Simple Combinatorial Algorithms for Combinatorial Bandits: Corruptions and Approximations

Haike Xu[1] **Jian Li**[1]

[1]Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University

Abstract

We consider the stochastic combinatorial semi-bandit problem with adversarial corruptions. We provide a simple combinatorial algorithm that can achieve a regret of $\tilde{O}\left(C + d^2K/\Delta_{min}\right)$ where C is the total amount of corruptions, d is the maximal number of arms one can play in each round, K is the number of arms. If one selects only one arm in each round, we achieves a regret of $\tilde{O}\left(C + \sum_{\Delta_i>0} 1/\Delta_i\right)$. Our algorithm is combinatorial and improves on the previous combinatorial algorithm by Gupta et al. [2019] (their bound is $\tilde{O}\left(KC + \sum_{\Delta_i>0} 1/\Delta_i\right)$), and almost matches the best known bounds obtained by Zimmert and Seldin [2021], Zimmert et al. [2019] (up to logarithmic factor). Note that the algorithms in Zimmert et al. [2019], Zimmert and Seldin [2021] require one to solve complex convex programs while our algorithm is combinatorial, very easy to implement, requires weaker assumptions and has very low oracle complexity and running time. We also study the setting where we only get access to an approximation oracle for the stochastic combinatorial semi-bandit problem. Our algorithm achieves an (approximation) regret bound of $\tilde{O}\left(d\sqrt{KT}\right)$. Our algorithm is very simple, only worse than the best known regret bound by $\sqrt{d}$, and has much lower oracle complexity than previous work.

1 INTRODUCTION

Stochastic multi-armed bandit (MAB) is a classical online learning problem [Lai and Robbins, 1985]. There are K bandit arms. By pulling an arm i, the player receives a random reward, which is an i.i.d. sample from an unknown distribution associated with arm i. The task of the player is to select one arm to pull in each round and maximize the cumulative reward (or minimize the regret). An important generalization to the classical MAB problem is the combinatorial multi-armed bandit problem (CMAB), in which the player selects a subset Z_t (also called a super-arm) of arms in round t, where Z_t belongs to some combinatorial family $\mathcal{M}$ (e.g., $\mathcal{M}$ contains all subsets of $[K]$ of size d). CMAB has important applications in a variety of application domains, such as online advertising, recommendation system, wireless networks, where each action is a combinatorial subset and the rewards are stochastic. CMAB has attracted significant attention in recent years [Gai et al., 2012, Kveton et al., 2015, Combes et al., 2015, Chen et al., 2016, Wang and Chen, 2018].

Recently, Lykouris et al. [2018] introduced a new model of stochastic bandits with adversarial corruptions (MAB-AC). Their model is motivated by the observation that in many applications most of the data is stochastic but a small proportion may be adversarially corrupted, e.g., click frauds, fake reviews and email spams. Theoretically, their model can be seen as an interpolation between the stochastic rewards and the fully adversarial ones. Lykouris et al. [2018] first presented an algorithm in this setting achieving a regret bound of $O\left(KC(\sum_{\Delta_i>0} 1/\Delta_i)\log^2(KT)\right)$, where C is total amount of corruptions and Δ_i is the gap of arm i (see their definitions in Section 2). The regret bound was improved to $O\left(KC + (\sum_{\Delta_i>0} 1/\Delta_i)\log(KT)\log T\right)$ by Gupta et al. [2019]. Later, Zimmert and Seldin [2021] realized that one can use online mirror descent (OMD) with Tsallis entropy regularization with power $\alpha = \frac{1}{2}$ (the technique was originally developed in Zimmert et al. [2019] to solve MAB in both stochastic and adverserial settings) can achieve the optimal regret $O\left(C + (\sum_{\Delta_i>0} 1/\Delta_i)\log T\right)$. In particular, their algorithm needs to solve the following constrained optimization problem: $x_t = \operatorname{argmin}_{x\in\Delta^{K-1}}\langle x, \hat{L}_{t-1}\rangle - \frac{4}{\eta_t}\sum_{i=1}^{K}\sqrt{x_i}$ where Δ^{K-1} is a probability simplex, $\hat{L}_t$ is a cumulative estimated loss, η_t is a learning rate. Additionally, their regret analysis uses "self-bounding" trick which requires the technical assumption that the optimal arm must be unique.

Zimmert et al. [2019] extended the above result to CMAB with adverserial corruptions (CMAB-AC) (assuming semi-bandit feedback). They used the "(Δ,C,T) self-bounding constraint" trick in Zimmert and Seldin [2021] and obtained a regret bound of $O\left(C + dK \log T/\Delta_{min}\right)$. At each time step t, their algorithm needs an oracle to solve a similar convex optimization problem over $Conv(\mathcal{M})$, the convex hull of all feasible super-arms. For similar reason, their analysis also requires the unique optimal super-arm assumption which says that $\exists \Delta_{min} > 0 \; s.t. \forall Z \neq Z^*, \Delta(Z) \geq \Delta_{min}$, where $\Delta(Z)$ is the difference between super-arm Z's and the optimal super-arm's mean reward and will be formal defined in section 2.

For many combinatorial problems such as the TSP problem and the maximum independent set problem, there is no efficient oracle that can provide exact optimal solutions for the offline optimization problem. [Kakade et al., 2009] first introduced the notion of approximation oracles, which, upon each query, returns an α-approximation for the offline optimization problem, and the notion of regret is extended to *α-approximate regret* in which the benchmark is α times the optimal value. Recently, several online optimization problems have been studied by using approximation algorithms as oracles (e.g., Kakade et al. [2009], Lin et al. [2015], Chen et al. [2016], Garber [2020], Hazan et al. [2018]).

1.1 OUR CONTRIBUTIONS

In this paper, we study the combinatorial multi-armed bandit problem with adversarial corruptions (CMAB-AC) and the combinatorial multi-armed bandit problem with approximation oracles (CMAB-APX). For both problems, we assume semi-bandit feedbacks (we can observe the rewards for all arms in the super-arm we choose in this round, but nothing else).

We first state our result for CMAB-AC. In the following theorems, we let K be the number of arms in the ground set, d be the maximal number of arms one feasible super-arm can contain, and C be the amount of corruption exerted by the adversary (see the formal definition in Section 2). For MAB, we define the gap Δ_i of arm i to be the difference between and the mean of the optimal arm and that of arm i. For combinatorial bandit, we generalize the definition to $\Delta_i = \max_{Z \in \mathcal{M}} \mu(Z) - \max_{i \in Z \wedge Z \in \mathcal{M}} \mu(Z)$. Furthermore, we adopt the notation $\Delta_{min} = \min_{\Delta_i > 0} \Delta_i$.

Theorem 1.1. *For CMAB-AC under semi-bandit feedback setting, CBARBAR algorithm can achieve the following expected pseudo-regret upper bound:*

$$O\left(C + \frac{d^2 K}{\Delta_{min}} \log^2 T\right) \tag{1}$$

The oracle complexity of CBARBAR is $O(K \log T)$.

