Online Learning for Markov Decision Processes in Nonstationary Environments: A Dynamic Regret Analysis

Yingying Li and Na Li

***Abstract*— In an online Markov decision process (MDP) with time-varying reward functions, a decision maker has to take an action before knowing the current reward function at each time step. This problem has received many research interests because of its wide range of applications. The literature usually focuses on static regret analysis by comparing the total rewards of an optimal offline stationary policy and the online policy. This paper studies a different measure, dynamic regret, which is the reward difference between an optimal offline (possibly non-stationary) policy and the online policy. The measure suits better the time-varying environment. To obtain meaningful regret analysis, we introduce a notion of total variation for the time-varying reward functions and bound the dynamic regret using the total variation. We propose an online algorithm, Follow the Weighted Leader (FWL), and prove that its dynamic regret can be upper bounded the total variation. We also prove a lower bound of dynamic regrets for any online algorithm. The lower bound matches the upper bound of FWL, demonstrating the optimality of the algorithm. Finally, we show via simulation that our algorithm FWL significantly outperforms the existing algorithms in literature.**

I. Introduction

In this paper, we consider a Markov decision process with time-varying and even adversarial reward functions. Specifically, at each time step $t = 1, \ldots, T$, the decision maker has to take an action $a_t \in A$ based on the observed state $s_t \in S$ *before* the (adversarial) environment reveals the reward function $r_t(s, a)$ for all $s \in S, a \in A$. The rule of taking actions is called a *policy*, and denoted by π_t. Then the system evolves to the next state s_{t+1} with transition probability $p_{s_t, s_{t+1}}(a_t)$. The transition probabilities remain the same and are available to the decision maker a priori. The goal is to maximize the total reward collected in T time steps by designing online algorithms based on only the history information. Such a problem is known as *online Markov decision processes (online MDP)* in literature.

Online MDP has attracted a lot of attention in recent years [1]–[12], partly due to its wide applications in data center scheduling [2], inventory control [1], [13], [14], resource allocation [2], [15], robotics tracking [3], queuing systems in networks [1], [6], [16], [17], etc.

Most papers in literature aim to design online MDP algorithms that are able to compete against the optimal stationary policy in hindsight, that is, to minimize the performance difference between the optimal stationary policy and the online policies [1]–[12]. We refer to such difference as the *static regret*, following the literature in online learning [18]–[20]. Various online MDP algorithms have been proposed to achieve $o(T)$ static regret regardless of the variation of the reward functions [1]–[5], [9]–[12].

However, when the reward function sequence $\{r_1, \ldots, r_T\}$ is highly nonstationary, the stationary policy may result in poor performance. For example, in a data center scheduling problem, if the electricity price suddenly increases, one might want to reduce the number of active servers to save energy instead of following the same allocation policy when the price is low.

To take into account the nonstationary nature of the problem, we focus on a dynamic benchmark: the optimal (possibly nonstationary) policies in retrospect of the MDP problem in T time steps, and we call as the *dynamic regret* the difference on the total reward between the optimal (possibly nonstationary) policies and the online policies, following the literature in online learning [18]–[20].

Intuitively, the dynamic regret should be larger than the static regret since the benchmark in the dynamic regret is more difficult to compete with. Indeed, we will show in this paper that when the environment changes drastically, no online algorithm can achieve sublinear $o(T)$ dynamic regret. Similar impossibility results have been established for online learning without an MDP structure in literature [20] [21].

Nevertheless, we are able to design online algorithms with sublinear dynamic regret when the variation of the environment is not huge. To formally characterize the environment's variation, we introduce the total variation of reward functions:

$$\text{Total variation of rewards} = \sum_{t=1}^{T+1} \|r_t - r_{t-1}\|_\infty$$

where $r_t = (r_t(i,a))_{i \in S, a \in A}$ and $r_0 = r_{T+1} = 0$; the infinity-norm is defined as $\|r\|_\infty = \max_{i \in S, a \in A} |r(i,a)|$. In addition, we propose an online algorithm FWL and show that the dynamic regret of FWL can be bounded by $C \sum_{t=1}^{T+1} \|r_t - r_{t-1}\|_\infty$, where the constant C depends on the number of states, the transition probabilities, and an algorithm parameter. Furthermore, we will show that no online algorithm can achieve a dynamic regret with better dependence on $\sum_{t=1}^{T+1} \|r_t - r_{t-1}\|_\infty$.

Finally, our simulation results show that our algorithm FWL significantly outperforms other online MDP algorithms in literature when the environment is highly nonstationary, and is comparable with existing algorithms when the environment is almost stationary.

The work was supported by NSF 1608509, NSF CAREER 1553407, AFOSR YIP, and ARPA-E through the NODES program. Y. Li and N. Li are with the School of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA 02138, USA (email: yingyingli@g.harvard.edu, nali@seas.harvard.edu).

A. Related works

[10] studies the adaptive regret for online MDP, whose benchmark is a policy sequence with at most $K = o(T)$ changes. Their benchmark is still weaker than ours because the optimal policies, as considered in the dynamic regret, might change T times in horizon T. To the best of our knowledge, there is a lack of study on the dynamic regret of online MDP algorithms.

In online learning without an MDP structure, the analysis of online algorithms against a dynamic benchmark has attracted much attention. For online convex optimization, there exist various online algorithms whose dynamic regrets are bounded by different environment-variation measures [18]–[20]. For expert problems, several algorithms have been proposed to minimize the adaptive regret [22]–[24].

In particular, Fixed-share Hedge algorithm proposed in [23] is in a similar spirit of our algorithm FWL: Fixed-share Hedge algorithm deals with the nonstationary rewards in expert problems by assigning decaying weights to the history policies, while our FWL assigns decaying weights to the history rewards.

Finally, we note that there are several variants of the online MDP frameworks studied in this paper, e.g., with unknown transition probabilities [10], with adversarially varying transition probabilities [6], [7], with bandit feedback [5], [6]. Nevertheless, most papers above focus on the static regret analysis. We leave it for future work to generalize our dynamic regret analysis to these frameworks.

B. Notations

For vector $x \in \mathbb{R}^n$, the infinity-norm is defined as $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$. For matrix $M \in \mathbb{R}^{n,m}$, the infinity-norm is $\|M\|_\infty = \max_{i,j} |M_{ij}|$. For integer n, $[n] = \{1, \dots, n\}$.

II. Problem formulation

Consider a Markov decision process (MDP) with time-varying reward functions. Reward functions are revealed to the decision maker in an online fashion, that is, reward functions are not available ahead of time and only revealed as time goes by. The MDP can be captured by a tuple (S, A, P) where $S = \{1, \dots, n\}$ denotes the finite state space, $A = \{a^1, \dots, a^m\}$ denotes the finite action space, and $P = \{p_{ij}(a),\ \forall\, a \in A,\ i, j \in S\}$ consists of transition probabilities, where $p_{ij}(a)$ represents the probability of jumping from state i to state j after taking action a. Such a problem is called as an *online Markov decision process (online MDP)* in literature. The protocol of online MDP is summarized below. At each time step $t = 1, 2, \dots,$

1) the decision maker observes the current state s_t
2) the decision maker takes an action a_t
3) the decision maker receives the reward $r_t(s_t, a_t)$, and observes the reward function $r_t(s, a)$ for all $s \in S$, and $a \in A$
4) the system evolves to the next state s_{t+1} with probability $p_{s_t, s_{t+1}}(a_t)$.

