Online Learning for Markov Decision Processes
Applied to Multi-Agent Systems

Mahmoud El Chamie  Behçet Açıkmese  Mehran Mesbahi

Abstract—Online learning is the process of providing online control decisions in sequential decision-making problems given (possibly partial) knowledge about the optimal controls for the past decision epochs. The purpose of this paper is to apply the online learning techniques on finite-state finite-action Markov Decision Processes (finite MDPs). We consider a multi-agent system composed of a learning agent and observed agents. The learning agent observes from the other agents the state probability distribution (pd) resulting from a stationary policy but not the policy itself. The state pd is observed either directly from an observed agent or through the density distribution of the multi-agent system. We show that using online learning, the learned policy performs at least as well as the one of the observed agents. Specifically, this paper shows that if the observed agents are running an optimal policy, the learning agent can learn the optimal average expected cost MDP policies via online learning techniques by using a descent gradient algorithm on the observed agents’ pd data.

I. INTRODUCTION

Online learning utilizes online convex optimization techniques to compute control actions in a sequence of decision epochs where each epoch is associated with a loss function. The goal of an online learning algorithm is to minimize regret that measures the difference between the cumulative loss of a sequence of control decisions and the loss of the optimal stationary decision chosen in hindsight [1]. Online learning has been successfully applied to many problems in interdisciplinary research areas such as artificial intelligence, game theory, information theory, and machine learning [2].

In this paper, we apply online learning to sequential stochastic decision-making models, specifically to finite Markov Decision Processes (MDPs) for agent-based systems. MDPs provide analytical models of stochastic processes with state and action spaces [3]. Learning in MDPs has taken considerable attention in artificial intelligence due to its practical applicability in agent-based systems [4]–[6]. For example, reinforcement learning (RL) is a method where an agent builds a model of the surrounding environment and learns, from this constructed model, the optimal actions that minimize costs (or maximize rewards) [7], [8]. The basic idea behind RL is that an agent learns through trial-and-error interactions with its dynamic environment [9].

These sequential decision-making problems are often formulated as infinite-horizon dynamic programming (DP) problems [10] with two common criteria for measuring the learning performance. The first is the discounted cost criterion that uses a discount factor $\gamma \in (0, 1)$ that leads to a mathematically tractable optimal control due to the contraction properties of the resulting DP operator [3]. The main limitation is how to choose the discount parameter $\gamma$ in real applications. Therefore an additional parameter fitting is necessary to optimize performance. The other performance criterion is the average expected costs. The average cost criterion is more intuitive but is less tractable mathematically and requires additional assumptions on the induced Markov chains (e.g., unichain, communicating, or multichain) [11]. The agent can thus learn, through experience, optimal policies using for example Q-learning for the discounted cost and R-learning for the average cost. These techniques are based on learning, indirectly, the transition probabilities by running the MDP model for “sufficiently long enough” time epochs. Techniques have been developed to speed up the learning process such as using exploitation versus exploration to overcome the uncertainty and the effects of initial conditions [12].

Contrary to online learning techniques in RL literature that deal with a single agent versus environment type of interactions [13]–[16], this paper studies the problem of using observations from other agents to construct a policy that optimizes the performance metric associated with the MDP. We assume that there is no direct communication between the agents and the actions of the observed agents are not directly known. We assume that only the state probability distribution resulting from the agents’ collective actions can be observed. We show that, if the state-action cost matrix and the transition model are fully known, we can apply an online algorithm to learn an optimal policy. Classical algorithms (e.g., value iteration, policy iteration, or linear programming) compute optimal policies offline by directly using the MDP model. On the other hand, the online learning algorithm proposed in this paper learns the decision policy by observing other agents. This approach also facilitates a distributed design of optimal policies as it is based on descent gradient methods.