Note that Zimmert et al. [2019] obtained a regret bound of $O\left(C + dK \log T/\Delta_{min}\right)$ and oracle complexity $O(T)$ for CMAB-AC, whose regret is better than ours. However, our algorithm has the following advantages: first, our algorithm is purely combinatorial and very easy to implement. The algorithm in Zimmert et al. [2019] needs to solve the following convex optimization problem:

$$x_t = \operatorname*{argmin}_{x \in Conv(\mathcal{M})} \langle x, \hat{L}_{t-1} \rangle + \eta_t^{-1} \psi(x) \tag{2}$$

$$\psi(x) = \sum_{i=1}^{K} -\sqrt{x_i} + \gamma(1 - x_i)\log(1 - x_i)$$

where $Conv(\mathcal{M})$ is the convex hull of the feasible super-arm set, $\hat{L}_{t-1}$ is a cumulative estimated loss, $\eta_t = \frac{1}{\sqrt{t}}$, and $0 \leq \gamma \leq 1$ is a parameter. Unless $Conv(\mathcal{M})$ has very special structure, solving (2) is either highly nontrivial or prohibitively expensive in practice (e.g., say $\mathcal{M}$ is the set of spanning trees). [1] Our algorithm only needs an oracle that can solve the corresponding offline weighted problem efficiently (see Section 2), hence applies to almost all known problems in PTIME. Furthermore, our algorithm only needs to query the oracle $O(K \log T)$ times in total. Additionally, the regret analysis in Zimmert et al. [2019] uses the "self-bounding" trick which requires the unique optimal super-arm assumption. On the contrary, our analysis does not need such assumption.

By slightly changing the parameter of our algorithm, we can achieve the following gap-dependent regret bound for stochastic multi-armed bandit (i.e., $d = 1$) with adversarial corruption (MAB-AC).

Theorem 1.2. *For MAB-AC, CBARBAR algorithm achieves the following expected pseudo-regret upper bound:*

$$O\left(C + \sum_{\Delta_i > 0} \frac{1}{\Delta_i} \log(TK) \log T\right) \tag{3}$$

The running time of CBARBAR is $O(T \log K + K \log T)$.

Our algorithm improves the previous combinatorial algorithms by Lykouris et al. [2018] and Gupta et al. [2019] with regret bounds of $O\left(KC(\sum_{\Delta_i > 0} 1/\Delta_i) \log^2(KT)\right)$ and $O\left(KC + (\sum_{\Delta_i > 0} 1/\Delta_i) \log(KT) \log T\right)$ respectively. The optimal regret bound for MAB-AC is obtained by Zimmert and Seldin [2021] with running time $O(N \cdot KT)$ where N is number of iteration for Newton's method to solve a convex program over the simplex. Our algorithm has a slightly worse regret bound, but is much simpler to implement and runs much faster. Also our analysis does not require the unique optimal arm assumption.

[1] In theory, if $Conv(\mathcal{M})$ has an efficient separation oracle, (2) can be solved in polynomial time via the ellipsoid algorithm [Grötschel et al., 1981]. However, the ellipsoid algorithm is quite slow in practice.

Next, we discuss our result for the stochastic combinatorial semi-bandit problem with approximation oracle CMAB-APX. Suppose the approximation oracle can guarantee to return an α-approximation of the offline optimization problem. Now we measure the algorithm performance by α-regret, defined as $Reg^{\alpha} = \sum_{t=1}^{T} \alpha\mu(Z^*) - \mu(Z_t)$ where Z^* is the optimal super-arm.

Theorem 1.3. *For CMAB-APX, CBAR-APX algorithm achieves the following expected α-regret upper bound:*

$$O\left(d\sqrt{KT\log(KT)}\right). \tag{4}$$

The oracle complexity of CBAR-APX is $O(K\log T)$.

To the best of our knowledge, the only previous result that can be applied to CMAB-APX is Chen et al. [2016]'s SDCB In fact, they studied a more general problem in which the cost function is a general function. When specialized to our CMAB-APX problem, their algorithm achieves a regret bound $O(\sqrt{dKT\log T})$ and oracle complexity $O(T)$. Comparing with their algorithm, our algorithm CBAR-APX is only worse by a $\sqrt{d}$ factor, but enjoys a much lower oracle complexity. Note that the oracle complexity of an online learning algorithm is a very important performance metric and has been studied in a number of works [Hazan et al., 2018, Garber, 2020, Ito et al., 2019]. Another closely related setting is the one considered in Garber [2020], Hazan et al. [2018]. They studied more general online linear optimization with approximation oracle and is different from our CMAB-APX : their action space is $\mathcal{A} \subseteq \mathcal{R}^K$ while ours is $\mathcal{M} \subseteq \{0,1\}^{[K]}$, their environment is adversarial while ours is stochastic, and they consider bandit or full-information feedback while we consider semi-bandit feedback. In particular, their best known regret bounds are $\tilde{O}(\sqrt{T})$ and $\tilde{O}(T^{\frac{2}{3}})$ for the full information and bandit settings respectively Hazan et al. [2018]. The oracle complexity is $O(T\log T)$ for the full information feedback and $\tilde{O}(T^{\frac{2}{3}})$ for the bandit setting.

1.2 RELATED WORK

The classical Multi-Armed Bandit (MAB) problem [Lai and Robbins, 1985] has been well-studied. It is known that one can achieve optimal regrets in both the stochastic and adverserial settings, by classical algorithms including the Upper Confidence Bound (UCB) algorithm [Auer et al., 2002a], the Active Arm Elimination (AAE) algorithm [Even-Dar et al., 2006], and EXP3 algorithm [Auer et al., 2002b]. The best of both worlds (stochastic and adverserial) setting has also attracted much attention in the recent years [Bubeck and Slivkins, 2012, Seldin and Slivkins, 2014, Auer and Chiang, 2016, Seldin and Lugosi, 2017, Zimmert and Seldin, 2021].

Recently, robustness of learning algorithms has attracted significant attention in machine learning community. There has been some recent work on improving the robustness of online learning algorithms as well, such as Kapoor et al. [2019], Niss and Tewari [2020], Lykouris et al. [2018], Gupta et al. [2019], Zimmert and Seldin [2021]. Kapoor et al. [2019] considered the kind of adversarial corruption where corruption happens with a fixed probability at each time step and Niss and Tewari [2020] considered the setting where the proportion of corruption is limited by at most ϵ fraction.

Combinatorial multi-armed bandit (CMAB) is an important extension of MAB and has been studied extensively in the literature [Gai et al., 2012, Kveton et al., 2015, Combes et al., 2015, Chen et al., 2016, Wang and Chen, 2018]. We only mention some previous works that are most related to ours. Kveton et al. [2015] and Combes et al. [2015] studied the stochastic combinatorial semi-bandit problem with exact oracle. Kveton et al. [2015] provided CombUCB1 algorithm which achieves a distribution independent regret $O\left(\sqrt{dKT\log T}\right)$ and a distribution dependent regret $O\left(dK\log(T)/\Delta_{min}\right)$. They also prove a regret lower bound $\Omega\left(\sqrt{dKT}\right)$ and $\Omega\left(dK\log(T)/\Delta_{min}\right)$. Combes et al. [2015] proposed the ESCB algorithm enjoying a regret of $O\left(\sqrt{d}K\log(T)/\Delta_{min}\right)$ under the assumption that each arm's reward is independent. Additionally, ESCB's oracle is rather complicated and typically very difficult to implement efficiently.

