Better Rates for Any Adversarial Deterministic MDP

Ofer Dekel OFERD@MICROSOFT.COM
Microsoft Research, 1 Microsoft Way, Redmond, WA 98052, USA

Elad Hazan EHAZAN@IE.TECHNION.AC.IL
Technion - Israel Inst. of Tech., Haifa 32000, Israel

Abstract

We consider regret minimization in adversarial deterministic Markov Decision Processes (ADMDPs) with bandit feedback. We devise a new algorithm that pushes the state-of-the-art forward in two ways: First, it attains a regret of $O(T^{2/3})$ with respect to the best fixed policy in hindsight, whereas the previous best regret bound was $O(T^{3/4})$. Second, the algorithm and its analysis are compatible with any feasible ADMDP graph topology, while all previous approaches required additional restrictions on the graph topology.

1. Introduction

A sequential decision making problem deals with a decision maker's extended interaction with his environment. The decision maker can take different actions that influence his state in the environment, and each action can have both short and long term effects on the decision maker's utility. For example, imagine a robotic vacuum cleaner that travels around the house and dynamically makes turn-by-turn decisions about its route. The robot is the decision maker, the house is the environment, the utility is the amount of dirt collected, and the robot's state in this example is his location. The robot knows its state and can take action to move to an adjacent state. Not all actions are available in all states and the consequence of each action depends on the current state.

Sequential decision making problems can be formulated in various ways, with different assumptions on the dynamics of the environment and on the information available to the decision maker. In this paper, we focus on one such setting: the *adversarial deterministic Markov decision process with bandit feedback*, abbreviated by *ADMDP*. Namely, we assume that the environment is adversarial, the state transition dynamics of the environment are deterministic, and the feedback observed by the decision maker is *bandit feedback* (all of these terms are explained below).

We model the sequential decision making problem as a game between a randomized game player (the decision maker) and a deterministic adversary (the environment). First, the player and the adversary are told the total number of steps in the game, T. Then, they are given a directed graph $G = (\mathcal{V}, \mathcal{E})$ called the *state transition graph*, where $\mathcal{V}$ is a finite set of vertices or *states*, and $\mathcal{E}$ is a set of directed edges, with at least one outgoing edge per state. Moreover, one of the states in $\mathcal{V}$ is designated as the *initial state*, and denoted by v_0.

Next, the adversary defines a sequence of T *loss functions*, $f_1, \ldots, f_T$, where each f_t takes the form $f_t : \mathcal{E} \mapsto [0, 1]$. In other words, the adversary assigns a loss value to each edge at each step in the game. The loss functions are not revealed to the player.

Finally, the player takes T steps along the edges of the graph, starting from the initial state v_0. Specifically, at each step $t \in 1, \ldots, T$, the player begins in state $V_{t-1} \in \mathcal{V}$, chooses one of V_{t-1}'s outgoing edges, and traverses that edge to a new state $V_t \in \mathcal{V}$. The chosen edge is called the player's *action*. Since the player has the power of randomization, he chooses his action by defining a distribution over V_{t-1}'s outgoing edges and sampling a concrete edge from that distribution. Therefore, the player's state after step t is a random variable. Some of the edges in $\mathcal{E}$ may be self-loops, in which case V_t can equal V_{t-1}. While traversing the edge (V_{t-1}, V_t), the player suffers a loss of $f_t(V_{t-1}, V_t)$. The player observes this loss value, but he does not observe the loss that he would have suffered had he chosen a different action or had he been in a differ-

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

ent state; this feedback model is commonly called the *bandit* feedback model.

We assume that the adversary has unlimited computational power and may even know the player's algorithm. The adversary may use random bits if he so desires, but since he already has unlimited computational power, random bits do not give him an additional advantage. Despite the adversary's power, he must choose the entire sequence of loss functions before the player makes a move and without knowing the player's random choices (in other words, the adversary knows the player's algorithm but he doesn't know the player's random bits). This type of adversary is called an *oblivious adversary* or a *non-adaptive adversary.* Although the adversary is oblivious, the player's loss at time t still depends on his past actions, since the loss depends on the player's state, which is determined by his past actions. Therefore, from the player's point of view, the environment does seem to react to his past actions.

The player's goal is to accumulate the smallest possible loss as he performs his T-step path over the graph. Although the loss functions are deterministic, the player's loss depends on his randomized actions. Therefore, our analysis focuses on the player's *expected* cumulative loss. Since the loss functions are adversarial, the player's loss is only meaningful when compared to a baseline (since the adversary could always assign the maximal loss to all actions at all states and at all steps); in this paper we compare the player's loss to the loss of the best fixed policy in hindsight.

Formally, let Π be the set of all fixed (deterministic) policies, where each $\pi \in \Pi$ is a mapping from $\mathcal{V}$ to itself that maps each state v to one of its outgoing neighbors. A deterministic policy π starts in the initial state v_0, transitions to $\pi(v_0)$, then to $\pi(\pi(v_0))$, and so on. This motivates us to define the shorthand

$$\pi^t(v) \;=\; \underbrace{\pi(\cdots\pi}_{t}(v)) \quad .$$

Using this notation, the loss accumulated by the fixed policy π equals $\sum_{t=1}^{T} f_t(\pi^{t-1}(v_0), \pi^t(v_0))$. We now define the player's (undiscounted) *regret*, denoted by $R_{\mathrm{ADMDP}}(T)$, as the difference between his expected cumulative loss and the loss of the best fixed policy in hindsight. Formally, $R_{\mathrm{ADMDP}}(T)$ is defined as

$$\mathbf{E}\left[\sum_{t=1}^{T} f_t(V_{t-1}, V_t)\right] \;-\; \min_{\pi\in\Pi}\sum_{t=1}^{T} f_t(\pi^{t-1}(v_0), \pi^t(v_0)) \quad .$$

We say that the player is *learning* if $R_{\mathrm{ADMDP}}(T)$ is upper-bounded by a sub-linear function of T, uniformly for all sequences of loss functions. A sub-linear regret implies that the average expected loss suffered by the player on each individual step tends to zero, so the player gets better with time. On the other hand, the player isn't learning if $R_{\mathrm{ADMDP}}(T)$ keeps growing linearly, even as T tends to infinity. Our technical goal is to prove bounds of the form $R_{\mathrm{ADMDP}}(T) = O(T^q)$ where $q \in [0, 1)$; a smaller value of q implies faster learning.

