Learning in Online MDPs: Is there a Price for Handling the Communicating Case?

Gautam Chandrasekaran[1] **Ambuj Tewari**[2]

[1]Department of Computer Science, University of Texas at Austin, Austin, Texas, USA
[2]Department of Statistics, University of Michigan, Ann Arbor, Michigan, USA

Abstract

It is a remarkable fact that the same $O(\sqrt{T})$ regret rate can be achieved in both the Experts Problem and the Adversarial Multi-Armed Bandit problem albeit with a worse dependence on number of actions in the latter case. In contrast, it has been shown that handling online MDPs with communicating structure and bandit information incurs $\Omega(T^{2/3})$ regret even in the case of deterministic transitions. Is this the price we pay for handling communicating structure or is it because we also have bandit feedback? In this paper we show that with full information, online MDPs can still be learned at an $O(\sqrt{T})$ rate even in the presence of communicating structure. We first show this by proposing an efficient follow the perturbed leader (FPL) algorithm for the deterministic transition case. We then extend our scope to consider stochastic transitions where we first give an inefficient $O(\sqrt{T})$-regret algorithm (with a mild additional condition on the dynamics). Then we show how to achieve $O\left(\sqrt{\frac{T}{\alpha}}\right)$ regret rate using an oracle-efficient algorithm but with the additional restriction that the starting state distribution has mass at least α on each state.

1 INTRODUCTION

In this work, we study online learning in Markov Decisions Processes. In this setting, we have an *agent* interacting with an adversarial *environment*. The agent observes the state of the environment and takes an action. The action incurs an associated loss and the environment moves to a new state. The state transition dynamics are assumed to be Markovian, i.e., the probability distribution of the new state is fully determined by the action and the old state. The transition dynamics are fixed and known to the learner in advance.

	No state	Ergodic	Communicating
Full Info	$\sqrt{T}$	$\sqrt{T}$	$\sqrt{T}$ THIS PAPER
Bandit Info	$\sqrt{T}$	$\sqrt{T}$[1]	$T^{2/3}$

Table 1: The dependence on time horizon T of the optimal regret, under full and bandit feedbacks, as the state transition dynamics become more complex.

However the losses are chosen by the adversary. The adversary is assumed to be oblivious (the entire loss sequence is chosen before the interaction begins). We assume that the environment reveals *full information* about the losses at a given time step to the agent after the corresponding action is taken. The total loss incurred by the agent is the sum of losses incurred in each step of the interaction. We denote the set of states by S and the set of actions by A. The objective of the agent is to minimized its total loss.

This setting was first studied in the seminal work of Even-Dar et al. [2009]. They studied the restricted class of *ergodic* MDPs where every policy induces a Markov chain with a single recurrent class. They designed an efficient (runs in polytime in MDP parameters and time of interaction) algorithm that achieved $O(\sqrt{T})$ regret with respect to the best stationary policy in hindsight. They assumed full information of the losses and that the MDP dynamics where known beforehand. This work was extended to *bandit feedback* by Neu et al. [2014][1]. They also achieved a regret bound of $O(\sqrt{T})$. Bandit feedback is a harder model in which the learner only receives information corresponding to the losses of the actions it takes.

In this paper we will look at the more general class of *communicating* MDPs, where, for any pair of states, there is a policy such that the time it takes to reach the second state from the first has finite expectation. In the case of bandit feedback with deterministic transitions, Dekel and Hazan

[1]with an additional assumption on the minimum stationary probability mass in any state

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

[2013] designed an algorithm that achieved $O\left(T^{2/3}\right)$ regret. This regret bound was proved to be tight by a matching lower bound in Dekel et al. [2013]. This regret lower bound was proved by a reduction from the problem of adversarial multi-armed bandits with *switching costs*. In this setting, the agent incurs an additional cost every time it switches the arm it plays. Their lower bound definitively proves that in the case of bandit information, online learning over communicating MDPs is *statistically harder* than the adversarial multi armed bandits problem for which we have $\tilde{O}(\sqrt{T})$ regret algorithms (Auer et al. [1995]). Their result gives rise to the natural question: is the high regret due to the communicating structure or bandit feedback(or both)? In the case of experts with switching cost, we know $O(\sqrt{T})$ regret algorithms such as FPL (Kalai and Vempala [2005]). Using this, we give an $O(\sqrt{T})$ algorithm for online learning in communicating MDPs with full information. Thus, we show that having communicating structure alone does not add any statistical price (see Table 1).

1.1 OUR CONTRIBUTIONS

In this paper, we show that online learning over MDPs with full information is *not statistically harder*[2][3] than the problem of online learning with expert advice. In particular, we design an efficient algorithm that learns to act in Communicating Deterministic MDPs (ADMDPs) with $O(\sqrt{T})$ regret under full information feedback against the best deterministic policy in hindsight. This is the first $O(\sqrt{T})$ regret algorithm for this problem. To achieve this bound, we designed a follow the perturbed leader (Kalai and Vempala [2005]) style algorithm where we achieve low regret with respect to the set of exponentially many policies with the additional guarantee that our algorithm does not switch policies too much. Since the number of policies is exponential, a naive implementation of FPL will not work. We had to carefully choose a polynomial number of perturbations (as opposed to exponential in naive FPL) such that the algorithm worked. We believe this is one of the sources of technical novelties in our paper.

