Efficient and Optimal Policy Gradient Algorithm for Corrupted Multi-armed Bandits

Jiayuan Liu
Carnegie Mellon University
Pittsburgh, United States
jiayuan4@andrew.cmu.edu

Siwei Wang
Microsoft Research Asia
Beijing, China
siweiwang@microsoft.com

Zhixuan Fang
Tsinghua University
Beijing, China
Shanghai Qi Zhi Institute
Shanghai, China
zfang@mail.tsinghua.edu.cn

ABSTRACT

In this paper, we consider the stochastic multi-armed bandits problem with adversarial corruptions, where the random rewards of the arms are partially modified by an adversary to fool the algorithm. We apply the policy gradient algorithm SAMBA to this setting, and show that it is computationally efficient, and achieves a state-of-the-art $O(K \log T/\Delta) + O(C/\Delta)$ regret upper bound, where K is the number of arms, C is the unknown corruption level, Δ is the minimum expected reward gap between the best arm and other ones, and T is the time horizon. Compared with the best existing efficient algorithm (e.g., CBARBAR), whose regret upper bound is $O(K \log^2 T/\Delta) + O(C)$, we show that SAMBA reduces one $\log T$ factor in the regret bound, while maintaining the corruption-dependent term to be linear with C. This is indeed asymptotically optimal. We also conduct simulations to demonstrate the effectiveness of SAMBA, and the results show that SAMBA outperforms existing baselines.

KEYWORDS

Multi-armed Bandits; Corruption; Policy Gradient

ACM Reference Format:
Jiayuan Liu, Siwei Wang, and Zhixuan Fang. 2025. Efficient and Optimal Policy Gradient Algorithm for Corrupted Multi-armed Bandits. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 8 pages.

1 INTRODUCTION

Multi-armed bandits (MAB) model requires the learning policy to learn from feedback to optimize decision-making in complex and uncertain environments [9]. In this model, there are K arms, and each arm a is associated with a reward distribution $\mathcal{F}_a$. In each round $t \in [T]$, a player can choose one arm a from the K arms to pull and observe a reward $R_a \sim \mathcal{F}_a$. Denote r_a the expected reward of arm a, $a^* = \text{argmax}_a\, r_a$ the optimal arm, and $r^* = \max_a r_a$ the highest expected reward. Then we let $\Delta_a = r^* - r_a$, $\Delta = \min_{\Delta_a > 0} \Delta_a$, and define cumulative regret as the expected difference between the cumulative reward from pulling the optimal arm and the cumulative reward of the algorithm, i.e., pulling arm a once incurs a regret of Δ_a. The aim of the player is to choose arms properly to minimize cumulative regret.

MAB captures the basic tradeoff between exploration and exploitation in online learning, and is widely adopted in real-world applications, e.g., when a news website picks an arriving article header to show to maximize the users' clicks, and when an investor chooses a stock for investment to maximize the total wealth [18]. Because of this, there is abundant research related to MAB problems, which proposes solutions including Upper Confidence Bound [4], Active Arm Elimination [8], Thompson Sampling [2], etc.

However, in some applications, such as a recommendation system that suggests restaurants to customers, while most inputs follow a stochastic pattern from a fixed distribution, some inputs would be corrupted, e.g., injected by fake reviews from the restaurant's competitors [16]. In addition, in machine learning applications, data may be imperfect or manipulated. Studying corruption bandits helps develop learning algorithms that remain effective even when data is corrupted, which is useful in fields such as federated learning [7] and distributed sensor networks [6].

Corruption also exists in other applications such as online advertising and cybersecurity. In this paper, we consider the stochastic multi-armed bandits problem with adversarial corruptions, where the rewards of the arms are partially modified by an adversary to fool the algorithm [10, 14]. At each time step t, before an arm is pulled, the adversary can make corruptions, i.e., shift the expected reward of any arm a to any corrupted value with cost $\max_a |r_a - r'_a(t)|$, where $r'_a(t)$ is the expected reward of arm a after such corruption. The only constraint for the adversary is that his total cost cannot exceed corruption level C, i.e., $\sum_t \max_a |r_a - r'_a(t)| \le C$, while this C also keeps unknown to the player.

Existing algorithms pay a high cost for robustness against adversarial corruptions. The current state-of-the-art *combinatorial algorithms* (i.e., those containing solely combinatorial operations such as enumeration and basic calculations, and with computational cost in each time step independent of the time horizon T) exhibit a regret upper bound of $O(\log^2 T + C)$ for corrupted bandits, e.g., Xu and Li [21]. This implies that the algorithm's regret is not tight (i.e., it has one more $\log T$ factor compared to the $\Omega(\log T)$ regret lower bound [16]), and suffers an $O(\log^2 T)$ regret even if there is no corruption.

In this paper, our aim is to solve the above challenge and find efficient bandit algorithms that can handle adversarial corruptions

Corresponding authors: Siwei Wang (siweiwang@microsoft.com), Zhixuan Fang (zfang@mail.tsinghua.edu.cn).

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

without overhead, i.e., the regret upper bound approaches the bound in standard MAB as corruption level C decays to zero. Recent work [20] proposes a combinatorial algorithm Stochastic Approximation Markov Bandit Algorithm (SAMBA) to solve the standard MAB problem. In this paper, we employ this algorithm to address the corrupted bandits problem.

We are interested in SAMBA due to its adoption of a Markovian policy, in which the distribution of the chosen arm at step $t+1$ depends solely on the distribution of the chosen arm at step t, as well as the chosen arm and the observation at step t. This is a desired property for corrupted bandits and reduces the complexity of the analysis. Based on such a property, we show that SAMBA achieves a regret upper bound of $O(\log T + C)$, which is a major improvement compared to the $O(\log^2 T + C)$ regret upper bound of the best existing combinatorial algorithms. Meanwhile, our regret upper bound matches i) the $\Omega(\log T)$ regret lower bound when there is no corruption; and ii) the $\Omega(C)$ regret lower bound with corruption level C. This shows that SAMBA is indeed asymptotically optimal. We also conduct experiments to compare the performance of SAMBA with other existing baselines, whose results demonstrate the empirical effectiveness of SAMBA.

1.1 Our Main Contribution

The aim of this research is to develop a combinatorial anti-corruption multi-armed bandits algorithm that is fast, easily implementable, and has a better performance guarantee than existing works. We employ SAMBA algorithm to tackle the corrupted bandits problem, marking the inaugural utilization of a policy gradient algorithm in this scenario.