Any fixed policy in Π takes at most $|\mathcal{V}|$ steps before returning to a state that it has previously visited. From that point on, the fixed policy repeats the same cycle of states over and over again. However, note that two policies in Π that lead to the same cycle can still accumulate different losses if they enter the cycle at different times or at different states. To demonstrate this concept, consider the complete graph over three states, $\mathcal{V} = \{v_0, v_1, v_2\}$. Notice that there are two policies that lead to the 2-cycle v_1, v_2: one that traverses the edge (v_1, v_2) on odd steps and one that does so on even steps. Now, define for all t

$$f_t(v, u) \;=\; \begin{cases} t \pmod 2 & \text{if } v = v_1 \\ t+1 \pmod 2 & \text{otherwise} \end{cases} \quad .$$

While the policy that traverses the edge (v_1, v_2) on odd steps suffers a total loss of $\theta(T)$, the policy that traverses the edge (v_1, v_2) on even steps suffers a total loss of $\theta(1)$. Therefore, it is insufficient to merely discover low-loss cycles in G, and we must also think of a policy's phase within a cycle.

We make some assumptions on the structure of G, but we prove that they are necessary. Specifically, we assume that G contains exactly one strongly connected component (see definition below), and we prove that without this assumption, there does not exist any algorithm that guarantees a sub-linear regret for all loss sequences. With this assumption alone, we present a new algorithm that guarantees a regret of $O(T^{2/3})$ against any sequence of loss functions. This proves that the assumption above is also a sufficient condition for learning in the ADMDP setting. To the best of our knowledge, our algorithm is the first to make do with the necessary condition on G, and all previous work on this topic requires more restrictive assumptions (see the related work section below for details). To simplify our presentation, we first solve the problem with the additional assumption that G is *aperiodic* (see definition below), but then we relax this assumption and default back to the necessary condition.

A more common way of formulating sequential decision making problems uses (stochastic) Markov decision processes (MDP). The classic MDP formulation assumes that the losses (or rewards) are stochastic (rather than adversarial) and the state transitions are noisy (rather than deterministically controlled by the player). Adversarial losses are strictly more general than stochastic ones, as the adversary is free to use random bits if he so chooses. Adversarial environments also generalize time-varying stochastic environments, strategic environments (like the ones encountered in multiplayer video games and online bidding systems), and malicious environments (like the ones encountered in spam and fraud detection systems). In this sense, our setting is strictly more powerful than the MDP setting. On the other hand, the deterministic state transitions in the ADMDP model are weaker than the noisy transitions in the MDP model. In some cases, this may be a limitation of the ADMDP setting. However, in general, a stochastic system is often well approximated by a deterministic system (Bertsekas, 2005, Chap. 2). Overall, the relatve strength of the ADMDP model versus the more traditional MDP model is not well understood.

Our assumption that the entire graph is known to the player beforehand is simply a matter of convenience. This assumption can be easily relaxed using ideas from Ortner (2010). Also, we chose to formulate the problem in terms of loss functions (rather then reward functions) for notational convenience; moving from rewards to losses and vice versa can be done by replacing each f_t with $1 - f_t$.

1.1. Related Work and Our Contribution

Our work is most closely related to the recent work in Arora et al. (2012), which addresses the same problem (with the minor technical difference that it formulates the game using rewards rather than losses). Arora et al. (2012) presents an algorithm called *MarcoPolo*, with a regret bound of $O(T^{3/4})$, which holds for any strongly connected aperiodic graph. In contrast, our new algorithm guarantees a better regret of $O(T^{2/3})$, without the restriction to aperiodic graphs. We note that the slow learning rate attained by MarcoPolo is not due to a loose analysis, but is an inherent limitation of the algorithm: MarcoPolo is built of a top-layer multi-armed bandit algorithm and a bottom layer online linear bandit algorithm; the latter is reset $\sqrt{T}$ times during the T steps, and must learn everything from scratch each time; this algorithmic construction implies that the $O(T^{3/4})$ regret guarantee is the best possible for MarcoPolo.

More generally, there is an abundance of previous research on MDPs with stochastic transitions and adversarial rewards (Even-Dar et al., 2009; Yu et al., 2009; Yu & Mannor, 2009; Neu et al., 2010). Moreover, the typical regret rates in this setting are $O(T^{1/2})$. Since stochastic transitions are strictly more general than deterministic transitions, it would seem that these papers solve a more general problem with a better regret guarantee. However, all of these papers make an additional assumption that turns out to be extremely restrictive when the transitions are deterministic. Specifically, the results for stochastic MDPs require that the state transition dynamics have a *unichain* structure, which means that *every* policy must lead to a Markov chain with a single recurrent class. In other words, they assume that the underlying Markov chain is uniformly ergodic under any policy. In the special case of deterministic transitions, the unichain assumption implies that any two cycles in the graph share a common vertex (Feinberg & Yang, 2008). Clearly, this assumption excludes most of the deterministic state transition graphs that one could imagine; for example, it excludes any graph with two self-loops. Therefore, these results are typically inapplicable in our deterministic setting. For more details on the limitations imposed by the unichain assumption in the ADMDP setting, see (Arora et al., 2012). As noted above, our new algorithm applies to any state transition graph that satisfies the necessary condition.