We prove a matching[2] regret lower bound in this setting. We also extend the techniques used in the previous algorithm to design an algorithm that runs in time exponential in MDP parameters that achieves $O(\sqrt{T})$ regret in the general class of communicating MDPs (albeit with an additional mild assumption[3]). Again, this is the first algorithm that achieves $O(\sqrt{T})$ regret against this large class of MDPs. Before this, $O(\sqrt{T})$ regret algorithms were only known for the case of ergodic MDPs. En route to this, we designed the Switch_Policy procedure(Algorithm 3) to catch the distribution induced to by a new policy in time $O(D^2)$ where D is the diameter of the MDP. This was subsequently used by Dai et al. [2022] where they prove analogous results to ours in the bandit case.

Finally, we study the problem of designing oracle-efficient algorithms. We give an $O\left(\sqrt{\frac{T}{\alpha}}\right)$ regret algorithm for communicating MDPs with a start state distribution having probability mass at least α on each state that is efficient when given access to an optimization oracle.

2 RELATED WORK

As mentioned in the introduction, a closely related problem is that of online learning with switching costs. In the case of full information, algorithms like FPL (Kalai and Vempala [2005]) achieves $O(\sqrt{T})$ regret with switching cost. In the case of bandit feedback, Arora et al. [2012b] gives an algorithm that achieves $O(T^{2/3})$ regret with switching cost. This was proved to be tight by Dekel et al. [2013] where they proved a matching lower bound.

Another related problem is that of designing oracle-efficient algorithms (studied in Dudík et al. [2016],Block et al. [2022],Haghtalab et al. [2022]). Designing oracle efficient algorithms is challenging since all the main computational steps in the algorithm need to be in the form of oracle calls and this restricts the design space of algorithms.

Subsequent to the release of an earlier version of this paper, Dai et al. [2022] gave an inefficient $O(T^{2/3})$ and oracle-efficient $O(T^{5/6})$ regret algorithm for online learning over Communicating[2] MDPs with bandit information. Their algorithms use our Switch_Policy procedure and thus requires the same assumption[3] as us.

3 PRELIMINARIES

An Online Markov Decision Process consists of a state space S, action space A, a transition probability matrix P where $P(s, a, s')$ is the probability of moving from state s to s' on action a and a sequence of loss functions(chosen by an oblivious adversary) $\ell_1, \ldots, \ell_T$ where each ℓ_t is a map from $S \times A$ to $[0, 1]$. In this paper, S and A will be finite sets.

In the case of Adversarial Deterministic MDPs(ADMDP), the transitions are deterministic and hence the ADMDP can also be represented by a directed graph G with vertices corresponding to states S. The edges are labelled by the actions. An edge from s to s' labelled by action a exists in the graph when the ADMDP takes the state s to state s' on action a. This edge will be referred to as (s, a, s').

A (stationary) policy π is a mapping $\pi : S \times A \to [0, 1]$ where $\pi(s, a)$ denotes the probability of taking action a when in state s. When the policy is deterministic, we overload the notation and define $\pi(s)$ to be the action taken when the state is s. The interaction starts in an arbitrary start

[2]up to polynomial factors in the number of states and actions

[3]assuming the existence of a state with a “do nothing" action

state $s_1 \in S$.

An algorithm $\mathcal{A}$ that interacts with the online MDP chooses the action to be taken at each time step. It maintains a probability distribution over actions denoted by $\mathcal{A}(. \mid s, \ell_1, \ldots, \ell_{t-1})$ which depends on the current state and the sequence of loss functions seen so far. The expected loss of the algorithm $\mathcal{A}$ is

$$L(\mathcal{A}) = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t)\right]$$

where $a_t \sim \mathcal{A}\left(. \mid s_t, \ell_1, \ldots, \ell_{t-1}\right), s_{t+1} \sim P(\cdot, s_t, a_t)$ For a stationary policy π, the loss of the policy is

$$L^{\pi} = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t)\right]$$

where $a_t \sim \pi(. \mid s_t), s_{t+1} \sim P(\cdot, s_t, a_t)$. The regret of the algorithm is defined as

$$R(\mathcal{A}) = L(\mathcal{A}) - \min_{\pi \in \Pi} L^{\pi} \ .$$

The total expected loss of the best policy in hindsight is denoted by L^*. Thus,

$$L^* = \min_{\pi \in \Pi} L^{\pi} \ .$$

For any stationary policy π, let $T(s' \mid M, \pi, s)$ be the random variable for the first time step in which s' is reached when we start at state s and follow policy π in MDP M. We define the diameter $D(M)$ of the MDP as

$$D(M) = \max_{s \neq s'} \min_{\pi} \mathbb{E}\left[T(s' \mid M, \pi, s)\right].$$

A *communicating MDP* is an MDP where $D(M) < \infty$.