Our primary contribution lies in three aspects. Firstly, we are the first to employ and analyze combinatorial policy gradient algorithms in the context of corrupted bandits. Secondly, we theoretically prove SAMBA's exceptional performance in the corrupted bandits setting. In addition, we demonstrate the empirical performance advantage of SAMBA over existing baselines. Our analysis is groundbreaking, as it is the first to prove that a combinatorial algorithm can achieve the optimal regret upper bound in the corrupted bandits setting. This result highlights the significance of our work in advancing the understanding and practicality of combinatorial approaches for dealing with corruption in bandit problems.

1.2 Related Work

Lykouris et al. [16] is the first to consider stochastic bandits with adversarial corruptions. They propose Fast-Slow Active Arm Elimination Race algorithm that achieves a high probability regret upper bound of $O\big(KC\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\big)$ when C is known, and Multi-layer Active Arm Elimination Race algorithm that achieves the same high probability regret upper bound when C is unknown. They also show that a linear degradation to the total corruption amount C is the best one can do, i.e., with corruption level C, any algorithm must suffer a regret lower bounded by $\Omega(C)$.

Gupta et al. [9] introduces a new algorithm called BARBAR, which reduces the regret upper bound to $O\big(KC+\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\big)$ when C is unknown. Liu et al. [15], Xu and Li [21] make some further improvements on BARBAR, providing the solutions under cooperative bandits setting [15] and combinatorial bandits setting [21]. In addition, they reduce the $O(KC)$ term in the regret upper bound to $O(C)$, i.e., the regret upper bound of Liu et al. [15] is $O\big(C+\frac{K\log^2 T}{\Delta}\big)$, and the regret upper bound for Xu and Li [21] is $O\big(C+\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\big)$.

Except for the combinatorial algorithms that come from traditional bandit literature, there is another type of non-combinatorial algorithms that come from "best-of-both-worlds (BOBW)" literature. In BOBW, the algorithm needs to ensure good regret performance under both the stochastic scenario and the totally adversarial scenario [5]. Some of the BOBW algorithms also perform well in corrupted bandits. For example, Zimmert and Seldin [24] uses Tsallis-INF algorithm with Tsallis entropy regularization and Jin and Luo [12] uses Follow-the-Regularized-Leader (FTRL) method [3, 22, 23] with a novel hybrid regularizer to solve the corrupted bandits problem, both of which are based on online mirror descent (OMD) method and lead to a regret upper bound of $O\big(C+\frac{K\log T}{\Delta}\big)$[1]. However, OMD algorithms are not combinatorial and require more computational power than combinatorial algorithms such as BARBAR and SAMBA. Specifically, in each time step, the OMD algorithms need to solve a convex optimization problem, and the regret analysis is based on the actions corresponding to the optimal points. In practice, one can only use optimization algorithms (e.g., gradient descent) to look for near-optimal points. Since there are totally T convex optimization problems to solve, to guarantee a similar regret bound, the gap between the approximate points and the optimal points should depend on T (e.g., $\frac{1}{T}$). Therefore, the complexity required for each step also depends on T. As a comparison, combinatorial algorithms (e.g., BARBAR and SAMBA) only need $O(K)$ additions or multiplications in each time step.

Recent findings show that sampling algorithms can be more computationally efficient than optimization algorithms [17, 19]. Honda et al. [11] incorporates this idea and uses follow-the-perturbed-leader-based (FTPL) method [1, 13] which replaces the procedure of solving the optimization problem in FTRL by multiple samplings and resamplings. However, FTPL does not completely solve the complexity challenge. Specifically, though the expected computation cost at each time step is $O(K)$, the variance of computation cost at each time step is $O(T)$, making it still non-combinatorial.

An overall comparison of different algorithms is given in Table 1. Note that the state-of-the-art combinatorial algorithms have $O(\log^2 T + C)$ regret upper bounds, while only non-combinatorial algorithms can achieve $O(\log T + C)$ regret upper bound. Our analysis shows that a combinatorial algorithm, SAMBA, can achieve an $O(\log T + C)$ regret upper bound, which matches the regret lower bound for the corrupted bandits.

2 PRELIMINARIES

2.1 Multi-armed Bandits

A multi-armed bandits instance is a tuple $(\mathcal{A}, \boldsymbol{r}, T)$. Here, i) $\mathcal{A} = \{1, 2, \cdots, K\}$ is the set of arms and K is the number of arms; ii)

[1]Though they claimed that their regret upper bound is $O\big(\frac{K\log T}{\Delta}+\sqrt{\frac{CK\log T}{\Delta}}\big)$, we emphasize that the definition of their regret is not the same as the one we and other corrupted bandits works use. In fact, there is an $O(C)$ gap between the two kinds of regret.

Table 1: Comparison of corrupted bandits algorithms

Algorithm	Known C	Combinatorial	Regret Bound
Fast-Slow AAE Race [16]	Yes	Yes	$O\left(KC\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\right)$
Multi-Layer AAE Race [16]	No	Yes	$O\left(KC\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\right)$
BARBAR [9]	No	Yes	$O\left(KC+\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\right)$
Cooperative Bandit Algorithm Robust to Adversarial Corruptions [15]	No	Yes	$O\left(C+\frac{K\log^2 T}{\Delta}\right)$
CBARBAR [21]	No	Yes	$O\left(C+\sum_{i\neq i^*}\frac{\log^2 T}{\Delta_i}\right)$
Tsallis-INF [24], FTRL [12], FTPL [11]	No	No	$O\left(C+\frac{K\log T}{\Delta}\right)$
SAMBA [20] (with our analysis)	**No**	**Yes**	$\boldsymbol{O\left(\frac{C}{\Delta}+\frac{K\log T}{\Delta}\right)}$
Regret Lower Bound [16]	–	–	$\Omega\left(C+\frac{K\log T}{\Delta}\right)$

$\boldsymbol{r}=[r_1,\cdots,r_K]\in[0,1]^K$ are the corresponding expected rewards of the K arms; and iii) T is the time horizon. At each time step $t\leq T$, the player must choose an arm $a(t)\in\mathcal{A}$ to pull. After that, he will receive a random reward (feedback) $R_{a(t)}(t)$. In this paper, for simplicity of analysis, we focus on the case that the random rewards are Bernoulli, i.e., $R_{a(t)}(t)$ are drawn from a Bernoulli distribution with mean $r_{a(t)}$ independently. Our results can be easily extended to the general bounded-reward case.