More generally, there is relevant previous work on the connections between reinforcement learning and individual sequence prediction in information theory (Farias et al., 2010), as well as work on regret minimization in stochastic and deterministic MDPs when the rewards are stochastic, rather than adversarial (see Szepesvari (2010) and the references therein, as well as (Ortner, 2010)).

2. Preliminaries

We begin with some mathematical background on directed graphs. Let $G = (\mathcal{V}, \mathcal{E})$ be a directed graph. A set of vertices $\mathcal{V}' \subseteq \mathcal{V}$ is *strongly connected* if G contains a directed path between any two vertices in $\mathcal{V}'$. We say that G is a strongly connected graph if its vertices $\mathcal{V}$ form a strongly connected set. A *cycle* in G from v to itself is a path in G that starts and ends at v. If G is strongly connected, it contains at least one cycle from any vertex v. A *strongly connected component* of G is a maximal strongly connected set of vertices. That is, if we add any vertex to a strongly connected component, it will no longer be strongly connected.

Assume that G is strongly-connected and consider the set of cycles of length at most $|\mathcal{V}|$ from v to itself. Let $\gamma(v)$ be the greatest common divisor (gcd) of the lengths of these cycles; this quantity is called the *period* of the vertex v. It can be shown that all of the vertices of a strongly connected graph have the same period (Bremaud, 1999, Chap. 2, Thm. 4.2), so period is actually a property of G rather than of v. For brevity, we denote the period of G by γ. If $\gamma = 1$, we say that G is an *aperiodic* graph.

A strongly connected graph G with period γ can always be arranged in a *cyclic structure* (Bremaud, 1999, Chap. 2, Thm. 4.1) of length γ. That is, the graph vertices can be uniquely partitioned into γ non-empty sets, called *cyclic classes* and denoted by $\mathcal{C}_0, \ldots, \mathcal{C}_{\gamma-1}$, such that the outgoing edges from the vertices in $\mathcal{C}_i$ all lead to vertices in $\mathcal{C}_{i+1}$ (for all i, where $i + 1$ is computed modulo γ). If G is aperiodic, then all of its vertices belong to a single cyclic class and the above holds trivially. In other words, if the vertices of G cannot be partitioned into multiple cyclic classes, as described above, then its period is necessarily $\gamma = 1$. From the above, we can conclude that the length of every cycle in a strongly connected graph with period γ is an integer multiple of γ.

If G is strongly connected and aperiodic (namely, $\gamma = 1$), there exists a critical length d such that for any $s \geq d$ there are paths in G of length s between any pair of vertices (Denardo, 1977). This result is a consequence of the *Frobenius coin exchange problem* in combinatorics, and specifically of Schur's theorem (Alfonsin, 2005). Moreover, the critical length d is never too big: letting $n = |\mathcal{V}|$, it is known that $d \leq n(n-1)$ (Denardo, 1977).

If G has a period of $\gamma > 1$, the upper bound on d can be generalized as follows: there exists a critical value d such that for any integer $s \geq d$ there is a path in G of length $s\gamma$ from any state v to any other state *in the same cyclic class*. To prove this generalization, let $\mathcal{C}_i$ denote the cyclic class to which v belongs. Construct a new graph $G' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' = \mathcal{C}_i$ and where $\mathcal{E}'$ contains the edge (u, v) if and only if there exists a path of length γ from u to v in G. Now note that G' is aperiodic: the greatest common divisor of cycles in G is γ so the greatest common divisor of the cycles in G' is 1. The generalized theorem applied to G follows from applying the original theorem to G'.

3. Necessary Conditions on G

As previously mentioned in the introduction, we make some assumptions on the topology of G. First, recall our assumption that every state has at least one outgoing edge - clearly this assumption is required for the ADMDP game to be well defined. Next, we assume that all of the states are reachable from v_0. This assumption is made without loss of generality, since otherwise we can simply remove the unreachable states without changing the sequential decision problem.

These two assumptions already imply that G containts at least one strongly connected component that is reachable from v_0. The reasoning is straightforward: there exists a path of length $|\mathcal{V}|$ that starts at v_0; this path must return to a state that it previously visited; this creates a cycle, which is a strongly connected set.

We add a third assumption on the topology of G: we assume that G contains exactly one strongly connected component, and no more. In contrast to the previous assumptions, which did not restrict generality, this assumption limits the graph topologies that we can handle. However, we prove this assumption is a necessary condition for any learning algorithm in this setting. Namely, if G contains two or more strongly connected components, a sub-linear upper bound on regret is unattainable by any algorithm.

Theorem 3.1. *If G has two (or more) strongly connected components, for any algorithm used by the player, there exists an oblivious sequence of loss functions that inflicts a regret of $\Omega(T)$.*

The proof is deferred to the appendix.

Fig. 1 shows an example of a graph that satisfies the conditions of our algorithm. It contains a single strongly connected component and has a period of 3.