2 PRELIMINARY

In this section, we formally introduce the combinatorial bandit problems we study in this paper. The ground set contains K stochastic arms, labeled from 1 to K. We use $[K]$ to denote the set $\{1,2,\ldots,K\}$. Whenever we play arm i, we receive a stochastic reward which is an i.i.d sample from a hidden distribution with unknown mean μ_i. $\mathcal{M} \subseteq \{0,1\}^{[K]}$ is a combinatorial family over $[K]$ (e.g., $\mathcal{M}$ is a matroid). Sometimes, we call a subset $Z \in \mathcal{M}$ a super-arm. Let d be the maximum number of arms a super-arm can contain. In other words, for all $Z \in \mathcal{M}$, $|Z| \le d$. For a super-arm $Z \in \mathcal{M}$, we define $\mu(Z) = \sum_{j\in Z}\mu_j$.

The player plays the game for T rounds. In round t, a player chooses a super-arm $Z_t \in \mathcal{M}$ to play and observes the reward $r_{t,i}$ from each arm $i \in Z_t$ (i.e., semi-bandit feedback). The reward the player receives is the summation of rewards of all the arms in Z_t, i.e., $\sum_{i\in Z_t} r_{t,i}$.

Adversarial Corruptions: After the environment generates the reward vector $R_t = [r_{t,i}]$, there is an adversary who can corrupt the reward after seeing R_t, and we only observe the corrupted reward in each round. More formally, we can write down the interaction procedure for generating a reward at a specific time step t:

- First, the environment stochastically generates the vector of reward $R_t = [r_{t,1}, ...r_{t,K}]$;
- Then, the adversary observes this reward vector R_t and modifies it to $\tilde{R}_t = [r_{t,1} + c_{t,1}, ...r_{t,K} + c_{t,K}]$;
- At last, the player observes the corresponding entry or entries in $\tilde{R}_t$.

For each t, we define the quantity $C_t = \|\tilde{R}_t - R_t\|_{[d]}$, which is the sum of the maximum d components of the vector c_t. C_t can be thought as the maximum amount of corruption exerted by the adversary on time t. The total amount of corruption is defined as $C = \sum_{t=1}^{T} C_t$. Note that when $d = 1$, $C_t = \|\tilde{R}_t - R_t\|_\infty$ which is defined in the same way as in Lykouris et al. [2018], Gupta et al. [2019].

For combinatorial bandit setting, there exists an optimal super-arm called Z^* which is $\operatorname*{argmax}_{Z \in \mathcal{M}} \mu(Z)$. The goal is to minimize the expected cumulative pseudo-regret $Reg = \sum_{t=1}^{T} \mu(Z^*) - \mu(Z_t)$. In the special case where we can choose exact one arm in each round (i.e., the standard multi-armed bandit setting), we assume that there exists an optimal arm i^* with mean μ^*. Again, our goal is to minimize the expected cumulative pseudo-regret $Reg = \sum_{t=1}^{T} \mu^* - \mu_{i_t}$.

To quantify the regret, we need to define the *gap* for each arm. For the multi-armed bandit setting (i.e., $d = 1$), we define gap for an arm i as $\Delta_i = \mu^* - \mu_i$. For the general combinatorial bandit setting, we denote $Z_i^* = \operatorname{argmax}_{i \in Z \wedge Z \in \mathcal{M}} \mu(Z)$ for each arm i, and define the gap for an arm i as $\Delta_i = \mu(Z^*) - \mu(Z_i^*)$. Define the gap for super-arm Z as $\Delta(Z) = \mu(Z^*) - \mu(Z)$.

Typically $|\mathcal{M}|$ contains exponential number of super-arms. We assume that we have access to the following oracles that can optimize over $\mathcal{M}$ (exactly or approximately).

Exact Oracle The player can ask two types of queries to the oracle. The first type is specified by a weighted vector $[a_i]_{i\in[K]}$ of length K. Upon such a query, the oracle returns the super-arm with the maximum weight, i.e., $\operatorname*{argmax}_{Z \in \mathcal{M}} \sum_{j\in Z} a_j$. The second type is specified by a weight vector $[a_i]_{i\in[K]}$ of length K along with an index i. The oracle returns the maximum weight super-arm that contains i. i.e., $\operatorname*{argmax}_{Z \in \mathcal{M} \wedge i \in Z} \sum_{j\in Z} a_j$. Note that such oracle exists for most known polynomial time solvable combinatorial problems, such as matroid, shortest path, intersection of two matroids.

Approximation Oracle In many cases, the optimization problem over $\mathcal{M}$ is NP-hard, such as vertex cover. In such cases, even we know the means μ_i exactly, we can not find the optimal solution efficiently (assuming P$\neq$NP). A reasonable assumption here is to assume an oracle that can answer the optimization problem approximately. In particular, the oracle can also support the aforementioned two types of queries, but the super-arm returned by the oracle is only guaranteed to be an α-approximate solution, i.e. the summation of reward of the returned super-arm is at least $\alpha \cdot OPT$ for a fixed constant $0 < \alpha < 1$, where OPT is the reward of the optimal super-arm. Such an oracle was first introduced in Kakade et al. [2009]. With such an approximation oracle, we measure the player's performance by α-regret defined as $Reg^\alpha = \sum_{t=1}^{T} \alpha\mu(Z^*) - \mu(Z_t)$.

3 ALGORITHMS

3.1 ALGORITHMS FOR CMAB-AC

In this section, we first describe our Algorithm CBARBAR-for CMAB-AC. The main regret bound is shown in Theorem 1.1. By slightly changing the parameter of the algorithm, we also obtain a gap-dependent regret bound in Theorem 1.2 for MAB-AC (i.e., $d = 1$).

Our algorithm's framework is inspired by the BARBAR algorithm in Gupta et al. [2019], and we adapt it to the new combinatorial bandit setting. However, we note there are several important differences (even for MAB-AC, $d = 1$). For example, in Algorithm 1, we restrict the decreasing rate of empirical gap between consecutive epochs (the third term in Line 14) and we distinguish the pulls into explorative ones (n_i^m times Z_i^m for each arm i, see Line 4) and exploitative ones (n_*^m times Z_*^m, see Line 3). One consequence of the seemingly minor parameter change is the new improved regret bound in Theorem 1.2.

See the pseudocode in Algorithm 1. Note that we choose different values for n_*^m for CMAB-AC and MAB-AC (Line 3). We use Δ_i^m to denote the estimated gap for arm i in epoch m, n_i^m to denote the expected number of pulls on arm Z_i^m, where Z_i^m is the super-arm with the largest confidence lower bound containing arm i.

CBARBAR algorithm proceeds by epochs. The length of epochs increase exponentially, so there are $O(\log T)$ epochs. In each epoch E_m, the algorithm estimates the mean of arm i, μ_i within precision Δ_i^m calculated at the end of the last epoch through pulling Z_i^m and then prepares a more accurate Δ_i^{m+1} for the next epoch. The algorithm also exploits Z_*^m to guarantee theoretical regret bound. Over iterations, the approximation error of $|\Delta_i^m - \Delta_i|$ decreases exponentially and Δ_i^m converges to the true value. If there is no adversarial corruption, estimating an arm within precision Δ_i^m requires pulling it roughly $\tilde{O}\left((\Delta_i^m)^{-2}\right)$ times. To tackle adversarial corruption, we use the following tricks in Algorithm 1.