The player can use the history information $\mathcal{H}_{t-1}$ to generate a random distribution $p(t)$ on the action set $\mathcal{A}$, and then draw his choice $a(t)$ from $p(t)$, where $\mathcal{H}_{t-1}=\{(a(\tau),R_{a(\tau)}(\tau))\}_{\tau\leq t-1}$ are the previous arm-reward pairs. The goal of the player is to choose the random distribution $p(t)$ properly to maximize the cumulative reward, or minimize the cumulative regret. The cumulative regret is defined as the expected reward gap between the real gain and the best one can do, i.e., always selecting the arm with the highest expected reward. By denoting $a^*=\arg\max_{a\in\mathcal{A}}r_a$ and $r^*=r_{a^*}$, the cumulative regret of policy π equals $Rg(T):=r^*T-\mathbb{E}\left[\sum_{t=0}^{T-1}\sum_{a\in\mathcal{A}}I_a(t)R_a(t)\right]=\sum_{a:a\neq a^*}(r^*-r_a)\mathbb{E}\left[\sum_{t=0}^{T-1}p_a(t)\right]$, where $I_a(t)=1$ if and only if arm a is pulled in round t. Let $\Delta_a=r^*-r_a$ and assume that $\Delta_a>0$ for any $a\neq a^*$ (i.e., there is one unique optimal arm), we can write the cumulative regret as $Rg(T)=\sum_{a:a\neq a^*}\Delta_a\mathbb{E}\left[\sum_{t=0}^{T-1}p_a(t)\right]$.

2.2 Corrupted Bandits

In a corrupted bandits instance, except for the basic components of the bandit model, there is another adversary who aims to fool the player. Specifically, the adversary is aware of the history information as well as the learning policy of the player. However, he cannot obtain the same randomness as used by the user. That is, the adversary knows the random distribution $p(t)$ of how the player will choose arm $a(t)$, but does not know the exactly chosen arm $a(t)$. Based on this knowledge, at each time step t, the adversary can change the expected reward of each arm from $\boldsymbol{r}$ to $\boldsymbol{r}'(t)$, at a cost of $c(t)=\max_{a\in\mathcal{A}}|r_a-r'_a(t)|$. In this case, if the player chooses to pull the arm $a(t)$, then his random reward (and feedback) $R_{a(t)}(t)$ is no longer drawn from Bernoulli distribution with mean $r_{a(t)}$, but from Bernoulli distribution with mean $r'_{a(t)}(t)$.

The goal of the adversary is to let the player suffer regret as much as possible, given the constraint that his total cost of corruption could not exceed the corruption level C. Here, the definition of cumulative regret is the same as classic MAB problems, i.e., we are still comparing the arms under their true expected rewards but not the corrupted expected rewards[2]. On the other hand, the goal of the player is to design algorithms such that the regret is still limited even if there is such an adversary. As in many existing works, we assume that the player does not know the corruption level C.

Note that our corruption method is slightly different from the existing literature, i.e., the adversary changes the expected reward but not the realized feedback. In fact, if we use a function to change the realized reward feedback R to $R'=f(R)$ (even for random functions) after seeing the feedback $R\sim\mathcal{D}$, we can get a new reward distribution $\mathcal{D}'$ where $R'\sim\mathcal{D}'$. Hence, our approach (directly changing the reward distribution to $\mathcal{D}'$) is more general than the classic approach. Moreover, the constraint on the adversary in the classic approach is $\sum_t|R_a(t)-R'_a(t)|\leq C$, while in our approach it is $\sum_t\left|\mathbb{E}[R_a(t)-R'_a(t)]\right|\leq C$, which is looser than the former one. As a result, the adversary in our approach could be more powerful than the classic one with the same C.

2.3 SAMBA Algorithm

The SAMBA algorithm [20] is described in Algorithm 1. The policy is a probability distribution vector $p(t)=[p_1(t),\ldots,p_K(t)]$ from which an arm is sampled in each round, and is initialized to $p_a(1)=1/K,\forall a\in[K]$. In each round t, an arm $a(t)$ is sampled from the distribution $p(t)$. The player then pulls arm $a(t)$ and gets a reward $R_{a(t)}(t)$ (a possibly corrupted reward in corrupted bandits).

[2]Most of the existing literature uses this definition, e.g., [9, 15, 16, 21]. As for those who compare the arms under their corrupted expected rewards, e.g., [12, 24], directly adding C to their regret upper bound leads to a regret bound under our definition.

Algorithm 1 SAMBA Algorithm

Require: $\alpha \in (0,1)$
Init: $p_a(1) \leftarrow 1/K$ for $a = 1, \ldots, K$
for $t = 1$ **to** T **do**
  Update leading arm: $a_l \leftarrow \arg\max_a p_a(t)$
  Draw $a(t)$ from randomly $p(t)$, observe $R_{a(t)}(t)$
  if $a(t) = a_l$ **then**
    for all $a' \neq a_l$ **do**
      $p_{a'}(t+1) \leftarrow p_{a'}(t) - \alpha \frac{p_{a'}^2(t) R_{a(t)}(t)}{p_{a_l}(t)}$
    end for
  else
    $p_{a(t)}(t+1) \leftarrow p_{a(t)}(t) + \alpha p_{a(t)}(t) R_{a(t)}(t)$
    for all $a \notin \{a(t), a_l\}$ **do**
      $p_a(t+1) \leftarrow p_a(t)$
    end for
  end if
  $p_{a_l}(t+1) \leftarrow 1 - \sum_{a' \neq a_l} p_{a'}(t+1)$
end for

The probabilities of all the non-leading arms $\forall a \neq a_l$ in $p(t)$ will be updated after the player observes the reward $R_{a(t)}(t)$ according to

$$p_a(t+1) \leftarrow p_a(t) + \alpha p_a(t)^2 \left(\frac{R_a(t) I_a(t)}{p_a(t)} - \frac{R_{a_l}(t) I_{a_l}(t)}{p_{a_l}(t)} \right) \quad (1)$$

where a_l is the current leading arm, i.e., the arm with the highest probability $p_a(t)$. Note that the update scheme applies importance sampling because the player can only observe the reward from the pulled arm. After updating the probability of the non-leading arms, the leading arm's probability is given by $p_{a_l}(t+1) = 1 - \sum_{a' \neq a_l} p_{a'}(t+1)$.

Walton and Denisov [20] prove that SAMBA achieves an $O(\log T)$ regret upper bound in the classic MAB model, which is stated in the following fact.