The rest of the paper is organized as follows. Section II provides preliminaries on MDPs and formulates the online learning MDP considered in this paper with two agents: a learning agent and an observed agent. Section III shows how the provided scenario can be applied to multi-agent systems with a large number (swarm) of agents. Section IV presents...
the main technical contribution of the paper, with corresponding proof given in Section V. Numerical simulations are provided in VI. Section VII concludes the paper.

II. PROBLEM FORMULATION

A. MDP with Average Expected Cost

The notation in this paper follows from the standard notation/assumptions in MDP literature [3]:

- Finite state set \( S = \{1, \ldots, n\} \), and finite action set \( A = \{1, \ldots, p\} \).
- Infinite time horizon; \( X_t \) and \( A_t \) are the random variables corresponding to the state and action at time \( t \).
- A non-stationary Markovian policy is given by \( \pi = \{Q_1, Q_2, \ldots\} \) where \( Q_t \in \mathbb{R}^{n \times p} \) is a decision matrix at time \( t \) whose elements are given by \( Q_t(s, a) := \text{Prob}[A_t = a|X_t = s] \). A stationary policy is given by a fixed decision matrix at every time epoch, i.e., \( \pi = \{Q, Q, \ldots\} \). Let \( \Pi \) be set of all feasible policies.
- The instantaneous cost matrix is stationary and is given for any time epoch by \( C \in \mathbb{R}^{n \times p} \) having the elements \( C(s, a) \) for a state \( s \) and action \( a \), and let \( C_{\max} = \max_{s,a} C(s, a) \).
- The MDP transition matrices are stationary and are given at time \( t \) by \( \{G_k \in \mathbb{R}^{n \times n}, k = 1, \ldots, p\} \) and whose elements are defined as
  \[
  G_k(i, j) := \text{Prob}[X_{t+1} = i|X_t = j, A_t = k].
  \] (1)

- Given a policy \( \pi \), the states \( \{X_1, X_2, \ldots\} \) induce a discrete-time Markov chain whose transition matrix at time \( t \) is \( M_t \in \mathbb{R}^{n \times n} \) and whose elements are defined as follows \( M_t(j, i) := \text{Prob}[X_{t+1} = j|X_t = i] \). With that definition, \( M_t \) is column stochastic, i.e., \( 1^T M_t = 1^T \), where \( 1 \) is the vector of all ones. Note that for stationary policies, \( M_t \) is independent of time.
- The state probability distribution (pd) at time \( t \) is given by the probability vector \( x_t \), where \( x_t(i) := \text{Prob}[X_t = i] \). Moreover, \( x_t \) satisfies the following dynamics:
  \[
  x_{t+1} = M_t x_t \quad \text{for} \quad t \geq 1,
  \] (2)

where \( M_t \) is the Markov chain transition matrix which is a linear function of the decision matrix \( Q_t \) and is given as follows:
  \[
  M_t(Q_t) = \sum_{k=1}^{p} G_k \odot (1(Q_t e_k)^T),
  \] (3)

where \( \odot \) is the Hadamard component-wise product, and \( e_k \) is the vector of all zeros except for the \( k \)-th component that is equal to 1, so \( Q_t e_k \) is simply the \( k \)-th column of \( Q_t \).
- The performance metric is the average cost given by
  \[
  v^\pi := \limsup_{N \to \infty} \frac{1}{N} \mathbb{E}_{x_1} \left[ \sum_{t=1}^{N} C(X_t, A_t) \right]
  \] (4)

\[
= \limsup_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} x_t^T C(Q_t)
  \] (5)

where \( C(Q_t) = (C \odot Q_t) 1 \), and \( \odot \) is the Hadamard component-wise product and \( 1 \) is the vector of all ones. Let \( v^* \) be an optimal cost, i.e.,

\[
v^* = \min_{\pi \in \Pi} \limsup_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} x_t^T C(Q_t)
  \] (6)

**Remark:** The objective of the MDP is to choose a policy \( \pi \) that minimizes the performance metric \( v^\pi \). “\( \limsup \)” is used in the definition of \( v^* \) as the “\( \lim \)” may not exist for general non-stationary policies. However, if \( \pi = \{Q, Q, \ldots\} \) is stationary, then the “\( \lim \)” is well defined.

