MATHEMATICS OF OPERATIONS RESEARCH
Vol. 34, No. 3, August 2009, pp. 726–736
ISSN 0364-765X | EISSN 1526-5471 | 09 | 3403 | 0726

DOI 10.1287/moor.1090.0396
© 2009 INFORMS

Online Markov Decision Processes

Eyal Even-Dar
Google Research, New York, New York 10011, evendar@google.com

Sham. M. Kakade
Toyota Technological Institute, Chicago, Illinois 60637, sham@tti-c.org

Yishay Mansour
School of Computer Science, Tel-Aviv University, 69978 Tel-Aviv, Israel, mansour@post.tau.ac.il

We consider a Markov decision process (MDP) setting in which the reward function is allowed to change after each time step (possibly in an adversarial manner), yet the dynamics remain fixed. Similar to the experts setting, we address the question of how well an agent can do when compared to the reward achieved under the best stationary policy over time. We provide efficient algorithms, which have regret bounds with no dependence on the size of state space. Instead, these bounds depend only on a certain horizon time of the process and logarithmically on the number of actions.

Key words: Markov decision process; no-regret algorithms
MSC2000 subject classification: Primary: 90C40, 68T05; secondary: 68Q32
OR/MS subject classification: Primary: dynamic programming/optimal control, Markov, finite state
History: Received August 16, 2006; revised September 29, 2008. Published online in *Articles in Advance* July 22, 2009.

1. Introduction.

Finite state and actions Markov decision processes (MDPs) are a popular and attractive way to formulate many stochastic optimization problems ranging from robotics to finance (Puterman [17], Bertsekas and Tsitsiklis [2], Sutton and Barto [18]). Unfortunately, in many applications the Markovian assumption made is only a relaxation of the real model. A popular framework that is not Markovian is the experts problem, in which during every round a learner chooses one of n decision-making experts and incurs the loss of the chosen expert. The setting is typically an adversarial one, where Nature provides the examples to a learner. The standard objective here is a myopic, backwards-looking one—in retrospect, we desire that our performance is not much worse than had we chosen any *single* expert on the sequence of examples provided by Nature. Expert algorithms have played an important role in computer science in the past decade, solving problems varying from classification to online portfolios (see Littlestone and Warmuth [13], Blum and Kalai [3], Helmbold et al. [8]).

There is an inherent tension between the objectives in an expert setting and those in a reinforcement learning (RL) setting. In contrast to the myopic nature of the expert algorithms, an RL setting typically makes the much stronger assumption of a fixed environment, and the forward-looking objective is to maximize some measure of the future reward with respect to this fixed environment. Therefore, in RL the past actions have a major influence on the current reward, whereas in the regret setting they have no influence. In this paper, we relax the Markovian assumption of the MDPs by letting the reward function be time dependent, and even chosen by an adversary as is done in the expert setting, but still keeping the underlying structure of an MDP.

The motivation of this work is to understand how to *efficiently* incorporate the benefits of existing experts' algorithms into a more adversarial reinforcement learning setting, where certain aspects of the environment could change over time. A naive way to implement an experts' algorithm is to simply associate an expert with each fixed policy. The running time of such algorithms is polynomial in the number of experts, and the regret (the difference from the optimal reward) is logarithmic in the number of experts. For our setting, the number of policies is huge, namely, for an MDP with state space S and action space A we have $|A|^{|S|}$ policies, which renders the naive experts' approach computationally infeasible. Another inherent problem in applying the best expert algorithms is that the current reward of the policy *depends* on the past actions, which is never the case in the standard expert setting. Furthermore, straightforward applications of standard regret algorithms produce regret bounds that are logarithmic in the number of policies, so they have linear dependence on the number of states. We might hope for a more effective regret bound that has *no dependence* on the size of state space (which is typically large).

1.1. Our model: Motivation.

The setting we consider is one in which the dynamics of the environment are known to the learner and are Markovian, but the reward function can change (adversarially) over time. We assume that after each time step the learner has full information, i.e., complete knowledge of the previous reward functions (over the entire environment), but does not know the future reward functions.

Many of the classical online problems can be cast in this setting. The basic idea is to have the online algorithm state as part of the MDP state. The changes in the state can be performed by actions (and incur the appropriate cost). Then, the MDP waits for a request, and the cost of the arriving request is modeled through the adversarial cost function. In the following, we describe in somewhat more detail the connections using three typical online problems: paging, k-server, and metrical task system (see Borodin and El-Yaniv [4] for an excellent exposition of the subject).

In the *paging problem* there is a memory that can hold k pages out of the N possible pages. A page request is a *hit* if it is in memory (and incurs a cost of zero) and a *miss* otherwise. The online algorithm can transfer pages to the memory at a cost of one per page. To model the paging problem as an MDP, each state is labeled by the set of pages, that it holds in memory (there are $\binom{N}{k}$ states). The MDP has two actions: *swap page*, which changes the state by replacing one page and incurs a cost of one, and *wait for request*, which waits for the next page request (there are $k(N-k)+1$ action in every state). The cost of a request for page p is zero for any state that includes page p and is one for any other state. (Note that unlike the classical paging model, we do not require bringing the page to memory in case of a miss, but the decision maker can later perform it. This difference between the two models is bounded by at most a factor of two in the cost, for any input sequence.)