Theorem 3.1 proves that having a single strongly connected component is a necessary condition. With this assumption in place, we can further simplify our presentation by assuming that all of the vertices in G are strongly connected. Given the previous assumptions, this assumption can be made without further loss of generality. To see why, assume that G is not strongly connected but contains exactly one strongly connected component. Note that any vertex that does not belong to the strongly connected component is unreachable after the first n steps (recall that $n = |\mathcal{V}|$). A precise proof of this is straightforward: a vertex that does not belong to a strongly connected component can only be visited once; therefore, any path of length n must end inside the strongly connected component; once the path enters the strongly connected component it can never leave. Therefore, the player can begin the game by performing n arbitrary actions (at worst, adding a constant to the cumulative loss). After that, he never has to worry about the vertices that are not part of the connected component. In other words, we assume, without any additional loss of generality, that G is strongly connected to begin with.

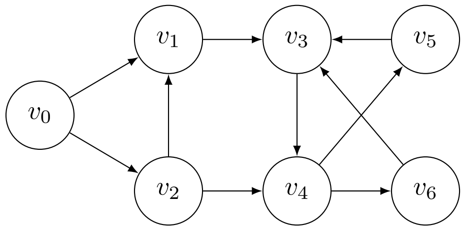

Figure 1. Example of a state transition graph that satisfies the conditions of our algorithm. It has a single strongly connected component and a period of $\gamma = 3$. Without loss of generality, states v_0, v_1, v_2 can be ignored, since each of them can only be visited once, resulting in a strongly connected graph.

4. An Algorithm for Aperiodic Graphs

In this section, we add the simplifying assumption that the state transition graph G is aperiodic. We will relax this assumption in a later section. As explained in Sec. 2, this assumption implies that there exists a path in G of length $d = n(n-1)$ between any pair of states.

We solve the ADMDP problem by reducing it to the *bandit shortest path* (BSP) problem. BSP is also modeled as a repeated game between an adversary and a player. The problem is defined by a directed graph $G^\star = (\mathcal{V}^\star, \mathcal{E}^\star)$ and two special vertices, a *source vertex* $s \in \mathcal{V}^\star$, and a *target vertex* $t \in \mathcal{V}^\star$. We assume that G contains at least one path from s to t.

The game is played for J rounds (the move from T to J is deliberate, and hints that our reduction will change the time scale of the original T-step ADMDP game). As in the ADMDP game, the adversary defines the entire sequence of loss functions before the player takes any action. To facilitate the presentation of our reduction, we denote the sequence of loss functions for the BSP problem by $g_1, \ldots, g_J$, where each $g_j : \mathcal{E} \mapsto \mathbb{R}_+$. On round j of the game, the player chooses a path P_j from s to t and suffers a loss equal to the sum of the losses on the edges in P_j. We overload our notation and for any path p we define

$$g_j(p) \;=\; \sum_{(u,w)\in p} g_j(u,w) \quad .$$

As its name implies, the bandit shortest path game is played with bandit feedback, so the player observes his loss but not the loss that he would have suffered had he chosen a different path. We note that our reduction actually results in an instance of the *semi-bandit shortest path* (SBSP) problem, where the player observes the loss along each edge of his path, but for simplicity, we present our results in terms of the BSP problem.

We define the player's regret by comparing his loss to the loss of the best fixed path from s to t. Formally, let $\mathcal{P}$ be the set of all paths from s to t, and define the player's regret as

$$R_{\mathrm{BSP}}(J) \;=\; \mathbf{E}\left[\sum_{j=1}^{J} g_j(P_j)\right] \;-\; \min_{p\in\mathcal{P}} \sum_{j=1}^{J} g_j(p) \quad .$$

Note that the BSP problem is a stateless online decision problem, which means that the player's choices on round j are not constrained by his choices in the past. In other words, by reducing ADMDP to BSP, we remove one of the most problematic aspects of ADMDP – the player's state.

4.1. The Reduction

We are now ready to construct the reduction from ADMDP to BSP. We use $[i]_k$ to denote the integer in $\{1, \ldots, k\}$ that satisfies $[i]_k - 1 = (i-1)$ modulo k. In other words,

$$[i]_k \;=\; \begin{cases} i & \text{if } i \le k \\ [i-k]_k & \text{otherwise} \end{cases} \quad .$$

We take the state transition graph $G = (\mathcal{V}, \mathcal{E})$ from the definition of the ADMDP problem and use it to define n^2 new graphs (where, again, $n = |\mathcal{V}|$): for each state $v \in \mathcal{V}$ and each integer $k \in \{1, \ldots, n\}$ we define the graph $G^{v,k} = (\mathcal{V}^{v,k}, \mathcal{E}^{v,k})$. The vertex set $\mathcal{V}^{v,k}$ includes $2 + (k-1)n$ vertices: two copies of v and $k-1$ copies of the entire set $\mathcal{V}$. The two copies of v are denoted by $v^{v,k,0}$ and $v^{v,k,k}$. The i-th copy of $u \in \mathcal{V}$ is denoted by $u^{v,k,i}$. Note that each vertex has three superscripts: the first two identify the graph, while the third distinguishes between the different copies of the original state.

Next, we construct the set of edges $\mathcal{E}^{v,k}$. For each of v's outgoing edges in G, $(v, u) \in \mathcal{E}$, we add the edge $(v^{v,k,0}, u^{v,k,1})$ to $\mathcal{E}^{v,k}$. For each of v's incoming edges in G, $(u, v) \in \mathcal{E}$, we add the edge $(u^{v,k,k-1}, v^{v,k,k})$ to $\mathcal{E}^{v,k}$. Finally, for each edge $(u, w) \in \mathcal{E}$ (including the incoming and outgoing edges of v), we add the set of edges $\{(u^{v,k,i-1}, w^{v,k,i})\}_{i=2}^{k-1}$ to $\mathcal{E}^{v,k}$.

