Improved Corruption Robust Algorithms for Episodic Reinforcement Learning

Yifang Chen [1] **Simon S. Du** [1] **Kevin Jamieson** [1]

Abstract

We study episodic reinforcement learning under unknown adversarial corruptions in both the rewards and the transition probabilities of the underlying system. We propose new algorithms which, compared to the existing results in (Lykouris et al., 2020), achieve strictly better regret bounds in terms of total corruptions for the tabular setting. To be specific, firstly, our regret bounds depend on more precise numerical values of total rewards corruptions and transition corruptions, instead of only on the total number of corrupted episodes. Secondly, our regret bounds are the first of their kind in the reinforcement learning setting to have the number of corruptions show up additively with respect to $\min\{\sqrt{T}, \text{PolicyGapComplexity}\}$ rather than multiplicatively. Our results follow from a general algorithmic framework that combines corruption-robust policy elimination meta-algorithms, and plug-in reward-free exploration sub-algorithms. Replacing the meta-algorithm or sub-algorithm may extend the framework to address other corrupted settings with potentially more structure.

1. Introduction

Reinforcement learning (RL) studies the problem where the learner interacts with the environment sequentially and aims to improve its decision making strategy over time. This problem has usually been modelled as a Markov Decision Process (MDP) with unknown transition functions. In this paper, we consider the classical episodic reinforcement learning with a finite horizon. Within each episode, the learner sequentially observes the current state at each stage, plays an action, receives the reward according to the current state-action pair, and then transitions to the next stage according to the underlying transition function.

The majority of the literature in learning in MDPs studies stationary environments, where the underlying unknown transition function and reward function are fixed. The rewards and the next states are independently and identically distributed given the current state and the learner's chosen action. Under this setting, the goal is to minimize the regret, which is the difference between the learner's cumulative rewards and the total rewards of the optimal policy (Brafman & Tennenholtz, 2002; Azar et al., 2017; Jin et al., 2018; Ok et al., 2018; Zanette & Brunskill, 2019; Simchowitz & Jamieson, 2019; Zhang et al., 2020). However, these techniques are vulnerable to corruptions on the rewards or the transitions. Recently, Rosenberg & Mansour (2019); Jin et al. (2020); Lee et al. (2020) gave provably efficient algorithms for the setting of adversarial rewards and fixed unknown transitions. Although their algorithms are robust to corruptions on rewards, they heavily rely on the assumption that the transitions are not corrupted.

The most relevant work is by Lykouris et al. (2020) who gave the first set of results on episodic reinforcement learning that achieve robustness to corruptions on both the rewards and the transition functions. Their regret is defined as the difference between the learner's accumulated rewards and the total rewards of the optimal fixed policy with respect to the *uncorrupted* underlying rewards and transition functions. Their algorithm is efficient and works for tabular RL and its linear variants. Unfortunately, their algorithm is not optimal in terms of the corruption level. Firstly, their corruption level C is defined as the total number of corrupted episodes. Ideally, we would like the regret to depend on more fine-grained characterizations of corruptions such as the total magnitude of corruptions on the rewards (C^r) and transition functions (C^p). Secondly, their regret bound scales $\tilde{\mathcal{O}}\left(C\sqrt{T} + C^2\right)$ in the worst case, where the corruption level C appears *both additively and multiplicatively*. They state in the paper that it is unclear whether one can obtain additive dependence alone in tabular RL. In this paper, we address this open problem.

Our contribution: To the best of our knowledge, this is the first work for the episodic tabular RL setting that obtains a regret bound that scales only *additively* with respect to the

[*]Equal contribution [1]Paul G. Allen School of Computer Science & Engineering, University of Washington. Correspondence to: Yifang Chen <yifangc@cs.washington.edu>, Simon S. Du <ssdu@cs.washington.edu>, Kevin Jamieson <jamieson@cs.washington.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

number of corruptions. This result is significant because it demonstrates that a learner can be highly robust to the corruptions, even though the magnitude and number of corrupted episodes are unknown to the learner. Our detailed contributions are shown as follows. Note that we omit all $\mathcal{S}, \mathcal{A}, H$ dependence for clarity.

- We first propose a corruption robust reward-free exploration algorithm EstAll such that for a given $\epsilon > 0$, EstAll returns $\left(\epsilon + (C^p + C^r)\epsilon^2\right)$-close estimations for all policies within a given policy set Π. If the total magnitude of corruptions to the transition functions satisfies $C^p \leq \tilde{\mathcal{O}}(1/\epsilon)$ then the algorithm requires a sample complexity of just $\tilde{\mathcal{O}}(\log|\Pi|/\epsilon^2)$. On the other hand, if $C^p > \tilde{\mathcal{O}}(1/\epsilon)$ then the algorithm will fail to complete within the expected sample complexity, providing the learner with a lower bound on the level of corruptions.
- We propose two meta-algorithms for RL inspired by the corruption robust algorithms for multi-armed bandits (Gupta et al., 2019; Bogunovic et al., 2020), both of which use EstAll as a sub-routine. The first meta-algorithm BARBAR-RL guarantees an $\tilde{\mathcal{O}}\big(\min\{\sqrt{T}, \text{PolicyGapComplexity}\} + (1+C^p)(C^p + C^r)\big)$ regret when the adversary must decide whether to corrupt the episode before seeing the learner's chosen deterministic policy at the current episode. The second meta-algorithm BrutePolicyElimination-RL guarantees an $\tilde{\mathcal{O}}\left(\sqrt{T} + (C^p + C^r)^2\right)$ regret when the adaptive adversary can decide when and how much to corrupt the episode after seeing the learner's chosen action and deterministic policy at each stage of the current episode.[1]
- Finally, comparing with (Lykouris et al., 2020) who defined the corruption level as the total number of corrupted episodes, our bounds depend on much finer definitions based on the magnitudes of corruptions on the reward and the transition (C^r and C^p).

Related Work: In addition to worst-case $\sqrt{T}$ dependent regret, Lykouris et al. (2020) also achieves an instance-dependent bound in terms of GapComplexity for tabular RL by using the UCB type algorithm and the analysis techniques developed in Simchowitz & Jamieson (2019). It remains unclear whether non-UCB type algorithms, for example, policy-elimination type methods, can also achieve the instance-dependent bound.

Other than the instance-dependent bounds, our regret bounds' dependency on $|\mathcal{S}|, |\mathcal{A}|$ and H are not optimal compared to the existing works including (Azar et al., 2017; Jin et al., 2018; Ok et al., 2018; Zanette & Brunskill, 2019; Zhang et al., 2020). Whether their techniques can be used in our framework or our policy-elimination-based methods require an entirely different analysis remains unclear.