Fact 1 (Walton and Denisov [20]). *If constant $\alpha < \frac{\Delta}{r^* - \Delta}$, then the SAMBA algorithm for multi-armed bandits problem without corruption ensures a regret $Rg(T) \leq \frac{K}{\alpha\Delta} \log T + Q_0 = O\left(\frac{K}{\Delta} \log T\right)$, where $Q_0 := \sum_{t=0}^{\infty} \mathbb{P}(p_{a^*}(t) \leq \frac{1}{2}) < \infty$ is proved in Walton and Denisov [20] to be a finite constant.*

3 REGRET OF SAMBA UNDER CORRUPTED BANDITS

Though SAMBA is not specially optimized for the corrupted bandits setting, we surprisingly find out that it works very well even when there is an adversary who tries to fool the algorithm by corruptions.

Theorem 2. *If constant $\alpha < \frac{\Delta}{r^* - \Delta}$, then the SAMBA algorithm for multi-armed bandits problem with adversarial corruption level C ensures a regret*

$$Rg(T) = O\left(\frac{K}{\Delta} \log T + \frac{C}{\Delta}\right).$$

Compared with the existing results, SAMBA achieves a more favorable regret bound by reducing one $\log T$ factor in the existing results (e.g. the $O(\frac{K}{\Delta} \log^2 T + C)$ bound in Liu et al. [15], Xu and Li [21]), resulting in improved performance as the time horizon T increases. In addition, SAMBA still maintains a linear dependence on the unknown corruption level C. This linear term ensures that SAMBA performs well even in the presence of high corruption levels. Due to the space limit, we only provide some technique highlights here, and defer the complete proof to Appendix A in the extended version of this paper, available online.

As we have mentioned before, the reason that we are interested in SAMBA is that it is a Markovian policy, in which the influence of one corruption only appears once. In fact, this is a very important and desired property to deal with corruptions, and most existing anti-corruption algorithms are trying to achieve this property. For example, BARBAR [9] divide the game into $\log T$ episodes, and only let the corruption in the i-th episode influence the arm chosen in the $(i+1)$-th episode. With this property, we only need to bound the "sudden" impact of a corruption, and this substantially reduce the complexity of analysis.

Another good property we found in SAMBA is that the "sudden" impact of a corruption scales linearly with the corruption cost. Roughly speaking, if there is a corruption with cost $c(t)$ at time step t and no corruptions after t, then it only requires about $\Theta(c(t))$ steps to counteract its influence, i.e., the probability distribution $p(t + d \cdot c(t))$ for some constant d becomes close to $p(t)$ (the probability distribution before corruption) as the corruption effect is mostly counteracted in $d \cdot c(t)$ steps. Then, by the Markovian property of SAMBA, we could imagine that the regret incurred by corruption is approximately the regret in the next $d \cdot c(t)$ steps, and hence also scales linearly with $c(t)$. In this way, we can finally show that the corruption dependent term of SAMBA is linear with C.

To better understand the above ideas, we first briefly recall how the analysis (without corruption) in Walton and Denisov [20] works. They divide the learning procedure into two cases: i) the case that $p_{a^*} \geq 1/2$; and ii) the case that $p_{a^*} < 1/2$. Our analysis on SAMBA for the corrupted bandits problem also follows these two cases.

3.1 The case when $p_{a^*} < 1/2$

When there is no corruption, Walton and Denisov [20] consider the case where the optimal arm a^* is not the leading arm a_l, and use $\mathbb{E}[p_{a^*}^{-1}(t)]$ to capture the trajectory of how $p_{a^*}(t)$ changes during the learning procedure. Specifically, when a^* is not a_l, from some calculations according to SAMBA's policy update rule, one can show that

$$p_{a^*}^{-1}(t+1) = \begin{cases} p_{a^*}^{-1}(t) - \dfrac{\alpha}{1+\alpha} p_{a^*}^{-1}(t) & \text{w.p. } r^* p_{a^*}(t) \\ p_{a^*}^{-1}(t) + \alpha \dfrac{p_{a^*}^{-1}(t)}{p_{a_l}(t) p_{a^*}^{-1}(t) - \alpha} & \text{w.p. } r_{a_l} p_{a_l}(t) \\ p_{a^*}^{-1}(t) & \text{otherwise} \end{cases}$$

Thus, when there is no corruption,

$$\begin{aligned} \mathbb{E}[p_{a^*}^{-1}(t+1) | H(t)] - p_{a^*}^{-1}(t) =& \alpha r_{a_l} \frac{p_{a_l}(t) p_{a^*}^{-1}(t)}{p_{a_l}(t) p_{a^*}^{-1}(t) - \alpha} - \frac{\alpha r^*}{1+\alpha} \\ \leq& \alpha(r^* - \Delta)(1+\epsilon) - \frac{\alpha r^*}{1+\alpha} \end{aligned} \quad (2)$$

where the last inequality holds because for the leading arm, $p_{a_l}(t) > 1/K$ and $r_{a_l} \leq r^* - \Delta$, and the constant $\epsilon > 0$ is chosen to satisfy

$(1+\epsilon)(1+\alpha) < \frac{r^*}{r^*-\Delta}$. Such $\epsilon > 0$ must exist because $\alpha < \frac{\Delta}{r^*-\Delta}$. Let constant $\xi := \alpha\frac{r^*}{1+\alpha} - \alpha(r^*-\Delta)(1+\epsilon) > 0$, we can get

$$\mathbb{E}\Big[p_{a^*}^{-1}(t+1)\Big|H(t)\Big] - p_{a^*}^{-1}(t) \le -\xi. \tag{3}$$

Note that at the beginning of the algorithm, $\mathbb{E}[p_{a^*}^{-1}(0)] = 1/K^{-1} = K$. When $p_{a^*}^{-1}(t) \le 2$, the arm a^* must be the leading arm. Hence, after at most $\lceil\frac{K-2}{\xi}\rceil$ steps, $\mathbb{E}[p_{a^*}^{-1}(t)]$ can become small enough, which results in $p_{a^*}(t)$ being large enough and a^* becoming the leading arm. Furthermore, if a^* again becomes a non-leading arm, say at time t', then $\mathbb{E}[p_{a^*}^{-1}(t')]$ is likely to be smaller than $\mathbb{E}[p_{a^*}^{-1}(0)]$ because in expectation the probability of sampling non-optimal arms $p_a(a \ne a^*)$ would be updated to a smaller value then. Thus, from the Markov property, the expected number of steps needed for a^* to become the leading arm again is smaller than the number of steps needed in the first time. The same reasoning applies to the future "non-leading to leading" transitions. From such intuition, they prove the regret that occurs when $p_{a^*} < \frac{1}{2}$ (not only when a^* is a non-leading arm) can be upper bounded by some constant Q_0, referring to [20] for details.