We will consider the following assumption on the MDP model:

**Assumption 1.** The MDP is unichain, where the Markov chain transition matrix \( M \) corresponding to every deterministic policy has a 2-norm ergodicity coefficient \( \tau_2(M) < 1 \).

An MDP is a unichain if the Markov chain transition matrix corresponding to every deterministic policy consists of a single ergodic class plus a possibly empty set of transient states. This is a standard assumption in MDPs with average costs, see for example [15], [17], and [3, p. 349] for the classification schemes of MDP classes. The \( p \)-norm ergodicity coefficient of a column stochastic matrix \( P \) is

\[
\tau_p(P) = \max_{||z||_p = 1, z^T 1 = 0} ||Pz||_p.
\]

Note that if in addition \( P \) is symmetric, \( \tau_2(P) \) simply reduces to the second largest eigenvalue in magnitude of \( P \). For more information on ergodicity coefficients and their properties we refer the reader to [18] and the references therein. Having \( \tau_2(M) < 1 \) ensures contracting properties of the Markov chain transition matrix, i.e., it implies that every policy \( \pi \) has a well-defined unique stationary distribution. Formally, for any initial state \( pd \) \( x_1 \), \( x_t \) converges to \( x_\infty \) as \( t \) goes to infinity. Since there is a finite number of deterministic decisions at a given time epoch, there exists \( \tau_2 \in \mathbb{R}_+ \) such that for all \( t \) we have \( \tau_2(M_t) \leq \tau_2 < 1 \). We assume \( \tau_2 \) is known in this paper, computing \( \tau_2 \) when it exists from a given MDP formulation (e.g., from the transition matrices \( \{G_k\} \)) is an interesting future direction, but is out of the scope of this paper.

B. Observing the State Probability Distribution

For simplicity, we first consider only two agents: a learning agent and an observed agent. The observed agent is running a (potentially optimal) policy, while the learner can only observe the state probability distribution \( x_t \) resulting from this (hidden) policy. The observed agent’s policy \( \bar{\pi} = (\bar{Q}, \bar{Q}, \ldots) \) is a stationary policy with a decision matrix \( \bar{Q} \in \mathbb{Q} \) where \( \mathbb{Q} \) is a closed, bounded, and non-empty set of decision matrices. For unconstrained MDPs, the set \( \mathbb{Q} \) is simply defined as follows:

\[
\mathbb{Q} = \{Q \in \mathbb{R}^{n \times p} : Q 1 = 1, Q \geq 0\}.
\] (7)

However, \( \mathbb{Q} \) can be more restricted. For example, in the presence of safety constraints of the form [19], [20]:

\[
x_t \leq d \quad \text{for} \quad t = 1, 2, \ldots
\]
where \( d \in \mathbb{R}^n \) is an upper-bound that constraints each element of the probability vector, the set \( Q \) is characterized by a more general set of linear (in-)equalities than those given in (7) [21].

**Remark:** This procedure can be extended to the case of multi-agent systems. In particular, if the system has a large number of agents, an approximation to \( \tilde{X}_t \) can be computed by the learning agent by counting the number of agents in each state and normalizing by the total number of agents as we will show in the next section.

Consequently we investigate a policy-learning problem as follows:

**Problem Formulation.** The learning agent knows the transition probabilities \( \{G_k, k = 1, \ldots, p\} \), the state-action cost matrix \( C \) for the state set \( S \) and the action set \( A \). At a given time epoch \( t \), the agent observes the state probability distribution of the observed agent (i.e., \( \tilde{X}_t \)) and then selects its decision matrix \( Q_t \) accordingly. The goal is to design an online policy \( \tilde{\pi} \) for the learning agent that performs at least as well as the one of the observed agent, i.e.,

\[
\hat{v} \leq \bar{v},
\]

where \( \bar{v} \) is the (hidden) policy of the observed agent.