Another classical example is the *k-server problem*. For simplicity, consider the case that the k servers are on a line graph with N nodes. The state includes the location of the k servers on the line (there are $\binom{N}{k}$ states). The actions are: *move server i left*, *move server i right*, and *wait for a request* (there are at most $2k+1$ actions in each state). Either move actions have a cost of one. When the decision maker performs *wait for a request*, the environment generates a request at node r. We model the effect of a request as a cost vector, where the cost in state $s=\{i_1,\dots,i_k\}$ is the minimal distance between one of the k servers in state s and the request in r, i.e., $\min_{i_j\in s}\{|i_j-r|\}$. Again, note that the dynamics are completely known to the decision maker, and the adversarial requests are modeled through the cost vectors. (Again, note that unlike the classical k-server model, we do not require changing states when the cost is not zero, but the decision maker can later make the state change. Again, the difference between the two models is bounded by a factor of two in the cost, for any request sequence.)

Both of the above examples, the paging problem and the k-server problem, are examples of a *metrical task system*. One can show that a general metrical task system can also be modeled in our setting. The states of the MDP and the metrical task system are the same states. The actions are either *move to state j* whose cost in state i is $d_{i,j}$, or *wait for a request*. When a request arrives, we can model it by a cost vector, where the cost in state i is the minimum over j of $d_{i,j}+c_j$, where c_j is the local cost in state j. As before, the dynamics are completely known, and state changes are not forced as a response to a request (and can have an effect of at most a factor of two).

Another classical motivating example is stochastic inventory control (Puterman [17]). At the beginning of each period the manager of a store has to decide how many items to order from the supplier based on the amount of items she currently holds. The manager's dilemma is that on one hand, holding the supply in store has an inventory cost, and on the other hand, running out of items is a clear revenue loss. The manager's goal is to maximize its profit, i.e., total revenue minus inventory cost. Although the manager can model the demand distribution, the item price and inventory cost can change between different periods due to the exogenous forces, and thus we can formulate these problem as an online MDP.

1.2. Related work. McMahan et al. [14] also considered a similar setting—they also assume that the reward function is chosen by an adversary and that the dynamics are fixed. However, they assume that the cost functions come from a finite set (but are not observable), and the goal is to find a min-max solution for the related stochastic game.

de Farias and Megiddo [6] considered a similar, yet different, problem. Similar to our setting, they have both a "state," which is affected by the past actions of the expert, and the assumption that after following an expert for a while the past is "forgotten." A few notable differences are that we can evaluate each expert even if we do not follow him because we can estimate its state, whereas in their works this is impossible because the current reward is influenced by the past in an unknown manner. Their main goal is to gain when Nature is not adversarial, which is a nonissue in our setting. More importantly, if one implements their algorithm in our setting it will be exponential in both time and space, whereas our algorithm is efficient and exploits the extra knowledge that it is given.

Nilim and El Ghaoui [16] studied robust MDPs, which have known reward distributions, and their transition matrices come from a convex set. They studied two possible models therein—one is that the transition matrix is stationary, chosen once by Nature at the beginning, and second, where Nature chooses at each timestep a matrix from the set. For this setting, they show how to compute the optimal policy using linear programming methods. This can be thought of as a game where the adversary chooses the transition matrix from a given set each round.

Following our initial publication, Yu et al. [20] studied a similar model where the transition matrix is known and stationary and the rewards are chosen by an adversary. Their algorithm is based on following the perturbed leader and is computationally more efficient. In their analysis they use a similar notion of mixing time and obtain similar dependence. An advantage of their algorithm is that the results hold with high probability.

2. The model. The online MDP, similar to the standard MDP, consists of state space S; actions available to the agent at each state A; a transition matrix P which specifies, for every s, s', and a, the probability of arriving at state s' from state s after performing action a; initial state distribution d_1 over S; and a sequence of reward functions $r_1, r_2, \ldots, r_T$, where r_t is the (bounded) reward function at time step t mapping $S \times A$ into $[0, 1]$.

The goal is to maximize the sum of *undiscounted* rewards over a T step horizon. We assume that the agent has complete knowledge of the transition model P, but at time t, the agent only knows the past reward functions $r_1, r_2, \ldots, r_{t-1}$. Hence, an algorithm $\mathcal{A}$ is a mapping from S and the previous reward functions $r_1, \ldots, r_{t-1}$ to a probability distribution over actions, so $\mathcal{A}(a \mid s, r_1, \ldots, r_{t-1})$ is the probability of taking action a at time t.

We define the return of an algorithm $\mathcal{A}$ as

$$V_{r_1, r_2, \ldots, r_T}(\mathcal{A}) = \frac{1}{T}\mathrm{E}\left[\sum_{t=1}^{T} r_t(s_t, a_t) \,\middle|\, d_1, \mathcal{A}\right],$$

where $a_t \sim \mathcal{A}(a \mid s_t, r_1, \ldots, r_{t-1})$ and s_t is the random variable that represents the state at time t, starting from initial state $s_1 \sim d_1$ and following actions $a_1, a_2, \ldots, a_{t-1}$. Note that we keep track of the expectation and not of a specific trajectory (and our algorithm specifies a distribution over actions at *every* state and at *every* time step t). This assumption is necessary because a specific trajectory makes the adversary too powerful, and also note that when we compute the value of the optimal policy we *must* consider its expected value because we do not have a specific trajectory in hand.

Ideally, we would like to find an algorithm $\mathcal{A}$ that achieves a large reward $V_{r_1, \ldots, r_T}(\mathcal{A})$ *regardless* of how the adversary chooses the reward functions. In general, this of course is not possible, and, as in the standard experts setting, we desire that our algorithm competes favorably against the best fixed stationary policy $\pi(a \mid s)$ in hindsight. Specifically, we would like to minimize the regret defined as

$$\mathcal{R}(\mathcal{A}) = \max_{r_1, r_2, \ldots, r_T} \max_{\pi} \mathrm{E}\left[\sum_{t=1}^{T} r_t(s_t, \pi(s_t)) \,\middle|\, d_1\right] - V_{r_1, r_2, \ldots, r_T}(\mathcal{A}),$$

where $\pi\colon S \to A$ is any deterministic policy.