While the literature on corrupted RL is limited, the corruption robust algorithms have been well studied in multi-arm bandits (MAB) settings, which is a special case of episodic tabular reinforcement learning with horizon $H = 1$. Corrupted MAB problems are relatively simpler than corrupted RL because we are no longer required to deal with the corruption on transition functions. In the MAB setting, obtaining a $\sqrt{T}$ regret bound with some C dependence terms, applying either additively or multiplicatively, is quite easily obtained by appealing to algorithms from the adversarial bandits literature such as the classical EXP-3 algorithm (Auer et al., 2002) that can achieve $\tilde{\mathcal{O}}(\sqrt{T})$ for adversarial rewards. Therefore, the majority of works in the corrupted MAB setting seek a Δ_a-dependent regret which scales only logarithmically with T, where Δ_a is the gap between the expected reward of action a and the optimal arm. Despite the simplified setting of corrupted MAB relative to RL, many of the techniques used in those works still provide inspiration for corrupted RL problems.

We will briefly review the most relevant corrupted MAB works here. Lykouris et al. (2018) achieves a $\tilde{\mathcal{O}}\left(\sum_{a\neq a^*}\frac{CK}{\Delta_a}\right)$ regret bound by using the multi-layer active arm elimination. Lykouris et al. (2020)'s corrupted RL work referenced above is built upon this technique. Gupta et al. (2019) achieves $\tilde{\mathcal{O}}(\sum_{a\neq a^*}\frac{1}{\Delta_a} + KC)$ by adopting a sampling strategy based on the estimated gap instead of eliminating arms permanently. One of our results is built on this technique by regarding each policy as an arm. Finally, Zimmert & Seldin (2019) achieves a near-optimal result $\tilde{\mathcal{O}}\left(\sum_{a\neq a^*}\frac{1}{\Delta_a} + \sqrt{\sum_{a\neq a^*}\frac{C}{\Delta_a}}\right)$ by using Follow-the-Regularized Leader with Tsallis Entropy. Note that their work actually solves a more difficult problem called best-of-both-worlds, which can achieve near-optimal result simultaneously for both adversarial and stochastic rewards. The similar technique has been adopted in Jin & Luo (2020), which achieves $\tilde{\mathcal{O}}(\text{GapComplexity} + \sqrt{C \cdot \text{GapComplexity}})$ when the transition function is known. Unfortunately, whether it is possible to extend such techniques to the unknown transition setting remains unclear. Besides the corrupted MAB setting, Lee et al. (2021) considers linear bandits which achieves an near-optimal result in terms of corruptions $\tilde{\mathcal{O}}\left(\min\{d\sqrt{T}, \text{GapComplexity}\} + C\right)$.

Note that all of these works presented above consider a weak adversary which must decide the corruption for each round (or episodes) before observing the learner's chosen action (or policy). Some works (e.g. (Liu & Shroff, 2019; Bogunovic et al., 2020)) consider a stronger adversary which

[1] This is a stronger adversary than the one studied in Lykouris et al. (2020).

can decide the corruption after seeing the learner's current behavior. In particular, Bogunovic et al. (2020) achieves a near-optimal regret $\tilde{\mathcal{O}}(\sqrt{dT}+Cd^{3/2}+C^2)$ for linear bandits by using arm elimination with an enlarged confidence bound. One of our results also considers this stronger adversary setting and adopts a similar technique.

Finally, our reward-free exploration sub-algorithm is based on the algorithm in Wang et al. (2020) by again using the trajectory synthesis idea. But just as in the original algorithm, this exploration sub-algorithm is inefficient. Algorithms proposed in Kaufmann et al. (2020) and Ménard et al. (2020) can efficiently achieve an ϵ-close estimation for each policy given a policy set Π when *no corruption* exists. But whether this type of algorithm can be made robust to corruptions at least as good as ESTALL remains unknown. We provide some discussion in Appendix E.

Structure of the paper: In Section 2, we formally define our settings and the regret objective. In Section 3, we describe the meta-algorithm BARBAR-RL for the non-cheated adversary and show a sketch analysis. We also briefly state the BRUTEPOLICYELIMINATION-RL algorithm and its result, postponing the details into the Appendix C because it essentially uses the same key techniques as ones in BARBAR-RL analysis. In Section 4, we give a formal description of the reward-free exploration algorithm ESTALL as well as its sketch analysis.

2. Preliminaries

Episodic reinforcement learning. Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, H, s_1)$ be an episodic *Markov Decision Process (MDP)* where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $P : \mathcal{S} \times \mathcal{A} \times [H] \to \Delta(\mathcal{S})$ is the transition operator which takes a state-action-step pair and returns a distribution over states, $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward distribution and the H is the episodic length. For convenience, we assume that the trajectory always starts from a single state s_0, that is, $P(s_1 = s) = 0$ for all $s \neq s_0$. It can be reduced from more general setting by adding an arbitrary starting state.

We have total T episodes. At each episode $t \in [T]$, a deterministic non-stationary policy π chooses an action $a \in \mathcal{A}$ based on the current state $s \in \mathcal{S}$ and the step $h \in [H]$. Formally, $\pi = \{\pi_h\}_{h=1}^H$ where for each $h \in [H]$, $\pi_h : \mathcal{S} \to \mathcal{A}$ maps a given state to an action. The policy π induces a random trajectory $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_H, a_H, r_H, s_{H+1}$ where $a_1 = \pi_1(s_1), r_1 \sim R(s_1, a_1), s_2 \sim P(\cdot|s_1, a_1, 1), a_2 = \pi_2(s_2), r_2 \sim R(s_2, a_2), \ldots, a_H = \pi_H(s_H), r_H \sim R(s_H, a_H), s_{H+1} \sim P(\cdot|s_H, a_H, H)$. We define the set of all possible policies as $\Pi = \mathcal{A}^{\mathcal{S}\times[H]}$.

Finally, we assume the bounded total reward that $r_h \geq 0$ for all $h \in [H]$ and $\sum_{h=1}^H r_h \in [0, H]$.

Episodic RL with corruption. When *no corruption* happens, all the samples are consistently generated by a *nominal* MDP $\mathcal{M}^* = (\mathcal{S}, \mathcal{A}, P^*, R^*, H, s_1)$. Here we assume the MDP is stationary, that is $P(\cdot|s, a, h) = P(\cdot|s, a, h'), R(s, a, h) = R(s, a, h')$ for all $h, h' \in [H]$.