An online policy is a policy where the learning agent adjusts its decision matrix \( Q_t \) at time \( t \) as a function of \( \tilde{X}_t \) of the other agents. We assume that the agents do not communicate and only the state \( pd \) resulting from their actions can be observed. Without loss of generality we assume that both the learning and observed agents start from the same initial state \( pd \), i.e., \( \tilde{x}_1 = x_1 \), but the results still hold for the general case.

**III. Observations in Multi-Agent Systems**

In the problem formulation, the learning agent observes the state \( pd \tilde{x}_t \) of an observed agent. In practical applications, the state \( pd \tilde{x}_t \) may not be directly observed, i.e., only a sample from the probability mass function (p.m.f.) over the states is observed and an estimate of the state \( pd \tilde{x}_t \) is computed. Notwithstanding this, we can benefit from the existence of many agents in the system to approximate effectively \( \tilde{x}_t \). For example, swarm coordination problems [22], [23] use a Markov chain for guiding large numbers of agents to a desired distribution. In this case, we can approximate the state \( pd \tilde{x}_t \) in multi-agent systems by considering the ensemble mean of the actual states of the agents, i.e., distribution of the agents. More formally, if there are \( K \) agents, then \( K \tilde{x}_t(i) \) describes the expected number of agents in the \( i \)th state. Let \( m_t = [m_t(1), \ldots, m_t(n)]^T \) denote the actual number of agents in each state. Then, \( m_t(i) \) is generally different from \( K \tilde{x}_t(i) \), although it follows from the independent and identically distributed agent realizations that

\[
\tilde{x}_t = \frac{E[m_t]}{K},
\]

and from the law of large numbers that \( m_t/K \to \tilde{x}_t \) as \( K \to \infty \) for all \( t \). The idea behind the multi-agent control is to control the propagation of the probability vector, \( \tilde{x} \), rather than controlling \( \{X_{t,k}\}_{k=1}^K \) directly, where \( X_{t,k} \) is the state of agent \( k \) at time \( t \). While, in general, \( m_t/K \neq \tilde{x}_t \), it will always be equal to \( \tilde{x} \) on average, and can be made arbitrarily close to \( \tilde{x} \) by using a sufficiently large number of agents. A block-diagram representation of the learning procedure in the presence of multi-agent system is given in Fig. 1. Therefore, the online learning results in this paper are applicable to scenarios involving an agent joining a swarm of agents, adversarial agent versus a swarm of agents, and swarm versus swarm type of interactions.

**IV. Main Results**

We associate with every iteration \( t \) a cost (loss) function \( w^t(Q) \) defined as follows:

\[
w^t(Q) := \tilde{x}_t^T \tilde{c}(Q) + \alpha_t||\tilde{x}_t - M(Q)\tilde{x}_{t-1}||, \quad t > 1
\]

where \( \alpha_t \geq 0 \), and \( w^t(Q) = \tilde{x}_t^T \tilde{c}(Q) \) for \( t = 1 \). The scalar \( \alpha_t \) is a weighting factor between the immediate cost of a policy \( \tilde{x}_t^T \tilde{c}(Q) \) and the shift from the observed state probability distribution \( ||\tilde{x}_t - M(Q)\tilde{x}_{t-1}|| \). The cost function \( w^t(Q) \) is a convex function in \( Q \) and is differentiable at each point \( Q \in \mathbb{Q} \) where \( ||\tilde{x}_t - M(Q)\tilde{x}_{t-1}|| \neq 0 \). The gradient of the cost function \( w^t \) with respect to the optimization variable \( Q \), \( \nabla w^t(Q) \), when it is differentiable is given as follows:

\[
\nabla w^t(Q) = \text{diag}(\tilde{x}_t)C + \frac{\alpha_t}{||z||} (\text{diag}(\tilde{x}_{t-1})G),
\]

