t$, counters, confidence bounds and $C_t$
6  while $\exists i \in [K]$ s.t. $\max_{\boldsymbol{\mu'} \in C_t} f(\boldsymbol{\mu'})_i - \min_{\boldsymbol{\mu'} \in C_t} f(\boldsymbol{\mu'})_i > \alpha$ and $LCB(\Delta_i) < \lambda$ do
    // Phase 2
7      Pull all arms once, update $t$, counters, confidence bounds and $C_t$
8  while $\exists i \in [K]$ s.t. $LCB(\Delta_i) < \lambda$ and $N_i < Tf(\hat{\boldsymbol{\mu}})_i$ and $t < T$ do // Phase 3
9      Pull arm $i$ the minimal number of times so $N_i \geq Tf(\hat{\boldsymbol{\mu}})_i$, update $t$ and counters.
10 Invoke $ALG$ for the remaining rounds // Phase 4
```

f is approximated correctly (w.h.p.) and pull arms with low opportunity cost solely for the minimal number of times. Consequently, the regret is bounded by the approximation error of f, which is $O(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T)$. In the remaining rounds, we execute ALG. The additional regret can only be up to the regret the ALG exhibits, which we assume to be bounded by $O(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T)$. That is well-justified since it is well known that algorithms for the classical setting can achieve a regret of $\tilde{O}(\sqrt{KT})$. Altogether, we obtain $O(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T)$. □

Under slight modifications our algorithms perform much better on some instances than others. We elaborate on this in Section B.

4 Lower Bound

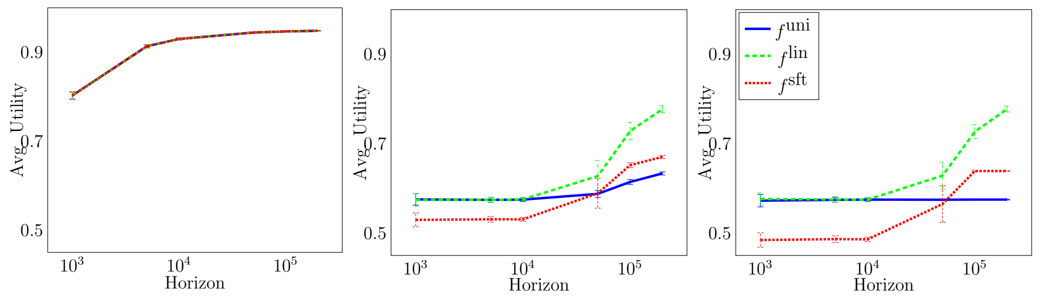
In the previous section, we presented Algorithm 2, which incurs a regret of $\tilde{O}\left(T^{2/3}\right)$ in the worst case. Here we show that this bound is asymptotically optimal by designing a family of R-O MAB instances that can mislead any algorithm.

Theorem 3. *Fix time horizon T, number of arms K, and Lipschitz constant L. For any algorithm, there exists a R-O MAB instance such that $\mathcal{R}(T) \geq \Omega(T^{2/3})$.*

Proof sketch. We construct a family of R-O MAB instances $\mathcal{F}$ that are almost identical, and show that any algorithm will have a high regret on at least one instance. The transfer cost is the same for all the instances and is 1. This means that the optimal algorithm pulls all the sub-optimal arms according to the fairness function. The instances differ in the expected rewards. Each instance consists of one optimal arm with expected reward of 1, a set of sub-optimal arms with expected reward of $\frac{1}{2}$ and a set of sub-optimal arms with expected reward of $\frac{1+\epsilon}{2}$ where $\epsilon = O(T^{-1/3})$. The fairness function f is piece-wise linear such that $f(\boldsymbol{\mu})_i = 0$ for $\mu_i \leq 0.5$ and $f(\boldsymbol{\mu})_i = \min\{\frac{L(\mu_i - 0.5)}{K}, \frac{L\epsilon}{K}\}$ for $0.5 < \mu_i \leq 1$. In order to distinguish between the instances in $\mathcal{F}$, $O(KT^{2/3})$ rounds are required. If the algorithm explores all the arms a number of times that is sufficient to distinguish between the instances in $\mathcal{F}$, there exists an instance in $\mathcal{F}$ such that at least one arm has an expected reward of $\frac{1}{2}$. The optimal algorithm does not pull this arm at all. Thus, the exploration yields a regret of $\Omega(T^{2/3})$. Otherwise, there is at least one arm that the algorithm pulls less than $O(1/\epsilon^2)$ times. There exists an instance in $\mathcal{F}$ that according to the construction of f, this arm should have been pulled $\frac{T^{2/3}L}{K}$ times but was pulled less than $8\log T$. Hence the regret is $\Omega\left(T^{2/3}\right)$. □

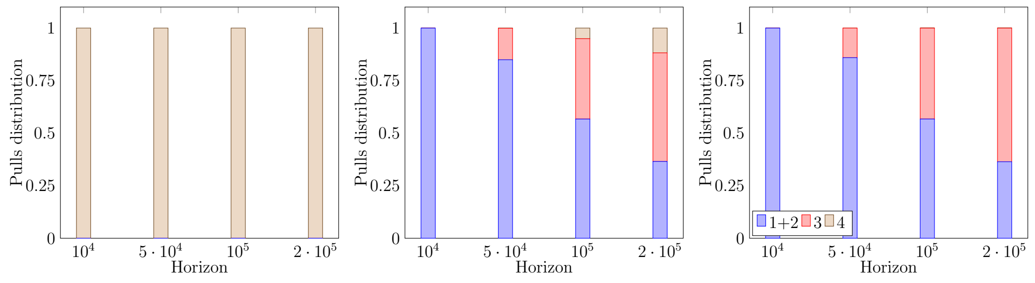
5 Experiments

In this section, we perform an empirical sensitivity analysis of Algorithm 2. We demonstrate how different fairness policies, i.e., pairs of a fairness function f and a transfer cost λ influence the course of the algorithm and the average utility. We consider several R-O MAB instances with $K = 6$ arms, expected values $\mu_i = 0.2 + (i-1) \cdot 0.15$ for every $i \in [K]$, and a varying horizon T. For fairness



(a) $\lambda = 0$ (b) $\lambda = 0.4$ (c) $\lambda = 0.8$

Figure 1: Average utility for different transfer costs.



(a) $\lambda = 0$ (b) $\lambda = 0.4$ (c) $\lambda = 0.8$

Figure 2: Round distributions per phase for f^{sft}.

functions, we consider f^{uni}, f^{lin}, and f^{sft} from Subsection 2.2. As for the transfer cost λ, we used 0, 0.4, and 0.8. We used UCB1 as the black box algorithm ALG in Algorithm 2. Each combination of fairness function, transfer cost and horizon was executed 200 times on a standard Mac computer. The entire process took several hours. In order to compare between the different horizons we present the *average* utility, i.e., the cumulative utility divided by the number of rounds.