First, we set a lower bound $2^{-\frac{m}{4}}$ for Δ_i^{m+1} (First term on Line 14). Small Δ_i^{m+1} means high precision and requires many pulls. Note that due to adversarial corruption, the arm may turn out to have a large gap and incur a large regret. Therefore, setting a gradually decreasing lower bound $2^{-\frac{m}{4}}$

can prevent our algorithm from pulling an arm too many times when the estimated gap is not sufficiently accurate. We also set another lower bound $\frac{\Delta_i^m}{2}$ for Δ_i^{m+1} (Third term on Line 14). This lower bound ensures that the next epoch is at most 4 times as long as the previous one. Otherwise, adversarial corruptions can make sub-optimal arms have high rewards and their Δ_i^{m+1} very small, rendering the next epoch exceedingly long. This is crucial for obtaining an $O(C)$ regret, and is one of our key technical observations. Third, we pull the currently best arm/super-arm n_*^m times in addition to estimating gaps for each arm (Line 3).

Algorithm 1: algorithm for CMAB-AC (CBARBAR)

```
   Input: confidence $\delta \in (0,1)$, time horizon $T$
1  $\Delta_i^1 \leftarrow 1$ for all $i \in [K]$, $Z_i^1$ be an arbitrary valid
     super-arm containing arm $i$, $Z_*^1$ be an arbitrary valid
     super-arm, and $\lambda \leftarrow 1024 \log_2 \left(\frac{8K}{\delta} \log_2 T\right)$
2  for epochs $m = 1, 2...$ do
3      $n_*^m \leftarrow \lambda 2^{\frac{m-1}{2}}$ for $d = 1$; or $n_*^m \leftarrow \lambda d^2 K 2^{\frac{m-1}{2}}$
         for general $d$
4      $n_i^m \leftarrow \lambda \left(\frac{\Delta_i^m}{d}\right)^{-2}$ for all $i \in [K]$
5      $N^m \leftarrow \sum_{i=1}^K n_i^m + n_*^m$ and
         $T_m \leftarrow T_{m-1} + N^m$
6      $q_i^m \leftarrow \frac{n_i^m}{N^m}$ and $q_*^m \leftarrow \frac{n_*^m}{N^m}$
7      for $t = T_{m-1} + 1$ to $T_m$ do
8          Sample $Z_i^m$ w.p. $q_i^m$ and $Z_*^m$ w.p. $q_*^m$
9      $\hat{\mu}_i^m \leftarrow \frac{1}{n_i^m} \sum_{t \in E_m} \tilde{R}_{t,i} \cdot \mathbb{I}[Z_t = Z_i^m]$
10     $\overline{r}_*^m \leftarrow \max_{Z \in \mathcal{M}} \sum_{j \in Z} \left(\hat{\mu}_j^m + \frac{1}{16d} \Delta_j^m\right)$
11     $\underline{r}_i^m \leftarrow \max_{Z \in \mathcal{M} \wedge i \in Z} \sum_{j \in Z} \left(\hat{\mu}_j^m - \frac{1}{16d} \Delta_j^m\right)$
12     $Z_i^{m+1} \leftarrow \operatorname{argmax}_{Z \in \mathcal{M} \wedge i \in Z} \sum_{j \in Z} \left(\hat{\mu}_j^m - \frac{1}{16d} \Delta_j^m\right)$
13     $Z_*^{m+1} \leftarrow \operatorname{argmax}_{Z \in \mathcal{M}} \sum_{j \in Z} \left(\hat{\mu}_j^m - \frac{1}{16d} \Delta_j^m\right)$
14     $\Delta_i^{m+1} \leftarrow \max\left(2^{-\frac{m}{4}}, \overline{r}_*^m - \underline{r}_i^m, \frac{\Delta_i^m}{2}\right)$
```

3.2 ALGORITHM FOR CMAB-APX

In this section, we describe our algorithm CBAR-APX for CMAB-APX which achieves the regret bound stated in Theorem 1.3. See the pseudocode in Algorithm 2.

Some Notations Our algorithm uses an approximation oracle A. We call the oracle $A(w)$ with w being the weight vector (each weight corresponds an arm). $A(w)$ returns a super-arm whose total weight is at least αOPT where OPT $= \operatorname{argmax}_{Z \in \mathcal{M}} \sum_{i \in Z} w_i$. For each i, we also need an oracle $A_i(w)$, which returns a super-arm whose total weight is at least αOPT_i where $\text{OPT}_i = \operatorname{argmax}_{Z \in \mathcal{M} \wedge i \in Z} \sum_{j \in Z} w_j$. We use $\hat{\mu}^m(Z)$ as an abbreviation for $\sum_{i \in Z} \hat{\mu}_i^m$ and K_m be the remaining arm set at the beginning of epoch m.

Algorithm CBAR-APX inherits the framework of algorithm CBARBAR which proceeds by epochs with gradually tightening precision. Because in this setting there is no adversarial corruption, we can utilize the action elimination mechanism. One challenge here is that because of the variance of the α-approx oracle, we may mis-delete an arm belonging to the unique optimal super-arm (say there is one). If one arm in the unique optimal super-arm is wrongly eliminated, the previous OPT is no longer in the remaining arm set, which would lead to the change of the benchmark and lose any theoretical guarantee on the regret. The idea we use to solve this problem is that our algorithm ensures that whenever we first eliminate an arm belonging to the optimal super-arm, we must already get a αOPT super-arm. Our algorithm remembers the previous Z_*^m and compare it with the super-arm returned by oracle in epoch $m+1$, as in Line 6. In this way, once an αOPT super-arm is found, CBAR-APX never chooses a super-arm worse than αOPT.

Algorithm 2: algorithm for CMAB-APX (CBAR-APX)

```
   Input: confidence $\delta \in (0,1)$, time horizon $T$
1  $K_1 \leftarrow K$, for all $i \in [K]$, set $Z_i^1$ be an arbitrary
     valid super arm containing arm $i$, and
     $\lambda \leftarrow 1024 \log_2 \left(\frac{8K}{\delta} \log_2 T\right)$
2  for epochs $m = 1, 2...$ do
3      For all $i \in K_m$, pull $Z_i^m$ for $\lambda d^2 2^{2m}$ times
4      $\hat{\mu}_i^m \leftarrow$ average reward of arm $i$ from $Z_i^m$ in $E_m$
5      $Z_i^{m+1} \leftarrow A_i(\hat{\mu}^m)$
6      $Z_*^{m+1} \leftarrow \operatorname{argmax}_{Z \in \{A(\hat{\mu}^m), A_i(\hat{\mu}^m), Z_*^m\}} \hat{\mu}^m(Z)$
7      For arm $i \in Z_*^{m+1}$, set $Z_i^{m+1} = Z_*^{m+1}$
8      $K_{m+1} \leftarrow$
         $\left\{i \middle| i \in K_m \wedge \hat{\mu}^m(Z_i^{m+1}) > \hat{\mu}^m(Z_*^{m+1}) - \frac{2^{-m}}{4}\right\}$
```

4 ANALYSIS

4.1 PROOF OF THEOREM 1.1

In this subsection, we analyze the performance of algorithm CBARBAR for CMAB-AC under semi-bandit feedback. We first need a lemma to bound the length of each epoch m, and the number of epochs.