where \( z = M\tilde{x}_{t-1} - \tilde{x}_t \) and \( G = [G_1^T \tilde{z} \ G_2^T \tilde{z} \ldots \ G_p^T \tilde{z}] \). With a little abuse of notation, when \( ||z|| = 0 \), we define \( \nabla w^t(Q) = \text{diag}(\tilde{x}_t)C \), this corresponds to a subgradient of the function \( w^t(Q) \), i.e., for a given \( Q \in \mathbb{Q} \), we have

\[
\text{Tr}(C^T \text{diag}(\tilde{x}_t)(P - Q)) \leq w^t(P) - w^t(Q), \quad \forall P \in \mathbb{Q}.
\]

Note that the cost function \( w^t(Q) \) is a design choice that impacts the performance of the proposed online MDP policy learning algorithm, Algorithm 1. The learning agent in Algorithm 1 observes the state \( pd \) of the observed agent, \( \tilde{x}_t \), and updates its policy according to the gradient iteration in line 3, where \( \nabla w^t(Q_t) \) is given by (10). The projection operator \( \text{Proj}_{[\cdot]} \) and the learning rate \( \frac{1}{2\gamma} \) in Eq. (11) are typically used to guarantee convergence properties in online
learning literature, see [1] for more details. The agent would then use \( Q_t \) as a lookup table to execute the action online.

**Algorithm 1** Online Learning MDP Optimal Policies: \( \hat{\pi} = (Q_1, Q_2, \ldots) \)

1. Choose any initial decision matrix \( Q_1 \in \mathbb{Q} \).
2. At a given time epoch \( t \), observe the density \( \tilde{x}_t \).
3. Update the decision matrix as follows:
   \[
   Q_{t+1} = \text{Proj}_{\mathbb{Q}} \left[ Q_t - \frac{1}{\sqrt{t}} \nabla w^T(Q_t) \right] \tag{11}
   \]
4. Act according to the corresponding decision matrix \( Q_t \).
5. Increment time \( t \) by 1 and go back to step 2.

We now present the main technical result of this paper.

**Theorem 1.** Let \( \hat{\pi} \) be the policy of the learning agent generated by Algorithm 1, with a fixed scalar \( \alpha_t = \alpha \) for all \( t \) in (9), and initial distribution \( x_1 = \tilde{x}_1 \). Suppose Assumption 1 holds. If \( \alpha \geq \frac{C_{\text{max}}}{1 - \tau_2} \), then
\[
v^\hat{\pi} \leq v^\hat{\pi}, \tag{12}
\]
where \( \hat{\pi} \) is the policy of the observed agent.

Theorem 1 shows that the learning agent can do at least as well as the observed agent under the average expected cost performance metric. We will prove the theorem in the following section, but first we give a corollary to Theorem 1:

**Corollary 1.** If the observed agent has an optimal stationary policy \( \pi^* = (Q^*, Q^*, \ldots) \), then
\[
v^\hat{\pi} = v^*, \tag{13}
\]
where \( v^* \) is given in (6).

**Proof.** We will show that \( v^\hat{\pi} \leq v^* \) and \( v^\pi \geq v^* \). The latter equation is straight forward by the definition of \( v^* \) being the cost of the optimal policy. It remains to show that \( v^\hat{\pi} \leq v^* \). Note that the observed agent’s policy is stationary, so we need to show that there exists a stationary policy that dominates nonstationary policies. In [24] it is shown that any finite state and action MDP has a Blackwell optimal policy that is average optimal and stationary. Therefore, there exists \( Q^* \) such that the policy \( \pi^* = (Q^*, Q^*, \ldots) \) is average optimal:
\[
v^\pi^* \leq v^* \quad \text{for all } \pi \in \Pi. \tag{14}
\]
Applying Theorem 1 we get that
\[
v^\hat{\pi} \leq v^\pi = v^\pi^* = v^*, \tag{15}
\]
and this completes the proof. \( \square \)

V. PROOF OF MAIN RESULT

We provide in this section the proof of Theorem 1. Recall that the performance metric \( v^\pi \) is given as follows:
\[
v^\pi = \limsup_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} x_t^T \tilde{c}(Q_t), \tag{16}
\]
where
\[
\tilde{c}(Q_t) = (C \odot Q_t)1,
\]
where \( \odot \) is the Hadamard component-wise product and \( 1 \) is the vector of all ones.