3.1 PRELIMINARIES ON ADMDPS

In this section, we use the graph G and the ADMDP interchangeably. A stationary deterministic policy π induces a subgraph G_π of G where (s, a, s') is an edge in G_π if and only if $\pi(s) = a$ and the action a takes state s to s'.

A communicating ADMDP corresponds to a strongly connected graph. This is because the existence of a policy that takes state s to s' also implies the existence of a path between the two vertices in the graph G.

The subgraph G_π induced by policy π in the communicating ADMDP is the set of transitions (s, a, s') that are possible under π. Each components of G_π is either a cycle or an initial path followed by a cycle. Start a walk from any state s by following the policy π. Since the set of states is finite, eventually a state must be repeated and this forms the cycle.

Let $N(s, a)$ be the next state after visiting state s and taking action a. Define $I(s)$ as

$$I(s) = \{(s', a) \mid N(s', a) = s\}.$$

The *period* of a vertex v in G is the greatest common divisor of the lengths of all the cycles starting and ending at v. In a strongly, connected graph, the period of each vertex can be proved to be equal(Bremaud [2000] Chap. 2, Thm 4.2). Thus, the period of a strongly connected G is well defined. If the period of G is 1, we call G *aperiodic*.

Let $\mathcal{C}_{(s,k)}$ be the set of all closed walks of G of length k such that the start vertex is s. The elements of $\mathcal{C}_{(s,k)}$ are represented by the sequence of edges in the walks.

Note that the cycles induced by any stationary deterministic policy π that are of length k and contain the vertex s will be in $\mathcal{C}_{(s,k)}$. However, $\mathcal{C}_{(s,k)}$ can also contain cycles not induced by policies(it can contain cycles that are not simple). We use $\mathcal{C}$ to denote $\bigcup_{s \in S, k \in [k]} \mathcal{C}_{(s,k)}$. We sometimes loosely refer elements of $\mathcal{C}$ as cycles. For a cycle c, we define $a_t(c)$ to be the action take by c in the t_{th} step if we start following c from the beginning of the interaction. Similarly, $s_t(c)$ is the state that you reach after following c for $t-1$ steps from the start of the interaction. We define $k(c)$ as the length of the cycle c.

The vertices of a strongly connected graph G with period γ can be partitioned into γ non-empty cycle classes, $C_1, \ldots, C_\gamma$ where each edge goes C_i to C_{i+1}.

Theorem 3.1. *If G is strongly connected and aperiodic, there exists a critical length d such that for any $\ell \geq d$, there exists a path of length ℓ in G between any pair of vertices. Also, $d \leq n(n-1)$ where n is the number of vertices in the graph.*

The above theorem is from Denardo [1977]. It guarantees the existence of a $d > 0$ such that there are paths of length d between any pair of vertices. The following generalization from Dekel and Hazan [2013] extends the result to periodic graphs.

Theorem 3.2 (Dekel and Hazan [2013])**.** *If G has a period γ, there exists a critical value d such that for any integer $\ell \geq d$, there is a path of in G of length $\gamma\ell$ from any state v to any other state in the same cycle class.*

Remark 3.3. *We can also find the paths of length $\ell \geq d$ from a given vertex s to any other vertex s' efficiently. This can be done by constructing the path in the reverse direction. We look at $P^{\ell-1}$ to see all the predecessors of s' that have paths of length $\ell - 1$ from s. We choose any of these as the penultimate vertex in the path and recurse.*

4 DETERMINISTIC TRANSITIONS

We now present our algorithm for online learning in ADMDPs when we have full information of losses. We use G

to refer to the graph associated to the ADMDP.

We assume that the ADMDP dynamics are known to the agent. This assumption can be relaxed as shown in Ortner [2010] as we can figure out the dynamics in poly($|S|, |A|$) time when the transitions are deterministic. We want to minimize regret against the class of deterministic stationary policies.

4.1 ALGORITHM SKETCH

We formulate the task of minimizing regret against the set of deterministic policies as a problem of prediction with expert advice. As observed earlier, deterministic policies induce a subgraph which is isomorphic to a cycle with an initial path. We keep an expert for each element of $\mathcal{C}_{(s,k)}$ for all states s and $k \leq s$. Note that we do not keep an expert for policies which have an initial path before the cycle. This is because the loss of these policies differ by at most $|S|$ compared to the loss of the cycle. Also, we make sure that the start state of the cycle is in the same cycle class as the start state of the environment. If this is not the case, our algorithm will never be *in phase* with the expert policy. Henceforth, we will refer to these experts as cycles.

The loss incurred by cycle $c \in \mathcal{C}_{(s,k)}$ at time t is equal to $\ell_t(s_t, a_t)$ where s_t and a_t are the state action pair traversed by the cycle c at time t if we had followed it from the start of the interaction.