In the *corrupted* setting, before episode t, the adversary decides whether to corrupt the episode, in which case the corresponding MDP $\mathcal{M}_t = (\mathcal{S}, \mathcal{A}, P_t, R_t, H, s_1)$ can be arbitrary. Notice that although the *nominal* MDP $\mathcal{M}^*$ is a stationary MDP, we generally allow the corrupted $\mathcal{M}_t$ to be non-stationary. We define the corruption numerically at episode t as

$$
\begin{aligned}
c_t^r &= \sum_{h=2}^{H} \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |R_t(s, a, h) - R^*(s, a)| \\
&\quad + \sup_{a\in\mathcal{A}} |R_t(s_0, a, 1) - R^*(s_0, a)| \\
c_t^p &= \sum_{h=2}^{H} \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \|P_t(\cdot|s, a, h) - P^*(\cdot|s, a)\|_1 \\
&\quad + \sup_{a\in\mathcal{A}} \|P_t(\cdot|s_0, a, 1) - P^*(\cdot|s_0, a)\|_1
\end{aligned}
$$

Notice that we define the corruption on transition and rewards separately because the main difficulty in RL setting comes from corruptions on the transition function. Also, compared to the corruption definition in Lykouris et al. (2020), which merely captures whether an episode has been corrupted or not, our definition is based on the real-valued magnitude of the corruption. Finally, both $\mathcal{M}^*$ and $\mathcal{M}_t$, as well as the corruption levels c_t^p, c_t^r are unknown to learner. The adversary can always adaptively decide to corrupt the current episode based on the learner's strategy and the observable history of the previous episode from 1 to $t-1$, which is the same setting as in Lykouris et al. (2020). But the adversary can be even stronger, that is, it can decide corruption c_t^p, c_t^r after seeing learner's *chosen policy* in each episode or even seeing learner's *state and chosen action* at each stage in each episode. Here we called it "cheated adversary". Otherwise, we call it "non-cheated adversary" for adversary who decides corruption before seeing learner's chosen deterministic policy.

Other Conventions and Notations. We use the superscript rp as a shorthand to suggest a term holds for both reward and transition corruptions simultaneously. We define the total corruption for any time interval $\mathcal{I}$ as $C_{\mathcal{I}}^{rp} = \sum_{t\in\mathcal{I}} c_t^{rp}$ and simply denote $C_{[0,T]}^{rp}$ as C^{rp}. For any policy π, we write the value function under $\mathcal{M}$ as

$V^{\mathcal{M},\pi}(s_1)$, and denote $V^{\mathcal{M}_t,\pi}(s_1)$ as $V_t^\pi(s_1)$, $V^{\mathcal{M}^*,\pi}(s_1)$ as $V_*^\pi(s_1)$. Also we denote $V^*(s_1) = \max_{\pi\in\Pi} V_*^\pi(s_1)$ and $\Delta_\pi = V^*(s_1) - V_*^\pi(s_1)$. Because we assume a deterministic start state, in the remainder of the paper we omit s_1.

Regret. In this paper, we will focus on the the regret that is only evaluated on the nominal MDP, defined as following,

$$\text{Reg}_T := \sum_{t=1}^{T} V^* - V_*^{\pi_t}$$

This is the same definition as in (Lykouris et al., 2020).

An ϵ-net for Policies. Using the same idea as in Section 5.1 of Wang et al. (2020), we can construct an ϵ-net of non-stationary policies, denoted as Π_ϵ. As proved in their work, Π_ϵ satisfied the following properties

$$|\Pi_\epsilon| \le \min\{(H/\epsilon+1)^{|\mathcal{S}|^2|\mathcal{A}|+|\mathcal{S}||\mathcal{A}|}, |\mathcal{A}|^{H|\mathcal{S}|}\} \quad (1)$$

$$V^* - \max_{\pi\in\Pi_\epsilon} V_*^\pi \le 8H^2|\mathcal{S}|\epsilon \quad (2)$$

The first property enables us to reduce the sample complexity when $H \gg |\mathcal{A}|, |\mathcal{S}|$. The second property ensures that, as long as ϵ is small enough, the best policy inside Π_ϵ is close to the true optimal policy. In the remainder of the paper, we will only consider policies inside $\Pi_{1/T}$ instead of the whole policy space $\mathcal{A}^{\mathcal{S}\times[H]}$.

3. Main Algorithms and Results

We present two algorithms: the first for the non-cheated and the second for the cheated. Recall that the difference between these settings is the strength of the adversary. The non-cheated must decide the corruption before seeing the learner's current action (or chosen policy) while the cheated can decide afterwards. Thus, for the more challenging setting of a cheated adversary, we expect a larger regret bound.

3.1. The Algorithm and the Result for Non-cheated Adversary

Algorithm 1 is based on the multi-arm bandits algorithm BARBAR proposed in Gupta et al. (2019). In BARBAR, instead of permanently eliminating an arm, the learner will continue pulling each arm with a certain probability defined by its estimated gap. Specifically, in an epoch m with length 2^{2m}, an arm a with an estimated gap $\hat{\Delta}_a^m$ will be pulled roughly $1/(\hat{\Delta}_a^m)^2$ times and suffer roughly $\frac{\text{total corruption in epoch m}}{2^{2m}(\hat{\Delta}_a^m)^2}$ amount of corruptions due to the randomness, so the estimation error of arm a will decrease when the the epoch length doubles, as long as the total amount of corruptions is sublinear. Therefore, close-to-optimal arms that suffered from large corruptions initially can recover and be correctly estimated later, instead of being permanently eliminated at the very beginning.