2.1. Mixing time. Before we provide our results, a few definitions are in order. For every stationary policy $\pi(a \mid s)$, we define P^{π} to be the transition matrix induced by π, where the component $[P^{\pi}]_{s,s'}$ is the transition probability from state s to state s' under π, i.e., $[P^{\pi}]_{s,s'} = \sum_{a\in A} \pi(a \mid s) P^{a}_{s,s'}$. Also, define $d_{\pi,t}$ to be the state distribution at time t when following π, i.e.

$$d_{\pi,t} = d_1 (P^{\pi})^t,$$

where we are treating d_1 as a row vector here.

We assume throughout this section that the MDP is a unichain model. Although it was shown recently that finding whether an MDP is unichain is NP-hard (Tsitsiklis [19]), a unichain is one of the basic models of MDP (see Puterman [17]); furthermore, the unichain model generalizes the ergodic model, which can be identified in polynomial time.

Because we assume that the MDP is unichain, every policy π has a well-defined unique stationary distribution, which we call d_{π}. More formally, for every initial state s, $d_{\pi,t}$ converges to d_{π} as t tends to infinity and $d_{\pi}P^{\pi} = d_{\pi}$. Furthermore, this implies that there exists some τ such that for *all* policies π, and distributions d and d',

$$\|dP^{\pi} - d'P^{\pi}\|_1 \le e^{-1/\tau}\|d - d'\|_1,$$

where $\|x\|_1 = \sum |x_i|$ denotes the l_1 norm of a vector x. We refer to τ as the *mixing time*, and for convenience assume that $\tau \ge 1$.

The parameter τ provides a bound on the planning horizon timescale because it implies that *every* policy achieves close to its average reward in $O(\tau)$ steps.[1] This parameter also governs how long it effectively takes to switch from one policy to another (after time $O(\tau)$ steps there is little information in the state distribution about the previous policy). This definition is related to the definition of flexibility made by de Farias and Megiddo [6], where in their terminology each expert is a policy in our setting and the environment is the MDP in our setting.

This assumption allows us to define the average reward of policy π in an MDP with reward function r as

$$\eta_r(\pi) = \mathrm{E}_{s\sim d_\pi,\, a\sim\pi(a\,|\,s)}[r(s, a)],$$

and the value $Q_{\pi,r}(s, a)$ is defined as

$$Q_{\pi,r}(s, a) \equiv \mathrm{E}\left[\sum_{t=1}^{\infty}(r(s_t, a_t) - \eta_r(\pi)) \,\middle|\, s_1 = s, a_1 = a, \pi\right],$$

where s_t and a_t are the state and actions at time t, after starting from state $s_1 = s$, then deviating with an immediate action of $a_1 = a$ and following π onwards. We slightly abuse notation by writing $Q_{\pi,r}(s, \pi') \equiv \mathrm{E}_{a\sim\pi'(a\,|\,s)}[Q_{\pi,r}(s, a)]$. These values satisfy the well-known recurrence equation

$$Q_{\pi,r}(s, a) = r(s, a) - \eta_r(\pi) + \mathrm{E}_{s'\sim P_{sa}}[Q_\pi(s', \pi)], \tag{1}$$

where $Q_\pi(s', \pi)$ is the next state value (without deviation).

If π^* is an optimal policy (with respect to r), then, as usual, we define $Q_r^*(s, a)$ to be the value of the optimal policy, i.e., $Q_r^*(s, a) = Q_{\pi^*,r}(s, a)$.

We now provide two useful lemmas. It is straightforward to see that the previous assumption implies a rate of convergence to the stationary distribution that is $O(\tau)$ for all policies. The following lemma states this more precisely.

Lemma 2.1. *For all policies* π,

$$\|d_{\pi,t} - d_\pi\|_1 \le 2e^{-t/\tau}.$$

Proof. Because π is stationary, we have $d_\pi P^\pi = d_\pi$, and so

$$\|d_{\pi,t} - d_\pi\|_1 = \|d_{\pi,t-1}P^\pi - d_\pi P^\pi\|_1 \le e^{-1/\tau}\|d_{\pi,t-1} - d_\pi\|_1,$$

which implies $\|d_{\pi,t} - d_\pi\|_1 \le e^{-t/\tau}\|d_1 - d_\pi\|_1$. The claim follows because for any distributions d and d', we have $\|d - d'\|_1 \le 2$. □

The following lemma derives a bound on the Q values as a function of the mixing time.

Lemma 2.2. *For any reward function* r *in* $[0, 1]$ *and policy* π, *we have* $Q_{\pi,r}(s, a) \le 3\tau$.

Proof. First, let us bound $Q_{\pi,r}(s, \pi)$, where π is used on the first step. For all t, including $t = 1$, let $d_{\pi,s,t}$ be the state distribution at time t starting from state s and following π. Hence, we have

$$\begin{aligned}
Q_{\pi,r}(s, \pi) &= \sum_{t=1}^{\infty}(E_{s'\sim d_{\pi,s,t}, a\sim\pi}[r(s', a)] - \eta_r(\pi)) \\
&\le \sum_{t=1}^{\infty}(E_{s'\sim d_\pi, a\sim\pi}[r(s', a)] - \eta_r(\pi) + 2e^{-t/\tau}) \\
&= \sum_{t=1}^{\infty} 2e^{-t/\tau} \le \int_0^\infty 2e^{-t/\tau} = 2\tau.
\end{aligned}$$

Using the recurrence relation (1) for the Q values and the fact that rewards are bounded in $[0, 1]$, we have that $Q_{\pi,r}(s, a) \le 1 + Q_{\pi,r}(s, \pi)$. The result follows because $1 + 2\tau \le 3\tau$. □

[1] If this timescale is unreasonably large for some specific MDP, then one could artificially impose some horizon time and attempt to compete with those policies that mix in this horizon time, as done by Kearns and Singh [11].