At each time step t, action a_t is chosen based on the available information, which consists of the current state s_t, the transition probabilities P, and the history reward functions $r_1, \dots, r_{t-1}$. [1] The rule of taking actions is called a *policy*, denoted as π. A policy π is a mapping from the state space S to the probability simplex on action space A, and $\pi(a|i)$ denotes the probability of taking action a given the current state i by following policy π. The algorithm is called an *online algorithm* if it selects policy π_t at each time t based on available information in online MDP problems. Formally, an online algorithm $\mathcal{A}$ can be characterized by:

$$\pi_t = \mathcal{A}(P, r_1, \dots, r_{t-1}), \quad \forall\, t = 1, 2, \dots$$

As a necessity to the Markov property, we require that the environment generates the reward function sequence in an *oblivious* way, that is, r_t does not depend on $s_1, \dots, s_{t-1}$ for all time step t. This requirement is adopted in almost all online MDP literature [1]–[12].

A. Performance metric: dynamic regret

The goal of the decision maker is to maximize the total (undiscounted) reward in T time steps, where $T \geq 1$ may not be available beforehand. Under policy $\{\pi_1, \dots, \pi_T\}$, the total reward given initial state i is denoted by $J_T(\pi_1, \dots, \pi_T)(i)$ and defined by

$$J_T(\pi_1, \dots, \pi_T)(i) = \mathop{\mathbb{E}}_{s_1 = i, a_t \sim \pi_t(\cdot|s_t)} \sum_{t=1}^{T} r_t(s_t, a_t) \qquad (1)$$

We write the total reward vector as $J_T(\pi_1, \dots, \pi_T) \in \mathbb{R}^n$.

When reward functions $r_1, \dots, r_T$ are available, the optimal policy sequence can be calculated by dynamic programming [17], and the optimal total reward is denoted as

$$J_T^*(i) = \max_{\{\pi_t\}_{t=1}^T} J_T(\pi_1, \dots, \pi_T)(i), \quad \forall\, i \in S$$

However, in the online setting, the reward functions are not revealed to the decision maker a priori, resulting in a much trickier problem. In this paper, we focus on designing algorithms utilizing only the available online information to ensure that the online performance is as close to the optimal performance J_T^* as possible. To capture the difference between the total reward produced by an online algorithm $\mathcal{A}$ and the optimal reward in hindsight, we introduce *dynamic regret*, which is formally defined as:

$$\text{dynamic regret of } \mathcal{A} = \|J_T^* - J_T(\mathcal{A})\|_\infty \qquad (2)$$

where $J_T(\mathcal{A})(i) = J_T(\pi_1, \dots, \pi_T)(i)$ and $\{\pi_t\}_{t=1}^T$ are the policies selected by online algorithm $\mathcal{A}$; the infinity-norm captures the maximal difference over all initial states.

Our performance metric is called *dynamic* regret because our benchmark J_T^* is generated by policies that are usually dynamic, i.e., nonstationary. This is in contrast to the previous regret adopted in literature, which we call as *static regret*. The static regret compares the online algorithm with the optimal stationary policy in hindsight, that is,

$$\text{static regret} = \max_{i \in S} \left[\max_{\pi} J_T(\pi, \dots, \pi)(i) - J_T(\mathcal{A})(i) \right]$$

[1] To guarantee the Markov property, we let a_t be independent of past states $s_1, \dots, s_{t-1}$ though they are also available. Such a restriction imposes no loss of generality since we can construct an equivalent Markov decision process with the augmented state $\tilde{s}_t = (s_1, \dots, s_t)$ to satisfy our restriction.

When the environment is under constant changes, it is more meaningful to compare with nonstationary policies than a stationary policy. For example, in a tracking problem, if the status of the target changes, the tracking policy is expected to change accordingly. Therefore, we will focus on dynamic regret in this paper.

B. Assumptions

This paper studies the unichain MDP model, which is a commonly adopted model in MDP literature and has wide applications [1]–[12]. This assumption ensures the existence of a scalar average reward, which will be further explained in Section II-C.

Assumption 1. *Every stationary policy defines a unichain Markov chain.*

Next, we introduce the second assumption in this paper.

Assumption 2. *There exists a state, denoted by n without loss of generality, that is recurrent in the Markov chain induced by any stationary policy.*

This assumption is also common in literature [17], [25] and satisfied in many applications, e.g. inventory control [13], and the queuing system [16]. Besides, this assumption is naturally satisfied in ergodic MDP [14]. Furthermore, this assumption is only for the proof of the regret bound and can be replaced by any other assumption guaranteeing (11).

C. Preliminaries: infinite-horizon average-reward MDP

We briefly introduce the infinite-horizon average-reward MDP problem below. Consider a MDP problem with the same reward function at each time step, i.e. $r_t = r$ for $t = 1, 2, \ldots$, and r is known a priori. In an infinite-horizon problem, the focus is on the average of the total reward because the total reward usually goes to infinity. Given a stationary policy π, the average reward of an infinite-horizon MDP with reward function r is defined as

$$g_r(\pi) = \lim_{T\to+\infty} \mathop{\mathbb{E}}_{s_1=i, a_t\sim\pi(\cdot|s_t)} \frac{1}{T}\sum_{t=1}^{T} r(s_t, a_t) \tag{3}$$

Under Assumption 1, the limit in (3) exists and the limiting value does not depend on the initial state i. The average reward $g_r(\pi)$ is also called the *gain* in literature [14], [26].

The goal of an average-reward MDP problem is to find a stationary policy that maximizes the average reward, that is,

$$\pi_r^* \in \arg\max_{\pi} g_r(\pi) \tag{4}$$

We refer to the policy π_r^* as the *average optimal policy*. The corresponding average reward, denoted by g_r^*, is referred as the *optimal gain*.

It is well-known that, under Assumption 1, the average-reward MDP (4) can be solved by the *Bellman equations* defined below:

$$h(i) + g = \max_{\pi} \left(r(i,\pi) + \sum_{j=1}^{n} p_{ij}(\pi) h(j) \right), \quad i \in S \tag{5}$$

where $r(i,\pi) = \mathbb{E}_{a\sim\pi(\cdot|i)}\, r(i,a) \in \mathbb{R}^n$ is the one step reward by following policy π, and $p_{ij}(\pi) = \mathbb{E}_{a\sim\pi(\cdot|i)}\, p_{ij}(a)$ is the transition probability by following policy π. The policy attaining the maximization in (5) is the average optimal policy defined by (4). Furthermore, with an additional requirement

$$h(n) = 0$$

the Bellman equations determine a unique solution (g_r^*, h_r^*), where g_r^* is the optimal gain, and h_r^* is usually called as the *bias* because up to a constant value, h_r^* captures the total difference between the reward and the optimal gain, that is, there exists $\lambda \in \mathbb{R}$ such that

$$h_r^*(i) + \lambda = \lim_{T\to+\infty} \mathop{\mathbb{E}}_{s_1=i, a_t\sim\pi_r^*(\cdot|s_t)} \sum_{t=1}^{T} [r(s_t,a_t) - g_r^*] \tag{6}$$

for all state i [14].

The average-reward MDP can be solved in polynomial time poly(n,m) by linear programming because the Bellman equations (5) together with the additional requirement $h(n) = 0$ can be converted to an equivalent linear program. Other computational methods for average-reward MDP include policy iteration and value iteration. For more details, we refer the reader to [14]. We note that the focus of this paper is not on the algorithm design for the average-reward MDP; instead, our online MDP algorithm will build on the existing average-reward MDP algorithms.