Algorithm 1 BARBAR-RL

1: **Input:** time horizon T, confidence $\delta_{overall}$
2: Construct a $1/T$-net for non-stationary policies, denoted as $\Pi_{1/T}$.
3: Initialize $S_1 = \{0\}, \Pi_0^1 = \Pi_{1/T}$. And for $j \in [\log T]$, initialize $\epsilon_j = 2^{-j}.\epsilon_{est}^j = \epsilon_j/128$
4: Set $\lambda_1 = 6|\mathcal{S}||\mathcal{A}|\log(H^2|\mathcal{S}||\mathcal{A}|T)$ and $\lambda_2 = 12\ln(8T/\delta_{overall})$
5: **for** epoch $m = 1, 2, \ldots$ **do**
6: Set $\delta_j^m = (|\Pi_j^m|\delta_{overall})/(5|\Pi_{1/T}|T)$ for all $j \in S_m$
7: Set $F_j^m = \frac{8|\mathcal{S}|^2H^4|\mathcal{A}|^2\ln(2|\Pi_j^m|/\delta_j^m)}{(\epsilon_{est}^j)^2}$ for all $j \in S_m$.
8: Set $n_j^m = 2\lambda_1\lambda_2 F_j^m$ for all $j \in S_m$.
9: Set $N_m = \sum_{j\in S_m} n_j^m$ and $T_m^s = T_{m-1}^s + N_{m-1}$
10: Initialize an independent sub-algorithm for each $j \in S_m$ as $\text{ESTALL}_j^m = \text{EstAll}(\epsilon_{est}^j, \delta_j^m, F_j^m, \Pi_j^m)$
11: **for** $t = T_m^s, T_m^s+1, \ldots, T_m^s+N_m-1$ **do**
12: Run ESTALL_j^m.CONTINUE with probability $q_j^t = n_j^m/N_m$
13: **end for**
14: **if** there exists unfinished ESTALL_j^m **then**
15: Set $T_m^s = t+1$ and repeat the whole process from line 10. ▷ So each repeat is a sub-epoch.
16: **else**
17: Obtain $\hat{r}_m(\pi)$ for all π.
18: **end if**
19: Set $\hat{r}_*^m = \max_{\pi\in\Pi_{1/T}}\{\hat{r}_m(\pi) - \frac{1}{16}\hat{\Delta}_\pi^{m-1}\}$
20: Set $j^m(\pi) = \inf\{j|2^{-j} < \max\{2^{-m}, \hat{r}_*^m - \hat{r}_\pi^m\}\}$ for all π, and let $\hat{\Delta}_\pi^m = 2^{-j^m(\pi)}$
21: Add π into $\Pi_{j^m(\pi)}^{m+1}$ for all π and set $S_{m+1} = \bigcup_\pi j^m(\pi)$
22: **end for**

In our algorithm, we regard each policy π as an arm and perform the same type of sampling strategy. We denote each repeat from Line 11 to 13 in epoch m as a sub-epoch E_m^k with length $\tilde{\mathcal{O}}(2^{2m})$. Then in any E_m^k, each policy π with estimated gap $\hat{\Delta}_\pi^m$ will be *simulated* roughly $1/(\hat{\Delta}_\pi^m)^2$ times and will suffer roughly $\frac{C_{E_m^k}^r + C_{E_m^k}^p}{2^{2m}(\hat{\Delta}_\pi^m)^2}$ amount of corruptions. While it suffices to rollout each π for $\mathcal{O}(1/\epsilon^2)$ episodes to get an ϵ-close estimation, this will result in a $\mathcal{O}(|\Pi_{1/T}|)$ dependence in regret. *In this work, we achieve an $\mathcal{O}(\log(|\Pi_{1/T}|))$ dependence by utilizing the shared information between policies.*

To be specific, at the end of each epoch m, we divide the policies into several subsets according to their current estimated policy gap (Line 19 to 21). For example, policies in Π_j^{m+1} all have estimated policy gaps close to 2^{-j}. These

subsets will be used for random sampling in the next epoch. And here we use S_{m+1} as a collection of the indices of these subsets.

Now suppose there exists a "perfect" oracle which guarantees an ϵ-close estimation on each policy uniformly inside some input policy set Π_{est}, with only $\mathcal{O}(\log(\Pi_{est})/\epsilon^2)$ sample complexity. Then, by calling such an oracle on each subset of polices Π_j^m, we will able to achieve the simulation goal stated above. Here we propose a reward-free exploration algorithm ESTALL as the sub-algorithm, whose performance is close to such a "perfect" oracle when the the amount of corruptions is relatively small, and still guarantees some sublinear regret otherwise. (See Section 4 for details)

ESTALL$_j^m$.INIT

– Start and run an independent sub-algorithm according to the inputs as described in Algorithm 2 until some policy π needs to interact with the environment.
– Suspend this sub-algorithm and set π awaiting.

ESTALL$_j^m$.FINISH

– Return "finish" when each $\pi \in \Pi_j^m$ gets an estimation $\hat{r}(\pi)$ as defined in Line 15 in Algorithm 2.

ESTALL$_j^m$.CONTINUE

If ESTALL$_j^m$ is suspended
—- Rollout the awaiting π once, which caused the suspension
—- Continue running the ESTALL$_j^m$ as described in Algorithm 2 until the next ROLLOUT is met, which means that there is some policy π' that needs to interact with the environment
—- Suspend the algorithm again and let π' be the new awaiting policy
Else ▷ ESTALL$_j^m$ has finished
—- Rollout any $\pi \in \Pi_j^m$ randomly
end

To be specific, at the beginning of each sub-epoch E_m^k, the learner initializes a set of parallel sub-algorithms denoted as $\{\text{ESTALL}_j^m\}$ corresponding to the constructed subset of policies (Line 10). Here δ_j^m and F_j^m set in Line 6 and 7 represent a failure probability and a parameter related to the number of roll-outs, given as inputs to ESTALL$_j^m$, which is described in Section 4 in detail. And n_j^m set in Line 8 is the expected number of times ESTALL$_j^m$ will interact with the environment. As described before, such an interaction strategy is carefully randomized according to the estimated gap of policies inside this sub-algorithm (Line 12). Then after roughly $n_j^m = \tilde{\mathcal{O}}(\log(|\Pi_j^m|)/\epsilon_j^2)$ interactions, ESTALL$_j^m$ returns one of the following conditions with probability at least $1 - \delta_j^m$:

- an $\left(\epsilon_j + (C^r_{E_m^k} + C^p_{E_m^k})\epsilon_j^2\right)$-close estimation on each π, denoted as $\hat{r}_m(\pi)$, when ESTALL$_j^m$ has finished. (from Theorem 4)
- an unfinished ESTALL$_j^m$, which implies that $(C^r_{E_m^k} + C^p_{E_m^k}) \geq \tilde{\Omega}(1/\epsilon_j)$. (from Theorem 3)

In the first case, we have achieved the desired uniform estimation with $\hat{r}_m(\pi)$ on each policy. (Line 16 and 17) The algorithm will then construct a new subset of policies and go to the next epoch. In the second case, we will repeat the sub-epoch until we successfully obtain uniform estimation on each policy. (Line 14 and 15) Due to the lower bound on $(C^r_{E_m^k} + C^p_{E_m^k})$, we can show that the regret caused by discarded sub-epochs can be upper bounded in terms of the amount of corruption.