We first present an efficient (running time polynomial in $|S|, |A|$ and T) algorithm to achieve $O(\sqrt{T})$ regret and switching cost against this class of experts. For this we used a *Follow the perturbed leader* (FPL) style algorithm.

We then use this low switching algorithm as a black box. Whenever, the black box algorithm tells us to switch policies at time t, we compute the state s that we would have reached if we had followed the new policy from the start of the interaction and moved $t + \gamma d$ steps. We then move to this state s in γd steps. Theorem 3.2 guarantees the existence of a path of this length. We then start following the new policy.

Thus, our algorithm matches the moves of the expert policies except when there is a switch in the policies. Thus, the regret of our algorithm differs from the regret of the black box algorithm by at most $O(\gamma d\sqrt{T})$.

4.2 FPL ALGORITHM

We now describe the FPL style algorithm that competes with the set of cycles described earlier with $O(\sqrt{T})$ regret and switching cost.

Algorithm 1: FPL algorithm for Deterministic MDPs

Sample perturbation vectors $\epsilon_i \in \mathbb{R}^{|S||A|}$ for $1 \leq i \leq |S|$ from an exponential distribution with parameter λ;

Sample a perturbation vector $\delta \in \mathbb{R}^{|S|^2}$ from the same distribution;

while $t \neq T + 1$ **do**

  $C_t = \arg\min_{s \in S, k \in [K], c \in \mathcal{C}_{(s,k)}} \delta(s,k) + \sum_{i=1}^{t-1} \ell_t(s_t(c), a_t(c)) + \sum_{i=1}^{\max(t, k(c)+1)} \epsilon_i(s_t(c), a_t(c))$;

  Adversary returns loss function ℓ_t;

4.2.1 Finding the leader: Offline Optimization Algorithm

First, we design an offline algorithm that finds the cycle (including start state) with lowest cumulative loss till time t given the sequence of losses $\ell_1 \ldots, \ell_{t-1}$. This is the $\arg\min$ step in Algorithm 1. Given (s, k), we find the best cycle among the cycles that start in state s and have length k. For this we use a method similar to that used in Arora et al. [2012a]. We then find the minimum over all (s, k) pairs to find the best cycle. Note that we only consider start states s which are in the same cycle class as the start state s_0 of the game.

We find the best cycle in $\mathcal{C}_{(s,k)}$ using Linear Programming. Let $n = |S||A|k$. The LP is in the space $\mathbb{R}^n$. Consider a cycle $c \in \mathcal{C}_{(s,k)}$. Let c_i denoted the i_{th} state in c. Also, let a_i be the action taken at that state. We associate a vector $x(c)$ with the cycle as follows.

$$x(c)_{s,a,i} = \begin{cases} 1 & \text{if } a = a_i \text{ and } s = c_i \\ 0 & \text{otherwise} \end{cases}$$

We construct a loss vector in $\mathbb{R}^n$ as follows.

$$l_{s,a,i} = \sum_{\substack{1 \leq j < t \\ (j-i) \equiv 0 \mod k}} \ell_j(s,a)$$

Our decision set $\mathcal{X} \subseteq \mathbb{R}^n$ is the convex hull of all $x(c)$ where $c \in \mathcal{C}_{(s,k)}$. Our objective is to find x in $\mathcal{X}$ such that $\langle x, l \rangle$ is minimized. The set $\mathcal{X}$ can be captured by the following polynomial sized set of linear constraints.

$$x \geq 0$$

$$\sum_{a \in A} x_{(s,a,1)} = 1$$

$$\forall s' \in S \setminus \{s\}, a \in A,\ x_{(s,a,1)} = 0$$

$$\forall (s', a') \notin I(s),\ x_{(s',a',k)} = 0$$

$$\forall s' \in S, 2 \leq i \leq k, \sum_{(s',a') \in I(s)} x_{(s',a',i-1)} = \sum_{a \in A} x_{(s,a,i)}$$

Once we get an optimal x for the above LP, we can decompose the mixed solution as a convex combination of at most $n+1$ cycles from Caratheodory Theorem. Also, these cycles can be recovered efficiently (Bazaraa et al. [2004]). Each of them will have same loss and hence we can choose any of them.

Once we have an optimal cycle for a given (s, k), we can minimize over all such pairs to get the optimal cycle. This gives us a polynomial time algorithm to get the optimal cycle.

Remark 4.1. *If the new cycle chosen has the same perturbed loss as the old cycle, we will not switch. This is to prevent any unnecessary switches caused by the arbitrary choice of cycle in each optimization step (as we choose an arbitrary cycle with non-zero weight in the solution).*

4.2.2 Regret of the FPL algorithm

We now state the bound on the regret and expected number of switches of Algorithm 1.

Theorem 4.2. *For appropriately chosen λ, the regret and the expected number of switches of Algorithm 1 can be bounded by*

$$O\left(|S|\sqrt{L^* \cdot \log |S||A|}\right)$$

where L^* is the cumulative loss of the best cycle in hindsight.