Our notation is unavoidably tedious, but the construction itself is quite straightforward. We illustrate our construction with a simple example in Fig. 2.

Finally, we unify the n^2 induced graphs into one big graph $G^\star = (\mathcal{V}^\star, \mathcal{E}^\star)$. Specifically, we add a source node s and a target node t, and for each $v \in \mathcal{V}$ and $k \in \{1, \ldots, n\}$ we add an edge from s to $v^{v,k,0}$ and

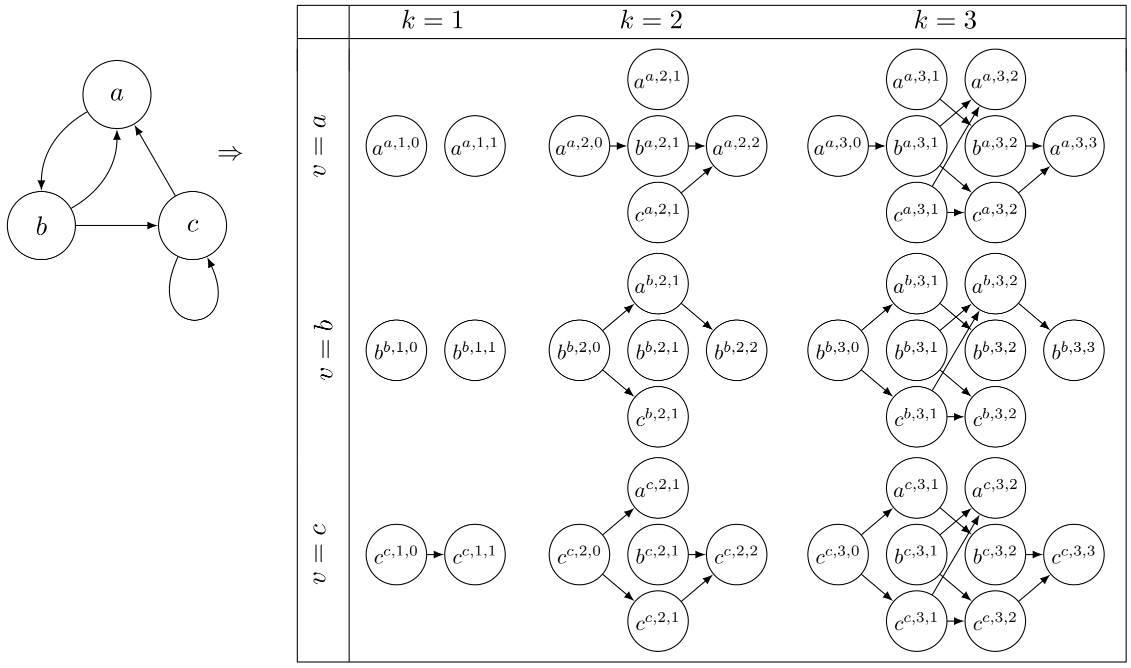

Figure 2. Example of a state transition graph G (left) and the induced graphs $G_{v,k}$, for all $v \in \mathcal{V}$ and $k \in \{1, \ldots, n\}$ (right). Note that the cycle $a \to b \to c$ in G induces the paths $a^{a,3,0} \to b^{a,3,1} \to c^{a,3,2} \to a^{a,3,3}$ in $G^{a,3}$, $b^{b,3,0} \to c^{b,3,1} \to a^{b,3,2} \to b^{b,3,3}$ in $G^{b,3}$, and $c^{c,3,0} \to a^{c,3,1} \to b^{c,3,2} \to c^{c,3,3}$ in $G^{c,3}$.

another edge from $v^{v,k,k}$ to t. More formally, we define

$$\mathcal{V}^\star = \big(\cup_{v,k} \mathcal{V}^{v,k} \big) \cup \{s, t\} \quad \text{and}$$
$$\mathcal{E}^\star = \big(\cup_{v,k} \mathcal{E}^{v,k} \big) \cup \{(s, v^{v,k,0})\}_{v,k} \cup \{(v^{v,k,k}, t)\}_{v,k} \quad .$$

The important thing to note is that each cycle of length k in the original graph G induces k disjoint paths in $G^\star$ from s to t, one for each phase (i.e., each starting point) of the cycle. Specifically, the cycle in G that includes the states $u_1, \ldots, u_k$ induces a path from s to t through each of the subgraphs $G^{u_1,k}, \ldots, G^{u_k,k}$. On the other hand, every path in $G^\star$ from s to t corresponds to a cycle and a phase in the original graph G.

Next, we describe how a ADMDP player uses an online algorithm for BSP to determine his actions in the ADMDP game. First, the player splits the T steps of the ADMDP game into epochs of length τ, where τ is a positive integer that we will specify later on. Assume, without loss of generality, that τ divides T, and let $J = T/\tau$ be the number of epochs. Note that epoch j starts on step $(j-1)\tau + 1$ and ends on step $j\tau$.

The player invokes the BSP algorithm once per epoch on the graph $G^\star$; at the beginning of epoch j, the BSP algorithm chooses a path P_j from s to t. By construction, there must be a state $u_1 \in \mathcal{V}$ such that P_j starts with the edge $(s, u_1^{u_1,k,0})$ and ends with the edge $(u_1^{u_1,k,k}, t)$. Therefore, let $(s, u_1^{u_1,k,0}, u_2^{u_1,k,1}, \ldots, u_k^{u_1,k,k-1}, u_1^{u_1,k,k}, t)$ denote the sequence of vertices in P_j. The player erases the superscripts from this sequence, as well as the source and target nodes, and is left with the state sequence $u_1, \ldots, u_k$. By construction, this sequence forms a feasible cycle in the state transition graph G, in one of the k possible phases.