3. Best expert algorithms. We first provide our assumption on the performance expert algorithms and later, for completeness, provide the weighted majority algorithm (Littlestone and Warmuth [13], Cesa-Bianchi et al. [5]).

Assumption 3.1 (Black Box Experts). *An optimized best expert algorithm $\mathcal{A}$ is an algorithm that guarantees that for any sequence of reward functions $r_1, r_2, \ldots, r_T$ over the action set A, the algorithm $\mathcal{A}$ selects a distribution q_t over A (using only the previous reward functions $r_1, r_2, \ldots, r_{t-1}$) such that for any $a \in A$,*

$$\sum_{t=1}^{T} \mathrm{E}_{a\sim q_t}[r_t(a)] \geq \sum_{t=1}^{T} r_t(a) - \sqrt{TM \log|A|},$$

where $r_t(a) \in [0, M]$. Furthermore, we also assume that decision distributions q_t do not change quickly:

$$\|q_t - q_{t+1}\|_1 \leq \sqrt{\frac{\log|A|}{t}}.$$

Next, we describe the weighted majority (WM) algorithm (Littlestone and Warmuth [13], Cesa-Bianchi et al. [5]), which satisfies this assumption (see Algorithm 1). We remark that many other popular algorithms, such as the exponential gradient (Kivinen and Warmuth [12]), also satisfy the assumption. The WM algorithm variant that we present here requires knowing the horizon time T in advance; however, there exist other variants such as doubling (Cesa-Bianchi et al. [5]) or changing the learning rate (Auer et al. [1]) that achieve the desired regret bounds without knowing T in advance.

Algorithm 1. Weighted Majority Algorithm

```
Weighted Majority (T)
Choose an initial distribution $P_1$;
for $t = 1$ to $T$ do
    Update $P_{t+1}(a) = P_t(a)\beta^{r_t(a)}/Z_t$, where $Z_t = \sum_{a\in A} P_t(a)\beta^{r_t(a)}$;
end
```

Proposition 1. *There exists a parameter $\beta > 0$ such that the weighted majority algorithm is an optimized best expert algorithm.*

A different flavor of a best expert algorithm that does not satisfy Assumption 3.1 but can be computationally more efficient was introduced in Kalai and Vempala [10] and Hannan [7]. Consider a setting where the action $a \in A \subset R^d$ is a vector of dimension d, and the reward function $r \in \mathcal{R} \subset R^d$ is also a vector of dimension d. The actual reward of performing action a under the reward function r is $r(a) = \sum_{i=1}^{d} r_i a_i$ (such a reward function is called *linear*). The *follow the perturbed leader* (FPL) algorithm works for problems that have the following properties: (1) the reward function is linear, and (2) There exists an oracle M (efficient algorithm) that computes the optimum of the static problem. We have the following associated parameters: (1) for any two decisions, we have $\|a - a'\|_1 \leq D$ (in the former setting each decision was associated with an expert), and (2) bounded reward, i.e., $r(a) \leq R$ for any $a \in A$ and $r \in \mathcal{R}$.

Algorithm 2. Follow the Perturbed Leader (FPL) Algorithm

```
FPL
for $t = 1$ to $\infty$ do
    Choose $p_t$ uniform at random in $[0, d\sqrt{t}]^d$;
    Use an oracle $M$ to find the optimal action $a_{t+1}$ for reward function $\sum_{i=1}^{t} r_t + p_t$;
    Use the action $a_{t+1}$ at time $t+1$;
    Observe $r_{t+1}$.
end
```

The following proposition bounds the regret of FPL algorithm (see Algorithm 2).

Proposition 2. *The FPL algorithm satisfies*

$$E\left[\sum_{t=1}^{T} r_t(a_t)\right] \geq \sum_{i=1}^{t} r_t(a) - 2\sqrt{DRdT}$$

for any action $a \in A$.

The major advantage of the FPL type of algorithms is that they can handle an exponential number of experts as long as the static problem can solved efficiently. A good motivating example is the shortest-path problem. In the shortest-path problem an agent is given a graph with two special nodes s and t in each time step, the agent chooses a path between s and t, and weight edges are revealed. The agent wants to minimize its average path distance and the regret is measured with respect to the optimal static path, which can be computed easily using Dijksta's algorithm. Hence, the FPL algorithm works efficiently, although the number of paths might be exponential and applying WM directly on the set of paths is computationally infeasible.

4. Online MDP: Idealized setting.

Every algorithm in the online MDP model faces two major obstacles. The first is computational because there is an exponential number of polices. The second is related to the MDP state because the current state (or distribution over the states) depends on the history (which is the algorithm's past actions). Therefore, the current reward depends not only on the current action, but also on past actions. This is in contrast to the common assumption in *every* best expert algorithm that the current reward is chosen arbitrarily. In this section, we concentrate only on the first problem and define an idealized setting, which makes the second problem regarding the MDP state irrelevant. In the next section, we will extend our solution to address both problems.

In our idealized setting, in *each* time step the algorithm chooses a policy and observes its return, i.e., the average reward $\eta_r(\pi)$. Therefore, there is no "MDP state" anymore, and the current reward is independent from the previous actions.