III. Online Algorithm: Follow the Weighted Leader

This section will propose a new online MDP algorithm, Follow the Weighted Leader (FWL), which is inspired by the classic online algorithm Follow the Leader (FTL) and its variants for MDP. In the following, we will first introduce FTL-based online MDP algorithms, then discuss the scenarios when they perform poorly. Then, we will present our FWL and explain why it performs well in different scenarios.

A. Classic algorithms: FTL for online MDP

Follow the Leader (FTL) has inspired various algorithms in online learning and online MDP problems [1], [10], [27]–[29]. Despite the variants, the main ideas of most FTL-based online MDP algorithms are similar and are summarized below: at time step $t = 2, 3, \ldots$

1) Compute the average of history rewards: [2]
$$\frac{1}{t-1}\sum_{k=1}^{t-1} r_k$$
2) Solve the average-reward MDP problem (4) with reward function $\frac{1}{t-1}\sum_{k=1}^{t-1} r_k$ and implement the average optimal policy:
$$\pi_t \in \arg\max_{\pi} g_{\frac{1}{t-1}\sum_{k=1}^{t-1} r_k}(\pi)$$

FTL-based online MDP algorithms perform well when the reward functions $r_1, \ldots, r_T$ are (almost) stationary. The

[2]Follow the Perturbed Leader, as one variant of FTL, adds a vanishing noise to the average reward and performs better in some scenarios.

reasons are discussed below. Let's consider a simple scenario when the reward function at time t is a static reward function r plus some i.i.d. zero-mean noise, that is, $r_t = r + \tau_t$. When the noise level is low and the horizon T is large, intuitively, the problem is similar to an average-reward MDP and the stationary policy π_r^* should perform well. In the online MDP problem when r is unknown, a good approximation of r would be the average of history rewards $\frac{1}{t-1}\sum_{k=1}^{t-1} r_k$, and the average optimal policy based on the average of history rewards would serve as a good approximation to policy π_r^*.

However, when the reward functions are changing drastically, FTL-based online MDP algorithms suffer poor performance. For example, consider a shifting environment when the reward function is r when $t \leq T/2$, then shifts to v when $T/2 < t \leq T$. Intuitively, the optimal policy should be close to π_v^* when $t > T/2$. However, the policy selected by FTL-based online algorithms are close to $\pi_{(r+v)/2}^*$ when $t > T/2$, which can be totally different from the optimal policy π_v^*, resulting in poor performance.

Intuitively, FTL-based online algorithms perform poorly because they put too much weight on the history reward a long time ago, thus adjusting slowly to the new environment. To deal with this, we introduce a new algorithm in Algorithm 1 in the next subsection.

Finally, we note that other online MDP algorithms, such as MDP-Expert, suffer from the same issue and perform poorly in the nonstationary environment.

B. Our algorithm: FWL for online MDP

FWL is summarized in Algorithm 1. The major computation burden comes from solving an average-reward MDP (Line 5) at each time step, which can be computed in polynomial time poly(n, m) as discussed in Section II-C.

Our algorithm FWL is similar to FTL-based online MDP algorithms except for one difference: FWL computes the policy based on a *weighted average* of history rewards, that is, in Line 4,

$$\begin{aligned}\hat{r}_t =& (1-\theta) r_{t-1} + \theta \hat{r}_{t-1} \\ =& (1-\theta) r_{t-1} + \theta(1-\theta) r_{t-2} + \cdots + \theta^{t-1} r_0\end{aligned}$$

where $t \geq 2$ and $\theta \in [0, 1)$. FWL focuses more on recent history rewards by assigning exponentially decaying weight to the history. In this way, FWL is able to adapt faster to the new environment thus enjoying a much better performance when the reward function is changing drastically.

Besides, if we allow the parameter θ to change with time, then FTL is a special case of FWL because the average history reward can be written as

$$\frac{1}{t-1}\sum_{k=1}^{t-1} r_k = \frac{1}{t-1} r_{t-1} + \frac{t-2}{t-1}\left(\frac{1}{t-2}\sum_{k=1}^{t-2} r_k\right)$$

where $t \geq 3$ and $\theta_t = \frac{t-2}{t-1}$ converges to 1. Thus, when the environment is almost stationary, FWL with a large (close to 1) θ will perform as well as FTL-based online MDP algorithms, matching our simulation result in Section V.

Finally, we note that the weighted average scheme adopted by FWL is sometimes called as exponential smoothing in time series literature [30].

Algorithm 1: Follow the Weighted Leader for MDP

1: **Inputs:** Transition probabilities P; initial policy π_0, parameter $\theta \in [0, 1)$
2: (Initialization): $\pi_1 = \pi_0$, $\hat{r}_1 = r_0 = 0$.
3: **for** $t = 2 : T$ **do**
4: Update the weighted average of history rewards:
$$\hat{r}_t = (1-\theta) r_{t-1} + \theta \hat{r}_{t-1}$$
5: Solve the average-reward MDP given reward $\hat{r}_t$ for the average optimal policy:
$$\pi_t \in \arg\max_{\pi} g_{\hat{r}_t}(\pi)$$
6: **end for**
7: **Ouputs**: π_t at each time step $t = 1, \ldots, T$.

IV. Regret Analysis

This section will first provide FWL's dynamic regret upper bounds, then present a fundamental lower bound for any online algorithm, followed by the proofs for the two results.

A. Regret Upper Bounds

Below is an upper bound of our FWL's dynamic regret.

Theorem 1. *Under Assumption 1 and 2, the dynamic regret of FWL is bounded by*

$$\begin{aligned}\|J_T(FWL) - J_T^*\|_\infty \leq& \kappa \sum_{t=1}^{T} \|r_t - r_{t+1}\|_\infty \\ &+ (2+\kappa)\sum_{t=1}^{T}\|r_t - \hat{r}_t\|_\infty + \kappa \sum_{t=1}^{T}\|\hat{r}_{t+1} - \hat{r}_t\|_\infty\end{aligned} \tag{7}$$

where $\hat{r}_t$ is defined in Algorithm 1, $\hat{r}_{T+1} = r_{T+1} = 0$, $\kappa = \frac{2n}{1-\rho}$ and $\rho \in [0, 1)$ is a constant determined by the transition probabilities P.

We defer the proof to Section IV-C but discuss insights of the regret bound in Theorem 1 below.

1) **First term** $\sum_{t=1}^{T}\|r_t - r_{t+1}\|_\infty$**.** This term represents the total fluctuation of the reward functions. When the reward fluctuation is large, it would be difficult to track the new reward functions, resulting in a large regret. When the reward fluctuation is small, it is easy to adapt to the new rewards so the regret should be small.

2) **Second term** $\sum_{t=1}^{T}\|r_t - \hat{r}_t\|_\infty$**.** This term represents the total error between the true reward r_t and our approximation $\hat{r}_t$. It is intuitive that the regret will be larger if the approximation error is larger.

3) **Third term** $\sum_{t=1}^{T}\|\hat{r}_t - \hat{r}_{t-1}\|_\infty$**.** This term represents the total fluctuation of our approximated reward $\hat{r}_t$. Since the policy π_t is chosen based on $\hat{r}_t$ (Line 5 in Algorithm 1), the change between $\hat{r}_t$ and $\hat{r}_{t-1}$ captures the change between policy π_t and π_{t-1}. Thus, the total fluctuation of $\hat{r}_t$ reveals the total fluctuation of π_t. This term shows that the performance of our algorithm suffers from the large fluctuation of the policy implemented.