Figure 1 demonstrates the behaviour of different fairness functions with respect to varying values of λ. As expected, the average utility decreases as the transfer cost increases. In Figure 1a, we see that the average utility of all the functions is the same, since ALG is played across all the rounds. As λ increases, the reward gaps Δ of more arms go below the transfer cost (see Line 5 in Algorithm 2), so the decision-maker should pull them $Tf(\boldsymbol{\mu})_i$ times (similar to optimal algorithm from Subsection 2.1). The expected utility increases with the horizon, since the proportion of approximation rounds (Lines 5 and 7) decreases with the horizon. For $\lambda = 0.8$, f^{sft} and f^{uni} use all the rounds for allocating opportunities. The difference in the average utility stems from the allocation differences. f^{uni} allocates evenly, while f^{sft} allocates more opportunities as the arm expected reward grows. Observe that $f^{\text{uni}}(\boldsymbol{\mu})_i > f^{\text{lin}}(\boldsymbol{\mu})_i$ for every $i \in [K]$ and the described $\boldsymbol{\mu}$; thus, the utility of f^{lin} is greater than f^{uni} for $\lambda > 0$. Figure 2 investigates the crux of the execution of Algorithm 2 for f^{sft}: The proportion of rounds devoted to phase 1 (Line 5), phase 2 (Line 7), phase 3, (Line 9), and phase 4 (invoking ALG, Line 10). For $\lambda = 0$, Algorithm 2 always invokes the black box algorithm ALG. For $\lambda = 0.4$ and $\lambda = 0.8$, the proportion of approximation rounds (phases 1 and 2) decreases as the horizon increases. The proportion of phase 3 rounds increases as the horizon grows. The reason is that the proportion of approximation rounds decreases, while the required number of opportunities grows linearly with the horizon. For $\lambda = 0.8$, the algorithm dedicates all rounds for opportunities. However, for $\lambda = 0.4$, it realizes that for arms 1,2, and 3 (with $\Delta_i > \lambda = 0.4$) paying the transfer cost λ per each withheld opportunities, and hence the remaining rounds are devoted to phase 4.

6 Discussion

We introduced a MAB problem that models decision making from the perspective of Corporate Social Responsibility and allocation of opportunities. Our modeling imitates many real-world scenarios where decision-makers are required to maximize their short-term utility while at the same time upholding fairness principles. With our framework, commercial companies can incorporate self-regulation in their algorithmic products, and provide opportunities as a form of social responsibility. We devised a no-regret algorithm and showed that its convergence rate is in fact optimal. We see considerable scope for follow-up work: Self-regulation for increasing subpopulation welfare can be incorporated in many other tasks, e.g., in classification or load balancing. Granting opportunities and investing in CSR can change subpopulations' expected rewards over the long-term. We are interested in the dynamic between the change in subpopulations' expected rewards and the fairness policy.

Broader Impact

The main beneficiaries of incorporating CSR considerations into algorithmic products will be weakened subpopulations. Opportunities are often denied from those subpopulations without suitable compensation or long-term vision; this is what we try to remedy in this paper. Society as a whole will benefit from securing the long-term well-being of weakened subpopulations and from closing the gaps.

If adopted as self-regulation, as we point out in Section 2.1, decision-makers can replace pulls (opportunities) with paying a transfer cost. Decision-makers can abuse the fairness policy by picking the transfer cost to be very low or choosing a very loose fairness function. That is, the number of opportunities derived by the fairness function will be very low while the transfer cost would not reflect the real cost of creating an opportunity. We acknowledge that our method is mostly relevant in cases where a company has commitment to a vision of fairness. We envision that a joint board or an independent organization (similar to Institutional Review Boards) will oversee the fairness policy selection in such firms to make sure it addresses not only the firm's revenue but also weakened subpopulations.

References

[1] Amazon scraps secret AI recruiting tool that showed bias against women - reuters. URL `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G`.

[2] How to hire with algorithms. URL `https://hbr.org/2016/10/how-to-hire-with-algorithms`.

[3] Using mobile to reach the Latin american unbanked | fico. URL `https://www.fico.com/en/node/8140?file=7900`.

[4] F. Abel. We know where you should work next summer: Job recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 230–230, 2015.

[5] A. Acharya and D. Sinha. Early prediction of students performance using machine learning techniques. *International Journal of Computer Applications*, 107(1), 2014.

[6] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1, 2012.

[7] I. Ajunwa and D. Greene. Platforms at work: Automated hiring platforms and other new intermediaries in the organization of work. *SP Vallas, and A. Kovalainen, Work and Labor in the Digital Age*, pages 61–91, 2019.

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

[9] Y. Bechavod and K. Ligett. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*, 2017.

[10] R. Berk. *Criminal justice forecasts of risk: A machine learning approach*. Springer Science & Business Media, 2012.

[11] R. Berk, R. Berk, and Drougas. *Machine learning risk assessments in criminal justice settings*. Springer, 2019.

[12] A. B. Carroll et al. The pyramid of corporate social responsibility: Toward the moral management of organizational stakeholders. *Business horizons*, 34(4):39–48, 1991.

[13] S.-C. Chow and M. Chang. Adaptive design methods in clinical trials–a review. *Orphanet journal of rare diseases*, 3(1):11, 2008.

[14] A. B. Diekman, E. R. Brown, A. M. Johnston, and E. K. Clark. Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers. *Psychological science*, 21(8):1051–1057, 2010.

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

[16] M. C. Fu. Alphago and monte carlo tree search: the simulation optimization perspective. In *Proceedings of the 2016 Winter Simulation Conference*, pages 659–670. IEEE Press, 2016.

[17] A. Fuster, P. Goldsmith-Pinkham, T. Ramadorai, and A. Walther. Predictably unequal? the effects of machine learning on credit markets. *The Effects of Machine Learning on Credit Markets (November 6, 2018)*, 2018.

[18] E. Garriga and D. Melé. Corporate social responsibility theories: Mapping the territory. *Journal of business ethics*, 53(1-2):51–71, 2004.

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

[20] M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth. Fair algorithms for infinite and contextual bandits. *arXiv preprint arXiv:1610.09559*, 2016.

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

[22] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

[23] A. Lambrecht and C. E. Tucker. Algorithmic bias? an empirical study into apparent gender-based discrimination in the display of stem career ads. *An Empirical Study into Apparent Gender-Based Discrimination in the Display of STEM Career Ads (March 9, 2018)*, 2018.

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

[25] T. Lux, R. Pittman, M. Shende, and A. Shende. Applications of supervised learning techniques on undergraduate admissions data. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 412–417, 2016.