Lemma 4.1. *When $d > 1$, for any epoch m, its length, N^m, has the following lower bound and upper bound*

$$\lambda d^2 K 2^{\frac{m-1}{2}} \leq N^m \leq 2\lambda d^2 K 2^{\frac{m-1}{2}}$$

For any arm i, the expected number of pulls of its representative super-arm Z_i^m also has an upper bound

$$n_i^m \leq \lambda d^2 2^{\frac{m-1}{2}}$$

Therefore, the number of epochs our algorithm has M is upper bounded by $2\log_2(\frac{T}{K})$.

In epoch m, we denote $\tilde{n}_i^m(\tilde{n}_*^m)$ to be the empirical number of pulls on arm $Z_i^m(Z_*^m)$, N^m to be the length of epoch m, $C_i^m = \sum_{t\in E_m} |c_{t,i}|$ to be the amount of corruption exerted by the adversary in epoch m on arm i, $C^m = \sum_{t\in E_m} |c_t|_{[d]}$ to be the total amount of corruption exerted in epoch m, measured by L-[d] norm.[2]

Lemma 4.2. *In epoch m, we have the following guarantee on the empirical mean, and we abuse the notation to allow i equals $*$ in the second inequality.*

$$Pr\left\{\forall i \in [K], |\hat{\mu}_i^m - \mu_i| \le \frac{2C_i^m}{N^m} + \frac{\Delta_i^m}{16d}\right\} \ge 1 - \frac{\delta}{2}$$

$$Pr\left\{\forall i \in [K]\cup\{*\}, \tilde{n}_i^m < 2n_i^m\right\} \ge 1 - \frac{\delta}{2}$$

Now we condition on the event that

$$\mathcal{E} = \left\{\forall m, i : |\hat{\mu}_i - \mu_i| \le \frac{2C_i^m}{N^m} + \frac{\Delta_i^m}{16d} \wedge \tilde{n}_i^m \le 2n_i^m\right\}$$

which happens with probability at least $1-\delta$ according to Lemma 4.2.

Definition 4.3. *For any epoch m, we define a metric to measure the cumulative corruption exerted on our environment until epoch m: $\rho_m = \sum_{s=1}^m 2C^s/(2^{m-s}N^s)$*

Proposition 4.4. *For any super-arm Z and one arm $i \in Z$, we have $\Delta(Z) \ge \Delta_i$.*

Lemma 4.5 (upper bound for Δ_i^{m+1})**.** *For any arm $i \in [K]$, the empirical estimation of its gap, Δ_i^{m+1}, used in epoch $m+1$ has the following upper bound:*

$$\Delta_i^{m+1} \le 2\left(\Delta_i + 2^{-\frac{m}{4}} + \rho_m\right)$$

Lemma 4.6 (lower bound for Δ_i^{m+1})**.** *For any arm $i \in [K]$, the empirical estimation of its gap, Δ_i^{m+1}, used in epoch $m+1$ has the following lower bound:*

$$\Delta_i^{m+1} \ge \max\left(2^{-\frac{m}{4}}, \Delta_i - \frac{4C^m}{N^m}\right)$$

Lemma 4.7 (regret incurred by the explorative super-arm Z_i^m)**.** *In epoch m, the representative super-arm Z_i^m of arm i, has the following upper bound on its gap:*

$$\Delta(Z_i^m) \le \frac{5}{4}\Delta_i + 2\rho_{m-1} + \frac{2^{-\frac{m-2}{4}}}{4}$$

Lemma 4.8 (regret incurred by the exploitive super-arm Z_*^m)**.** *In epoch m, the current best super-arm Z_*^m has the following upper bound on its gap:*

$$\Delta(Z_*^m) \le 2\rho_{m-1} + \frac{2^{-\frac{m-2}{4}}}{4}$$

[2]The L-[d] norm of a vector is the summation of its maximal d coordinates' absolute value

Proposition 4.9. *When $d > 1$, the cumulative corruption is upper bounded by $\sum_m \lambda d^2 K 2^{\frac{m-1}{2}} \rho_m \le O(C)$*

Now, we have all the tools to prove Theorem 1.1.

Proof of Theorem 1.1. We set the confidence parameter $\delta = 1/T$, so the case where $\mathcal{E}$ does not hold only contributes an $O(1)$ to the expected regret. In this case, $\lambda \le O(\log(TK))$. In the following, we assume that $\mathcal{E}$ happens.

Recall that $Reg = \sum_{t=1}^T \mu(Z^*) - \mu(Z_t)$ and we can divide it by epochs. We define epoch regrert by $Reg_m = \sum_{t\in E_m} \mu(Z^*) - \mu(Z_t)$ and then $Reg = \sum_{m=1}^M Reg_m$.

For a fixed epoch m, we further decompose the regret into the exploitative part (Z_*^m) and the explorative part (Z_i^m), as follows: $Reg_m = \tilde{n}_*^m \Delta(Z_*^m) + \sum_{i=1}^K \tilde{n}_i^m \Delta(Z_i^m)$

According to Lemma 4.2, $\forall i \in [K]\cup\{*\}, \tilde{n}_i^m \le 2n_i^m$, so we use n_i^m instead in the following analysis.

Upper bound for exploitative regret $n_*^m\Delta(Z_*^m)$. By Lemma 4.8, we have $\Delta(Z_*^m) \le 2\rho_{m-1} + \frac{2^{-\frac{m-2}{4}}}{4}$. Basing on the level of magnitude of the two terms, we divide it into two cases:

Case 1: $\frac{1}{2}\Delta(Z_*^m) \le 2\rho_{m-1}$ This means that the regret caused by pulling super-arm Z_*^m is at most a constant multiple of ρ_{m-1}. For this case, by Proposition 4.9 that the summation over regret in all M epochs can be upper bounded by $\sum_{m=1}^M \lambda d^2 K 2^{\frac{m-1}{2}} \times 4\rho_{m-1} \le O(C)$

Case 2: $\frac{1}{2}\Delta(Z_*^m) \le \frac{2^{-\frac{m-2}{4}}}{4}$ Note that in epoch m, $n_*^m = \lambda d^2 K 2^{\frac{m-1}{2}}$. Then regret in this case for epoch m is upper bounded by $\lambda d^2 K 2^{\frac{m-1}{2}} * \frac{2^{-\frac{m-2}{4}}}{2} \le \lambda d^2 K 2^{\frac{m-2}{4}} \le O(\frac{\lambda d^2 K}{\Delta(Z_*^m)}) \le O(\frac{\lambda d^2 K}{\Delta_{min}})$

Upper bound for explorative regret $\sum_{i=1}^K n_i^m \Delta(Z_i^m)$. For an arm i, by Lemma 4.7, we have $\Delta(Z_i^m) \le \frac{5}{4}\Delta_i + 2\rho_{m-1} + \frac{2^{-\frac{m-2}{4}}}{4}$. Similarly, we divide it into 3 cases here.

Case 1: $\frac{1}{3}\Delta(Z_i^m) \le 2\rho_{m-1}$: Analysis for this case is similar to the previous case 1.