Now let's consider what happens if there is an adversary to deploy corruptions. We also consider the case where a^* is not the leading arm first. In such case, $p_{a^*}^{-1}(t+1)$ is updated as

$$\begin{cases} p_{a^*}^{-1}(t) - \dfrac{\alpha}{1+\alpha}p_{a^*}^{-1}(t) & \text{w.p. } r'_{a^*}(t)p_{a^*}(t) \\ p_{a^*}^{-1}(t) + \alpha\dfrac{p_{a^*}^{-1}(t)}{p_{a_l}(t)p_{a^*}^{-1}(t)-\alpha} & \text{w.p. } r'_{a_l}(t)p_{a_l}(t) \\ p_{a^*}^{-1}(t) & \text{otherwise} \end{cases}$$

Then, we can derive

$$\begin{aligned} &\mathbb{E}[p_{a^*}^{-1}(t+1)|H(t)] - p_{a^*}^{-1}(t) \\ \le&\alpha(r_{a_l}+c(t))\frac{p_{a_l}(t)p_{a^*}^{-1}(t)}{p_{a_l}(t)p_{a^*}^{-1}(t)-\alpha} - \frac{\alpha(r^*-c(t))}{1+\alpha} \\ \le&\alpha(r^*-\Delta+c(t))(1+\epsilon) - \frac{\alpha(r^*-c(t))}{1+\alpha} \\ =&-\xi+\alpha c(t)\big(1+\epsilon+\frac{1}{1+\alpha}\big) \end{aligned}$$

where the first inequality is because $r'_{a_l}(t) \le r_{a_l} + c(t)$ and $r_{a^*}(t) \ge r_{a^*} - c(t)$. We can see that, except for the regular bias $-\xi$, the corruption $c(t)$ can increase $\mathbb{E}[p_{a^*}^{-1}(t)]$ by at most $\alpha c(t)\big(1+\epsilon+\frac{1}{1+\alpha}\big)$. Hence, one can imagine that if the total corruption level is upper bounded by C, then we need to run the algorithm for an extra number of $O(\frac{C}{\xi})$ time steps to counteract the influence of corruption. Here, we omit the technical details, which are relegated to Appendix A.

3.2 The case when $p_{a^*} \ge 1/2$

On the other hand, if $p_{a^*} \ge 1/2$, then a^* must be the leading arm. In this case, [20] uses $\mathbb{E}[p_a(t)]$ to capture the trajectory of how $p_a(t)$ changes during the learning procedure for each arm $a \ne a^*$. Specifically, one can show that for $a \ne a^*$,

$$\mathbb{E}[p_a(t+1) - p_a(t)|H(t)] \le \alpha p_a(t)^2(r_a - r_{a^*}) \le -\alpha\Delta p_a(t)^2. \tag{4}$$

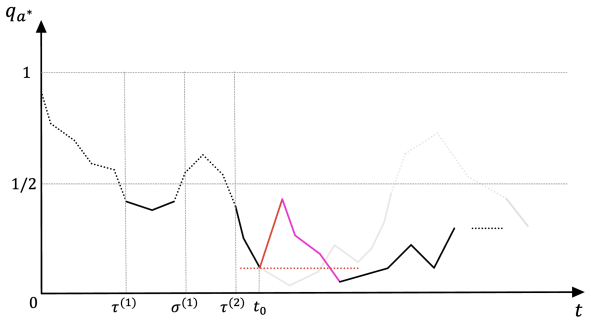


Figure 1: Recovery process.

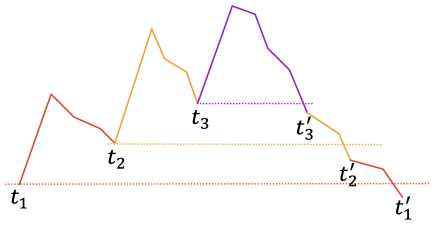


Figure 2: Consecutive corruptions.

Let $q_{a^*} := 1 - p_{a^*} = \sum_{a:a\ne a^*} p_a$. From (4) and Jensen's inequality,

$$\mathbb{E}[q_{a^*}(t+1)|H(t)] - q_{a^*}(t) \le \sum_{a:a\ne a^*} -\alpha\Delta p_a(t)^2 \le -\frac{\alpha\Delta}{K}q_{a^*}(t)^2. \tag{5}$$

Thus, $\mathbb{E}[q_{a^*}(t)]$ is in the same order as $\frac{K}{2K+\alpha\Delta t}$ (whose trajectory can be easily verified to satisfy the above equation). Hence, by taking the sum, the regret that occurs when $p_{a^*} \ge 1/2$ is upper bounded by $O(\frac{K}{\alpha\Delta}\log T)$. Details can be found in Appendix A.

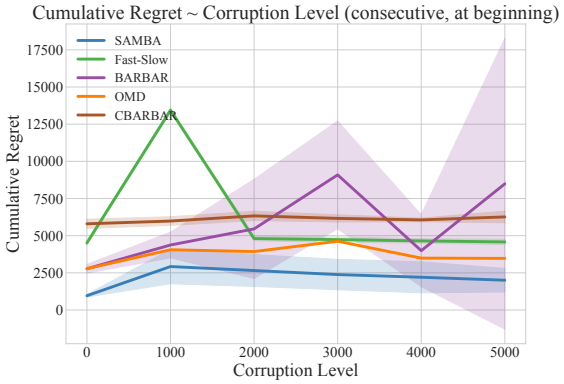
Now, we consider the case where there are adversarial corruptions. When $p_{a^*} \ge 1/2$, then a^* is the leading arm and for any $a \ne a^*$, we have

$$\begin{aligned} \mathbb{E}[p_a(t+1) - p_a(t)|H(t)] &\le \alpha p_a(t)^2\Big((r_a + c(t)) - (r_{a^*} - c(t))\Big) \\ &\le \alpha(2c(t) - \Delta)p_a(t)^2. \end{aligned}$$