[26] A. Mas-Colell, M. D. Whinston, J. R. Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.

[27] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[28] Northpointe. Practitioner's guide to compas core, 2015. URL `https://assets.documentcloud.org/documents/2840784/Practitioner-s-Guide-to-COMPAS-Core.pdf`.

[29] R. Oentaryo, E.-P. Lim, M. Finegold, D. Lo, F. Zhu, C. Phua, E.-Y. Cheu, G.-E. Yap, K. Sim, M. N. Nguyen, et al. Detecting click fraud in online advertising: a data mining approach. *The Journal of Machine Learning Research*, 15(1):99–140, 2014.

[30] V. Patil, G. Ghalme, V. Nair, and Y. Narahari. Achieving fairness in the stochastic multi-armed bandit problem. *arXiv preprint arXiv:1907.10516*, 2019.

[31] A. Pérez-Martín, A. Pérez-Torregrosa, and M. Vaca. Big data techniques to measure credit banking risk in home equity loans. *Journal of Business Research*, 89:448–454, 2018.

[32] A. Slivkins. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12 (1-2):1–286, 2019. ISSN 1935-8237. doi: 10.1561/2200000068. URL `http://dx.doi.org/10.1561/2200000068`.

[33] H. R. Varian and H. R. Varian. *Microeconomic analysis*, volume 3. Norton New York, 1992.

[34] A. Waters and R. Miikkulainen. Grade: Machine learning support for graduate admissions. *AI Magazine*, 35(1):64–64, 2014.

[35] J. White. *Bandit algorithms for website optimization*. " O'Reilly Media, Inc.", 2012.

[36] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

[37] C. Zeng, Q. Wang, S. Mokhtari, and T. Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 2025–2034, 2016.

[38] S. Zhang, W. Xiong, W. Ni, and X. Li. Value of big data to finance: observations on an internet credit service company in china. *Financial Innovation*, 1(1):17, 2015.

[39] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*, 2017.

A Omitted Proofs

A.1 Optimal Algorithm

Lemma 1. *Fix an arbitrary instance $\langle K, T, \boldsymbol{\mu}, f, \lambda \rangle$ and let OPT be an optimal algorithm for that instance. For every sub-optimal arm i, if $\Delta_i < \lambda$ then OPT pulls i exactly $Tf(\boldsymbol{\mu})_i$ times; if $\Delta_i > \lambda$, OPT does not pull i at all. If $\Delta_i = \lambda$, OPT pulls arm i between zero and $Tf(\boldsymbol{\mu})_i$ times.*

Proof. Note that the utility of R-O MAB instance depends on the number of times each are was pulled rather than the order. Hence, we only care about the vector $\boldsymbol{N} = (N_1, \dots, N_K)$. We prove that any vector $\boldsymbol{N}$ that violates the structure described above is sub-optimal.

Denote by $\boldsymbol{N^*} = (N_1^*, \dots, N_K^*)$ a vector of counts where for every sub-optimal arm i, if $\Delta_i < \lambda$, $N_i^* = T(\boldsymbol{\mu})_i$ and if $\Delta_i > \lambda$, $N_i^* = 0$. If $\Delta_i = \lambda$, $0 \le N_i^* \le Tf(\boldsymbol{\mu})_i$. Note that $\boldsymbol{N^*}$ is not unique. Let $\boldsymbol{N} = (N_1, \dots, N_K)$ be a vector that violates the structure of $\boldsymbol{N^*}$.

Since $\boldsymbol{N}$ violates the structure of $\boldsymbol{N^*}$, there exists $i, j \in [K], i \ne j$ such that $N_i < N_i^*$ and $N_j > N_j^*$. We prove that by moving pulls from j to i in a way specified below the expected utility increases. Let $\boldsymbol{N'} = (N_1', \dots, N_K')$ be the modified vector after moving pulls from j to i. That is, for all $l \in [K], l \ne i, j$, $N_l = N_l'$. Additionally, we show that either $N_i' = N_i^*$ or $N_j' = N_j^*$.

If $0 \le N_i < N_i^*$ it implies that $\Delta_i \le \lambda$. Otherwise $N_i^* = 0$. If $\Delta_j < \lambda$ it implies that $N_j > Tf(\boldsymbol{\mu})_j$. In this case moving $m = \min\{N_j - Tf(\boldsymbol{\mu})_j, Tf(\boldsymbol{\mu})_i - N_i\}$ pulls from j to i changes the expected utility by $m(\lambda + \mu_i - \mu_j) = m(\lambda - \Delta_i + \Delta_j)$. Since $\Delta_i < \lambda$ and $\Delta_j \ge 0$ the expected utility increases. By the definition of m either $N_i' = N_i^*$ or $N_j' = N_j^*$. Otherwise, if $\Delta_j > \lambda$, it implies that $\mu_i > \mu_j$. By moving $m = \min\{N_j, Tf(\boldsymbol{\mu})_i - N_i\}$, the expected utility changes by at least -$m(\mu_i - \mu_j)$. Since $\mu_i > \mu_j$ the expected utility increases. By the definition of m either $N_i' = N_i^*$ or $N_j' = N_j^*$.

We showed that every vector that violates the structure of $\boldsymbol{N^*}$ can be modified such that the expected utility increases and at least one more entry does not violate the structure of $\boldsymbol{N^*}$. Repeating this process at most $K - 1$ times yields $\boldsymbol{N^*}$ and increases the utility.

□

A.2 Regret of Fairness-Aware-ETC

Theorem 1. *Fix any arbitrary instance of R-O MAB, and let $N = 8L^{2/3}T^{2/3}\log^{1/3}T$. Algorithm 1 has a regret of $O(KL^{2/3}T^{2/3}\log^{1/3}T)$.*

Proof. We define the clean event to be the event that $|\hat{\mu}_{i,t} - \mu_i| \le r_t(i)$ holds for all arms simultaneously. Where $r_t(i) = \sqrt{\frac{2\log T}{N_{i,t}}}$. We will argue separately the clean event, and the "bad event" – the complement of the clean event. The regret is -

$$\mathcal{R}(T) = \mathcal{R}(T|\text{clean event})\mathbb{P}(\text{clean event}) + \mathcal{R}(T|\text{bad event})\mathbb{P}(\text{bad event}).$$

We now bound the probabilities for the clean event and for the "bad event" at the end of the exploration phase. That is, after each arm was pulled N times. Using Hoeffding's inequality, $\mathbb{P}(|\hat{\mu}_{i,t} - \mu_i| \le r_t(i)) \ge 1 - \frac{2}{T^4}$. Using the union bound, the probability for the clean event is

$$\mathbb{P}(\forall i \in [K]\, |\hat{\mu}_{i,t} - \mu_i| \le r_t(i)) \ge 1 - \frac{2K}{T^4}.$$

Since the "bad event" complements the clean event the probability of the "bad event" is at most $\frac{2K}{T^4}$. That the regret of the bad event is bounded by the highest possible regret - T. Combining the two last statement together we obtain that $\mathcal{R}(T|\text{bad event})\mathbb{P}(\text{bad event}) < \frac{2K}{T^3}$.