4) **Tradeoff on θ.** There are two terms in (7) depending on θ: the total approximation error $\sum_{t=1}^{T}\|r_t-\hat{r}_t\|_\infty$ and the total fluctuation of the approximated reward $\sum_{t=1}^{T}\|\hat{r}_t-\hat{r}_{t-1}\|_\infty$. On the one hand, to reduce the fluctuation of the approximated rewards, we should increase the parameter θ to add more inertia during the update (Line 4 Algorithm 1). In the extreme case when $\theta=1$, we have zero fluctuation because $\hat{r}_t=\hat{r}_{t-1}$. On the other hand, if θ is too large, our approximation $\hat{r}_t$ assigns too little weight to the recent history, resulting in the slow adaptation to the new environment and thus a large approximation error. This introduces a tradeoff on the choice of θ. Generally speaking, we recommend choosing a larger θ to reduce fluctuation if the environment is almost stationary, and choosing a smaller θ to adapt faster when the environment is changing fast.

5) **Value of ρ.** The specific form of ρ is the following:

$$\rho=\max_{i\in S,\pi}\mathbb{P}(s_{n+1}\neq n,\ldots,s_2\neq n\mid s_1=i,\pi) \tag{8}$$

It can be shown by Assumption 2 that $\rho\in[0,1)$ [17] [14]. We note that ρ only depends on the transition probabilities. Furthermore, the main use of ρ is to determine the constant factor in the relation $\|h_r^*-h_v^*\|_\infty=O(\|r-v\|_\infty)$ (Lemma 3). Even if $\rho=1$, as long as the relation $\|h_r^*-h_v^*\|_\infty=O(\|r-v\|_\infty)$ holds, Theorem 1 still holds, just with a different factor.

6) **The reason behind $\|r_{T+1}-r_T\|_\infty$.** This term, in the first part of the regret upper bound, captures the regret resulted from not knowing the horizon T beforehand. If the decision maker knew the online MDP problem would terminate at T, or equivalently $0=r_{T+1}=r_{T+2}=\ldots$, (s)he might have chosen a different action to reduce the regret.

Finally, we show that even with a poorly chosen θ, the regret can be still bounded by the total fluctuation of the reward functions, as shown in the corollary below. The proof is deferred to Appendix D.

Corollary 1. *Under Assumption 1 and 2, for any $\theta\in[0,1)$, the dynamic regret of FWL is bounded by*

$$\|J_T(FWL)-J_T^*\|_\infty\leq\left(3\kappa+\frac{2+\kappa}{1-\theta}\right)\sum_{t=1}^{T+1}\|r_t-r_{t-1}\|_\infty$$

where $\kappa=\frac{2n}{1-\rho}$, $\rho\in[0,1)$ is a constant determined by the transition probabilities P, and $r_0=r_{T+1}=0$.

Corollary 1 shows that the dynamic regret of FWL can be upper bounded by a constant factor times the total variation of the reward $\sum_{t=1}^{T+1}\|r_t-r_{t-1}\|_\infty$, where the constant factor $2\kappa+\frac{2+\kappa}{1-\theta}$ only depends on P, n and θ. Therefore, FWL achieves $O(1)$ dynamic regret if the reward function r_t only changes for a constant number of times, which is much better than classic online MDP algorithms in literature according to our discussion in Section III-A.

In Corollary 1, the optimal θ to minimize the regret is 0. This is due to the proof techniques to deal with the worst case scenario and is not fundamental. The simulation section will provide more insights on the choice of θ.

B. Fundamental Lower Bound

We show that FWL reaches the optimal dependence on the total variation of the reward functions in the following theorem. The proof is deferred to Section IV-D.

Theorem 2. *Consider an online MDP problem with any transition probabilities and the number of actions satisfying $m\geq 2$. When $T\geq 1$, for any online algorithm $\mathcal{A}$, and any reward variation budget $K_T>0$, there exists a reward sequence $r_1,\ldots,r_T$ where $\sum_{t=1}^{T+1}\|r_t-r_{t-1}\|_\infty\leq K_T$ and $r_0=r_{T+1}=0$, such that the dynamic regret is lower bounded by*

$$\|J_T(\mathcal{A})-J_T^*\|_\infty\geq\Omega(K_T)$$

Remark 1. *When $m=1$, there is only one possible action, which is also the optimal action. The problem is only interesting when there are multiple actions, i.e. $m\geq 2$.*

C. Proof of Theorem 1

The proof takes three steps.

1) Introduce an approximation of J_T^*, denoted by W_1 and defined in (9), and bound the approximation error $\|J_T^*-W_1\|_\infty$
2) Bound the difference between FWL's performance and W_1, i.e., $\|J_T(FWL)-W_1\|_\infty$.
3) Derive the regret bound by the triangle inequality.

Before the proof, we introduce some shorthand notations. For r_t, we will write π_t^* for $\pi_{r_t}^*$, h_t^* for $h_{r_t}^*$, and g_t^* for $g_{r_t}^*$ to avoid double subscripts. In addition, we define a Bellman operator B and an optimal Bellman operator B^* as

$$B(r,h,\pi)(i)=r(i,\pi)+\sum_{j=1}^{n}p_{ij}(\pi)h(j)$$

$$B^*(r,h)=\max_\pi B(r,h,\pi)$$

Now we can write the Bellman equations in the vector form: $ge+h=\max_\pi B(r,h,\pi)=B^*(r,h)$, where $e=(1,\ldots,1)$.

Step 1: Introduce the approximation W_1 and bound $\|J_T^*-W_1\|_\infty$. One major difficulty in analyzing the dynamic regret defined in (2) is to characterize J_T^*, which can be computed by dynamic programming but has limited properties for theoretical analysis. To overcome the difficulty, we introduce an auxiliary vector

$$W_1=\sum_{t=1}^{T}g_t^*e+h_1^* \tag{9}$$

which is composed by some well-studied concepts: the bias and gains of average-reward MDP problems.

The intuition behind the approximation W_1 is the following. If the reward functions are the same at each time step, that is, $r_1=\cdots=r_T$, and T is relatively large, then the optimal total reward is almost (up to a constant scalar)

$$J_T^*\approx Tg_1^*e+h_1^*(i)=g_1^*e+\cdots+g_T^*e+h_1^*=W_1$$

according to the interpretation of the bias in (6).

Next we bound the approximation error. Remember that J_T^* can be calculated by dynamic programming. Following

the literature, we define the optimal reward-to-go vector from $t = k$ to $t = T$ as $V_k^* \in \mathbb{R}^n$, then the dynamic programming method can be written by the Bellman operator:

$$V_k^* = B^*(r_k, V_{k+1}^*),\ k = 1, \ldots, T \tag{10}$$

where $V_{T+1}^* = 0$. The optimal total reward is $J_T^* = V_1^*$. Thus we only need to bound $\|W_1 - V_1^*\|_\infty$.

Lemma 1.

$$\|J_T^* - W_1\|_\infty \leq \sum_{t=1}^{T} \|h_{t+1}^* - h_t^*\|_\infty$$

where $h_{T+1}^* = 0$.