To achieve the desired switching bound, we grouped the policies into polynomial number of groups and showed that probability of the current policy switching to a policy in any of these groups is at most λ. Then, taking a union bound over all the groups, we achieved the desired regret bound. We prove this result in the supplementary section.

4.3 PUTTING IT TOGETHER

We have described the FPL style algorithm that achieves low regret and low switching. We now use Algorithm 1 as a sub-routine to design a low regret algorithm for the online ADMDP problem.

Recall that for a cycle c, $s_t(c)$ is the state you would reach if you followed the cycle c from the start. This can be computed efficiently.

We now state the regret bound of Algorithm 2.

Theorem 4.3. *Given a communicating ADMDP with state space S, action space A and period γ, the regret of Algorithm 2 is bounded by*

$$\textit{Regret} \leq O\left(|S|^3 \cdot \gamma\sqrt{L^* \cdot \log |S||A|}\right)$$

where L^* is the total loss incurred by the best stationary deterministic policy in hindsight.

Algorithm 2: Low regret algorithm for communicating ADMDPs

```
t=1;
s_0 is the start state of the environment;
Let c_1 be the cycle chosen by Algorithm 1 at t = 1;
if s_1 ≠ s_0(c_1) then
    Spend γd steps to move to state s_0(c_1);
    c_{1+γd} = c_1;
    t = 1 + γd;
while t ≠ T + 1 do
    Choose action a_t = a_t(c_t);
    Adversary returns loss function ℓ_t and next state
      s_{t+1};
    Feed ℓ_t as the loss to Algorithm 1 ;
    if Algorithm 1 switches cycle to c_{t+1} then
        if s_{t+1} ≠ s_{t+1}(c_{t+1}) then
            Spend γd steps to move to state
              s_{t+γd}(c_{t+1});
            c_{t+γd} = c_{t+1};
            t = t + γd;
    else
        t = t + 1;
```

Proof. We spend γd steps whenever Algorithm 1 switches. In all other steps, we receive the same loss as the cycle chosen by Algorithm 1. Thus, the regret differs by at most $\gamma d \cdot N_s$. From Theorem 4.2, we get that the total regret of our algorithm in the deterministic case is $O\left(|S| \cdot \gamma d\sqrt{T \log |S||A|}\right)$ where d is the critical length in the ADMDP. Note that d is at most $O(|S|^2)$. Thus, we get that

$$\text{Regret} \leq O\left(|S|^3 \cdot \gamma\sqrt{L^* \cdot \log |S||A|}\right)$$

□

Remark 4.4. *To achieve the first order regret bound, we set λ in terms of L^*. We need prior knowledge of L^* to directly do this. This can be circumvented by using a doubling trick.*

4.4 REGRET LOWER BOUND FOR DETERMINISTIC MDPS

We now state a matching regret lower bound (up to polynomial factors).

Theorem 4.5. *For any algorithm $\mathcal{A}$ and any $|S| > 3, |A| \geq 1$, there exists an MDP M with $|S|$ states and $|A|$ actions and a sequence of losses $\ell_1, \ldots, \ell_t$ such that*

$$R(\mathcal{A}) \geq \Omega\left(\sqrt{|S|T \log |A|}\right)$$

where $R(\mathcal{A})$ is the regret incurred by $\mathcal{A}$ on M with the given sequence of losses.

5 STOCHASTIC TRANSITIONS

In the previous sections, we only considered deterministic transitions. We now present an algorithm that achieves low regret for the more general class of communicating MDPs (with an additional mild restriction). This algorithm achieves $O(\sqrt{T})$ regret but takes exponential time to run (exponential in $|S|$).

Assumption 5.1. *The MDP M has a state s^* and action a such that*

$$Pr(s_{t+1} = s^* \mid s_t = s^*, a_t = a) = 1$$

In other words, there is some state s^* in which we have a deterministic action that allows us to stay in the state s^*. This can be interpreted as a state with a "do nothing" action where we can wait before taking the next action.

We now state a theorem that guarantees the existence of a number ℓ^* such that all states can be reached from s^* in exactly ℓ^* steps with a reasonably high probability.

Theorem 5.2. *In MDPs satifying Assumption 5.1, we have $\ell^* \leq 2D$ and state s^* such that, for all target states s', we have policies $\pi_{s'}$ such that*

$$p_{s'} = Pr[T(s' \mid M, \pi_{s'}, s^*) = \ell^*] \geq \frac{1}{4D}$$

Let $p^* = \min_s p_s$. Clearly, $p^* \geq \frac{1}{4D}$

5.1 ALGORITHM

We extend the algorithm we used in the deterministic MDP case.

We use a low switching algorithm (FPL) that considers each policy $\pi \in \Pi$ as an expert. We know from Kalai and Vempala [2005] that FPL achieves $O(\sqrt{T \log n})$ regret as well as switching cost. At time t, we receive loss function ℓ_t from the adversary. Using this, we construct $\hat{\ell}_t$ as

$$\hat{\ell}_t(\pi) = \mathbb{E}\left[\ell_t(s_t, a_t)\right]$$

where $s_1 \sim d_1, a_t \sim \pi(s_t, .)$

In other words, $\hat{\ell}_t(\pi)$ is the expected loss if we follow the policy π from the start of the game. d_1 is the initial distribution of states.