We start by describing and analyzing the naive approach. The naive algorithm uses an optimized best expert algorithm (Assumption 3.1), where each deterministic policy is an expert and the loss function for a policy π at time t is $\eta_{r_t}(\pi)$.

Because there is no state in the idealized setting, we can use the standard best experts algorithms directly, with $|A|^{|S|}$ experts where the reward $\eta_r(\pi)$ is bounded by one. Substituting these quantities in the best expert performance guarantees we obtain the following proposition.

Proposition 3. *Let $\mathcal{A}$ be an optimized best expert algorithm for the idealized settings. Then, for any sequence of reward functions $r_1, r_2, \ldots, r_T$, for any stationary policy π,*

$$E\left[\frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t)\right] \geq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) - \sqrt{|S|\log|A|/T},$$

where π_t is the policy selected by $\mathcal{A}$ at time t. Also, the algorithm $\mathcal{A}$ has space and time complexity $O(|A|^{|S|})$.

The above approach is clearly computationally infeasible due to the large space and time complexities. However, the MDP has a structure that one can exploit to give efficient algorithms. Algorithm FPL_in_MDPs (see Algorithm 3) is the FPL algorithm adapted to MDPs.[2]

Algorithm 3. FPL in Idealized Setting

```
FPL_in_MDPs
for $t = 1$ to $\infty$ do
Choose $p_t$ uniform at random in $[[0, |S||A|\sqrt{t}]^{|S||A|}]$;
Calculate the optimal policy, $\pi_t$ for the MDP with reward function $r = \sum_{i=1}^{t-1} r_i + p_t$;
Use $\pi_t$ at time $t$;
end
```

Proposition 4. *Algorithm FPL_in_MDPs in the idealized settings, for any sequence of reward functions $r_1, r_2, \ldots, r_T$, for any stationary policy π,*

$$E\left[\frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t)\right] \geq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) - \sqrt{|S||A|/T},$$

where π_t is the policy selected by FPL_in_MDPs at time t. Also, FPL_in_MDPs has space complexity $O(|A||S|)$ and the time complexity is polynomial in $|A|$ and $|S|$.

[2] The analysis of this algorithm in MDPs was also done independently by McMahan et al. [15].

Proof. The `FPL_in_MDPs` algorithm requires the reward function to be linear and its complexity is the complexity of solving the optimal static problem. We first show that the reward function is linear, i.e., $\eta_{\bar{r}_{[1,T]}}(\pi) = \sum_{t=1}^{T} \eta_{r_t}(\pi)$, where $\bar{r}_{[1,T]} = \sum_{t=1}^{T} r_t$. Namely,

$$\begin{aligned}\sum_{t=1}^{T}\eta_{r_t}(\pi) &= \sum_{t=T}^{t}\sum_{s\in S} d_\pi(s)\pi(a\mid s)r_t(s,a)\\ &= \sum_{s\in S}\sum_{t=1}^{T} d_\pi(s)\pi(a\mid s)r_t(s,a)\\ &= \sum_{s\in S} d_\pi(s)\pi(a\mid s)\sum_{t=1}^{T} r(s,a)\\ &= \eta_{\bar{r}_{[1,T]}}(\pi).\end{aligned}$$

Next, we would like to calculate the algorithm parameters. We know that the reward function is bounded by $R=1$; we also have that for any two policies π, π', $\|d_\pi - d_{\pi'}\| \le 2$, and that the dimension is $d = |S||A|$. Substituting in Proposition 2 gives the desired bound. The space complexity is $|S||A|$ because we only need to store the cumulative reward for every state action pair. The time complexity of computing the static problem is that of computing an optimal MDP policy, and it can be done in time polynomial in $|S|$ and $|A|$. □

Next, we present our algorithm `MDP-E`, which is as efficient as the `FPL_in_MDPs`, algorithm and its regret bound is slightly better. In contrary to `FPL_in_MDPs`, the `MDP-E` algorithm is not general and directly exploits the MDP structure. Later, we show that this algorithm translates to a good algorithm in the general setting as well. The `MDP-E` algorithm is intuitive and simple to describe. The algorithm uses optimized best expert algorithms (Assumption 3.1), but instead of using them on individual policies (as in Proposition 3), it associates each state with an optimized best expert algorithm where the individual experts correspond to the actions in the state. The policy of `MDP-E` is defined by the product of all best expert algorithms distributions. The immediate question is what loss function should we feed to each expert in each state. It turns out that Q_{π_t, r_t} is an appropriate feedback. Note that although the best expert algorithms are "local," they receive "global" information through the loss of Q_{π_t, r_t}.

Algorithm 4. MDP Expert Algorithm

```
MDP-E
Put in every state a best expert algorithm $B_s$;
for $t = 1$ to $\infty$ do
Let $a_t(s)$ be the distribution over action of $B_s$ at time $t$;
Let $\pi_t$ be $\pi_t(s) = a_t(s)$;
Use policy $\pi_t$ and obtain $r_t$ from the environment;
Feed $B_s$ with gain function $Q_{\pi_t, r_t}(s,\cdot)$
end
```

Theorem 4.1. *For any sequence of rewards $r_1, r_2, \ldots, r_T$, the* `MDP-E` *algorithm has the following performance guarantee: For any policy π,*

$$E\left[\sum_{t=1}^{T}\eta_{r_t}(\pi_t)\right] \ge \sum_{t=1}^{T}\eta_{r_t}(\pi) - \sqrt{3\tau T\log|A|},$$

where $\pi_1, \pi_2, \ldots, \pi_T$ is the sequence of policies selected by `MDP-E` *in response to $r_1, r_2, \ldots, r_T$.*

Before proving the above theorem, we provide a technical lemma (which is a variant of a result in Kakade [9]). The lemma motivates why our choice to feed each experts algorithm B_s the reward Q_{π_t, r_t} is appropriate.