Recall that any state in G can be reached from any other state in exactly d steps. The player spends the first d steps in the epoch moving from his current state towards the cycle $u_1, \ldots, u_k$. Specifically, on step $(j-1)\tau + d$ the player needs to arrive in state $u_{[d]_k}$. Although the player suffers a loss on these d initial steps, and this loss is accounted for in the regret analysis, the player simply ignores it. Since there are J epochs in the entire game, the player ignores the loss on at most Jd steps.

The player spends the rest of the epoch traversing the cycle $u_1, \ldots, u_k$ over and over again. On these steps, the player keeps track of the sum of losses. At the end of the epoch, the BSP algorithm expects a feedback from the environment, which represents the loss assigned to its chosen path P_j. The player provides the accumulated sum of losses as the feedback. The BSP algorithm uses this feedback to update itself and then chooses the next cycle P_{j+1}.

Our first technical goal is to prove that, from the point of view of the BSP algorithm, it is playing a standard BSP game against a predefined sequence of loss functions $g_1, \ldots, g_J$. To this end, for each $j \in \{1, \ldots, J\}$, $v \in \mathcal{V}$, and $k \in \{1, \ldots, n\}$, define $g_j(s, v^{v,k,0}) = 0$ and $g_j(v^{v,k,k}, t) = 0$. Also, for each j, v, k as above, $(u, w) \in \mathcal{E}$, and $i \in \{2, \ldots, k-1\}$, define

$$g_j(u^{v,k,i-1}, w^{v,k,i}) \ = \sum_{t=(j-1)\tau+1+d}^{j\tau} f_t(u, w) \ \mathbb{1}_{[t]_k = i} \ . \quad (1)$$

The above is a complete definition of g_j, for every edge in $\mathcal{E}^\star$. Note that this definition is independent of the player's actions, and can therefore be specified at the beginning of the game (obliviously). The following lemma proves that the feedback provided to the BSP algorithm equals $g_j(P_j)$.

Lemma 4.1. *Let P_j be the path chosen by the BSP algorithm on epoch j and let g_j be as defined in (1). Assume that the player uses P_j to transition through the ADMDP graph G and accumulates losses, as described in the reduction above. Then the loss accumulated by the player and reported to the BSP algorithm at the end of epoch j equals $g_j(P_j) = \sum_{(u,w)\in P_j} g_j(u, w)$.*

Proof. Recall that the actual loss of the player, aside from the first d steps in the epoch in which the player is moving to the start vertex, is given by

$$\sum_{t=(j-1)\tau+1+d}^{j\tau} f_t(V_{t-1}, V_t) \ \ ,$$

where V_t is the state at time t. Let $(s, u_1^{u_1,k,0}, u_2^{u_1,k,1}, \ldots, u_k^{u_1,k,k-1}, u_1^{u_1,k,k}, t)$ denote the sequence of vertices in P_j that correspond to the phased cycle $u_1, \ldots, u_k$ in the original graph. Thus, for the time periods in this epoch we have $(V_t, V_{t+1}) = (u_{[t]_k}, u_{[t+1]_k})$, hence

$$\begin{aligned} &\textstyle\sum_{t=(j-1)\tau+1+d}^{j\tau} f_t(V_{t-1}, V_t) \\ &= \textstyle\sum_{t=(j-1)\tau+1+d}^{j\tau} f_t(u_{[t]_k}, u_{[t+1]_k}) \\ &= \textstyle\sum_{i=1}^{k-1} \sum_{t=(j-1)\tau+1+d}^{j\tau} f_t(u_i, u_{i+1}) 1_{[t]_k = i} \\ &= \textstyle\sum_{i=1}^{k-1} g_j(u_i, u_{i+1}) = g_j(P_j) \ \ . \end{aligned}$$

□

4.2. Regret Upper Bound

After specifying the reduction to BSP, we state and prove our main result. Let $R_{ADMDP}(T)$ be the regret incurred by the ADMDP player that uses the technique defined in the previous section and let R_{BSP} be the regret of the underlying BSP algorithm. The next theorem bounds $R_{ADMDP}(T)$ in terms of R_{BSP}. After stating and proving this theorem, we derive concrete bounds on $R_{ADMDP}(T)$ using known bounds on R_{BSP}.

Theorem 4.1. *Let $\mathcal{A}$ be a given algorithm for the BSP (or SBSP) problem with a regret bound of:*

$$R^{\mathcal{A}}_{\text{BSP}}(J) = \mathbf{E}\left[\sum_{j=1}^{J} g_j(P_j)\right] - \min_{p\in\mathcal{P}} \sum_{j=1}^{J} g_j(p) \ \ .$$

Then, the regret of our ADMDP algorithm is bounded by

$$R_{ADMDP}(T) \le \frac{T}{J} \cdot R^{\mathcal{A}}_{\text{BSP}}(J) + Jd + n^2 + \frac{T}{J} \ \ .$$

Proof. Recall that there exists a path of length d between any two states in G, and that $d < n^2$. Assume w.l.o.g. that $n^2 \le \tau \le T$, otherwise the theorem holds trivially.