Theorem 1. *By running this algorithm in the non-cheated setting, with probability at least $1 - \delta_{overall}$, the regret is bounded by*

$$\begin{aligned}
&\tilde{\mathcal{O}}\left(|\mathcal{S}|^2|\mathcal{A}|^{\frac{3}{2}}H^2\min\{\sqrt{H}, \sqrt{|\mathcal{S}||\mathcal{A}|}\}\ln(1/\delta_{overall})(\star)\right)\\
&+\tilde{\mathcal{O}}\left(|\mathcal{S}|^2|\mathcal{A}|^2H^2\ln(1/\delta_{overall})C^p\right)\\
&+\tilde{\mathcal{O}}\left(|\mathcal{S}||\mathcal{A}|\ln(1/\delta_{overall})C^r\right)\\
&+\tilde{\mathcal{O}}\left(\frac{(C^p)^2}{H}+\frac{C^pC^r}{H^2}\right)
\end{aligned}$$

where $\tilde{\mathcal{O}}$ hides log *factors on $T, |\mathcal{S}|, |\mathcal{A}|, H$, and*

$$\star = \min\{\sqrt{T}, \frac{1}{\min_{\pi\in\Pi}\Delta_\pi}\}.$$

We note that the PolicyGapComplexity, $\frac{1}{\min_{\pi\in\Pi}\Delta_\pi}$, has also been used in some previous work (Jaksch et al., 2010). If we let Π be all deterministic policies, the PolicyGapComplexity will be close to the GapComplexity defined in Simchowitz & Jamieson (2019) in some non-trivial cases, for example, when all the policies visit a subset of states at step 2 with uniform probability. Otherwise, it can be much larger than the GapComplexity. We postpone the discussion on their relation to Appendix B.7.

The dependence on $|\mathcal{S}|, |\mathcal{A}|, H$ is not optimal compared to many existing tabular RL results without

corruptions, but compared to Lykouris et al. (2020), our result scales better in terms of H. Most importantly, this is the first result we are aware of in the corrupted setting where the amount of corruptions contributes only additively to the regret bound instead of multiplying $\sqrt{T}$ as in Lykouris et al. (2020). Conceptually, our result also suggests that corruptions on transition functions have much more influence on the regret than the corruptions on rewards.

Finally, we provide some intuition for why the $\tilde{\mathcal{O}}\left(\frac{(C^p)^2}{H}+\frac{C^pC^r}{H^2}\right)$ terms appear in the bound: Suppose in some epoch there was more than $\mathcal{O}(\sqrt{N_m})$ amount of corruptions, but all the sub-algorithms still happened to finish (e.g., if the adversary changed the transition function in an undetectable way). Furthermore, in the next epoch, the adversary manipulates the corruptions to force the algorithm to restart the sub-algorithms again and again. Under this described scenario, the algorithm is repeatedly using the data from previous corrupted epochs without any chance to correct them, which causes the $(C^p)^2$ and C^pC^r terms. In addition, since c_t^p scales with the horizon H and this regret term depends on the number of times the learner restarts sub-algorithms, when the total corruption budget C^p is fixed, we will have H in the denominators.

3.2. The Algorithm and the Result for Cheated Adversary

Algorithm Overview: In Algorithm 1, we avoid permanently eliminating an policy. Instead, we use a random policy sampling strategy to ensure that, the corruptions that affected any given policy estimation in the early stages can be corrected for later. However, in the cheated setting, the randomness of policy sampling no longer works because now the adversary decides when to corrupt after seeing the sampled policy. Thus, we propose BRUTE-FORCE POLICY ELIMINATION , which is based on the traditional policy elimination method that permanently eliminates policies, but with an enlarged confidence range of $\tilde{\mathcal{O}}(\sqrt{HT})$. Therefore, the best policy will never be eliminated as long as $C^p+C^r\le\tilde{\mathcal{O}}(\sqrt{HT})$. But such a brute-force method will lead to a regret that scales like $(C^r)^2$ instead of C^r. As before, we still need a uniform estimation of each policy with only a $\mathcal{O}(\log|\Pi|/\epsilon^2)$ sample complexity. Fortunately, the same approach still works, which is, running a set of sub-algorithms in parallel and restarting them when there is an unfinished one. The algorithm and analysis techniques are very similar as in the non-cheated adversary case, and therefore, we postpone the details into Appendix C.

Theorem 2. *By running this algorithm in the cheated setting, with probability at least $1-\delta_{overall}$, the regret is upper bounded by*

$$\tilde{\mathcal{O}}\left(|\mathcal{S}|^2|\mathcal{A}|^{3/2}H^2\min\{\sqrt{H},\sqrt{|\mathcal{S}||\mathcal{A}|}\}\ln(1/\delta_{overall})\sqrt{T}\right)+\tilde{\mathcal{O}}\left(\frac{(C^r)^2}{|\mathcal{S}||\mathcal{A}|H^3}+|\mathcal{S}||\mathcal{A}|H(C^p)^2\right)$$

Compared with Theorem 1, Theorem 2 suffers an additional $\frac{(C^r)^2}{H^3|\mathcal{S}||\mathcal{A}|}$ regret and also has additional $H^2|\mathcal{S}||\mathcal{A}|$ multiplicative dependence on $(c^p)^2$ terms, to account for the cheated adversary.

3.3. Analysis Sketch for Theorem 1

We give a proof sketch for Theorem 1 here and postpone the details to Appendix B.

Step 1: Let Γ_m denote the number of sub-epochs in epoch m. Firstly, appealing to standard concentration inequalities and the random policy sampling strategy, we show that the following events hold with high probability. Note that to aid the exposition, the events defined below are somewhat different than the ones defined in the Appendix.

$$\mathcal{E}_{est}:=\left\{\forall m,\pi:\begin{array}{l}|\hat{r}^m(\pi)-V_*^\pi|/4\\ \le\lambda_1\lambda_2(C^r_{E_m^{\Gamma_m}}+C^p_{E_m^{\Gamma_m}})/N_m+\hat{\Delta}_\pi^{m-1}/64\end{array}\right\}$$

$$\mathcal{E}_{unfinished}:=\left\{\forall m,\forall k\in[\Gamma_m-1]:\begin{array}{l}C^p_{E_m^k}\\ \ge\sqrt{\frac{\ln(10T|\Pi_{1/T}|/\delta_{overall})}{16\lambda_1\lambda_2}N_m}\end{array}\right\}$$

Here $\mathcal{E}_{est}$ suggests that, at the end of epoch m, we can have $\tilde{\mathcal{O}}\left(\hat{\Delta}_\pi^{m-1}+(C^p_{E_m^{\Gamma_m}}+C^r_{E_m^{\Gamma_m}})\epsilon_m^2\right)$-close estimation on every policy. And $\mathcal{E}_{unfinished}$ suggests that for each unfinished sub-epochs E_m^k, its length can always be upper bounded by $\tilde{\mathcal{O}}\left((C^p_{E_m^k})^2\right)$.