The clean event implies that for all $i \in [K]$, $|\hat{\mu}_i - \mu_i| \le O\left(\sqrt{\frac{\log T}{N}}\right) = O\left(L^{-1/3}T^{-1/3}\log^{1/3}T\right)$.

Thus, with respect to l_1 norm, $\|\boldsymbol{\mu} - \boldsymbol{\mu}'\|_1 \le O\left(KL^{-1/3}T^{-1/3}\log^{1/3}T\right)$, since f is L-Lipschitz, $T\|f(\hat{\boldsymbol{\mu}}) - f(\boldsymbol{\mu})\|_1 \le O\left(KL^{2/3}T^{2/3}\log^{1/3}T\right)$.

Next, we examine several cases at the end of the exploration phase and their effect on the regret. For arm i if $\Delta_i < \lambda$ the optimal algorithm pulls arm i exactly $Tf(\mu)_i$ times.

- If $\hat{\Delta}_i < \lambda$, Algorithm 1 will pull arm i, $\max\{N, Tf(\hat{\boldsymbol{\mu}})_i\}$ times. If $Tf(\boldsymbol{\mu})_i, Tf(\hat{\boldsymbol{\mu}})_i < N$ the regret is bounded by $N(\Delta_i + \lambda) \leq O\left(L^{2/3}T^{2/3}\log^{1/3} T\right)$. Otherwise, the regret is bounded by $(\Delta_i + \lambda)T\,|f(\boldsymbol{\mu})_i - f(\hat{\boldsymbol{\mu}})|$.
- If $\hat{\Delta}_i > \lambda$. Given the clean event it implies that $\lambda - \Delta_i \leq O\left(\sqrt{\frac{\log T}{N}}\right)$. In this situation, Algorithm 1 will not pull arm i anymore. If $N \geq Tf(\boldsymbol{\mu})_i$, the regret is be bounded by $N\Delta = O\left(L^{2/3}T^{2/3}\log^{1/3} T\right)$. If $N < Tf(\boldsymbol{\mu})_i$, the regret is $(\lambda - \Delta_i)N \leq O(\sqrt{N\log T}) \leq O(L^{2/3}T^{2/3}\log^{1/3} T)$.

For arm i if $\Delta_i > \lambda$ the optimal algorithm does not pull arm i at all.

1. If $\hat{\Delta}_i > \lambda$, Algorithm 1 will not pull arm i anymore. The regret is bounded by $N\Delta_i \leq O\left(L^{2/3}T^{2/3}\log^{1/3} T\right)$.
2. If $\hat{\Delta}_i < \lambda$. Given the clean event it implies that $\Delta_i - \lambda \leq O\left(\sqrt{\frac{\log T}{N}}\right) = O\left(L^{-1/3}T^{-1/3}\log^{1/3} T\right)$. Algorithm 1 will play arm i, $\max\{N, Tf(\hat{\boldsymbol{\mu}})_i\}$ times. If $N \geq Tf(\hat{\boldsymbol{\mu}})_i$, the regret is bounded by $N(\Delta_i + \lambda) = O\left(L^{2/3}T^{2/3}\log^{1/3} T\right)$. If $N < Tf(\hat{\boldsymbol{\mu}})_i \leq Tf(\boldsymbol{\mu})_i$, the regret is bounded by $(\Delta_i - \lambda)Tf(\hat{\boldsymbol{\mu}})_i \leq O\left(T^{2/3}\log^{1/3} T\right)$. Otherwise, if $N < Tf(\hat{\boldsymbol{\mu}})_i$ and $Tf(\boldsymbol{\mu})_i \leq Tf(\hat{\boldsymbol{\mu}})_i$ the regret is bounded by $(\Delta_i - \lambda)Tf(\hat{\boldsymbol{\mu}})_i + \lambda T(f(\hat{\boldsymbol{\mu}})_i - f(\boldsymbol{\mu})_i) \leq O(L^{2/3}T^{2/3}\log^{1/3} T)$

The analysis of the cases above implies that after the loop that starts on Line 1 ends, for every sub-optimal arm i, $|N_i^* - N_i| \leq O\left(L^{2/3}T^{2/3}\log^{1/3} T\right) + T\,|f(\boldsymbol{\mu})_i - f(\hat{\boldsymbol{\mu}})_i|$. Note that $T\sum_{i=1}^{K}|f(\hat{\boldsymbol{\mu}})_i - f(\boldsymbol{\mu})_i| = T\left\|f(\hat{\boldsymbol{\mu}}) - f(\boldsymbol{\mu})\right\|_1 \leq TL\left\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\right\|_1 \leq KL^{2/3}T^{2/3}\log^{1/3} T$. Hence the regret in the end of the exploration phase is bounded by $O\left(KL^{2/3}T^{2/3}\log^{1/3} T\right)$.

The remaining rounds (less than T) are allocated an arbitrary arm with the highest observed expected reward. Given the clean event, the reward gap between the optimal arm and the sub-optimal arm the algorithm commits to is less than $O\left(L^{-1/3}T^{-1/3}\log^{1/3} T\right)$. Hence the regret before allocating the remaining rounds is bounded by $O(T^{2/3}\log^{1/3} T)$.

Putting it all together, The regret for the event case is bounded by -

$$\mathcal{R}(T|\text{clean case}) \leq O\left(KL^{2/3}T^{2/3}\log^{1/3} T\right)$$

The total regret is bounded by -

$$\mathcal{R}(T) \leq (1 - \frac{2K}{T^4})O\left(KL^{2/3}T^{2/3}\log^{1/3} T\right) + \frac{2K}{T^4}\cdot T \leq O\left(KL^{2/3}T^{2/3}\log^{1/3} T\right)$$

□

A.3 Lower Bound

Theorem 3. *Fix time horizon T, number of arms K, and Lipschitz constant L. For any algorithm, there exists a R-O MAB instance such that $\mathcal{R}(T) \geq \Omega(T^{2/3})$.*

Proof. We construct a family of R-O MAB instances $\mathcal{F}$ that any algorithm will incur a high regret. For a given K, T and L. Let $\lambda = 1$ and $\epsilon = c \cdot T^{-1/3}$. We build $\mathcal{F}$ in the following way -

$$\mathcal{I}_{J,S} = \begin{cases} \mu_i = 1 & i = j \\ \mu_i = \frac{1}{2} & i \neq j, i \in S \\ \mu_i = \frac{1+\epsilon}{2} & i \neq j, i \notin S \end{cases} \qquad f(\boldsymbol{\mu})_i = \begin{cases} 0 & \mu_i \leq \frac{1}{2} \\ \frac{L(\mu_i - 0.5)}{K} & \frac{1}{2} < \mu_i \leq \frac{1}{2} + \epsilon \\ \frac{L\epsilon}{K} & \frac{1}{2} + \epsilon < \mu_i \leq 1 \end{cases}$$

$\mathcal{I}_{J,S}$ describes a vector of expected rewards. The vector is composed of arm j with expected reward 1, subset of arms, S, with expected reward $\frac{1}{2}$ and the remaining arms have expected reward of $\frac{1+\epsilon}{2}$. The size of $\mathcal{F}$ is $K \cdot 2^{K-1}$. The fairness function, f is piece-wise linear with 3 pieces. The first piece, for values less or equal to $\frac{1}{2}$ is constant and equal to 0. The second piece, for values between $\frac{1}{2}$ to $\frac{1}{2} + \epsilon$ is linear that goes from 0 to $\frac{L\epsilon}{K}$. The third piece, for values greater than $\frac{1}{2} + \epsilon$ is constant and equal to $\frac{L\epsilon}{K}$. In Claim 1 we prove that f is L-Lipschitz and sums to at most one.