Case 2: $\frac{1}{3}\Delta(Z_i^m) \le 2^{-\frac{m-2}{4}}$: Note that in epoch m, for a single arm i, $n_i^m \le \lambda d^2 2^{\frac{m-1}{2}}$. The remaining analysis is similar to the previous case 2, whose regret upper bounded is $O(\frac{\lambda d^2}{\Delta(Z_i^m)}) \le O(\frac{\lambda d^2}{\Delta_i})$.

Case 3: $\frac{1}{3}\Delta(Z_i^m) \le \frac{5}{4}\Delta_i$: Here we need to utilize the second term in Lemma 4.6 $\Delta_i^m \ge \Delta_i - \frac{4C^{m-1}}{N^{m-1}}$.

- If $\frac{4C^{m-1}}{N^{m-1}} \leq \frac{1}{2}\Delta_i$, then we have $\Delta_i^m \geq \frac{1}{2}\Delta_i$, so regret can be bounded by $\frac{\lambda d^2 \Delta(Z_i^m)}{(\Delta_i^m)^2} \leq \frac{\frac{15}{4}\lambda d^2 \Delta_i}{\frac{1}{4}(\Delta_i)^2} \leq O\left(\frac{\lambda d^2}{\Delta_i}\right)$
- Otherwise, we have $\frac{8C^{m-1}}{N^{m-1}} \geq \Delta_i$, then the summation of all arms' regret belonging to this case is upper bounded by $N^m \Delta(Z_i^m) \leq \frac{15}{4} N^m \Delta_i \leq 30 N^m \frac{C^{m-1}}{N^{m-1}} \leq 60\sqrt{2} N^{m-1} \frac{C^{m-1}}{N^{m-1}} \leq O(C^{m-1})$. The third inequality uses the property $\lambda d^2 K 2^{\frac{m-1}{2}} \leq N^m \leq 2\lambda d^2 K 2^{\frac{m-1}{2}}$ from Lemma 4.1 to get $N^m \leq 2\sqrt{2} N^{m-1}$

Note there are at most $O\left(\log \frac{T}{K}\right)$ epochs. By summarizing all cases, we prove that the expected regret for our algorithm is upper bounded by $O\left(C + \frac{d^2 K \log^2(T)}{\Delta_{min}}\right)$.

Next, for the oracle complexity, there are $O(K)$ calls in each epoch and at most $O(\log \frac{T}{K})$ epochs, so the $O(K \log T)$ oracle complexity follows. □

4.2 PROOF OF THEOREM 1.2

In this subsection, we analyze the performance of algorithm 1 for MAB-AC. This setting is a special case of CMAB-AC with $d = 1$. Recall that the only change in algorithm 1 is that now $n_*^m \leftarrow \lambda 2^{\frac{m-1}{2}}$ which is K times smaller than before. Most previous lemmas still holds in this new setting, and the only changes are different versions of Lemma 4.1 and Proposition 4.9, as follows.

Lemma 4.10. *When $d = 1$, the length for any epoch m, N^m, satisfies $\lambda 2^{\frac{m-1}{2}} \leq N^m \leq \lambda K 2^{\frac{m-1}{2}}$ and for any arm i, the expected number of pulls of its representative super-arm Z_i^m also has an upper bound $n_i^m \leq \lambda 2^{\frac{m-1}{2}}$. Additionally, $N^m \leq 4N^{m-1}$. Therefore, the number of epochs M has an upper bound of $2\log_2 T$.*

Proposition 4.11. *When $d = 1$, the cumulative corruption is upper bounded by $\sum_{m=1} \lambda 2^{\frac{m-1}{2}} \rho_m \leq O(C)$*

The proof is almost the same as Proposition 4.9.

To eliminate the $\frac{K}{\Delta_{min}}$ dependency in Theorem 1.1, we need to use another way to calculate regret.

Proof of Theorem 1.2. Note in the multi-armed bandit setting, Z_*^m and Z_i^m all refer to a single arm. Furthermore, Z_i^m is exactly the i-th arm.

Similarly, we set the confidence parameter $\delta = \frac{1}{T}$, so the case $\mathcal{E}$ doesn't hold only contribute $O(1)$ regret to the regret expectation. In this case $\lambda \leq O(\log(TK))$ In the following, we assume that $\mathcal{E}$ happens.

Again, we calculate regret by epoch. For a fixed epoch m, we can decompose the regret into the exploitative part and the explorative part.

Upper bound for exploitative regret: $n_*^m \Delta(Z_*^m)$. By Lemma 4.8, we have $\Delta(Z_*^m) \leq 2\rho_{m-1} + \frac{2^{-\frac{m-2}{4}}}{4}$ and we consider the following two cases:

Case 1: $\frac{1}{2}\Delta(Z_*^m) \leq 2\rho_{m-1}$ By proposition 4.11, the summation over regret in all M epochs can be upper bounded by $\sum_{m=1}^M \lambda 2^{\frac{m-1}{2}} \times 4\rho_{m-1} \leq O(C)$

Case 2: $\frac{1}{2}\Delta(Z_*^m) \leq \frac{2^{-\frac{m-2}{4}}}{4}$ Note that in epoch m, $n_*^m = \lambda 2^{\frac{m-1}{2}}$. Then regret in this case for epoch m is upper bounded by $\lambda 2^{\frac{m-1}{2}} * \frac{2^{-\frac{m-2}{4}}}{2} \leq \lambda 2^{\frac{m-2}{4}} \leq O(\frac{\lambda}{\Delta(Z_*^m)}) \leq O(\frac{\lambda}{\Delta_{min}})$

Upper bound for explorative regret: $\sum_{i=1}^K n_i^m \Delta(Z_i^m)$. Instead of using Lemma 4.7, we use the property that $\Delta(Z_i^m) = \Delta_i$, so we only need to consider the previous case 3.

For an arm i, we utilize the second term in Lemma 4.6 $\Delta_i^m \geq \Delta_i - \frac{4C^{m-1}}{N^{m-1}}$.

- If $\frac{4C^{m-1}}{N^{m-1}} \leq \frac{1}{2}\Delta_i$, then we have $\Delta_i^m \geq \frac{1}{2}\Delta_i$, so regret can be bounded by $\frac{\lambda \Delta(Z_i^m)}{(\Delta_i^m)^2} \leq \frac{\lambda \Delta_i}{\frac{1}{4}(\Delta_i)^2} \leq O(\frac{\lambda}{\Delta_i})$
- Otherwise, we have $\frac{8C^{m-1}}{N^{m-1}} \geq \Delta_i$, then the summation of all arms' regret belonging to this case is upper bounded by $N^m \Delta(Z_i^m) = N^m \Delta_i \leq 8N^m \frac{C^{m-1}}{N^{m-1}} \leq 32 N^{m-1} \frac{C^{m-1}}{N^{m-1}} \leq O(C^{m-1})$. The second inequality uses the property that $N^m \leq 4N^{m-1}$ from Lemma 4.10.

By summarizing all cases, we prove that the expected regret for our algorithm is upper bounded by $O\left(C + \sum_{i=1}^K \frac{1}{\Delta_i} \log(TK) \log(T)\right)$ □

4.3 PROOF OF THEOREM 1.3

In this section, we analyze the performance of algorithm CBAR-APX for CMAB-APX under semi-bandit feedback.