Step 2: Now we can decompose the regret into

$$\mathrm{Reg}\le\underbrace{\frac{3}{2}\sum_{m=1}^{M}\sum_{j\in S_m}\mathring{\Delta}_j^m n_j^{m,\Gamma_m}}_{\text{NON-REPEAT TERM}}+\underbrace{\frac{3}{2}\sum_{m=1}^{M}\sum_{k=1}^{\Gamma_m-1}\sum_{j\in S_m}\mathring{\Delta}_j^m n_j^{m,k}}_{\text{REPEAT TERM}}+\mathcal{O}(\text{Low order terms induced by }\epsilon\text{-net of policies})$$

where $\mathring{\Delta}_j^m = \max_{\pi\in\Pi_j^m}\left(\max_{\hat{\pi}\in\Pi_{1/T}} V_*^{\hat{\pi}} - V_*^{\pi}\right)$. The non-repeat term represents the sub-epochs where the sub-algorithms complete and estimate all the policy values successfully. Given $\mathcal{E}_{est}$, by using similar techniques as in Gupta et al. (2019), we have $\mathring{\Delta}_j^m \le \mathcal{O}(\epsilon_j) + \mathcal{O}\left(\lambda_1\lambda_2\sum_{s=1}^{m-1}\frac{\left(HC^p_{E_s^{\Gamma_s}}+C^r_{E_s^{\Gamma_s}}\right)}{16^{m-s-1}N_s}\right)$, where the second term is a discounted corruption rate. It matches our intuition that the influence from early corrupted estimations will decay as we doubling the epoch. Thus we can bound the non-repeat term by $\tilde{\mathcal{O}}(\sqrt{T}+C^r+C^p)$. The repeat term represents the regret from sub-epochs when the sub-algorithms restart. Fortunately, according to $\mathcal{E}_{unfinished}$, this only occurs when the corruption level is beyond some threshold. In this case, intuitively, discarding the data collected in the sub-epoch won't hurt too much since the estimation itself is not accurate. Thus the repeat term can by upper bounded by $\tilde{\mathcal{O}}(C^p(C^r+C^p))$.

4. The Sub-algorithm and the Results

In this section, we give a detailed description for a reward-free exploration algorithm ESTALL. As stated in the previous section, we use this algorithm as a black-box sub-algorithm and any improvements in this sub-algorithm would improve the overall regret bounds as well. In a sub-epoch E_m^k, we run a set of independent copies in parallel, each denoted as ESTALL_j^m. As described in ESTALL_j^m.CONTINUE, for each copy ESTALL_j^m, we will run it offline until some policy needs to interact with the environment. In this case, we will suspend the algorithm and make the policy awaiting hold until the next ESTALL_j^m.CONTINUE has been called. Then we will again continue running ESTALL_j^m offline and repeat the process above until finished.

4.1. Algorithms

This algorithm follows the same idea as one in Wang et al. (2020). That is, we adaptively build an exploration policy set $\Pi_{\mathcal{D}}$ and collect samples by only implementing the policies inside $\Pi_{\mathcal{D}}$, as shown in ROLLOUT (Algorithm 4). Then we are able to evaluate many policies simultaneously on the collected data, as shown in SIMULATE (Algorithm 3). The original version in Wang et al. (2020), however, requires $\mathcal{O}(poly(|\mathcal{S}||\mathcal{A}|H)\log(\Pi)/\epsilon_{est}^3)$ to get a uniform ϵ_{est}-close estimation on each policy values. This is because the original algorithm allocates $\mathcal{O}(poly(H)\log(\Pi)/\epsilon_{est}^2)$ independent sub-algorithms called SIMONE, each with sample complexity $\mathcal{O}(poly(|\mathcal{S}||\mathcal{A}|H)/\epsilon_{est})$, and all the data collected in each SIMONE will only be used to simulate one corresponding trajectory of any π.

We improve this algorithm in terms of ϵ_{est} by the fact

Algorithm 2 ESTALL

Input: target estimation error ϵ_{est}, confidence parameter δ_{est}, number of simulate trajectories $F_{est} \ge \frac{8|\mathcal{S}|^2H^4|\mathcal{A}|^2\log(2|\Pi_{est}|/\delta_{est})}{\epsilon_{est}^2}$ and policy set Π_{est}.
Set $\tau = 6$, which is a parameter related to ROLLOUT
Initialize empty buffers $\mathcal{D}_{s,a}$ for all $(s,a)\in\mathcal{S}\times\mathcal{A}$ and let $\mathcal{D} = \{\mathcal{D}_{s,a}\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$.
Initialize an empty exploration policy set $\Pi_{\mathcal{D}}$
for $\pi\in\Pi$ **do**
  $\{z_i^\pi\}_{i\in[F]} \leftarrow \text{SIMULATE}(\pi, \mathcal{D}, F_{est})$
  if $\exists(s,a), \sum_{i=1}^{F_{est}} \mathbf{1}[z_i^\pi \text{is } \textit{Fail} \text{ at } (s,a)] \ge \frac{\tau\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}$ **then**
    $\{z_i^\pi\}_{i\in[F_{est}]}, \mathcal{D} \leftarrow \text{ROLLOUT}(\pi, \tau, \mathcal{D}, F_{est})$
    $\Pi_{\mathcal{D}} \leftarrow \Pi_{\mathcal{D}}\bigcup\{\pi\}$
    ▷ Note that $\Pi_{\mathcal{D}}$ is not used in actual algorithm implement, but just for analysis convenience
  end if
end for
for each trajectory $z = (s_1,a_1,r_1),(s_2,a_2,r_2),\dots$ in $\{z_i^\pi\}_{(i,\pi)\in|F_{est}|\times\Pi_{est}}$ **do**
  Calculate

$$r(z) = \begin{cases} 0 & z \text{ is Fail} \\ \sum_{h=1}^H r_h & \text{otherwise} \end{cases}$$

end for
Calculate $\hat{r}(\pi) = \frac{1}{F_{est}}\sum_{i=1}^{F_{est}} r(z_i^\pi)$ for all $\pi\in\Pi$
return $\{\hat{r}(\pi)\}_{\pi\in\Pi}$

that, due to the properties of an MDP, data collected in the one trajectory can be used to simulate different independent trajectories of any π. Therefore, instead of updating exploration policy set $\Pi_{\mathcal{D}}$ based on the failure number on a whole trajectory, we do updates based on the failure number on each state-action pairs. (Line 7 in Algorithm 2) Then we show that the size of $\Pi_{\mathcal{D}}$ is at most $\tilde{\mathcal{O}}(poly(|\mathcal{S}||\mathcal{A}|))$ and each $\pi \in \Pi_{\mathcal{D}}$ will interact with environment $\tilde{\mathcal{O}}\left(poly(|\mathcal{S}||\mathcal{A}|H)\log(1/\delta_{est})/\epsilon_{est}^2\right)$ times.