Following the optimal algorithm described in Section 2.1, for every instance in $\mathcal{F}$ the optimal algorithm pulls all the sub-optimal arms according to the fairness function. This means the optimal algorithm does not pull at all the arms in S and pulls $\frac{TL\epsilon}{2K}$ times sub-optimal arms not in S. The algorithm pulls the optimal arm the remaining rounds $(T - TL\epsilon(K-1-|S|)/2K)$. In order to distinguish between two instances in $\mathcal{F}$, the decision-maker must pull each arm at least $1/8\epsilon^2$ times, otherwise with positive probability the decision-maker is unable to distinguish between at least two instances.

We use the following notation for simplicity - $\mathcal{I}_1 = \mathcal{I}_{K,[K-1]}, \mathcal{I}_{K,i} = \mathcal{I}_{K,[K-1]\setminus\{i\}}$. $\mathcal{I}_1$ is an instance where the K'th arm is the optimal arm and all the other arms have an expected reward of $\frac{1}{2}$. $\mathcal{I}_{K,i}$ is an instance where the K'th arm is the optimal arm and all the other arms expect arm $i \neq K$ have an expected reward of $\frac{1}{2}$ and arm i has an expected reward of $\frac{1+\epsilon}{2}$.

Fix an algorithm ALG. If ALG pulls each arm at least $1/8\epsilon^2$ times, the utility of ALG -

$$\begin{aligned} \mathcal{U}_{ALG}(\mathcal{I}_1) &\leq (T - (K-1)/8\epsilon^2) + (K-1)/8\epsilon^2 \cdot \frac{1}{2} \\ &= T - \frac{(K-1)T^{2/3}}{16c^2}. \end{aligned}$$

On the other hand, the optimal algorithm will pull only arm K and hence, the utility of OPT is T. The regret is then - $\mathcal{R}(\mathcal{I}_1) = \mathcal{U}_{OPT}(\mathcal{I}_1) - \mathcal{U}_{ALG}(\mathcal{I}_1) \geq \frac{(K-1)T^{2/3}}{16c^2} = \Omega(T^{2/3})$.

Otherwise, there exists arm i that ALG draws less than $1/8\epsilon^2$ times. With probability at least 0.01, in the instance that generated the sequence of rewards $i \notin S$. That is, $\mu_i = \frac{1+\epsilon}{2}$. For example $\mathcal{I}_{K,[K-1]\setminus\{i\}}$. Denote by N_i the number of times arm i is pulled by ALG.

$$\begin{aligned} \mathcal{U}_{ALG}(\mathcal{I}_{K,i}) &\leq (T - N_i) + N_i\left(\frac{1+\epsilon}{2}\right) - (T \cdot f(\mathcal{I}_{K,i})_i - N_i) \\ &= T + N_i\left(\frac{1+\epsilon}{2}\right) - T \cdot f(\mathcal{I}_{K,i})_i \\ &\leq T + \frac{1}{8\epsilon^2}\left(\frac{1+\epsilon}{2}\right) - \frac{TL\epsilon}{2K} \\ &= T - T^{2/3}\left(\frac{cL}{2K} - \frac{1}{16c^2}\right) + \Theta\left(T^{1/3}\right) \end{aligned} \tag{3}$$

The optimal algorithm will pull arm i, $TL\epsilon/2K$ times and arm K the remaining rounds. The utility of OPT is -

$$
\begin{aligned}
\mathcal{U}_{OPT}(\mathcal{I}_{K,i}) &= (1 - \frac{L\epsilon}{2K})T + \frac{TL\epsilon}{2K}\left(\frac{1+\epsilon}{2}\right) \\
&= T - \frac{TL\epsilon}{4K} + \frac{TL\epsilon^2}{4K} \\
&= T - \frac{T^{2/3}cL}{4K} + \Theta(T^{1/3})
\end{aligned}
$$

Finally, observe that

$$
\begin{aligned}
\mathcal{R}(ALG, \mathcal{I}_{K,i}) &= \mathcal{U}_{OPT}(\mathcal{I}_{K,i}) - \mathcal{U}_{ALG}(\mathcal{I}_{K,i}) \\
&\geq T^{2/3}\left(\frac{cL}{4K} - \frac{1}{16c^2}\right) + \Theta(T^{1/3}) \\
&= \Omega(T^{2/3})
\end{aligned}
$$

□

Claim 1. *f is L-Lipschitz and sums to at most one.*

Proof. f **is L-Lipschitz**. f is continuous and piece-wise linear, hence it is $\sup|f'(x)| - Lipschitz$. $\sup|f'(x)| = \frac{L}{K} < L$, and therefore f is $L - Lipschitz$.

f **sums to at most 1** - notice that

$$
\max \sum_{i=1}^{K} f(\boldsymbol{\mu})_i \leq \sum_{i=1}^{K} \max_{\boldsymbol{\mu}\in[0,1]^K} f(\boldsymbol{\mu})_i = L\epsilon.
$$

$L\epsilon \leq 1$ if $\epsilon < 1/L$. In the example above, $\epsilon = cT^{-1/3}$, for large enough T we obtain $\epsilon < 1/L$. □

A.4 General Black Box Algorithm

Theorem 2. *Fix any arbitrary instance of R-O MAB, and let $\alpha = K^{2/3}L^{2/3}T^{-1/3}\log^{1/3}T$, $\beta = T^{-1/3}\log^{1/3}T$. Then, Algorithm 2 has a regret of $O(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T)$.*

Proof. We analyze the regret that stems from the different phases: Approximating the reward gap (Phase 1), approximating f (Phase 2), granting opportunities (Phase 3), and invoking ALG (Phase 4).