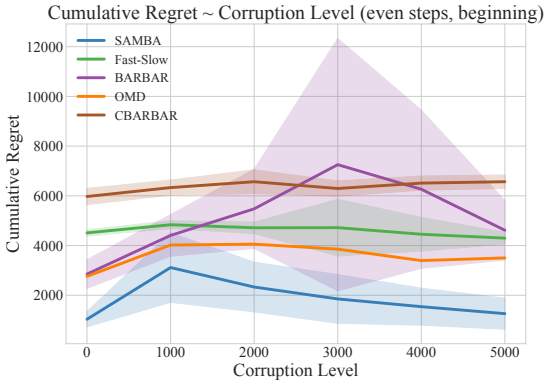
That is, except for regular bias $-\Delta_a p_a^2(t)$, the corruption $c(t)$ can increase $\mathbb{E}[p_a(t)]$ for at most $2\alpha c(t)p_a^2(t)$. However, this increase is not a constant, and one cannot directly obtain how many time steps are needed to counteract the influence of corruption. The trick here is to notice that after corruption, p_a becomes larger, and hence its decreasing rate αp_a^2 becomes larger than $\alpha p_a^2(t_c)$ before it fully recovers from the corruption, where t_c is the time step that the corruption occurs. Formally, we first define the recovery process as follows.

Definition 3 (Recovery process). *The recovery process of a corruption at time t_c is a time interval $[t_c+1, t'_c]$ on process $\{q_{a^*}(t)\}$ such that t'_c is the first time step satisfying $t'_c \ge t_c + 1$ and $q_{a^*}(t'_c) \le q_{a^*}(t_c)$.*

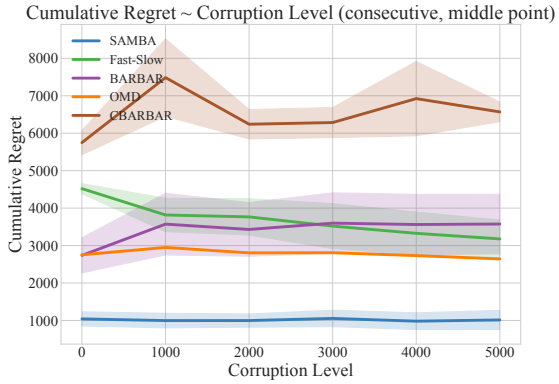
Roughly speaking, the recovery process of corruption at time t_c is the time steps required to let $q_{a^*}(t)$ fall below $q_{a^*}(t_c)$.



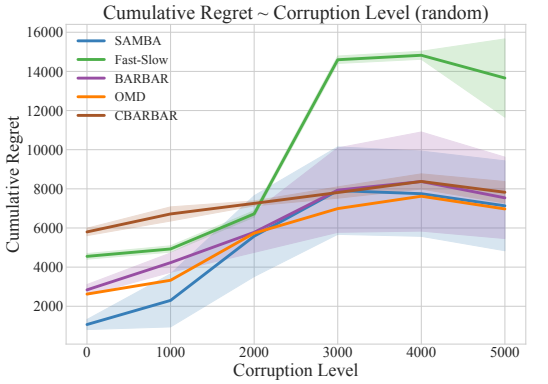
(a) Corruption scheme 1: consecutive, at beginning.



(b) Corruption scheme 2: even steps, at beginning.



(c) Corruption scheme 3: consecutive, in the middle.



(d) Corruption scheme 4: at random steps.

Figure 3: Comparison of different algorithms: the cumulative regrets under different corruption levels and different corruption schemes. SAMBA achieves the lowest cumulative regret in most settings, particularly outperforming baselines when $C = 0$, demonstrating its $O(\log T)$ regret versus $O(\log^2 T)$ for others. However, as corruption C increases, SAMBA's advantage diminishes, consistent with its regret bound of $O(C + \log T)$, while OMD shows worse performance due to its high complexity and large constant factors.

If there is a large corruption in only one step, say step t_0 with corruption level $c(t_0) > \Delta/4$, then $\mathbb{E}[q_{a^*}]$ may increase after t_0 and subsequently gradually decrease, as shown in Figure 1. What we want to do is to upper bound the expected number of steps during the recovery process after corruption $c(t_0)$ (colored magenta in Figure 1). Here we use the optional stopping theorem to give such a bound. Let $\phi = \min\{t > t_0 : q_{a^*}(t) \le q_{a^*}(t_0)\}$. When $t_0 < t \le \phi$, it holds that $q_{a^*}(t) \ge q_{a^*}(t_0)$. Then,

$$\mathbb{E}[q_{a^*}(t+1)|H(t)] - q_{a^*}(t) \le -\frac{\alpha\Delta}{2K} q_{a^*}(t)^2 \le -\frac{\alpha\Delta}{2K} q_{a^*}(t_0)^2.$$

Thus, $\{q_{a^*}(t)|t > t_0\}$ is a supermartingale. From the optional stopping theorem,

$$\mathbb{E}[q_{a^*}(\phi \wedge t)] + \frac{\alpha\Delta}{2K} q_{a^*}(t_0)^2 \mathbb{E}[\phi \wedge t]$$
$$\le \mathbb{E}[q_{a^*}(\phi \wedge (t_0+1))] + \frac{\alpha\Delta}{2K} q_{a^*}(t_0)^2 \mathbb{E}[\phi \wedge (t_0+1)].$$

Here, $\wedge$ denotes the pairwise minimum. Then, applying the monotone converge theorem, we get

$$\mathbb{E}[\phi - t_0 - 1] \le \lim_{t\to\infty} \mathbb{E}[\phi \wedge t] - \mathbb{E}[\phi \wedge (t_0+1)]$$
$$\le \frac{2K}{\alpha\Delta q_{a^*}(t_0)^2} (\mathbb{E}[q_{a^*}(t_0+1)] - \mathbb{E}[q_{a^*}(\phi)]). \tag{6}$$

Therefore,

$$\mathbb{E}[\phi - t_0] \le \frac{2K}{\alpha\Delta q_{a^*}(t_0)^2} \left((2c(t) - \Delta) \frac{\alpha}{K} q_{a^*}(t_0)^2 + \frac{\alpha\Delta}{2K} q_{a^*}(t_0)^2 \right) + 1$$
$$= \frac{4c(t)}{\Delta}. \tag{7}$$

If there are consecutive corruptions (other corruptions come before recovery from the previous corruption), then the total extra regret incurred by these corruptions is upper bounded by the regret calculated by considering these corruption steps separately from "inner" corruptions to "outer" corruptions. Here we use Figure 2 as an example. Corruptions are made at the time steps t_1, t_2, t_3.