Proof. The proof is by induction. Define $W_k = h_k^* + \sum_{t=k}^{T} g_t^* e$ for $1 \leq k \leq T$ and $W_{T+1} = 0$. Notice that h_k^* satisfies the Bellman equation: $h_k^* + g_k^* e = B^*(r_k, h_k^*)$ when $1 \leq k \leq T$. Thus, when $1 \leq k \leq T$, we can write W_k as

$$\begin{aligned} W_k &= h_k^* + \sum_{t=k}^{T} g_t^* e = B^*(r_k, h_k^*) + \sum_{t=k+1}^{T} g_t^* e \\ &= B^*(r_k, h_k^* + \sum_{t=k+1}^{T} g_t^* e) = B^*(r_k, h_k^* + W_{k+1} - h_{k+1}^*) \end{aligned}$$

where $e = (1, \ldots, 1)$, the third equality is by $g_t^* \in \mathbb{R}$ $\sum_{j=1} p_{ij}(a) = 1$.

Comparing with (10), and by the nonexpansiveness of the Bellman operator (Lemma 2), we have

$$\begin{aligned} \|V_k^* - W_k\|_\infty &\leq \|V_{k+1}^* - (W_{k+1} - h_{k+1}^* + h_k^*)\|_\infty \\ &\leq \|V_{k+1}^* - W_{k+1}\|_\infty + \|h_{k+1}^* - h_k^*\|_\infty \end{aligned}$$

Then, by induction, the proof is done. □

Lemma 2. *The operator B^* is non-expansive in the sense that for any reward r and any vector h and h':*

$$\|B^*(r, h) - B^*(r, h')\|_\infty \leq \|h - h'\|_\infty$$

In addition, for any reward functions r, r' and vectors h and h', the operator B satisfies

$$\|B(r, h, \pi) - B(r', h', \pi)\|_\infty \leq \|r - r'\|_\infty + \|h - h'\|_\infty.$$

Proof. By symmetry and the definition of the infinite norm, it suffices to show $B^*(r, h)(i) - B^*(r, h')(i) \leq \|h - h'\|_\infty$ holds for any $i \in A$. For any i, suppose action a attains the maximization in $B(r, h)(i)$ and action a' attains the maximization in $B(r, h')(i)$, then

$$\begin{aligned} &B^*(r, h)(i) - B^*(r, h')(i) = r(i, a) \\ &+ \sum_{j=1}^{n} p_{ij}(a) h(j) - \left(r(i, a') + \sum_{j=1}^{n} p_{ij}(a') h'(j) \right) \\ &\leq r(i, a) + \sum_{j=1}^{n} p_{ij}(a) h(j) - \left(r(i, a) + \sum_{j=1}^{n} p_{ij}(a) h'(j) \right) \\ &= \sum_{j=1}^{n} p_{ij}(a)(h(j) - h'(j)) \leq \|h - h'\|_\infty \end{aligned}$$

by a' is the maximizer, and $\sum_j p_{ij}(a) = 1$.

The proof for B is straightforward.

$$\begin{aligned} &|B(r, h, \pi)(i) - B(r', h', \pi)(i)| \\ &= |r(i, \pi) - r'(i, \pi) + \sum_{j=1}^{n} p_{ij}(\pi)(h(j) - h'(j))| \\ &\leq \|r - r'\|_\infty + \|h - h'\|_\infty \end{aligned}$$

□

The only remaining thing is to show that

$$\sum_{t=1}^{T} \|h_{t+1}^* - h_t^*\|_\infty = O(\sum_{t=1}^{T} \|r_{t+1} - r_t\|_\infty) \tag{11}$$

This is done by two supportive lemmas whose proofs are deferred to Appendix C and A respectively.

Lemma 3. *Consider two reward functions r and v. Under Assumption 2,*

$$\|h_r^* - h_v^*\|_\infty \leq \frac{n}{1 - \rho} \|r - v\|_\infty + \frac{n}{1 - \rho} |g_r^* - g_v^*|$$

where ρ is defined in (8)*. In addition, $\rho \in [0, 1)$.*

Lemma 4. *Suppose Assumption 1 holds. Consider two reward functions r and v, then,*

$$|g_r^* - g_v^*| \leq \|r - v\|_\infty$$

Finally, by applying results above, we can derive a bound for $\|J_T^* - W_1\|_\infty$ in the next lemma. The proof is deferred to the appendix.

Lemma 5. *Under Assumption 1 and 2, we have*

$$\|J_T^* - W_1\|_\infty \leq \frac{2n}{1 - \rho} \sum_{t=1}^{T} \|r_{t+1} - r_t\|_\infty$$

where $r_{T+1} = 0$ and ρ is defined in (6)*.*

Step 2: Bound $\|J_T(FWL) - W_1\|_\infty$. Notice that $J_T(FWL)$ can also be calculated by the backward induction. We denote the reward-to-go starting from time step k as V_k, and the backward induction can be written as

$$V_k = B(r_k, V_{k+1}, \pi_k)$$

where $V_{T+1} = 0$ and π_k is the policy selected by FWL at time step k. The total reward generated by FWL is

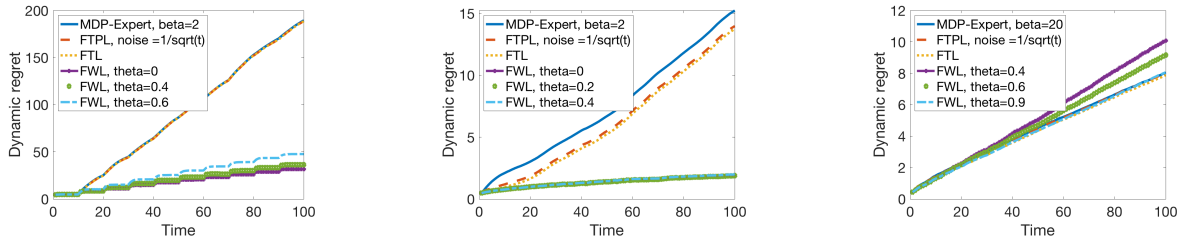
$$J_T(FWL) = V_1$$

Thus it suffices to bound $\|W_1 - V_1\|_\infty$.

Lemma 6. *Under Assumption 1, 2,*

$$\begin{aligned} \|J_T(FWL) - W_1\|_\infty &\leq \sum_{t=1}^{T} \frac{2n}{1 - \rho} \|\hat{r}_{t+1} - \hat{r}_t\|_\infty \\ &+ (\frac{2n}{1 - \rho} + 2) \sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty \end{aligned}$$

where $r_{T+1} = \hat{r}_{T+1} = 0$, and ρ is defined in (6)*.*



(a) Shifting rewards (b) Drifting rewards (c) Oscillating rewards

Fig. 1: The dynamic regrets of FWL with different θ, FTL, FTPL and MDP-Expert in three different scenarios.

Proof. We first introduce an auxiliary vector $\hat{W}_1$ defined as $\hat{W}_1 = \sum_{t=1}^{T} \hat{g}_t^* e + \hat{h}_1^*$, and $(\hat{g}_t^*, \hat{h}_t^*)$ is the optimal gain and bias with respect to reward function $\hat{r}_t$ for $t = 1, \ldots, T+1$ and $\hat{r}_1 = \hat{r}_{T+1} = 0$. Intuitively, $\hat{W}_1$ should be close to W_1 if the approximated reward $\hat{r}_t$ is close to true reward r_t. This is formally proved below.

$$\begin{aligned}\|W_1 - \hat{W}_1\|_\infty &\le \sum_{t=1}^{T} |g_t^* - \hat{g}_t^*| + \|h_1^* - \hat{h}_1^*\|_\infty \\ &\le \sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty + \frac{2n}{1-\rho}\|r_1 - \hat{r}_1\|_\infty \\ &\le (\frac{2n}{1-\rho} + 1) \sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty\end{aligned}$$

by Lemma 4 and Lemma 3.