We feed $\hat{\ell}_t$ as the losses to FPL.

We can now rewrite L^π as

$$L^\pi = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t)\right] = \sum_{t=1}^{T} \mathbb{E}\left[\ell_t(s_t, a_t)\right] = \sum_{t=1}^{T} \hat{\ell}_t(\pi)$$

where $s_1 \sim d_1$ and $a_t \sim \pi(s_t, .)$. Let π_t be the policy chosen by FPL at time t. We know that

$$\mathbb{E}\left[\sum_{t=1}^{t} \hat{\ell}_t(\pi_t)\right] - \sum_{t=1}^{t} \hat{\ell}_t(\pi) \leq O(\sqrt{T \log |\Pi|})$$

for any deterministic policy π.

We need our algorithm to receive loss close to the first term in the above sum. If this is possible, we have an $O(\sqrt{T})$ regret bound for online learning in the MDP. We now present an approach to do this.

5.1.1 Catching a policy

When FPL switches policy, we cannot immediately start receiving the losses of the new policy. If this was possible, then the regret of our algorithm will match that of FPL. When implementing the policy switch in our algorithm, we suffer a delay before starting to incur the losses of the new policy (in an expected sense). Our goal now is to make this delay as small as possible. This coupled with the fact that FPL has a low number of switches will give us good regret bounds. Note that this was easily done in the deterministic case using Theorem 3.2. Theorem 5.2 acts somewhat like a stochastic analogue of Theorem 3.2 and we use this to reduce the time taken to catch the policy.

5.2 ANALYSIS

The following lemma shows that the Switch_Policy routine works correctly. That is, after the execution of the routine, the state distribution is exactly the same as the state distribution of the new policy.

Lemma 5.3. *If Switch_Policy terminates at time t, we have that*

$$Pr[S_t = s \mid T_{switch} = t] = d_\pi^t(s)$$

where $d_\pi^t(s)$ is the distribution of states after following policy π from the start of the game.

We now bound the expected loss of the algorithm in the period that FPL chooses policy π

Lemma 5.4. *Let the policy of FPL be π from time t_1 to t_2. We have that*

$$\mathbb{E}\left[\sum_{t=t_1}^{t_2} \ell_t(s_t, a_t)\right] \leq 48 \cdot D^2 + \sum_{t=t_1}^{t_2} \hat{\ell}_t(\pi)$$

We are now ready to bound the regret of Algorithm 3

Theorem 5.5. *The regret of Algorithm 3 is at most $O\left(D^2\sqrt{T \log |\Pi|}\right)$*

Algorithm 3: Low Regret Algorithm For Communicating MDPs

Function `Switch_Policy(`s,π,t_0`)`**:**
- $Done = 0$
- $t = t_0 + 1$ `// ` t_0 `is the time that` B `switched policy`
- $S_t = s$ `// ` S_t `stores the state at time` t
- **while** $Done \neq 1$ **do**
  - Move to state s^* using the best policy `// Say this step takes` k `steps`
  - $t = t + k$
  - Sample $T_{t+\ell^*}$ from $d_\pi^{t+\ell^*}(.)$
  - We set $T_{t+\ell^*}$ as the target state
  - Use policy $\pi_{T_{t+\ell^*}}$ guaranteed by Corollary 5.2 to move ℓ^* steps from s^*
  - $t = t + \ell^*$
  - **if** $S_t = T_t$ **then**
    - Consider a Bernouli Random Variable I such that $I = 1$ with probability $\frac{p^*}{p_{S_t}}$.
    - **if** $I = 1$ **then**
      - Start following π and set $Done$ to 1
      - Let the time at this happens be T_{switch}
    - **else**
      - $I = 0$
    - Continue
  - **else**
    - Continue

Function `Main`**:**
- Let π_1^{FPL} be the expert chosen by FPL at time 1
- $\pi_1 = \pi_1^{\text{FPL}}$
- Let S_1 be the start state.
- $t = 1$
- **while** $t \neq T + 1$ **do**
  - Sample a_t from $\pi_t(s_t, .)$
  - Adversary returns loss function ℓ_t and next state s S_{t+1}=s
  - Compute $\hat{\ell}_t$ and feed it as the loss to FPL as discussed before
  - **if** *FPL switches policy* **then**
    - Switch_Policy($s, \pi_{t+1}^{\text{FPL}}, t+1$) `// Call the switch policy function to catch the new policy`
    - $\pi_{t+k} = \pi_{t+1}^{\text{FPL}}$ `// ` k `is the number of steps taking by Switch Policy`
    - $t = t + k$
  - **else**
    - $\pi_{t+1} = \pi_t$
    - $t = t + 1$

Proof. We condition on the number of switches made by FPL. Let N_s be the random variable corresponding to the number of switches made by FPL. We refer to Algorithm 3 as $\mathcal{A}$.