Lemma 4.1. *For any policies π and π',*

$$\eta_r(\pi') - \eta_r(\pi) = \mathrm{E}_{s\sim d_{\pi'}}[Q_{\pi,r}(s,\pi') - Q_{\pi,r}(s,\pi)].$$

Proof. Note that by definition of stationarity, if the state distribution is at $d_{\pi'}$, then the next state distribution is also $d_{\pi'}$ if π' is followed. More formally, if $s \sim d_{\pi'}$, $a \sim \pi'(a\mid s)$, and $s' \sim P_{sa}$, then $s' \sim d_{\pi'}$. Using this and Equation (1), we have

$$\begin{aligned}\mathrm{E}_{s\sim d_{\pi'}}[Q_{\pi,r}(s,\pi')] &= \mathrm{E}_{s\sim d_{\pi'}, a\sim\pi'}[Q_{\pi,r}(s,a)]\\ &= \mathrm{E}_{s\sim d_{\pi'}, a\sim\pi'}[r(s,a) - \eta_r(\pi) + \mathrm{E}_{s'\sim P_{sa}}[Q_{\pi,r}(s',\pi)]]\end{aligned}$$

$$= \mathrm{E}_{s\sim d_{\pi'},\, a\sim\pi'}[r(s,a) - \eta_r(\pi)] + \mathrm{E}_{s\sim d_{\pi'}}[Q_{\pi,r}(s,\pi)]$$
$$= \eta_r(\pi') - \eta_r(\pi) + \mathrm{E}_{s\sim d_{\pi'}}[Q_{\pi,r}(s,\pi)].$$

Rearranging terms proves the lemma. □

Now we complete the proof of the theorem.

Proof of Theorem 4.1. Using the assumed regret in Assumption 3.1,

$$\begin{aligned} E\left[\sum_{t=1}^{T}\eta_{r_t}(\pi) - \sum_{t=1}^{T}\eta_{r_t}(\pi_t)\right] &= \sum_{t=1}^{T}\mathrm{E}_{s\sim d_\pi}[Q_{\pi_t,r_t}(s,\pi) - Q_{\pi_t,r_t}(s,\pi_t)] \\ &= \mathrm{E}_{s\sim d_\pi}\left[\sum_{t=1}^{T} Q_{\pi_t,r_t}(s,\pi) - Q_{\pi_t,r_t}(s,\pi_t)\right] \\ &\le \mathrm{E}_{s\sim d_\pi}[\sqrt{3\tau T\log|A|}] = \sqrt{3\tau T\log|A|}, \end{aligned}$$

where the first equality is by Lemma 4.1, in the second equality we used the fact that d_π does not depend on the time, and the last inequality uses the regret bound of Assumption 2 with $M \le 3\tau$ (by Lemma 2.2). □

5. Online MDP: The general setting. In this section, we derive our main result showing how to use any generic experts algorithm in the general setting, where the current reward is influenced by the previous policies.

Whereas in the previous section we use only the regret part of Assumption 3.1, for the general setting we will also use the "slow change" condition. Intuitively, our experts algorithms will be using a similar policy for significantly long periods of time.

Note that although we moved from the idealized settings to the general settings and our target function had changed, our algorithm MDP-E remained (surprisingly) unchanged. We now state our main theorem.

Theorem 5.1. *For any sequence of reward functions $r_1, r_2, \ldots, r_T$, for any stationary policy π,*

$$V_{r_1,r_2,\ldots,r_T}(\texttt{MDP-E}) \ge V_{r_1,r_2,\ldots,r_T}(\pi) - 4\tau^2\sqrt{\frac{\log|A|}{T}} - \sqrt{\frac{3\tau\log|A|}{T}} - \frac{4\tau}{T}.$$

Note that the regret vanishes at the rate $O(1/\sqrt{T})$, as is also the case with stateless experts algorithms. Furthermore, the bound does *not depend* on the size of the state space, but only on the mixing time. Note that although our bounds depend on the mixing time, which is the maximum over all policies' mixing time, the bounds are actually better because they depend only on the mixing time of the policies that were actually used.

5.1. The analysis. The analysis has two parts. First, we use the performance bounds of the algorithm in the idealized setting. Then, we take into account the slow change of the policies to show that the actual performance is similar to the instantaneous performance.

Taking mixing into account. First, we relate the values V to the sums of average reward used in the idealized setting. We call an algorithm $\mathcal{A}$ *slowly changing* if for any reward sequence $r_1, r_2, \ldots, r_T$, for any state we have $\|\pi_t(\cdot\mid s) - \pi_{t+1}(\cdot\mid s)\|_1 \le \sqrt{\log|A|/t}$, where π_t is the policy $\mathcal{A}$ selects at time t.

Theorem 5.2. *For any reward sequence $r_1, r_2, \ldots, r_T$, for any slowly changing algorithm $\mathcal{A}$,*

$$\left|V_{r_1,r_2,\ldots,r_T}(\mathcal{A}) - \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t)\right| \le 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{2\tau}{T},$$

where $\pi_1, \pi_2, \ldots, \pi_T$ is the sequence of policies selected by $\mathcal{A}$ in response to $r_1, r_2, \ldots, r_T$.