Lemma 2, which we prove below, guarantees that after each arm was pulled at most $8T\beta$ the algorithm is certain (w.h.p.) for every arm if its cost of opportunity is bigger or smaller than the transfer cost by a factor of β. Thus the regret that stems from this phase is bounded by $O(TK\beta)$.

The second phase approximates f only for entries with low cost of opportunity, i.e., $LCB(\Delta_i) < \lambda$. Lemma 3 shows that after each arm is pulled at most $8T\alpha$ times, f is approximated up to a factor of α in the relevant entries. An immediate consequence of Lemma 3 is that after at most $8T\alpha$ rounds, $\sum_{i\in[K];LCB(\Delta_i)\leq\lambda} \max_{\boldsymbol{\mu'}\in C_t} f(\boldsymbol{\mu'})_i - \min_{\boldsymbol{\mu'}\in C_t} f(\boldsymbol{\mu'})_i \leq \alpha$. Thus the regret from this phase is bounded by $O(TK\alpha)$. The regret from the first two phases is bounded by $O(TK\max\{\alpha,\beta\})$.

In the third phase, the algorithm pulls each arm with low opportunity cost until the condition in Line 9 is met; therefore,

1. For arm i with $\Delta_i < \lambda$, the regret stems form the approximation error of f which is bounded by $O(T\alpha)$.

2. For arm i with $\Delta_i > \lambda$, if arm i is pulled during the third phase. This happens when the reward gap is very close to the transfer cost and $T\beta$ rounds are not sufficient to approximate Δ_i correctly. If $N_i \leq 8T\max\{\alpha,\beta\}$, the regret is also bounded by this term. Otherwise, assume $N_i > 8T\max\{\alpha,\beta\}$. Note that the regret is bounded by $Tf(\hat{\boldsymbol{\mu}})_i(\Delta_i - \lambda) + \lambda T\,|f(\hat{\boldsymbol{\mu}})_i - f(\boldsymbol{\mu})_i|$. The second part of this expression is bounded by the approximation

error of f, i.e., $O(T\alpha)$. We bound the first part of the regret by bounding $\Delta_i - \lambda$. To do so, we now look into the two possible cases that stopped the loop in Line 5. The term the clean event is similar to the clean event defined in Subsection A.2.

(a) If $UCB(\Delta_i) < \lambda + \beta$, given the clean event, $\lambda - \beta < \lambda < \Delta_i \le UCB(\Delta_i) \le \lambda + \beta$. This implies that $\Delta_i - \lambda \le 2\beta$. Thus, the first part of the regret is bounded by $2T\beta$.

(b) If $LCB(\Delta_i) > \lambda - \beta$, given the clean event, $\lambda - \beta < LCB(\Delta_i) \le \lambda < \Delta_i \le \lambda + \beta$. This implies that $\Delta_i - \lambda \le 2\beta$ and therefore the first part of the regret is bounded by $2T\beta$.

To summarize, if arm i with $\Delta_i > \lambda$ is pulled in the third phase, the regret associated with this arm is bounded by $O(T(\alpha + \beta))$.

Eventually, Phase 4 (invoking ALG) contributes the regret of ALG.

The regret from the approximation phases is bounded by $O\left(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T\right)$. The regret from filling the fairness requirements is also bounded by this term. Ultimately, the total regret is bounded by $O\left(K^{5/3}L^{2/3}T^{2/3}\log^{1/3}T\right)$, assuming that this term is an upper bound on the regret of ALG w.r.t. a vanilla stochastic bandit setting. □

Lemma 2. *Fix any arbitrary instance of R-O MAB and let $\beta = T^{-1/3}\log^{1/3}T$. Then after at most $8T\beta$ rounds, for every $i \in [K]$, either $UCB(\Delta_i) < \lambda + \beta$ or $LCB(\Delta_i) > \lambda - \beta$*

Proof. With high probability after pulling each arm $T\beta = T^{2/3}\log^{1/3}$ times, with high probability $\left|\hat{\Delta}_i - \Delta_i\right| < 2\sqrt{\frac{2\log T}{8T\beta}} = T^{-1/3}\log^{1/3}T = \beta$. Therefore, $UCB(\Delta_i) - LCB(\Delta_i) \le 2\beta$ and hence at least one of the following happens - $LCB(\Delta_i) \ge \lambda - \beta$ or $UCB(\Delta_i) \le \lambda + \beta$. □

Lemma 3. *Fix any arbitrary instance of R-O MAB and let $\alpha = K^{2/3}L^{2/3}T^{-1/3}\log^{1/3}T$ and a hyper-cube $C \subseteq [0,1]^K$. Then after pulling each arm at most $8T\alpha$ times, for every $i \in [K]$, $\max_{\boldsymbol{\mu}' \in C} f(\boldsymbol{\mu}')_i - \min_{\boldsymbol{\mu}' \in C} f(\boldsymbol{\mu}')_i \le \alpha$.*

Proof. After pulling each arm $8T\alpha$ times, with high probability for every $i \in [K]$, $|\hat{\mu}_i - \mu_i| \le K^{-1/3}L^{-1/3}T^{-1/3}\log^{1/3}T$. That is $|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}| < K^{2/3}L^{-1/3}T^{-1/3}\log^{1/3}T$. Since f is L-Lipschitz, we obtain that $\max_{\boldsymbol{\mu}' \in C} f(\boldsymbol{\mu}')_i - \min_{\boldsymbol{\mu}' \in C} f(\boldsymbol{\mu}')_i \le K^{2/3}L^{2/3}T^{-1/3}\log^{1/3}T = \alpha$. □

A.5 Efficient Oracle for Computing Equation (4) for Softmax

Proposition 1. *Let $f^{\text{sft}}(\boldsymbol{\mu})_i = \frac{\exp^{\mu_i}}{\sum_{l=1}^{K}\exp^{\mu_l}}$. For every hyper-cube $C \subseteq [0,1]^K$, the term*

$$\max_{\mu \in C} f^{\text{sft}}(\boldsymbol{\mu})_i - \min_{\mu \in C} f^{\text{sft}}(\boldsymbol{\mu})_i \tag{4}$$

can be computed efficiently.

Proof. Let

$$f^{\text{sft}}(\boldsymbol{\mu}) = \frac{\exp^{\mu_i}}{\exp^{\mu_i} + \sum_{l \ne j}\exp^{\mu_l}} = \frac{x}{x+y}$$

for $x = \exp^{\mu_i}$ and $y = \frac{\sum_{l \ne j}\exp^{\mu_l}}{\exp^{\mu_i}}$. Due to Proposition 2 below, the function $g(x,y) \overset{\text{def}}{=} \frac{x}{x+y}$ is monotonically increasing in x and monotonically decreasing in y. Consequently,

- $\max_{\boldsymbol{\mu} \in C} f^{\text{sft}}(\boldsymbol{\mu})_i$ is obtained for $\mu_i = \max C[i]$ and $\mu_j = \min C[j]$ for all $j \in [K], j \ne i$.