Here F_{est} is the number of trajectories we at least need to simulate each $\pi \in \Pi$ in order to get a desired estimation. Therefore, we need to rollout each $\pi \in \Pi_{\mathcal{D}}$ at least F_{est} times. However, while this number is sufficient for simulating $\pi \in \Pi_{\mathcal{D}}$ enough times, it does not account for the fact that other policies in $\Pi_{\mathcal{D}}$ may need additional data to simulate on. As a consequence we need to repeat the F_{est} rollouts τ times to ensure we have enough data ($\tau = 6$ suffices).

Algorithm 3 SIMULATE($\pi, \mathcal{D}, F$)

for $(s,a) \in \mathcal{S} \times \mathcal{A}$ **do**
  Mark all elements in $\mathcal{D}_{s,a}$ as unused,
end for
for $h \in [H]$ **do**
  for simulated trajectory $i \in [F]$ **do**
    if all elements in $\mathcal{D}_{S_h,\pi(s_h)}$ are marked as used **then**
      Mark *Fail* at s_h for i-th trajectory simulation of π, denote as $Fail(s_h, \pi_h(s_h), i)$
    else
      Set (s_{h+1}^i, r_h^i) to be the first unused element in $\mathcal{D}_{S_h,\pi_h(s_h)}$ and mark it as used
    end if
  end for
end for
return
$(s_1^i, \pi(s_1)^i, r_1^i), (s_2^i, \pi(s_2)^i, r_2^i), \ldots, (s_H^i, \pi(s_H)^i, r_H^i)$
or
$(s_1^i, \pi(s_1)^i, r_1^i), (s_2^i, \pi(s_2)^i, r_2^i), \ldots,$
$(Fail(s_h, \pi(s_h), i))$,
for all simulated trajectory $i \in [F]$.

4.2. Results and Sketch Analysis

Theorem 3 (Sample complexity)**.** *Suppose* $F_{est} \geq \frac{8|\mathcal{S}|^2H^4|\mathcal{A}|^2\log(2|\Pi_{est}|/\delta_{est})}{\epsilon_{est}^2}$ *and* $\tau \geq 6$. *If the* $C_{est}^p \leq \frac{\epsilon_{est}F_{est}}{2|\mathcal{S}||\mathcal{A}|H^2}$, *then with probability at least* $1-\delta_{est}$, *the number of (non-simulated) roll-outs in the environment is at most*

$$|\mathcal{S}||\mathcal{A}|F_{est}\tau log(H|\mathcal{S}||\mathcal{A}|/\epsilon_{est})$$

times. This also implies that if the algorithm interacts more than the above number of times, then with probability at least $1-\delta_{est}$, $C_{est}^p > \frac{\epsilon_{est}F_{est}}{2|\mathcal{S}||\mathcal{A}|H^2}$.

Algorithm 4 ROLLOUT(π,τ,$\mathcal{D}$,F)

for $j \in [F\tau]$ **do**
  Sample the j-th trajectory for π and collect H samples denoted as $z_j^\pi = (s_1, a_1, r_1), (s_2, s_2, r_2), \ldots, (s_H, a_H, r_H)$.
  for $h \in [H]$ **do**
    Update $\mathcal{D}_{s_h,a_h} \leftarrow \mathcal{D}_{s_h,a_h} \cup \{(s_{h+1}, r_h)\}$
  end for
end for
return updated $\mathcal{D}$ and the uniformly chosen F trajectories $\{z_j^\pi\}_{j\in[F]}\}$.

Proof Sketch: Here we provide a proof sketch for the non-corrupted setting and postpone the details including how to deal with $C_{est}^p \leq \frac{\epsilon_{est}F_{est}}{2|\mathcal{S}||\mathcal{A}|H^2}$ into Appendix D. Notice that, every time the condition in Line 7 in Algorithm 2 is satisfied, we will add the corresponding π into the exploration set $\Pi_{\mathcal{D}}$ and rollout π in the environment $F_{est}\tau$ times. So the key is to show that, without the presence of corruptions, the number of times the condition in Line 7 in Algorithm 2 has been satisfied scales like $\mathcal{O}(\log|\Pi_{est}|)$ and not $\mathcal{O}(|\Pi_{est}|)$.

Define $f^\pi(s,a)$ as the random variable describing the total number of times a single trajectory induced by π visits (s,a) under the MDP $\mathcal{M}^*$. If $\sum_{i=1}^{F_{est}} \mathbf{1}[z_i^\pi \text{is } \textit{Fail} \text{ at } (s,a)] \geq \frac{\tau\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}$ for some fixed (s,a) and π, then there are only two cases. In case 1, $|\mathcal{D}_{s,a}| = 0$ and $\mathbb{E}[f^\pi(s,a)] \geq \Omega\left(\frac{\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}\right)$. So calling ROLLOUT$(\pi, \tau, \mathcal{D}, F_{est})$ will make $|\mathcal{D}_{s,a}|$ increase to at least $o\left(\frac{\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}\right)$ with high probability. In case 2, $|\mathcal{D}_{s,a}|$ is roughly smaller than $2\mathbb{E}[f^\pi(s,a)]F_{est}$. So calling ROLLOUT$(\pi, \tau, \mathcal{D}, F_{est})$ will make $|\mathcal{D}_{s,a}|$ double with high probability. (Notice here we say "roughly" because in the actual proof, we consider some lower bound of $|\mathcal{D}_{s,a}|$ instead of $|\mathcal{D}_{s,a}|$ directly.) Thus, $|\mathcal{D}_{s,a}|$ starting in the worst case at about $\frac{\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}$ will eventually double until it reaches HF_{est}, at which time the simulation will never fail. Therefore, the total number of polices added into $|\Pi_{\mathcal{D}}|$ due to the failure at (s,a) is about $\log_2((HF_{est})/(\frac{\epsilon_{est}}{|\mathcal{S}||\mathcal{A}|H}F_{est}) = \log_2(H^2|\mathcal{S}||\mathcal{A}|/\epsilon_{est}))$. Noting that there are $|\mathcal{S}||\mathcal{A}|$ number of state-action pairs, and $F_{est}\tau$ trajectories are taken per added policy, we conclude the proof.