Thus it suffices to bound the difference between V_1 and $\hat{W}_1$. We will bound $\|V_1 - \hat{W}_1\|_\infty$ by induction. Define $\hat{W}_k = \sum_{t=k}^{T} \hat{g}_t^* e + \hat{h}_k^*$ for $1 \le k \le T$ where $e = (1, \ldots, 1)$. Similar to the discussion on W_k, we can write $\hat{W}_k$ as

$$\begin{aligned}\hat{W}_k &= \hat{h}_k^* + \sum_{t=k}^{T} \hat{g}_t^* e = B(\hat{r}_k, \hat{h}_k^*, \pi_k) + \sum_{t=k+1}^{T} \hat{g}_t^* e \\ &= B(\hat{r}_k, \hat{h}_k^* + \sum_{t=k+1}^{T} \hat{g}_t^* e, \pi_k) \\ &= B(\hat{r}_k, \hat{h}_k^* + \hat{W}_{k+1} - \hat{h}_{k+1}^*, \pi_k)\end{aligned}$$

by the definition of π_k in Line 5 of Algorithm 1, where $e = (1, \ldots, 1)$.

Now we can bound $\|V_k - \hat{W}_k\|_\infty$ in a similar way to the proof of Lemma 5. For $1 \le k \le T$,

$$\begin{aligned}&\|V_k - \hat{W}_k\|_\infty \\ &= \|B(r_k, V_{k+1}, \pi_k) - B(\hat{r}_k, \hat{h}_k^* + \hat{W}_{k+1} - \hat{h}_{k+1}^*, \pi_k)\|_\infty \\ &\le \|r_k - \hat{r}_k\|_\infty + \|V_{k+1} - \hat{W}_{k+1}\|_\infty + \|\hat{h}_{k+1}^* - \hat{h}_k^*\|_\infty \\ &\le \|r_k - \hat{r}_k\|_\infty + \|V_{k+1} - \hat{W}_{k+1}\|_\infty + \frac{2n}{1-\rho}\|\hat{r}_{k+1} - \hat{r}_k\|_\infty\end{aligned}$$

Then, the proof is completed by deduction and Lemma 2. □

Step 3: derive the bound by the triangle inequality. The upper bound can be derived by Lemma 5 and Lemma 6 and the triangle inequality: $\|J_T(FWL) - J_T^*\|_\infty \le \|J_T(FWL) - W_1\|_\infty + \|W_1 - J_T^*\|_\infty$.

D. Proof of Theorem 2

Proof. For any $K_T > 0$, we construct two reward functions:

$$r^1(s,a) = \frac{K_T}{T+1}\mathbf{1}_{(a=a^1)} \quad r^2(s,a) = \frac{K_T}{T+1}\mathbf{1}_{(a=a^2)}$$

where $\mathbf{1}_{(a=a^1)}$ is the indicator function taking value 1 only when $a = a^1$. We have $\|r^1 - r^2\|_\infty \le \frac{K_T}{(T+1)}$.

Consider an i.i.d. randomly generated reward sequence: at each time step t, $r_t = r^1$ with probability 1/2, and $r_t = r^2$ with probability 1/2. It is easy to verify that for any realization of the reward sequence, the variation of rewards is at most K_T: $\sum_{t=1}^{T+1} \|r_t - r_{t-1}\|_\infty \le K_T$.

Since the reward function does not depend on states, the online MDP problem becomes a classic best expert problem. The optimal total reward is $J_T^*(i) = \frac{T}{T+1}K_T, \ \forall i \in S$.

For any online algorithm $\mathcal{A}$, at time step t, given state s_t, if $\mathcal{A}$ selects policy π_t, then

$$\begin{aligned}\mathbb{E}\, r_t(s_t, \pi_t) &= 1/2\frac{K_T}{T+1}\pi_t(a^1|s_t) + 1/2\frac{K_T}{T+1}\pi_t(a^2|s_t) \\ &\le \frac{K_T}{2(T+1)}\end{aligned}$$

Thus, the total reward by online algorithm $\mathcal{A}$ is $\mathbb{E}\, J_T(\mathcal{A})(i) = \mathbb{E}[\sum_{t=1}^{T} r_t(s_t, \pi_t)|s_1 = i] \le \frac{T}{2(T+1)}K_T$. The expected regret is $\mathbb{E}(J_T^*(i) - J_T(\mathcal{A})(i)) \ge \frac{T}{2(T+1)}K_T, \ \forall i \in S$. Therefore, there must exist a reward sequence $r_1, \ldots, r_T$, such that when $T \ge 1$,

$$\|J_T^* - J_T(\mathcal{A})\|_\infty \ge \frac{T}{2(T+1)}K_T \ge \frac{1}{4}K_T$$

□

V. Numerical Experiments

In this section, we compare our algorithm FWL with three online MDP algorithms in literature in three different scenarios. Specifically, we compare with FTL introduced in Section III-A, FTPL and MDP-Expert proposed in [1].

The underlying Markov decision processes is constructed randomly as follows. Consider state space $S = \{1, \ldots, 10\}$, and action space $A = \{a^1, a^2, a^3\}$. The transition probabilities in each scenario are generated randomly by first uniformly randomly selecting nonzero entries' indexes then generating these entries i.i.d. from Unif$[0, 1]$ then doing the normalization. When $a = a^1, a^2, a^3$, each row of $P(a)$ has 5, 6, 3 nonzero entries respectively. Notice that we do not

require the transition probabilities to satisfy Assumption 1 and 2.

Next we introduce reward generation methods for three different scenarios.

Scenario 1: Shifting environment When time step t satisfies $\mod(t, 10) = 1$, then generate a new reward function r_t with each entry i.i.d. following Unif$[0, 10]$; when $\mod(t, 10) \neq 1$, let $r_t = r_{t-1}$.

Scenario 2: Drifting environment Let r_t equal to the previous reward function plus a random drifting term, that is, $r_t = r_{t-1} + \tau_t$, with τ_t being the random matrix whose entries are i.i.d. from Unif$[0, 0.5]$. The first reward function r_1 is a random matrix whose entries are i.i.d. from Unif$[0, 1]$.

Scenario 3: Oscillating environment Given two randomly selected reward function r and v, $r_t = r$ with probability 1/2 and $r_t = v$ with probability 1/2.

The plots in Figure 1 are based on the average dynamic regret of each algorithm after 20 repeated experiments.

Firstly, we observe that in highly nonstationary scenarios such as the shifting and drifting environment, our algorithm FWL performs much better than other online MDP algorithms in literature. This matches our intuition in Section III. In addition, the optimal choice of θ is close to 0, since with a smaller θ, FWL adapts faster to the changing rewards thus enjoying a better performance.

In the oscillating scenario, FWL achieves better performance given a larger θ, because a larger θ reduces fluctuation and helps utilize more history information. The classic online MDP algorithms also perform well in this scenario.

VI. Conclusion

In this paper, we study the online Markov decision processes with a focus on the *dynamic regret*. We propose FWL whose dynamic regret upper bound matches the fundamental lower bound. In simulation, our algorithm FWL-MDP outperforms the existing algorithms in literature.

There are many interesting future directions, e.g. generalizing the framework to handle unknown transition probabilities, and even adversarial transition probabilities; dealing with bandit feedback on reward; studying the benefits of the predictions on future rewards, etc.

References

[1] E. Even-Dar, S. M. Kakade, and Y. Mansour, "Online markov decision processes," *Mathematics of Operations Research*, vol. 34, no. 3, pp. 726–736, 2009.