Because the above holds for all slowly changing $\mathcal{A}$ (including those $\mathcal{A}$ that are a constant policy π), then combining this with Theorem 4.1 (once with $\mathcal{A}$ as MDP-E and once with $\mathcal{A}$ as π) completes the proof of Theorem 5.1. We continue and prove the above theorem.

The following simple lemma bounds the distance in the next state distribution as a function of the distance between the policies used.

Lemma 5.1. *Let π and π' be such that $\|\pi(\cdot\mid s) - \pi'(\cdot\mid s)\|_1 \le \epsilon$. Then, for any state distribution d, we have $\|dP^{\pi} - dP^{\pi'}\|_1 \le \varepsilon$.*

Proof. Consider the case when d is a delta function on s. The difference in the next state distributions, $\|dP^{\pi} - dP^{\pi'}\|_1$, is

$$\begin{aligned}\sum_{s'} |[P^{\pi}]_{s,s'} - [P^{\pi'}]_{s,s'}| &= \sum_{s'}\sum_{a} |P_{s,a}(s')(\pi(a\mid s) - \pi'(a\mid s))| \\ &\le \sum_{s',a} P_{s,a}(s')|\pi(a\mid s) - \pi'(a\mid s)| \\ &= \sum_{a} |\pi(a\mid s) - \pi'(a\mid s)| \le \varepsilon.\end{aligned}$$

Linearity of expectation leads to the result for an arbitrary distribution d. □

Analogous to the definition of $d_{\pi,t}$, we define

$$d_{\mathcal{A},t} = \Pr[s_t = s \mid d_1, \mathcal{A}],$$

which is the probability that the state at time t is s given that $\mathcal{A}$ has been followed.

Lemma 5.2. *Let $\pi_1, \pi_2, \ldots, \pi_T$ be the sequence of policies selected by an optimized best expert algorithm $\mathcal{A}$ in response to $r_1, r_2, \ldots, r_T$. Then,*

$$\|d_{\mathcal{A},t} - d_{\pi_t}\|_1 \le 2\tau^2\sqrt{\frac{\log|A|}{t}} + 2e^{-t/\tau}.$$

Proof. Let $k \le t$. Because $\mathcal{A}$ is an optimized best expert algorithm (Assumption 3.1), we have that

$$\|\pi_k(\cdot\mid s) - \pi_t(\cdot\mid s)\|_1 \le \sum_{i=k}^{t-1}\sqrt{\frac{\log|A|}{i}} \le (\sqrt{t}-\sqrt{k})\sqrt{\log|A|} \le 2(t-k)\sqrt{\frac{\log|A|}{t}}.$$

Using this with $d_{\mathcal{A},k} = d_{\mathcal{A},k-1}P(\pi_k)$ and $d_{\pi_t}P^{\pi_t} = d_{\pi_t}$, we have

$$\begin{aligned}\|d_{\mathcal{A},k} - d_{\pi_t}\|_1 &= \|d_{\mathcal{A},k-1}P^{\pi_k} - d_{\mathcal{A},k-1}P^{\pi_t} + d_{\mathcal{A},k-1}P^{\pi_t} - d_{\pi_t}\|_1 \\ &\le \|d_{\mathcal{A},k-1}P^{\pi_t} - d_{\pi_t}\|_1 + \|d_{\mathcal{A},k-1}P^{\pi_k} - d_{\mathcal{A},k-1}P^{\pi_t}\|_1 \\ &\le \|d_{\mathcal{A},k-1}P^{\pi_t} - d_{\pi_t}P^{\pi_t}\|_1 + 2(t-k)\sqrt{\log|A|/t} \\ &\le e^{-1/\tau}\|d_{\mathcal{A},k-1} - d_{\pi_t}\|_1 + 2(t-k)\sqrt{\log|A|/t},\end{aligned}$$

where we used in the second inequality Lemma 5.1, and in the third inequality our mixing time assumption. Recursing on the above equation leads to

$$\begin{aligned}\|d_{\mathcal{A},t} - d_{\pi_t}\| &\le 2\sqrt{\log|A|/t}\sum_{k=2}^{t}(t-k)e^{-(t-k)/\tau} + e^{-t/\tau}\|d_1 - d_{\pi_t}\| \\ &\le 2\sqrt{\log|A|/t}\sum_{k=1}^{\infty}ke^{-k/\tau} + 2e^{-t/\tau} \\ &\le 2\sqrt{\log|A|/t}\int_0^{\infty}ke^{-k/\tau}\,dk + 2e^{-t/\tau} \\ &= 2\tau^2\sqrt{\log|A|/t} + 2e^{-t/\tau},\end{aligned}$$

which completes the proof of the lemma. □

We are now ready to prove the mixing theorem.

Proof of Theorem 5.2. By definition of V,

$$\begin{aligned}V_{r_1,r_2,\ldots,r_T}(\mathcal{A}) &= \frac{1}{T}\sum_{t=1}^{T}E_{s\sim d_{\mathcal{A},t},\,a\sim\pi_t}[r_t(s,a)] \\ &\le \frac{1}{T}\sum_{t=1}^{T}E_{s\sim d_{\pi_t},\,a\sim\pi_t}[r_t(s,a)] + \frac{1}{T}\sum_{t=1}^{T}\|d_{\mathcal{A},t} - d_{\pi_t}\|_1 \\ &\le \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t) + \frac{1}{T}\sum_{t=1}^{T}\left(2\tau^2\sqrt{\frac{\log|A|}{t}} + 2e^{-t/\tau}\right) \\ &\le \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t) + 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{2\tau}{T},\end{aligned}$$

where we have bounded the sums by integration in the second to last step. A similar argument for the lower bound leads to the result. □