We first deal with $c(t_3)$, then $c(t_2)$, and finally $c(t_1)$. The length of the recovery process for $c(t_3)$ (colored purple) can be bounded directly by the previous derivation. After dealing with $(t_3, t_3']$, we can remove this interval and combine the rest together. Then, we consider the interval $(t_2, t_3] \cup (t_3', t_2']$ as a whole and apply the optional stopping theorem to it, which holds because $q_{a^*}(t_3') \leq q_{a^*}(t_3)$. The same analysis holds for $c(t_1)$ (details can be found in Appendix A). In this way, we can upper bound the expected number of total recovery steps needed by $\sum_{t=0}^{T-1} \frac{4c(t)}{\Delta} = \frac{4C}{\Delta}$.

From the above analysis, we know that in both cases ($p_{a^*} < 1/2$ or $p_{a^*} \geq 1/2$), the influence of corruption C would be counteracted by $O(C)$ time steps, leading to an additional regret of $O(C)$. Therefore, along with Fact 1, we can get the final regret upper bound as $O(K \log T + C)$. The formal proofs can be found in Appendix A.

Remark 1. *In BARBAR (and CBARBAR), the algorithm is divided into* $\log T$ *phases (the length of each phase keeps doubling). To ensure that the algorithm is robust against corruptions, any arm should be pulled* $O(\log T)$ *times in each phase (so that the empirical mean is accurate enough) to detect the corruptions. This leads to a* $O(\log^2 T)$ *regret even when there is no corruption. In SAMBA, the algorithm and analysis are based on expectations but not accurate empirical means. Thus, we do not require pulling each arm* $O(\log T)$ *times in each phase to detect the influence of corruption, and instead, only a constant number of pulls in each phase is enough. In this way, we reduce one* $\log T$ *factor in the regret upper bound (note that when there is no corruption, every arm is pulled* $\Theta(\log T)$ *times, which is enough to guarantee good performance).*

4 SIMULATION

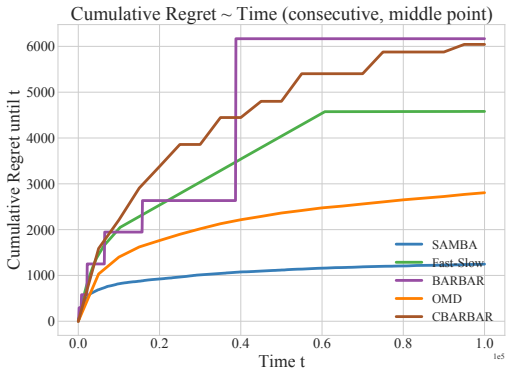
We then conduct experiments to compare the empirical performance of SAMBA with four baseline algorithms. We set the parameters to $T = 100,000$, $K = 9$ and the 9 arms are of mean rewards $0.1, 0.2, \ldots, 0.9$ respectively, $\alpha = 0.05$ in SAMBA, and $\delta = 1/T$ in Fast-Slow AAE. We test with five different corruption levels $C = 1000, 2000, \ldots, 5000$ and on four different corruption schemes:

(1) All corruption added at the beginning consecutively, i.e., at steps $0, 1, 2, 3, \ldots$;
(2) All corruption added at the the even steps at the beginning, i.e., at steps $0, 2, 4, 6, \ldots$;
(3) All corruption added concentratedly in the middle, i.e., at steps $T/4, T/4+1, T/4+2, \ldots$;
(4) All corruptions added at random steps among the first $T/10 = 10,000$ steps.

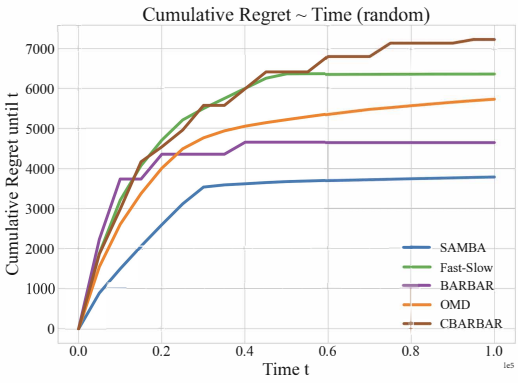
Table 2: The average single-run time (in seconds) and standard deviation (SD) of different algorithms.

	SAMBA	Fast-Slow	BARBAR	CBARBAR	OMD
Time (s)	2.2594	1.2942	0.7401	0.7823	1733.3
SD (s)	0.01422	0.01057	0.00701	0.00684	10.903

First, we compare the time costs of these corrupted bandits algorithms, and the results are shown in Table 2. We can see that the combinatorial algorithms have a much lower time cost than the OMD methods, e.g., SAMBA runs more than 500x faster than



(a) Corruption scheme 3: consecutive, in the middle.



(b) Corruption scheme 4: at random steps.

Figure 4: Comparison of different algorithms: the trend of their cumulative regret with the time when $C = 2000$ under corruption schemes 3 and 4.

OMD. This indicates the efficiency of our algorithm, i.e., it is a *combinatorial* algorithm with asymptotically optimal regret upper bound.

Then, we consider the cumulative regret under different corruption levels. The experiment result is shown in Figure 3. It shows the mean and standard deviation of the cumulative regret for the four algorithms under different settings. Each experiment runs for 100 times, except the one on the OMD algorithm which runs very slow due to its requirement of solving an optimization problem in each step. We can see that SAMBA performs the best in terms of cumulative regret in most settings. Specifically, when $C = 0$, SAMBA outperforms the baselines, which demonstrates SAMBA's $O(\log T)$ regret advantage over other algorithms' $O(\log^2 T)$ regret. However, it seems that SAMBA's performance advantage over baseline algorithms decreases as the corruption level C increases. This actually matches SAMBA's regret bound of $O(C + \log T)$. When C is large, the regret is determined primarily by C rather than the $\log T$ term. As for OMD, it has a much higher time complexity, and performs worse than SAMBA when the corruption level is small,

because some large constant factors appear in the regret upper bound.

In addition, we compare the cumulative regret of different algorithms over time. The curves for two corruption schemes and corruption level $C = 2,000$ are shown in Figure 4. Here, BARBAR and CBARBAR are implemented by selecting $n_a(m)$ times of arm a in phase m, where $n_a(m)$ is predetermined before phase m. Thus, the non-optimal arms are sampled together, leading to a step-like curve. In Figure 4a, the consecutive corruptions in the middle incur a regret surge (a large number of non-optimal arms selected after the concentrated corruptions) for BARBAR, while SAMBA actually converges quickly and tolerates the abrupt corruptions in the middle well.