- $\min_{\boldsymbol{\mu} \in C} f^{\text{sft}}(\boldsymbol{\mu})_i$ is obtained for $\mu_i = \min C[i]$ and for all $j \in [K], j \ne i$ $\mu_j = \max C[j]$.

Combining these facts, we proved that Equation 4 can be computed efficiently.

□

Algorithm 3: Constant function utility maximization algorithm

```
Input: Black-box bandit algorithm $ALG$
1 for $i = 1, \ldots T\gamma$ do // Phase 1
2     for $j = 1 \ldots K$ do
3         pull arm $j$ if $LCB(\Delta_j) \leq \lambda$
4 Invoke $ALG$ for the remaining rounds // Phase 2
```

Proposition 2. *Let $g(x, y) = \frac{x}{x+y}$, where $x, y > 0$. Then g is monotonically increasing in x and monotonically decreasing in y.*

Proof. Since g is differentiable, so it is suffice to show that the partial derivatives are always positive for x and always negative for y. Observe that

$$\frac{\partial g}{\partial x} = \frac{y}{(x+y)^2}, \qquad \frac{\partial g}{\partial y} = \frac{-x}{(x+y)^2}.$$

Since $y > 0$, $\frac{\partial g}{\partial x}$ is always positive; thus, g is monotonically increasing in x. Additionally, since $x > 0$, $\frac{\partial g}{\partial y}$ is always negative; thus, g is monotonically decreasing in y. □

B Special Cases

In this section, we present two private cases that achieve better regret than the worst-case regret of $\tilde{O}(T^{2/3})$ presented in Section 3.

First, if f has very loose fairness requirements, i.e., $f(\boldsymbol{\mu})_i \leq T^{-\gamma}$ for all $\boldsymbol{\mu}$ and $\gamma \geq \frac{1}{2}$. In such a case, pulling each arm for $O(T^{1-\gamma})$ rounds and invoking a black-box algorithm for the remaining rounds achieves a regret of $O(\max\{KT^{1-\gamma}, \mathcal{R}(ALG)\})$. Formally,let $\gamma \geq \frac{1}{2}$, $f \in \mathcal{F}^{\text{min}}_{T,\gamma}$ if for all $\boldsymbol{\mu} \in [0,1]^K$ and for all $i \in [K]$, $f(\boldsymbol{\mu})_i \leq T^{-\gamma}$.

Proposition 3. *Fix any R-O MAB instance with horizon T and fairness function $f \in \mathcal{F}^{\text{min}}_{T,\gamma}$, pulling all arms $O(T^{1-\gamma})$ and invoking a black-box bandit algorithm for the remaining rounds achieves a regret bounded by $O(\max\{KT^{1-\gamma}, \mathcal{R}(ALG)\})$.*

Proof. For every arm i after it was pulled $T^{1-\gamma}$ times, $Tf(\boldsymbol{\mu})_i \leq N_i$. Thus, the regret that stems from this phase is bounded by $O\left(KT^{1-\gamma}\right)$. The optimal algorithm pulls the optimal arm for the remaining rounds. Thus any additional regret is a consequence of ALG and therefore the regret is bounded by $O(\mathcal{R}(ALG))$. □

The second case is a generalization of f^{uni} which was presented in Section 2.2. Formally, $f \in \mathcal{F}^{\text{const}}$ if there exists $\gamma \leq \frac{1}{K}$ such that for all $\boldsymbol{\mu} \in [0,1]^K$ and for all $i \in [K]$, $f(\boldsymbol{\mu})_i = \gamma$. Algorithm 3 is a variation of Successive Elimination [32] that achieves a regret of $O(\max\{\sqrt{KT \log T}, \mathcal{R}(ALG)\})$.

To illustrate the motivation to use Algorithm 3 instead of Algorithm 2 consider the following R-O MAB instance. Fix horizon T, number of arms K and define f to be $f(\boldsymbol{\mu})_i = \frac{1}{T}$ for every $\boldsymbol{\mu} \in [0,1]^K$ and every $i \in [K]$. Let $\lambda = \frac{1}{2}$. Set $\mu_1 = 1, \mu_2 = \frac{1+T^{-1/3}}{2}$ and set the expected rewards of the other arms arbitrarily. Note that $\Delta_2 = \frac{1-T^{-1/3}}{2}$. Thus, the first phase of Algorithm 2 pulls each arm $O\left(T^{2/3}\right)$ and incur a $\Omega\left(T^{2/3}\right)$ regret. Algorithm 3 will pull each arm exactly once and then invoke the black-box algorithm. Thus, in this case the regret is bounded by the regret of ALG.

Algorithm 3 leverages the fact that functions in $\mathcal{F}^{\text{const}}$ has no approximation error and thus can combine phases 1-3 in Algorithm 2 to one phase. Note that if an arm is pulled $T\gamma$ times we can stop pulling it. In the first phase (Lines 1-3) arms are pulled either at most $T\gamma$ times or until the algorithm is certain that $\Delta_i > \lambda$. In the second phase (Line 4), the black box algorithm is invoked and thus the regret is also bounded by the regret of the black box algorithm.

Proposition 4. *For any R-O MAB instance with fairness function $f \in \mathcal{F}^{\text{const}}$, Algorithm 3 achieves a regret of $O\left(\max\{\sqrt{KT \log T}, \mathcal{R}(ALG)\}\right)$.*

Proof. We start by analyzing the regret of the first phase. For arm i with $\Delta_i < \lambda$, given the clean event, arm i is pulled exactly $T\gamma \leq \frac{T}{K}$ times in the first phase, as in the optimal algorithm.