Theorem 4 (Estimation correctness)**.** *Suppose* $F_{est} \geq \frac{8|\mathcal{S}|^2H^4|\mathcal{A}|^2\log(2|\Pi_{est}|/\delta_{est})}{\epsilon_{est}^2}$ *and* $\tau \geq 6$. *Then for all* $\pi \in \Pi$, *with probability at least* $1-\delta_{est}$,

$$\left|\hat{r}(\pi) - V^\pi(s_1)\right| \leq (1+\tau)\epsilon_{est} + (HC_{est}^p + C_{est}^r)/F_{est}$$

Proof Sketch: We provide a proof sketch here and postpone the details until Appendix D. By definition, $\hat{r}(\pi) =$

$\frac{1}{F_{est}}\sum_{i=1}^{F_{est}} r(z_i^\pi)$ and $\{r(z_i^\pi)\}_{i=1}^{F_{est}}$ is a sequence of independent random variables. We denote their expected values $\mathbb{E}[r(z_i^\pi)]$ as $\{V_i^\pi\}_{i=1}^{F_{est}}$. Here V_i^π is not a true value function but an "average value function" whose rewards and transition functions are the average of rewards and transition functions generated by the MDPs under different times (so some are corrupted).

Now, for those $\pi \in \Pi_{\mathcal{D}}$, we can use Hoeffding's inequality to directly bound $\left|\hat{r}(\pi) - \frac{1}{F_{est}}\sum_{i=1}^{F_{est}} V_i^\pi\right|$. For those $\pi \notin \Pi_{\mathcal{D}}$, if none of them are failed, we can again use Hoeffding's inequality to directly bound $\left|\hat{r}(\pi) - \frac{1}{F_{est}}\sum_{i=1}^{F_{est}} V_i^\pi\right|$. Otherwise, because the policy *fails* at most $\epsilon_{est}\tau F/H|\mathcal{S}||\mathcal{A}|$ times at each (s,a) according to Line 7 in Algorithm 2, there will be at most $\tau\epsilon_{est}F_{est}/H$ trajectories with *fails* when computing $\hat{r}(\pi)$. Thus, $\hat{r}(\pi)$ is changed at most by $\tau\epsilon_{est}$ from the no-failure case and we get the following,

$$\text{Prob}\left[\left|\hat{r}(\pi) - \frac{\sum_{i=1}^{F_{est}} V_i^\pi}{F_{est}}\right| \geq (1+\tau)\epsilon_{est}\right] \leq \delta_{est}/2|\Pi_{est}|$$

Now we can decompose out target result into,

$$\left|\hat{r}(\pi) - V^\pi\right| \leq \left|\hat{r}(\pi) - \frac{\sum_{i=1}^{F_{est}} V_i^\pi}{F_{est}}\right| + \left|\frac{\sum_{i=1}^{F_{est}} V_i^\pi}{F_{est}} - V^\pi\right|$$

The first term can be upper bounded by the previous results. The second term can be upper bounded by the total corruptions. Finally, by taking union bound over all policy in Π_{est}, we get our target result.

5. Discussion

Since our bound in the non-cheated setting scales like $\mathcal{O}((C^p)^2)$, one natural open question is to obtain an $\mathcal{O}(C^p)$ regret bound. Second, the computational complexity of our algorithms scale with $|\Pi|$ due to the reward-free exploration sub-algorithm we use. Thus, finding an efficient algorithm is also an interesting problem. Finally, our algorithm is not instance-dependent, so whether we can achieve some regret of the form $\tilde{\mathcal{O}}\left(\text{GapComplexity} + (C^p+1)(C^p+C^r)\right)$ also remains open.

Acknowledgements

The work of KJ is supported in part by NSF RI 1907907. YC want to thank Yuanhao Wang, Chen-yu Wei and Daogao Liu for inspiring discussions.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pp. 263–272. PMLR, 2017.

Bogunovic, I., Losalka, A., Krause, A., and Scarlett, J. Stochastic linear bandits robust to adversarial attacks, 2020.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

Gupta, A., Koren, T., and Talwar, K. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory*, pp. 1562–1578. PMLR, 2019.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 4868–4878, 2018.

Jin, C., Jin, T., Luo, H., Sra, S., and Yu, T. Learning adversarial Markov decision processes with bandit feedback and unknown transition. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4860–4869. PMLR, 13–18 Jul 2020.

Jin, T. and Luo, H. Simultaneously learning stochastic and adversarial episodic mdps with known transition. *Advances in Neural Information Processing Systems*, 33, 2020.

Kaufmann, E., Ménard, P., Domingues, O. D., Jonsson, A., Leurent, E., and Valko, M. Adaptive reward-free exploration, 2020.

Lee, C.-W., Luo, H., Wei, C.-Y., and Zhang, M. Bias no more: high-probability data-dependent regret bounds for adversarial bandits and mdps. *Advances in Neural Information Processing Systems*, 2020.

Lee, C.-W., Luo, H., Wei, C.-Y., Zhang, M., and Zhang, X. Achieving near instance-optimality and minimax-optimality in stochastic and adversarial linear bandits simultaneously, 2021.

Liu, F. and Shroff, N. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pp. 4042–4050. PMLR, 2019.

Lykouris, T., Mirrokni, V., and Paes Leme, R. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 114–122, 2018.

Lykouris, T., Simchowitz, M., Slivkins, A., and Sun, W. Corruption robust exploration in episodic reinforcement learning, 2020.

Ménard, P., Domingues, O. D., Jonsson, A., Kaufmann, E., Leurent, E., and Valko, M. Fast active learning for pure exploration in reinforcement learning, 2020.

Ok, J., Proutiere, A., and Tranos, D. Exploration in structured reinforcement learning. In *32nd Conference on Neural Information Processing Systems (NIPS), DEC 02-08, 2018, Montreal, CANADA*, volume 31. Neural Information Processing Systems (NIPS), 2018.

Rosenberg, A. and Mansour, Y. Online convex optimization in adversarial markov decision processes. In *International Conference on Machine Learning*, pp. 5478–5486. PMLR, 2019.

Simchowitz, M. and Jamieson, K. G. Non-asymptotic gap-dependent regret bounds for tabular mdps. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1151–1160, 2019.

Wang, R., Du, S. S., Yang, L. F., and Kakade, S. M. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning?, 2020.

Xu, H., Ma, T., and Du, S. S. Fine-grained gap-dependent bounds for tabular mdps via adaptive multi-step bootstrap. *arXiv preprint arXiv:2102.04692*, 2021.

Zanette, A. and Brunskill, E. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pp. 7304–7312. PMLR, 2019.

Zhang, Z., Ji, X., and Du, S. S. Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon, 2020.

Zimmert, J. and Seldin, Y. An optimal algorithm for stochastic and adversarial bandits. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 467–475. PMLR, 2019.