We also conduct experiments with varying numbers of arms to compare the performance of different algorithms. The tested values of K (number of arms) are 6, 8, 10, 15, 20, and 30. The mean reward for each arm is uniformly distributed in the range $[0, 1]$. The mean cumulative regrets (with $T = 100,000$) are summarized in Table 3. The corruption level is set to $C = 3,000$, with corruption scheme 3.

Table 3: Comparison of the mean cumulative regret under different algorithms and different number of arms (K), with corruption level 3000 and corruption scheme 3.

Algorithm	Number of Arms (K)					
	6	8	10	15	20	30
SAMBA	629.9	884.2	1054.8	1722.7	2947.4	3534.4
Fast-Slow	1473.7	2265.3	3519.8	7955.8	10269.1	12661.2
BARBAR	2010.5	8813.9	3599.2	4800.7	8675.4	10695.7
CBARBAR	4189.2	6901.2	6285.2	11874.3	12668.2	14644.0
OMD	2038.3	2839.5	3322.7	4575.6	8460.1	10189.2

The experimental results indicate that as the number of arms K increases, the cumulative regrets scale approximately linearly with K, aligning well with the theoretical bounds. In all cases, SAMBA achieves the lowest mean cumulative regret, consistently outperforming the other algorithms.

5 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we apply a policy gradient algorithm SAMBA to the stochastic multi-armed bandits problem with adversarial corruptions. Our analysis is the first result of a combinatorial algorithm that achieves an asymptotically optimal regret upper bound of $O(C + \log T)$, establishing our method as the state-of-the-art in the corrupted bandits setting. We have also conducted simulations, demonstrating that SAMBA outperforms existing baselines.

There are several directions for future work. For example, it would be interesting to generalize SAMBA as well as our analysis to the combinatorial bandit setting or linear bandit setting, and it would also be valuable to validate the algorithm's performance in real-world applications, e.g., to conduct experiments on actual systems or design large-scale simulations that capture realistic complexities.

ACKNOWLEDGMENTS

This work was done when Jiayuan Liu was an intern at Microsoft Research Asia and a student at Tsinghua University. The work of Siwei Wang is supported in part by the National Natural Science Foundation of China Grant 62106122. The work of Zhixuan Fang is supported by Tsinghua University Dushi Program and Shanghai Qi Zhi Institute Innovation Program SQZ202312.

REFERENCES

[1] Jacob D Abernethy, Chansoo Lee, and Ambuj Tewari. 2015. Fighting bandits with a new kind of smoothness. *Advances in Neural Information Processing Systems* 28 (2015).
[2] Shipra Agrawal and Navin Goyal. 2017. Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)* 64, 5 (2017), 1–24.
[3] Jean-Yves Audibert, Sébastien Bubeck, et al. 2009. Minimax Policies for Adversarial and Stochastic Bandits.. In *COLT*, Vol. 7. 1–122.
[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47 (2002), 235–256.
[5] Sébastien Bubeck and Aleksandrs Slivkins. 2012. The best of both worlds: Stochastic and adversarial bandits. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 42–1.
[6] Gang Chen and Yaoyao Zhou. 2023. Dynamic Estimation Over Distributed Sensing Network With Communication Delays. *IEEE Transactions on Industrial Informatics* (2023).
[7] YAO Duanyi, Songze Li, XUE Ye, and Jin Liu. 2023. Constructing Adversarial Examples for Vertical Federated Learning: Optimal Client Corruption through Multi-Armed Bandit. In *The Twelfth International Conference on Learning Representations*.
[8] Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. 2006. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. *Journal of machine learning research* 7, 6 (2006).
[9] Anupam Gupta, Tomer Koren, and Kunal Talwar. 2019. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory*. PMLR, 1562–1578.
[10] Jiafan He, Dongruo Zhou, Tong Zhang, and Quanquan Gu. 2022. Nearly optimal algorithms for linear contextual bandits with adversarial corruptions. *arXiv preprint arXiv:2205.06811* (2022).
[11] Junya Honda, Shinji Ito, and Taira Tsuchiya. 2023. Follow-the-Perturbed-Leader Achieves Best-of-Both-Worlds for Bandit Problems. In *International Conference on Algorithmic Learning Theory*. PMLR, 726–754.
[12] Tiancheng Jin and Haipeng Luo. 2020. Simultaneously learning stochastic and adversarial episodic mdps with known transition. *Advances in neural information processing systems* 33 (2020), 16557–16566.
[13] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307.
[14] Sayash Kapoor, Kumar Kshitij Patel, and Purushottam Kar. 2019. Corruption-tolerant bandit learning. *Machine Learning* 108, 4 (2019), 687–715.
[15] Junyan Liu, Shuai Li, and Dapeng Li. 2021. Cooperative stochastic multi-agent multi-armed bandits robust to adversarial corruptions. *arXiv preprint arXiv:2106.04207* (2021).
[16] Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. 2018. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 114–122.
[17] Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan. 2019. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences* 116, 42 (2019), 20881–20885.
[18] Aleksandrs Slivkins et al. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12, 1-2 (2019), 1–286.
[19] Haoran Sun, Katayoon Goshvadi, Azade Nova, Dale Schuurmans, and Hanjun Dai. 2023. Revisiting sampling for combinatorial optimization. In *International Conference on Machine Learning*. PMLR, 32859–32874.
[20] Neil Walton and Denis Denisov. 2023. Regret Analysis of a Markov Policy Gradient Algorithm for Multiarm Bandits. *Mathematics of Operations Research* 48, 3 (2023), 1553–1588.
[21] Haike Xu and Jian Li. 2021. Simple combinatorial algorithms for combinatorial bandits: corruptions and approximations. In *Uncertainty in Artificial Intelligence*. PMLR, 1444–1454.
[22] Alexander Zimin and Gergely Neu. 2013. Online learning in episodic Markovian decision processes by relative entropy policy search. *Advances in neural information processing systems* 26 (2013).
[23] Julian Zimmert and Tor Lattimore. 2019. Connections between mirror descent, Thompson sampling and the information ratio. *Advances in Neural Information Processing Systems* 32 (2019).
[24] Julian Zimmert and Yevgeny Seldin. 2021. Tsallis-inf: An optimal algorithm for stochastic and adversarial bandits. *The Journal of Machine Learning Research* 22, 1 (2021), 1310–1358.