[2] X. Wei, H. Yu, and M. J. Neely, "Online learning in weakly coupled markov decision processes: A convergence time study," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, p. 12, 2018.

[3] G. Neu and V. Gómez, "Fast rates for online learning in linearly solvable markov decision processes," *arXiv preprint arXiv:1702.06341*, 2017.

[4] G. Neu, A. Gyorgy, and C. Szepesvári, "The adversarial stochastic shortest path problem with unknown transition probabilities," in *Artificial Intelligence and Statistics*, 2012, pp. 805–813.

[5] G. Neu, A. Antos, A. György, and C. Szepesvári, "Online markov decision processes under bandit feedback," in *Advances in Neural Information Processing Systems*, 2010, pp. 1804–1812.

[6] J. Y. Yu and S. Mannor, "Online learning in markov decision processes with arbitrarily changing rewards and transitions," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*. IEEE, 2009, pp. 314–322.

[7] ——, "Arbitrarily modulated markov decision processes," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. IEEE, 2009, pp. 2946–2953.

[8] Y. Abbasi, P. L. Bartlett, V. Kanade, Y. Seldin, and C. Szepesvári, "Online learning in markov decision processes with adversarially chosen transition probability distributions," in *Advances in neural information processing systems*, 2013, pp. 2508–2516.

[9] A. Zimin and G. Neu, "Online learning in episodic markovian decision processes by relative entropy policy search," in *Advances in neural information processing systems*, 2013, pp. 1583–1591.

[10] J. Y. Yu, S. Mannor, and N. Shimkin, "Markov decision processes with arbitrary reward processes," *Mathematics of Operations Research*, vol. 34, no. 3, pp. 737–757, 2009.

[11] T. Dick, A. Gyorgy, and C. Szepesvari, "Online learning in markov decision processes with changing cost sequences," in *International Conference on Machine Learning*, 2014, pp. 512–520.

[12] P. Guan, M. Raginsky, and R. M. Willett, "Online markov decision processes with kullback–leibler control cost," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1423–1438, 2014.

[13] Y. He, M. C. Fu, and S. I. Marcus, "A simulation-based policy iteration algorithm for average cost unichain markov decision processes," in *Computing Tools for Modeling, Optimization and Simulation*. Springer, 2000, pp. 161–182.

[14] M. L. Puterman, "Markov decision processes: Discrete stochastic dynamic programming (wiley series in probability and statistics)," 2005.

[15] Y. Wang, S. Chen, H. Goudarzi, and M. Pedram, "Resource allocation and consolidation in a multi-core server cluster using a markov decision process model," in *Quality Electronic Design (ISQED), 2013 14th International Symposium on*. IEEE, 2013, pp. 635–642.

[16] A. Zadorojniy, G. Even, and A. Shwartz, "A strongly polynomial algorithm for controlled queues," *Mathematics of Operations Research*, vol. 34, no. 4, pp. 992–1007, 2009.

[17] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 2005, vol. 1, no. 3.

[18] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," *arXiv preprint arXiv:1603.04954*, 2016.

[19] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Artificial Intelligence and Statistics*, 2015, pp. 398–406.

[20] Y. Li, G. Qu, and N. Li, "Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit," *arXiv preprint arXiv:1801.07780*, 2018.

[21] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations research*, vol. 63, no. 5, pp. 1227–1244, 2015.

[22] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790, 2007.

[23] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine learning*, vol. 32, no. 2, pp. 151–178, 1998.

[24] E. Hazan and C. Seshadhri, "Adaptive algorithms for online decision problems," in *Electronic colloquium on computational complexity (ECCC)*, vol. 14, no. 088, 2007.

[25] D. P. Bertsekas, "A new value iteration method for the average cost dynamic programming problem," *SIAM journal on control and optimization*, vol. 36, no. 2, pp. 742–759, 1998.

[26] ——, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 2012, vol. 2, no. 4.

[27] J. Hannan, "Approximation to bayes risk in repeated play," *Contributions to the Theory of Games*, vol. 3, pp. 97–139, 1957.

[28] M. Hutter and J. Poland, "Adaptive online prediction by following the perturbed leader," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 639–660, 2005.

[29] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[30] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. springer, 2016.

[31] Y. Li and N. Li. (2018) Online learning for markov decision processes in nonstationary environments: A dynamic regret analysis (extended version). [Online]. Available: https://nali.seas.harvard.edu/files/nali/files/2018acconlinemdp.pdf

APPENDIX

A. Proof of Lemma 3

This part is based on the connection with the stochastic shortest path problem (λ-SSP) introduced in [25]. We briefly explain the connection below. Since this paper considers rewards instead of costs, we will define a stochastic longest path problem, λ-SLP. Given any MDP and $\lambda \in \mathbb{R}$, λ-SLP. can be constructed below.

Now, we define a λ-SLP with state space $\tilde{S} = [n] \cup \{\tau\}$ where τ is the termination state. Action space is $\tilde{A} = A$. Policy is denoted as $\tilde{\pi}(a|i)$. The transition probability is defined as

$$\tilde{p}_{ij}(a) = \begin{cases} p_{ij}(a), & j \neq \tau, j \neq n \\ 0, & j = n \\ p_{in}(a), & j = \tau \end{cases} \tag{12}$$

when $i \neq \tau$. And $p_{\tau\tau}(a) = 1$ for all $a \in A$.

The reward function is

$$\tilde{r}(i,a) = \begin{cases} r(i,a) - \lambda, & i \neq \tau \\ 0, & i = \tau \end{cases} \tag{13}$$

where λ can be any value.

In the rest of the paper, we will use $\tilde{\mathbb{E}}$ for the expectation with respect to $\tilde{p}$, $\tilde{\mathbb{P}}$ for the probability with respect to $\tilde{p}$, $\tilde{h}$ for the reward-to-go of SLP etc.

For any policy $\pi : S \to \Delta_A$ for the original MDP, we can define a policy $\tilde{\pi} : S \cup \tau \to \Delta_A$ for SLP: let $\tilde{\pi}(a|i) = \pi(a|i)$ for any $i \in [n]$, and $\tilde{\pi}(a|\tau)$ be any distribution on A. Notice that $\tilde{\pi}$ is the same to π when $i \in [n]$, and is not important when $i = \tau$. In the following, we will sometimes abuse the notation and call $\tilde{\pi}$ as π.

When analyzing SLP, we usually need properness assumption (Assumption 7.2.1 in [17]).

Assumption 3. *For any* $\{\tilde{\pi}_t\}_{t=1}^{+\infty}$,

$$\rho = \max_{i=1,\dots,n} \tilde{\mathbb{P}}(\tilde{X}_{n+1} \neq \tau | \tilde{X}_1 = i, \{\tilde{\pi}_t\}_{t=1}^{+\infty}) < 1$$

It is known that Assumption 2 and 1 can guarantee Assumption 3 ([25] and Section 7.4 in [17]).

Necessary concepts for λ-SLP: For the λ-SLP,

$\tilde{h}^*(r,\lambda) =$ optimal total reward vector of λ-SLP

$\tilde{h}(r,\lambda,\pi) =$ total reward vector of λ-SLP following π

Since $\tilde{p}_{in}(a) = 0$ for all $a \in A$ and $i \in S$, the Bellman operator for λ-SLP coincides with the Bellman operators of λ-SLP defined below:

$$F(r,\lambda,h)(i) = \max_{a \in A} \left[r(i,a) - \lambda + \sum_{j=1}^{n} \tilde{p}_{ij}(a) h(j) \right]$$

where we abuse the notation and call the policy of λ-SLP as π because π and $\tilde{\pi}$ are the same when state $i \in S$.