$$L(\mathcal{A}) = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t)\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t) \mid N_s\right]\right]$$

After each switch, Lemma 5.4 tells us that the Algorithm suffers at most $48 \cdot D^2$ extra average loss to the loss of the algorithm FPL. Thus,

$$L(\mathcal{A}) \leq \mathbb{E}\left[48 \cdot D^2 \cdot N_s + \sum_{t=1}^{T} \hat{\ell}_t(\pi_t)\right]$$

π_t is the policy chosen by algorithm FPL at time t. Since FPL is a low switching algorithm, we have $N_s \leq O(\sqrt{T \log |\Pi|}$. The second term in the expectation is atmost $L^\pi + O(\sqrt{T \log |\Pi|})$ for any deterministic policy π. This is because FPL is a low regret algorithm. Thus, we have

$$L(\mathcal{A}) - L^\pi \leq O(D^2 \sqrt{T \log |\Pi|})$$

for all stationary π.

Thus, $R(\mathcal{A}) \leq O\left(D^2 \sqrt{T \log |\Pi|}\right)$ □

When Π is the set of stationary deterministic policies, we get that $|\Pi| \leq |A|^{|S|}$. Thus, we get the following theorem.

Theorem 5.6. *Given a communicating MDP satisfying Assumption 5.1 with $|S|$ states, $|A|$ action and diameter D, the regret of Algorithm 3 can be bounded by*

$$\textit{Regret} \leq O\left(D^2 \sqrt{T|S| \log |A|}\right)$$

In fact, since we are using FPL as the expert algorithm, we can get first-order bounds similar to Theorem 4.2. In a setting with n experts with m being the total loss of the best expert, we can derive that the regret and number of switches can be bounded by $O(\sqrt{m \cdot \log n})$.Thus, using this, we get the following first order regret bounds for Algorithm 3

Corollary 5.7. *Given a communicating MDP satisfying Assumption 5.1 with $|S|$ states, $|A|$ action and diameter D, the regret of Algorithm 3 can be bounded by*

$$\textit{Regret} \leq O\left(D^2 \sqrt{L^* \cdot |S| \log |A|}\right)$$

where L^* is the total expected loss incurred by the best stationary deterministic policy in hindsight.

5.3 ORACLE-EFFICIENT ALGORITHM ASSUMING EXPLORING STARTS

In this section, we design an oracle-efficient algorithm for the stochastic case assuming that the initial distribution over states, d_1 has probability mass at least α on every state. That is, $Pr[S_1 = s] \geq \alpha$ for all $s \in S$.

Here, we assume that we have an oracle $\mathcal{O}$ that can find the stationary deterministic policy with minimum cumulative loss at no computational cost. We use this oracle and the ideas from the previous sections to design an FPL style algorithm low regret algorithm.

Before, proceeding to a detailed discussion of the oracle and the algorithm, we give a few comments on the motivation for designing such algorithms. Oracle efficient reductions are standard for those online problems where the offline problem is computationally challenging. The benefit is in showing that handling the online case does not offer any additional complexity (up to poly factors) over solving the offline problem (where there is no learning, just optimization). In our case, it is not clear if the offline problem (calculating the best stationary problem in hindsight) can be computed efficiently. Theorem 4.12 from Mundhenk et al. [2000] states that the decision version of the problem is P-hard and in NP, but nothing further is known about the hardness to the best of our knowledge. Any future improvements in the offline problem immediately improves our online algorithm as well, and this is the main advantage of designing an oracle efficient algorithm.

5.3.1 Oracle

The oracle $\mathcal{O}$ takes in loss functions $\ell_1, \ldots, \ell_T$ and outputs the stationary deterministic policy with the lowest expected cumulative loss. That is, it returns the policy $\pi = \arg\min_{\Pi} L^{\pi}$, where

$$L^{\pi} = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(s_t, a_t)\right]$$

with $s_1 \sim d_1$ and $a_t = \pi(s_t)$.

We say that an algorithm is *oracle-efficient* if it runs in polynomial time when given access to the oracle $\mathcal{O}$. We now describe our oracle-efficient algorithm.

5.3.2 Algorithm

We use Algorithm 4 as our black box experts algorithm. We prove that Algorithm 4 has low regret.

Theorem 5.8. *The regret and expected number of switches can be bounded by* $O\left(\sqrt{\frac{L^*|S|\log|S||A|}{\alpha}}\right)$.