Recall the definition of $R_{\text{ADMDP}}(T)$ as

$$\mathbf{E}\left[\sum_{t=1}^{T} f_t(V_{t-1}, V_t)\right] - \min_{\pi\in\Pi} \sum_{t=1}^{T} f_t(\pi^{t-1}(v_0), \pi^t(v_0)) \ \ .$$

As detailed previously, the optimal policy π is, up to an initialization phase of length at most $|V|^2 = n^2$, a phased cycle in the graph which we denote as $P^* = (v_1, v_2, ..., v_k)$. Thus, we can write the loss of π as

$$\begin{aligned} &\textstyle\sum_{t=1}^{T} f_t(\pi^{t-1}(v_0), \pi^t(v_0)) \\ &\ge \textstyle\sum_{j=1}^{J} (g_j(P^*) + \sum_{t=(j-1)\tau+1}^{d} f_t(\pi^{t-1}(v_0), \pi^t(v_0))) \\ &\ge \textstyle\sum_{j=1}^{J} g_j(P^*) \ \ . \end{aligned}$$

In the last inequality we used the fact that the losses are nonnegative. Here $J = \frac{T}{\tau}$ is the partition of time into epochs of length $\tau \ge d$. We assume that T is an integer multiple of τ, else incur an addition τ regret by ignoring the last at most τ game steps.

By construction, the loss incurred by our algorithm is exactly equal to the loss incurred by the paths chosen by the BSP algorithm with the addition of the first d steps of each epoch, in which the ADMDP algorithm incurs an additional loss of at most d. Thus,

$$\mathbf{E}\left[\sum_{t=1}^{T} f_t(V_{t-1}, V_t)\right] \le \sum_{j=1}^{J} g_j(P_j) + J \cdot d \ \ .$$

Combining the previous two observations, we have

$$\begin{aligned} &R_{\text{ADMDP}}(T) \\ &\le \textstyle\sum_{j=1}^{J} g_j(P_j) + J \cdot d - \sum_{j=1}^{J} g_j(P^*) + n^2 + \tau \\ &= \tau \cdot R^{\mathcal{A}}_{\text{BSP}}(J) + Jd + n^2 + \tau \\ &= \tfrac{T}{J} \cdot R^{\mathcal{A}}_{\text{BSP}}(J) + Jd + n^2 + \tfrac{T}{J} \ \ , \end{aligned}$$

where the first equality holds since the edge costs of the BSP problem are now bounded by $\tau = \frac{T}{J}$ rather than one, as the costs per edge are summed up along the epoch. By the additive definition of regret, this increases the regret of the BSP algorithm by a multiplicative factor of at most $\frac{T}{J}$. $\square$

The final regret bound we obtain depends on the underlying BSP (or SBSP) algorithm used in our construction. Possible choices are the BSP algorithms of Abernethy et al. (2012); Dani et al. (2007) or the SBSP algorithm of Audibert et al. (2011).

Theorem 4.2 (Audibert et al. (2011))**.** *For the SBSP problem on graphs with n vertices and m edges, the algorithm presented in Audibert et al. (2011) runs in polynomial time and guarantees a regret of $R_{SBSP}(J) = O(m\sqrt{J})$.*

Thus, we obtain

Corollary 4.1. *The ADMDP algorithm attains $R_{ADMDP}(T) = O(n^2 m T^{2/3})$.*

Proof. The graph $G^\star$ constructed in the reduction has $O(n^2 m)$ edges, where n, m are the number of vertices and edges in G. Thus, applying Theorem 4.1, we get

$$\begin{aligned} R_{ADMDP}(T) &\leq \tfrac{T}{J} \cdot O(n^2 m\sqrt{J}) + \tfrac{T}{J} + J \cdot d + n^2 \\ &= O(\tfrac{n^2 m T}{\sqrt{J}} + \tfrac{T}{J} + J n^2) \quad . \end{aligned}$$

The corollary is obtained by taking $J = T^{2/3}$. $\square$

5. Extension to Periodic Graphs

In the previous section, we focused only on strongly connected aperiodic graphs. In this section, we deal with graphs with arbitrary period. Recall our assumption (made without loss of generality) that G is strongly connected and that our goal is to compete with the best fixed policy in hindsight. When a strongly connected graph has a period of $\gamma \geq 1$, we know that the length of every cycle in the graph is a multiple of γ (see Sec. 2). This already gives us some useful information about the path induced by the best fixed policy. Moreover, we know that G has a cyclic structure. Let C_0 be the cyclic class that contains the initial vertex v_0. The cyclic structure of G implies that the best fixed policy starts in C_0 and necessarily returns to C_0 every γ steps.

These two facts allow us to refine our reduction. Instead of defining a graph $G^{v,k}$ for each state $v \in \mathcal{V}$ and each cycle length $k \in \{1, \dots, n\}$, we only need $G^{v,k}$ for each cycle length $k \in \{\gamma, 2\gamma, \dots, \lfloor n/\gamma \rfloor \gamma\}$ and for each $v \in C_0$ (for a total of $\lfloor n/\gamma \rfloor |C_0| \leq n^2/\gamma$ graphs). As in the previous section, we connect a common source s and target t to each graph, and name the resulting graph $G^\star$. As before, each path in $G^\star$ from s to t corresponds to a phased cycle in G that is attainable by a fixed policy, and vice versa.

As before, we split the T steps into epochs, making sure that the epoch length τ is a multiple of γ (increasing τ to a multiple of γ does not effect the regret rate). Additionally, recall that we begin each epoch by taking d steps toward the chosen cycle; now we must take $d\gamma$ steps. These choices guarantee that we start each epoch at a vertex in C_0 and that we finish the initial $d\gamma$ steps of the epoch in a vertex of C_0. We can now apply the reduction exactly as before.

6. Conclusions

We have modeled the sequential decision making problem as an ADMDP - an MDP with deterministic state transitions, adversarial losses, and bandit feedback. We presented a new algorithm that significantly improves on the state-of-the-art in terms of regret bounds. Moreover, it applies to the most general class of state transition graphs, whereas all previous algorithms relied on generality-limiting assumptions.