Lemma 4.12. *We define event $\mathcal{E}$ as the following:*

$$\mathcal{E} = \left\{\forall m, i \in K_m, |\hat{\mu}_i^m - \mu_i| \leq 2^{-m}/(16d)\right\}$$

Then, we have $Pr\{\mathcal{E}\} \geq 1 - \delta$ and once $\mathcal{E}$ happens, we can guarantee that $\forall Z \in \mathcal{M} \cap \{0,1\}^{K_m}, |\hat{\mu}^m(Z) - \mu(Z)| \leq 2^{-m}/16$

The proof follows from standard concentration inequality.

Now we condition on the event $\mathcal{E}$ which happens with probability at least $1 - \delta$ according to Lemma 4.12.

Lemma 4.13 (lower bound for $\mu(Z_*^m)$ before the first deletion of an optimal arm)**.** *If in epoch m_1, we first delete an arm $i \in Z^*$, then for $m \le m_1$, we have $\mu(Z_*^{m+1}) \ge \alpha\text{OPT} - 2^{-m}/8$*

The following simple lemma is the key to our analysis.

Lemma 4.14 (lower bound for $\mu(Z_*^m)$ after the first deletion of an optimal arm)**.** *If in epoch m_1, we first delete an arm $i \in Z^*$, then for $m \ge m_1$, we have $\mu(Z_*^{m+1}) \ge \alpha\text{OPT} + 2^{-m}/8$*

Proof. We prove this by induction. First, in epoch m_1, the optimal super-arm Z^* is still admissible for the oracle. The deleted arm i satisfies

$$\hat{\mu}^{m_1}(Z_i^{m_1+1}) \ge \alpha\hat{\mu}^{m_1}(Z^*) \ge \alpha\text{OPT} - 2^{-m_1}/16$$

Because arm i is deleted, super-arm $Z_*^{m_1+1}$ satisfies

$$\begin{aligned}\mu(Z_*^{m_1+1}) &\ge \hat{\mu}^{m_1}(Z_*^{m_1+1}) - 2^{-m_1}/16\\ &\ge \hat{\mu}^{m_1}(Z_i^{m_1+1}) + 3\cdot 2^{-m_1}/16\\ &\ge \alpha\text{OPT} + 2^{-m_1}/8\end{aligned}$$

Now, we assume that this argument is valid for epochs before m' for some $m' > m_1$. We have the following

$$\begin{aligned}\mu(Z_*^{m'+1}) &\ge \hat{\mu}^{m'}(Z_*^{m'+1}) - 2^{-m'}/16\\ &\ge \hat{\mu}^{m'}(Z_*^{m'}) - 2^{-m'}/16 \qquad (5)\\ &\ge \mu(Z_*^{m'}) - 2^{-m'}/8\end{aligned}$$

Next we apply the induction argument for $\mu(Z_*^{m'})$

$$\begin{aligned}\mu(Z_*^{m'}) - \frac{2^{-m'}}{8} &\ge \left(\alpha\text{OPT} + \frac{2^{-(m'-1)}}{8}\right) - \frac{2^{-m'}}{8} \qquad (6)\\ &\ge \alpha\text{OPT} + 2^{-m'}/8\end{aligned}$$

Line (5) holds because according to Line 6 Algorithm 2, $Z_*^{m'}$ is always a candidate for $Z_*^{m'+1}$. □

Corollary 4.15. *For any epoch m, we have $\mu(Z_*^{m+1}) \ge \alpha\text{OPT} - 2^{-m}/8$.*

Lemma 4.16 (lower bound on $\mu(Z_i^m)$)**.** *For any epoch $m+1$, and any arm $i \in K_{m+1}$, the corresponding super-arm satisfies that $\mu(Z_i^{m+1}) \ge \alpha\text{OPT} - 2^{-m}/2$*

Proof of Theorem 1.3. Recall the definition of alpha regret and we divide it by epochs:

$$Reg = \sum_{t=1}^{T} \alpha\text{OPT} - \mu(Z_t) = \sum_{m=1}^{M}\sum_{t\in E_m} \alpha\text{OPT} - \mu(Z_t).$$

Similarly, we set the confidence parameter $\delta = 1/T$, so the case where $\mathcal{E}$ does not hold only contributes $O(1)$ to the expected regret. In this case $\lambda \le O(\log(TK))$). In the following, we assume that $\mathcal{E}$ happens.

Fixing an epoch m, which has length $\lambda|K_m|d^2 2^{2m}$, by Lemma 4.16, all the super-arms $\{Z_i^m\}_{i\in K_m}$ pulled in epoch m satisfy that $\mu(Z_i^m) \ge \alpha\text{OPT} - 2^{-(m-1)}/2$, so the regret incurred in epoch m is at most $O(\lambda|K_m|d^2 2^m)$.

Finally, we take a summation over M epochs. Note the number of epochs M satisfies that $\lambda|K_M|d^2 2^{2M} \le T$.

$$\begin{aligned}\text{Reg} &\le \sum_{m=1}^{M} \lambda|K_m|d^2 2^m \le O\left(\lambda K d^2 \cdot \sqrt{T/(\lambda K d^2)}\right)\\ &\le O(d\sqrt{KT\log(KT)})\end{aligned}$$

Next, for the oracle complexity, there are $O(K)$ calls in each epoch and at most $O(\log T)$ epochs, so the $O(K\log T)$ oracle complexity follows. □

5 EXPERIMENT

We run several experiments to test the empirical performance of our algorithm CBARBAR on CMAB-AC and MAB-AC. We compare our algorithm with the current best algorithms on each problem: HYBRID from Zimmert et al. [2019] on CMAB-AC and Tsallis-INF from Zimmert and Seldin [2021] on MAB-AC.

In practice, we made several minor changes to our CBARBAR algorithm in Algorithm 1: First, we set $\lambda = 12$ on line 1. Second, we set the base used in line 3 and line 14 to be 4 instead of 2. Third, on line 9, we also utilize statistics from super-arm other than Z_i^m to estimate μ_i.

Experiment setup: It is unclear how to implement HYBRID for more involved combinatorial family, we consider the simplest m-set problem for CMAB. We use K to denote the number of arms, d to denote the size of one super-arm. MAB problem is a special case with $d = 1$. For each problem, we set d optimal arms with mean reward $\frac{1}{2} + \Delta$ and all the other sub-optimal arms with mean reward $\frac{1}{2} - \Delta$. There are T rounds in each experiment.

We use stochastic arms with adversarial corruption in our experiment. Since it is difficult to construct truly adversarial corruption, we design two heuristics to simulate such adversarial corruption. A corruption heuristic consists of two parts: when to add corruption and how to set the corrupted reward. For the second part, our heuristic believes that, by greedy, if the adversarial decides to add corruption on one round, he should minimize the optimal arms' rewards and maximize the sub-optimal arms' rewards using the quota, so we swap the mean reward between the optimal arm and suboptimal arm on such a round. Then, we design two different heuristics to decide when to add corruption:

BEGIN: we add all corruptions at the beginning until the quota exhausts. Usually the first few rounds have larger weights in deciding which arm is the optimal, so adding corruption in the beginning has larger impact.

SUPPRESS: Since the action produced by algorithm is a distribution of all super-arms, we calculate the ratio p of selecting the optimal arm/arms. We let p_0 be the ratio of selecting the optimal arm/arms using uniform distribution. We add corruption once $p > p_0$ and continue until $p < \frac{p_0}{3}$. One intuition behind this heuristic is to only use corruption on those important rounds where there is a decent chance of selecting the optimal arm/arms. Another intuition is that the algorithm is relatively less "robust" or easy to be misled when p is small, so it is efficient to add corruption on this situation.