PROOF OF THEOREM 5.1. We first relate $V_{r_1, r_2, \ldots, r_T}(\pi)$ to $\sum_{t=1}^{T} \eta_{r_t}(\pi)$:

$$\begin{aligned} V_{r_1, r_2, \ldots, r_T}(\pi) &= \frac{1}{T}\sum_{t=1}^{T} E_{s\sim d_{\pi,t},\, a\sim\pi}[r_t(s,a)] \\ &\leq \frac{1}{T}\sum_{t=1}^{T} E_{s\sim d_{\pi},\, a\sim\pi}[r_t(s,a)] + \frac{1}{T}\sum_{t=1}^{T}\|d_{\pi,t}-d_{\pi}\|_1 \\ &\leq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) + \frac{1}{T}\sum_{t=1}^{T} 2e^{-t/\tau} \\ &\leq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) + \frac{2\tau}{T}. \end{aligned}$$

Using this inequality and Theorem 5.2 we are ready to complete our proof:

$$\begin{aligned} V_{r_1, r_2, \ldots, r_T}(\pi) - V_{r_1, r_2, \ldots, r_T}(\mathcal{A}) &\leq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) + \frac{2\tau}{T} - V_{r_1, r_2, \ldots, r_T}(\mathcal{A}) \\ &\leq \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) + \frac{2\tau}{T} - \left(\frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t) - 4\tau^2\sqrt{\frac{\log|A|}{T}} - \frac{2\tau}{T}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi) - \frac{1}{T}\sum_{t=1}^{T}\eta_{r_t}(\pi_t) + 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{4\tau}{T} \\ &\leq \sqrt{\frac{3\tau\log|A|}{T}} + 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{4\tau}{T}, \end{aligned}$$

where the first inequality is due to the previous bound, the second is due to Theorem 5.2, and the last inequality is due to Theorem 4.1. □

6. Conclusions and open problems.

We view this work as a first step in bridging between reinforcement learning and adversarial online learning. We present an efficient low-regret algorithm for an online MDP setting. The importance of our extension of the standard MDP literature (Sutton and Barto [18]) is in relaxing the Markovian assumption, which is implied by the MDP model and in many cases is only a relaxation of the true non-Markovian world. An open problem that remains is extending our full-information result to banditlike settings, where one can observe only its reward for the current state action pair. Another interesting research direction is to significantly relax the assumptions regarding both the mixing time and unichain model.

Acknowledgments. This work was supported in part by the Information Society Technology Programme of the European Community under the PASCAL Network of Excellence, IST-2002-506778, by a grant from the Israel Science Foundation and an IBM faculty award. This publication reflects only the authors' views. This work was done while the first author was a graduate student at Tel Aviv University.

References

[1] Auer, P., N. Cesa-Bianchi, C. Gentile. 2002. Adaptive and self-confident on-line learning algorithms. *J. Comput. System Sci.* **64** 48–75.
[2] Bertsekas, D. P., J. N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.
[3] Blum, A., A. Kalai. 1999. Universal portfolios with and without transaction costs. *Machine Learning* **35** 193–205.
[4] Borodin, A., R. El-Yaniv. 1998. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, UK.
[5] Cesa-Bianchi, N., Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, M. K. Warmuth. 1997. How to use expert advice. *J. ACM* **44**(3) 427–485.
[6] de Farias, D. P., N. Megiddo. 2006. Combining expert advice in reactive environments. *J. ACM* **53**(5) 762–799.
[7] Hannan, J. 1957. Approximation to Bayes risk in repeated play. M. Dresher, A. W. Tucker, P. Wolde, eds. *Contributions to the Theory of Games, III*. Princeton University Press, Princeton, NJ, 97–139.
[8] Helmbold, D. P., R. E. Schapire, Y. Singer, M. K. Warmuth. 1998. On-line portfolio selection using multiplicative updates. *Math. Finance* **8**(4) 325–347.
[9] Kakade, S. M. 2003. On the sample complexity of reinforcement learning. Ph.D. thesis, University College London, London.
[10] Kalai, A., S. Vempala. 2005. Efficient algorithms for on-line optimization. *J. Comput. System Sci.* **71**(3) 291–307.

[11] Kearns, M., S. Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* **49**(2–3) 209–232.

[12] Kivinen, J., M. Warmuth. 1997. Additive versus exponentiated gradient updates for linear prediction. *J. Inform. Comput.* **132**(1) 1–64.

[13] Littlestone, N., M. K. Warmuth. 1994. The weighted majority algorithm. *Inform. Comput.* **108**(2) 212–261.

[14] McMahan, H. 2003. Planning in the presence of cost functions controlled by an adversary. *Proc. 20th Internat. Conf. Machine Learning* (*ICML*), Washington, DC, 536–543.

[15] McMahan, H., G. Gordon, A. Blum. 2003. Personal communication.

[16] Nilim, A., L. El Ghaoui. 2005. Robust solutions to Markov decision problems with uncertain transition matrices. *Oper. Res.* **53** 780–798.

[17] Puterman, M. 1994. *Markov Decision Processes*. Wiley-Interscience, New York.

[18] Sutton, R., A. Barto. 1998. *Reinforcement Learning. An Introduction*. MIT Press, Cambridge, MA.

[19] Tsitsiklis, J. N. 2007. NP-hardness of checking the unichain condition in average cost MDPs. *Oper. Res. Lett.* **35**(3) 319–323.

[20] Yu, J. Y., S. Mannor, N. Shimkin. 2009. Markov decision processes with arbitrary reward processes. *Math. Oper. Res.* **34**(3) 737–757.