Lemma 7. *$h_r^* = \tilde{h}^*(r, g_r^*)$ and $\tilde{h}^*(r, g_r^*)(n) = 0$.*

Proof. Notice that $\tilde{h}^*(r,\lambda)$ is the unique fixed point for the Bellman equation of SLP, that is,

$$\tilde{h}^*(r,\lambda) = F(r,\lambda,\tilde{h}^*(r,\lambda))$$

Now, remember that h_r^* and g_r^* satisfy the Bellman equation of MDP:

$$h_r^* = B(r - g_r^*, h_r^*)$$

We require $h_r^*(n) = 0$, then there is a unique h_r^* satisfying the Bellman equation. In addition, the Bellman equation becomes

$$h_r^* = F(r, g_r^*, h_r^*)$$

This means that $h_r^* = \tilde{h}^*(r, g_r^*)$ and $\tilde{h}^*(r, g_r^*)(n) = 0$. □

Now, we are ready to prove Lemma 3.

Proof of Lemma 3:

First, we show that in SLP/SSP problems, the total optimal reward/cost is bounded.

Lemma 8. *Under Assumption 3, the total optimal reward of λ-SLP is bounded by*

$$\|\tilde{h}^*(r,\lambda)\|_\infty \leq \frac{n\|r - \lambda\|_\infty}{1-\rho}$$

where ρ is defined in Assumption 3.

Proof. This is a classic result whose proof can be found in p407 [17]. □

Next, we show that the change of the optimal total reward can be bounded by the change of stage reward.

Lemma 9.

$$\|\tilde{h}^*(r,\lambda) - \tilde{h}^*(r',\lambda')\|_\infty \leq \frac{n(\|r - r'\|_\infty + |\lambda - \lambda'|)}{1-\rho}$$

Proof. We define the total reward of λ-SLP problem with reward function $r - \lambda$ following policy π as $\tilde{h}(r,\lambda,\pi)$. For any $i \in [n]$, we have

$$\begin{aligned} &\left[\tilde{h}^*(r,\lambda) - \tilde{h}^*(r',\lambda')\right](i) = \left[\tilde{h}(r,\lambda,\pi_r^*) - \tilde{h}(r',\lambda',\pi_{r'}^*)\right](i) \\ &\leq \left[\tilde{h}(r,\lambda,\pi_r^*) - \tilde{h}(r',\lambda',\pi_r^*)\right](i) \\ &= \tilde{h}(r - r', \lambda - \lambda', \pi_r^*)(i) \leq \tilde{h}^*(r - r', \lambda - \lambda')(i) \\ &\leq \frac{n(\|r - r'\|_\infty + |\lambda - \lambda'|)}{1-\rho} \end{aligned}$$

by $\pi_{r'}^*$ maximize $\tilde{h}(r',\lambda',\pi)$, the definition, and Lemma 8.

Similarly, we can bound $\left[\tilde{h}^*(r',\lambda') - \tilde{h}^*(r,\lambda)\right](i)$. □

Using the lemma above, we can bound $\|h_k^* - h_{k+1}^*\|_\infty$ by $\|r_k - r_{k+1}\|_\infty$ and $|g_k^* - g_{k+1}^*|$.

Corollary 2. *For $0 \leq k \leq T$,*

$$\|h_k^* - h_{k+1}^*\|_\infty \leq \frac{n(\|r_k - r_{k+1}\|_\infty + |g_k^* - g_{k+1}^*|)}{1-\rho}$$

where $h_{T+1}^* = 0$, $r_{T+1} = 0$, $g_{T+1}^* = 0$.

B. Proof of Lemma 5

Proof.

$$\begin{aligned}\|J_T^* - W_1\|_\infty &\le \sum_{t=1}^{T} \|h_{t+1}^* - h_t^*\|_\infty \\ &\le \sum_{t=1}^{T} \frac{n}{1-\rho}\left[\|r_{t+1} - r_t\|_\infty + |g_{t+1}^* - g_t^*|_\infty\right] \\ &\le \sum_{t=1}^{T} \frac{2n}{1-\rho}\|r_{t+1} - r_t\|_\infty\end{aligned}$$

where the first inequality is by Lemma 1; the second is by Lemma 3 and $g_{T+1}^* = 0$, $r_{T+1} = 0$, $h_{T+1}^* = 0$; the last one is by Lemma 4. □

C. Proof of Lemma 4

Proof. First we prove an upper bound of $g_r^* - g_v^*$:

$$\begin{aligned}&g_r^* - g_v^* \\ &= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\sum_{k=1}^{T} r(x_t, \pi_r^*) - \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\sum_{k=1}^{T} v(x_t, \pi_v^*) \\ &\le \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\sum_{k=1}^{T} [r(x_t, \pi_r^*) - v(x_t, \pi_r^*)] \\ &\le \|r - v\|_\infty\end{aligned}$$

where the first equality is by definition, the first inequality is by π_v^* is the optimal average policy while π_r^* is not when the reward is v, and the last inequality is by defintion. We can prove an upper bound of $g_v^* - g_r^*$ by similar arguments. □

D. Proof of Corollary 1:

Under Theorem 1, it suffices to show that $\sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty$ and $\sum_{t=1}^{T} \|\hat{r}_t - \hat{r}_{t-1}\|_\infty$ can be bounded by $\sum_{t=1}^{T} \|r_t - r_{t+1}\|_\infty$. First we bound $\sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty$ below.

$$\begin{aligned}&\sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty \le \sum_{t=1}^{T} \|r_t - r_{t-1}\|_\infty + \sum_{t=1}^{T} \|r_{t-1} - \hat{r}_t\|_\infty \\ &= \sum_{t=1}^{T} \|r_t - r_{t-1}\|_\infty + \theta \sum_{t=1}^{T} \|r_{t-1} - \hat{r}_{t-1}\|_\infty\end{aligned}$$

Since $r_0 = \hat{r}_0 = \hat{r}_1 = 0$, by moving to the left hand side the term $\theta \sum_{t=1}^{T} \|r_{t-1} - \hat{r}_{t-1}\|_\infty$ then dividing it by $1 - \theta$, we have

$$\sum_{t=1}^{T} \|r_t - \hat{r}_t\|_\infty \le \frac{1}{1-\theta} \sum_{t=1}^{T} \|r_t - r_{t-1}\|_\infty$$

In addition, we can bound $\sum_{t=1}^{T} \|\hat{r}_{t+1} - \hat{r}_t\|_\infty$ by

$$\begin{aligned}\sum_{t=1}^{T} \|\hat{r}_{t+1} - \hat{r}_t\|_\infty &= \sum_{t=1}^{T-1} \|\hat{r}_{t+1} - \hat{r}_t\|_\infty + \|\hat{r}_T - \hat{r}_{T+1}\|_\infty \\ &\le 2\sum_{t=1}^{T-1} \|\hat{r}_{t+1} - \hat{r}_t\|_\infty \\ &= 2(1-\theta)\sum_{t=1}^{T-1} \|r_t - \hat{r}_t\|_\infty \\ &\le 2\sum_{t=1}^{T} \|r_t - r_{t-1}\|_\infty\end{aligned}$$

where $\hat{r}_{T+1} = \hat{r}_1 = 0$. □