Algorithm 4: Black Box FPL algorithm used for Communicating MDPs with exploring starts

Sample perturbation vectors $\epsilon \in \mathbb{R}^{|S||A|}$ from an exponential distribution with parameter λ;
while $t \neq T+1$ **do**

$$\pi_t = \arg\min_{\pi \in \Pi} \mathbb{E}\left[\epsilon(s_1, a_1) + \sum_{i=1}^{t-1} \ell_t(s_t, a_t)\right]$$

with $s_1 \sim d_1$ and $a_t = \pi(s_t)$;
Adversary returns loss function ℓ_t;

The rest of the algorithm is the same as Algorithm 3. The exploring starts assumption allows us to get an efficient low regret, low switching algorithm assuming access to the oracle $\mathcal{O}$:

Theorem 5.9. *Given a communicating MDP satisfying Assumption 5.1 with a start distribution with at least probability α on every state, and given access to the oracle $\mathcal{O}$, we have an efficient algorithm with*

$$Regret \leq O\left(D^2\sqrt{\frac{L^*|S|\log|S||A|}{\alpha}}\right)$$

Proof. The proof is exactly the same as that of Theorem 5.5 except that we use the switching cost bound from Theorem 5.8. □

6 CONCLUSION

We considered learning in a communicating MDP with adversarially chosen costs in the full information setting. We gave an efficient algorithm that achieves $O(\sqrt{T})$ regret when transitions are deterministic. We also presented an inefficient algorithm that achieves a $O(\sqrt{T})$ regret bounds for the general stochastic case with an extra mild assumption. Our result show that in the full information setting there is *no statistical price* (as far as the time dependence is concerned) for the extension from the vanilla online learning with experts problem to the problem of online learning with communicating MDPs.

Several interesting questions still remain open. First, what are the best lower bounds in the general (i.e., not necessarily deterministic) communicating setting? In the deterministic setting, diameter is bounded polynomially by the state space size. This is no longer true in the stochastic case. The best lower bound in terms of diameter and other relevant quantities ($|S|$, $|A|$ and T) still remains to be worked out. Second, is it possible to design an efficient algorithm beyond the deterministic case with fewer assumptions? The source of inefficiency in our algorithm is that we run FPL with each

policy as an expert and perturb the losses of each policy independently. It is plausible that an FPL algorithm that perturbs losses (as in the deterministic case) can also be analyzed. However, there are challenges in its analysis as well as in proving that it is computationally efficient. For example, we are not aware of any efficient way to compute the best deterministic policy in hindsight for the general communicating case. This leads us to another open question: are there any oracle-efficient $O(\sqrt{T})$ regret algorithms that do online learning over communicating MDPs. Dai et al. [2022] give an oracle-efficient $O(T^{5/6})$ regret algorithm but that works for bandits as well and does not use the additional information that is there in the full information case.

Acknowledgements. AT acknowledges the support of NSF via grant IIS-2007055. Thanks to Csaba Szepesvari and Brian Denton for helpful discussions about the computational complexity of finding the best stationary policy in hindsight.

References

Raman Arora, Ofer Dekel, and Ambuj Tewari. Deterministic mdps with adversarial rewards and bandit feedback. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, page 93–101, Arlington, Virginia, USA, 2012a. AUAI Press. ISBN 9780974903989.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2, 06 2012b.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331, 1995. doi: 10.1109/SFCS.1995.492488.

Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows*. Wiley-Interscience, USA, 2004. ISBN 0471485993.

Adam Block, Yuval Dagan, Noah Golowich, and Alexander Rakhlin. Smoothed online learning is as easy as statistical learning, 2022. URL https://arxiv.org/abs/2202.04690.

P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 2000.

Yan Dai, Haipeng Luo, and Liyu Chen. Follow-the-perturbed-leader for adversarial markov decision processes with bandit feedback. *Advances in Neural Information Processing Systems, NeurIPS 2022*, 2022.

Ofer Dekel and Elad Hazan. Better rates for any adversarial deterministic mdp. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–675–III–683. JMLR.org, 2013.

Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: $t^{2/3}$ regret. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 10 2013. doi: 10.1145/2591796.2591868.

Eric V. Denardo. Periods of connected networks and powers of nonnegative matrices. *Mathematics of Operations Research*, 2(1):20–24, 1977. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3689120.

Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E. Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design, 2016. URL https://arxiv.org/abs/1611.01688.

Eyal Even-Dar, Sham. M. Kakade, and Yishay Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/40538442.

Nika Haghtalab, Yanjun Han, Abhishek Shetty, and Kunhe Yang. Oracle-efficient online learning for beyond worst-case adversaries, 2022. URL https://arxiv.org/abs/2202.08549.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2004.10.016. URL https://www.sciencedirect.com/science/article/pii/S0022000004001394. Learning Theory 2003.

Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon markov decision process problems. *J. ACM*, 47(4):681–720, 2000. doi: 10.1145/347476.347480. URL https://doi.org/10.1145/347476.347480.

Gergely Neu, András György, Csaba Szepesvári, and András Antos. Online markov decision processes under bandit feedback. *IEEE Trans. Autom. Control.*, 59(3):676–691, 2014. doi: 10.1109/TAC.2013.2292137. URL https://doi.org/10.1109/TAC.2013.2292137.

Ronald Ortner. Online regret bounds for markov decision processes with deterministic transitions. *Theoretical Computer Science*, 411(29):2684–2695, 2010. ISSN 0304-3975. doi: https://doi.org/10.1016/j.tcs.2010.04.005. URL https://www.sciencedirect.com/

science/article/pii/S0304397510002008.
Algorithmic Learning Theory (ALT 2008).