Several interesting questions remain open. Is a regret bound of $O(T^{1/2})$ possible in the ADMDP setting or is $O(T^{2/3})$ the best bound possible? Can our algorithm and analysis be extended to the more general case of stochastic state transitions, without reintroducing the restrictive unichain assumption? Finally, can we allow the adversary to influence the state transition dynamics? We leave these questions for future research.

Acknowledgments

We thank Alexander Holroyd for directing us to the coin exchange problem. Part of this research was conducted when the second author was a visiting researcher at Microsoft Research, Redmond. Elad Hazan is supported by the Technion-Microsoft Electronic Commerce Research Center.

References

Abernethy, J., Hazan, E., and Rakhlin, A. Interior-point methods for full-information and bandit online learning. *IEEE Transactions on Information Theory*, 58(7):4164–4175, 2012.

Alfonsin, J. Ramirez. *The Diophantine Frobenius problem.* Oxford University Press, 2005.

Arora, R., Dekel, O., and Tewari, A. Deterministic

MDPs with adversarial rewards and bandit feedback. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pp. 93–101, 2012.

Audibert, Jean-Yves, Bubeck, Sébastien, and Lugosi, Gábor. Minimax policies for combinatorial prediction games. *Journal of Machine Learning Research - Proceedings Track*, 19:107–132, 2011.

Bertsekas, D. P. *Dynamic Programming and Optimal Control*. Athena Scientific, Third edition, 2005.

Bremaud, P. *Markov chains : Gibbs fields, Monte Carlo simulation and queues*. Springer, 1999.

Dani, Varsha, Hayes, Thomas P., and Kakade, Sham. The price of bandit information for online optimization. In *Advances in Neural Information Processing Systems 20*, 2007.

Denardo, E. V. Periods of connected networks and powers of nonnegative matrices. *Mathematics of Operations Research*, 2(1):20–24, 1977.

Even-Dar, E., Kakade, S., and Mansour, Y. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Farias, V. F., Moallemi, C. C., Roy, B. Van, and Weissman, T. Universal reinforcement learning. *IEEE Transactions on Information Theory*, 56(5):2441–2454, 2010.

Feinberg, E. A. and Yang, F. On polynomial cases of the unichain classification problem for Markov decision processes. *Operations Research Letters*, 36(5): 527–530, 2008.

Neu, G., György, A., Szepesvári, C., and Antos, A. Online Markov decision processes under bandit feedback. In *Advances in Neural Information Processing Systems 23*, pp. 1804–1812, 2010.

Ortner, R. Online regret bounds for Markov decision processes with deterministic transitions. *Theoretical Computer Science*, 411(29-30):2684–2695, 2010.

Szepesvari, C. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1), 2010.

Yu, J. Y., Mannor, S., and Shimkin, N. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757, 2009.

Yu, Jia Yuan and Mannor, Shie. Arbitrarily modulated markov decision processes. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 2946–2953, 2009.

Appendix

Proof of Theorem 3.1. Let v be a state in one strongly connected component and let v' be a state in another strongly connected component. G cannot have both a path from v to v' and a path from v' to v; without loss of generality, assume that there is no path from v to v', and there may or may not be a path from v' to v.

Define $f : \mathcal{E} \mapsto [0, 1]$ and $f' : \mathcal{E} \mapsto [0, 1]$ as follows: f assigns a zero loss to the outgoing edges of v and a loss of 1 to all other edges; f' assigns a zero loss to the outgoing edges of v' and loss of 1 to all other edges. We use f and f' to define two oblivious sequences of loss functions: sequence one is $f_t \equiv f$ for all t; sequence two is $f_t \equiv 1$ (the constant function 1) for $t \leq T/2$ and $f_t \equiv f'$ for $t > T/2$. We show that one of these sequences always inflicts a linear regret.

By assumption, a path exists from v_0 to v and v belongs to a strongly connected set. Therefore, there exists a fixed policy that starts at v_0, takes the shortest path to v, and from there takes the shortest cycle back to v. This policy visits v $\Theta(T)$ times. Similarly, there exists another fixed policy that visits v' a linear number of times. Therefore, for both loss sequences, the best fixed policy suffers a loss of zero $\Theta(T)$ times. Also note that we have set the problem up such that the player can never attain a smaller loss than the loss of the best fixed policy, so regret on any subset of steps is nonnegative..

We need the theorem to hold for any possible algorithm, so we deal with two (overlapping) cases that together cover all possibilities. We ask: what does the algorithm do if it observes a loss of 1 on all steps? At least one of the following statements must be true: (1) with positive probability the algorithm does not visit v in the first $T/2$ steps, (2) with positive probability, the algorithm reaches v on or before step $T/2$.

If (1) is true, we examine the player's expected loss against sequence one, defined above. With probability $p > 0$, the player does not visit v in the first $T/2$ steps. In that case, his loss on the first $T/2$ steps equals $T/2$. The difference between this loss and the loss of the best fixed policy is $\Theta(T)$. On the last $T/2$ steps, the player's regret is nonnegative. With probability $1 - p$, the player's regret on all rounds is nonnegative. Overall, in expectation over both cases, the regret is $\Theta(T)$.

If (2) is true, we examine the player's expected loss against sequence two, defined above. Our definition of the second sequence guarantees that that the player observes a loss of 1 on the first $T/2$ rounds, so with probability $p > 0$, the player reaches v within the first $T/2$ steps. By assumption, there is no path from v to v', so the player accumulates a loss of T, and his regret is $Omega(T)$. With probability $1 - p$, the player regret on all rounds is nonnegative. Overall, the expected regret is $\Theta(T)$. $\square$