For arm i with $\Delta_i > \lambda$, if $\Delta_i - \lambda \leq O\left(\sqrt{\frac{\log T}{T\gamma}}\right)$, arm i is pulled $T\gamma$ times and the regret will be bounded by $O\left(\sqrt{\frac{T\log T}{K}}\right)$. Otherwise, $\Delta_i - \lambda > O\left(\sqrt{\frac{\log T}{T\gamma}}\right)$. Arm i is pulled as long as $\Delta_i - \lambda \leq O\left(\sqrt{\frac{\log T}{N_i}}\right)$. Thus the regret of the first phase is bounded by $O(\sqrt{KT\log T})$. The regret that stems from the second phase is bounded by the regret of ALG. Combining the two together we get that $\mathcal{R}($ Algorithm 3, $ALG) \leq O(\max\{\sqrt{KT\log T}, \mathcal{R}(ALG)\})$. □

C Full Version of Algorithm 2

In this section we present the full version of Algorithm 2, brought here as Algorithm 4. The algorithm relies heavily on the confidence intervals of the rewards and of the rewards gaps. The confidence intervals of the rewards are used to estimate f's variability inside the hyper-cube of expected rewards. The confidence interval of the expected reward of arm i, given that it was pulled N_i times is $[\hat{\mu}_i - \sqrt{\frac{2\log T}{N_i}}, \hat{\mu}_i + \sqrt{\frac{2\log T}{N_i}}]$. The confidence intervals of the reward gaps Δ_i are used to determine the most probable optimal algorithm and follow it. They are defined to be $LCB(\Delta_i) = \max_{j\in[K]}\hat{\mu}_j + \hat{\mu}_i - 2\sqrt{\frac{2\log T}{N_i}}$, $UCB(\Delta_i) = \max_{j\in[K]}\hat{\mu}_j + \hat{\mu}_i + 2\sqrt{\frac{2\log T}{N_i}}$. Note that using the formulas above the upper confidence bound can be above one and the lower confidence bound can be less than zero. This is not possible in our settings. The bounds are trimmed all through the algorithm to be between zero and one. Note that in order to calculate the confidence interval of the reward gaps we use the fact that in the first two phases all the arms are pulled the same number of times.

The algorithm starts with initialization phase (Lines 1-5). In Line 1 the time variable is initialized. We then initialize the arms' data - the number of pulls is set to zero (Line 3), the confidence interval of the reward gap is set to [0, 1] (Line 4). Lastly, the hyper-cube of the rewards, C_1 is initialize with $[0,1]^K$ (Line 5).

After the initialization the first phase of the algorithm starts (Line 6). If there is an arm that the algorithm is not certain with high probability whether its reward gap is lower or higher than λ by more than β, all arms are pulled (Line 7). After arm i is pulled we increase the number of times arm i was pulled by one (Line 9), increase the time by one (Line 10) and update the expected reward based on the obtained reward (r_t) and the previous expected reward (Line 11). Possibly, the desired approximation is not achieved before T steps. In this case the execution is stopped once T rounds were played. This is done in Line 12. After pulling all the arms we update the estimators of the reward gaps Δ_i. In Line 14 we find the arm with the maximal expected reward. Then for each arm, the confidence interval of the reward gap is calculated (Lines 16-18) and the cube of probable expected values is updates (Line 19).

The second phase starts in Line 20. If there is an arm with low opportunity cost, i.e. $LCB(\Delta_i) < \lambda$, and the variability of f for this entry in the hyper-cube is high, all arms are pulled once (Line 21). As in the previous phase, first we pull each arm, update its counter, the time and the expected reward and then update the estimators. In Line 26, we assure that even if the desired approximation is not achieved by T rounds the execution ends.

In the third phase, which starts in Line 33, we ensure that each arm with low opportunity cost is pulled a sufficient number of times with respect to the fairness function f. M_i denotes the number of additional times arm i should be pulled. M_i is the difference between the number of times arm i should be pulled according to the fairness function $(Tf(\hat{\boldsymbol{\mu}})_i)$ and the number of times it was already pulled (N_i) rounded down to an integer. Arm i is then pulled M_i times and the counter and time are updated accordingly (Lines 36-38). To be precise, in order not to pull more than T times, M_i is set to be the minimum between $T - t$ and $\lfloor Tf(\hat{\boldsymbol{\mu}})_i - N_i \rfloor$.

If there are any remaining rounds, ALG is invoked until the end of the execution (Line 39).

Algorithm 4: Self-regulated Utility Maximization

```
Input: Black-box bandit algorithm $ALG$, allowed approximation error parameters $\alpha$ and $\beta$
$t = 1$
for $i = 1, \dots K$ do // Initialization
    $N_i \leftarrow 0$
    $LCB(\Delta_i) \leftarrow 0, UCB(\Delta_i) \leftarrow 1$
    $C_1[i] \leftarrow [0, 1]$
while $\exists j \in [K]$ s.t $UCB(\Delta_j) > \lambda + \beta$ and $LCB(\Delta_j) < \lambda - \beta$ do // Phase 1
    for $i = 1, \dots, K$ do
        Pull arm $i$, receive a reward $r_t$
        $N_i \leftarrow N_i + 1$
        $t \leftarrow t + 1$
        $\hat{\mu}_{i,t} \leftarrow ((N_i - 1)\hat{\mu}_{i,t-1} + r_t)/N_i$
        if $t > T$ then
            End execution
    $j^* \leftarrow \arg\max_{j \in [K]} \hat{\mu}_j$
    for $i = 1, \dots, K$ do
        $c_i \leftarrow \sqrt{\frac{2 \log T}{N_i}}$
        $LCB(\Delta_i) \leftarrow \max\{0, \hat{\mu}_{j^*} - \hat{\mu}_i - 2c_i\}$
        $UCB(\Delta_i) \leftarrow \min\{1, \hat{\mu}_{j^*} - \hat{\mu}_i + 2c_i\}$
        $C_t[i] \leftarrow [\max\{0, \hat{\mu}_i - c_i\}, \min\{1, \hat{\mu}_i + c_i\}]$
while $\exists j \in [K]$ s.t. $\max_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_j - \min_{\boldsymbol{\mu}' \in C_t} f(\boldsymbol{\mu}')_j > \alpha$ and $LCB(\Delta_j) < \lambda$ do
 // Phase 2
    for $j = 1, \dots, K$ do
        Pull arm $i$, receive a reward $r_t$
        $N_i \leftarrow N_i + 1$
        $t \leftarrow t + 1$
        $\hat{\mu}_{i,t} \leftarrow ((N_i - 1)\hat{\mu}_{i,t-1} + r_t)/N_i$
        if $t > T$ then
            End execution
    $j^* \leftarrow \arg\max_{j \in [K]} \hat{\mu}_j$
    for $j = 1, \dots, K$ do
        $c_i \leftarrow \sqrt{\frac{2 \log T}{N_i}}$
        $C_t[i] \leftarrow [\max\{0, \hat{\mu}_i - c_i\}, \min\{1, \hat{\mu}_i + c_i\}]$
        $LCB(\Delta_i) \leftarrow \hat{\max}\{0, \mu_{j^*} - \hat{\mu}_i - 2c_i\}$
for $i = 1, \dots K$ do // Phase 3
    if $LCB(\Delta_i) < \lambda$ and $N_i < Tf(\boldsymbol{\mu}')_i$ then
        $M_i \leftarrow \min\{T - t, \lfloor Tf(\hat{\boldsymbol{\mu}})_i - N_i \rfloor\}$
        Pull arm $i$ $M_i$ times
        $N_i \leftarrow N_i + M_i$
        $t \leftarrow t + M_i$
Invoke $ALG$ for the remaining rounds // Phase 4
```