In our experiment, we record the larger one between two regrets under the above two heuristics. Our experiment is run on AMD EPYC 7K62 48-Core Processor. CMAB-AC algorithms are implemented by Python and MAB-AC algorithms are implemented by C++. Running time is tested for the $C = 0$ case.

5.1 EMPIRICAL EXPERIMENTS FOR CMAB-AC

We first run the experiment with $d = 3, K = 7, T = 10^7, \Delta = 0.1$. Each specific experiment setting is repeated for at least 96 times, and even up to 720 times for those settings with high standard deviation.

CMAB-AC	Time	C=0	C=6000	C=30000
HYBRID	45min	800	10544	44982
CBARBAR	5min	9816	17046	43278

Table 1: Regret and running time comparison between CBARBAR and HYBRID on CMAB-AC with $d = 3, K = 7, T = 10^7, \Delta = 0.1$ and different corruption amount

The results show that for the stochastic case ($C = 0$), HYBRID indeed performs better than our CBARBAR due to the advantage of a logarithmic factor in the theoretical regret complexity bound, and quantitatively our regret is 12 times larger than HYBRID. However, as the corruption C becomes larger, two algorithms have comparable empirical regret because both of them achieve a linear C regret term. Even for this small test case, we can see CBARBAR runs significantly faster compared with HYBRID. Our CBARBAR has running time complexity $O(Td \log K + Kd \log T)$, while HYBRID needs to solve a convex optimization at each time step.

Next, we run the experiment on a larger instance with $d = 3, K = 100, T = 3 \cdot 10^7, \Delta = 0.3$. We terminate HYBRID after 12 hours. Each specific experiment setting is repeated by 96 times.

For larger problem instance, the running time of CBARBAR only increases slightly, while the running time of HYBRID

CMAB-AC	Time	C=0	C=30000	C=120000
HYBRID	>12h	N/A	N/A	N/A
CBARBAR	15min	99227	170748	358256

Table 2: Regret and running time comparison between CBARBAR and HYBRID on CMAB-AC with $d = 3, K = 100, T = 3 \cdot 10^7, \Delta = 0.3$ and different corruption amount

increases dramatically.

5.2 EMPIRICAL EXPERIMENT FOR MAB-AC

We run the experiments with $K = 10, T = 10^7, \Delta = 0.1$ and $K = 100, T = 10^7, \Delta = 0.3$, and report the average of 512 runs.

MAB-AC	Time	C=0	C=6000	C=30000
Tsallis-INF	6.3s	395	12974	62167
CBARBAR	2.2s	6967	16851	60262

Table 3: Regret and running time comparison between CBARBAR and Tsallis-INF on MAB-AC with $d = 1, K = 10, T = 10^7, \Delta = 0.1$ and different corruption amount

MAB-AC	Time	C=0	C=6000	C=30000
Tsallis-INF	41.3s	1702	16604	65771
CBARBAR	2.2s	30650	42116	96005

Table 4: Regret and running time comparison between CBARBAR and Tsallis-INF on MAB-AC with $d = 1, K = 100, T = 10^7, \Delta = 0.3$ and different corruption amount

Our CBARBAR's regret is 17 times larger than Tsallis-INF for the stochastic case and is only somewhat larger than Tsallis-INF for large C case. For running time, CBARBAR is much faster than Tsallis-INF and the difference becomes more evident for large K. Although Tsallis-INF needs to solve a constrained optimization problem at each step, Newton's method is relatively efficient. CBARBAR enjoys running time complexity $O(T \log K + K \log T)$ and Tsallis-INF's is $O(N * TK)$, where N is the number of iterations performed by Newton's method. The running time improvement may be apparent for large K.

6 CONCLUSION

We consider the stochastic combinatorial semi-bandit problem with adversarial corruptions CMAB-AC. We provide simple combinatorial algorithms that can achieve better regret than previous combinatorial algorithms, and almost match the best known regret bound (up to logarithmic factors) achieved by convex-programming-based algorithms. Our algorithm is simpler to implement, can handle more general combinatorial families, and our analysis does not require the unique solution assumption. We also study the the stochastic combinatorial semi-bandit problem where we only get access to an approximation oracle CMAB-APX. We propose a simple algorithm that almost match the best known regret bound. Our algorithm is quite simple and has only logarithmic oracle complexity.

Acknowledgements

Xu and Li are supported in part by the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016 and the Zhongguancun Haihua Institute for Frontier Information Technology, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology.

References

Peter Auer and Chao-Kai Chiang. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *Conference on Learning Theory*, pages 116–120. PMLR, 2016.

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

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b.

Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *Conference on Learning Theory*, pages 42–1. JMLR Workshop and Conference Proceedings, 2012.

Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1659–1667, 2016.

Richard Combes, Sadegh Talebi, Alexandre Proutière, and Marc Lelarge. Combinatorial bandits revisited. In *NIPS 2015-Twenty-ninth Conference on Neural Information Processing Systems*, 2015.

Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6), 2006.

Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, 2012.

Dan Garber. Efficient online linear optimization with approximation algorithms. *Mathematics of Operations Research*, 2020.

Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory*, pages 1562–1578. PMLR, 2019.

Elad Hazan, Wei Hu, Yuanzhi Li, and Zhiyuan Li. Online improper learning with an approximation oracle. *Advances in Neural Information Processing Systems*, 31: 5652–5660, 2018.

Shinji Ito, Daisuke Hatano, Hanna Sumita, Kei Takemura, Takuro Fukunaga, Naonori Kakimura, and Ken-Ichi Kawarabayashi. Oracle-efficient algorithms for online linear optimization with bandit feedback. In *Advances in Neural Information Processing Systems*, pages 10590–10599, 2019.

Sham M Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1088–1106, 2009.

Sayash Kapoor, Kumar Kshitij Patel, and Purushottam Kar. Corruption-tolerant bandit learning. *Machine Learning*, 108(4):687–715, 2019.

Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pages 535–543. PMLR, 2015.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Tian Lin, Jian Li, and Wei Chen. Stochastic online greedy learning with semi-bandit feedbacks. In *NIPS*, pages 352–360, 2015.

Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–122, 2018.

Laura Niss and Ambuj Tewari. What you see may not be what you get: Ucb bandit algorithms robust to ε-contamination. In *Conference on Uncertainty in Artificial Intelligence*, pages 450–459. PMLR, 2020.

Yevgeny Seldin and Gábor Lugosi. An improved parametrization and analysis of the exp3++ algorithm for stochastic and adversarial bandits. In *Conference on Learning Theory*, pages 1743–1759. PMLR, 2017.

Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *International Conference on Machine Learning*, pages 1287–1295. PMLR, 2014.

Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pages 5114–5122. PMLR, 2018.

Julian Zimmert and Yevgeny Seldin. Tsallis-inf: An optimal algorithm for stochastic and adversarial bandits, 2021.

Julian Zimmert, Haipeng Luo, and Chen-Yu Wei. Beating stochastic and adversarial semi-bandits optimally and simultaneously. In *International Conference on Machine Learning*, pages 7683–7692. PMLR, 2019.