Let \( \hat{\pi} = (Q, Q, \ldots) \) be the policy of the observed agent. Let \( \hat{x}_t \) be the state probability distribution corresponding to the observed policy with the following dynamics:
\[
\hat{x}_{t+1} = \hat{M}\hat{x}_t \quad \text{for } t \geq 1,
\]
where \( \hat{M} = \sum_{k=1}^{p} G_k \odot (1(\hat{Q} e_k)^T) \).

The following lemma provides an upper bound on the performance of the online policy from Algorithm 1.

**Lemma 1.** We can bound the total expected cost of the online learned policy \( \hat{\pi} = (Q_1, Q_2, \ldots) \) as follows:
\[
\sum_{t=1}^{N} x_t^T \tilde{c}(Q_t) \leq \sum_{t=1}^{N} \tilde{x}_t^T \tilde{c}(Q_t) + C_{\text{max}} \sum_{t=2}^{N} ||x_t - \tilde{x}_t||, \tag{18}
\]
where \( x_{t+1} = M(Q_t)x_t \) for \( t \geq 1 \).

**Proof.** The proof applies Cauchy-Schwarz inequality
\[
\sum_{t=1}^{N} x_t^T \tilde{c}(Q_t) = \sum_{t=1}^{N} \tilde{x}_t^T \tilde{c}(Q_t) + \sum_{t=1}^{N} (x_t^T - \tilde{x}_t^T) \tilde{c}(Q_t)
\]
\[
\leq \sum_{t=1}^{N} \tilde{x}_t^T \tilde{c}(Q_t) + C_{\text{max}} \sum_{t=2}^{N} ||x_t - \tilde{x}_t||
\]

We will show next that \( ||x_t - \tilde{x}_t|| \) in Lemma 1 can be bounded from above using the terms \( ||\tilde{x}_t - M(Q_{t-1})\tilde{x}_{t-1}|| \) that appear in \( w^t(Q) \):

**Lemma 2.** Suppose Assumption 1 holds. Then \( ||x_t - \tilde{x}_t|| \) can be bounded from above as follows:
\[
||x_t - \tilde{x}_t|| \leq \sum_{k=2}^{t} \tau_2^{t-k} ||M(Q_{k-1})\tilde{x}_{k-1} - \tilde{x}_k||
\]
where \( \tau_2 \in (0, 1) \).

**Proof.**
\[
||x_t - \tilde{x}_t|| = ||M(Q_{t-1})x_{t-1} - \tilde{x}_t|| \\
\leq ||M(Q_{t-1})x_{t-1} - \tilde{x}_t|| + ||M(Q_{t-1})(x_{t-1} - \tilde{x}_{t-1})|| \\
\leq ||M(Q_{t-1})\tilde{x}_{t-1} - \tilde{x}_t|| + \tau_2||x_{t-1} - \tilde{x}_{t-1}||,
\]
where the last inequality above is due to the fact that the vector \( v = x_{t-1} - \tilde{x}_{t-1} \) sums to 0, i.e., \( v^T1 = 0 \), and \( \tau_2(M_t) \leq \tau_2 < 1 \) due to Assumption 1. We can repeat the above argument recursively to bound \( ||x_{t-1} - \tilde{x}_{t-1}|| \) from above, and thus we have
\[
||x_t - \tilde{x}_t|| \leq \sum_{k=2}^{t} \tau_2^{t-k} ||M(Q_{k-1})\tilde{x}_{k-1} - \tilde{x}_k||.
\]
\( \square \)
Let $w^t(Q)$ be given as in (9), then the following lemma applies the online convex programming results to Algorithm 1 to upper bound the cumulative loss function:

**Lemma 3.** Algorithm 1 provides a non-stationary policy $\hat{\pi} = (Q_1, Q_2, \ldots)$ such that

$$\sum_{t=1}^{N} w^t(Q_t) \leq \sum_{t=1}^{N} w^t(\hat{Q}^*) + \epsilon(\sqrt{N}),$$  \hspace{1cm} (19)

where $\hat{Q}^* = \arg\min_{Q \in Q} \sum_{t=1}^{N} w^t(Q)$, and $\epsilon(\sqrt{N})$ is a linear function of $\sqrt{N}$.

**Proof.** $Q$ is closed and bounded set, $w^t(Q)$ is a convex function, and $\nabla w^t(Q)$ is a bounded gradients, then the lemma is a direct consequence of a standard online learning result given in [1, Theorem 1] with a regret until time $N$ defined as $R_N^\pi := \sum_{t=1}^{N} w^t(Q_t) - \min_{Q \in \mathcal{Q}} \sum_{t=1}^{N} w^t(Q)$.

We are now ready to give the following proposition:

**Proposition 1.** Suppose Assumption 1 holds. Let $\hat{\pi} = (Q_1, Q_2, \ldots)$ be the policy generated by Algorithm 1. Then

$$\sum_{t=1}^{N} x_t^T \mathbf{c}(Q_t) \leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + \epsilon(\sqrt{N}).$$  \hspace{1cm} (20)

**Proof.** Using Lemma 1 we have

$$\sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(Q_t) \leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + C_{\max} \sum_{t=1}^{N} ||x_t - \tilde{x}_t||$$  \hspace{1cm} (21)

then apply Lemma 2 to get an upper bound on the expression in (21)

$$\leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + C_{\max} \sum_{t=2}^{N} \tau_{t-1} ||M(Q_{t-1})\tilde{x}_{t-1} - \tilde{x}_t||$$  \hspace{1cm} (22)

$$= \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q})$$

$$+ C_{\max} \frac{1}{1-\tau_2} \sum_{t=2}^{N} (1-\tau_2^{N+1-t}) ||M(Q_{t-1})\tilde{x}_{t-1} - \tilde{x}_t||,$$  \hspace{1cm} (23)

let $L := \frac{C_{\max}}{1-\tau_2}$, then for $\alpha_t = \alpha \geq L \geq C_{\max} (1-\tau_2^{N+1-t})$, an upper bound to the above expression in (23) is as follows:

$$\leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + \sum_{t=2}^{N} \alpha_t ||M(Q_{t-1})\tilde{x}_{t-1} - \tilde{x}_t||,$$  \hspace{1cm} (24)

and apply Lemma 3 to (24) to get

$$\leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + \sum_{t=2}^{N} \alpha_t ||M(Q*)\tilde{x}_{t-1} - \tilde{x}_t||+\epsilon(\sqrt{N}),$$  \hspace{1cm} (25)

the first two terms in (25) are simply equal to $\min_{Q \in \mathcal{Q}} \sum_{t=1}^{N} w^t(Q) \leq \sum_{t=1}^{N} w^t(Q)$, using $||M(Q)\tilde{x}_{t-1} - \tilde{x}_t|| = 0$, we get an upper bound to (25):

$$\leq \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) + \epsilon(\sqrt{N}),$$  \hspace{1cm} (26)

and this completes the proof. \qed

As a result of Proposition 1, we have

$$\limsup_{N \to \infty} \left( \frac{1}{N} \sum_{t=1}^{N} x_t^T \mathbf{c}(Q_t) - \frac{1}{N} \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q}) \right) \leq 0$$

and the theorem follows because the performance metric is given by $v^\pi = \limsup_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} x_t^T \mathbf{c}(Q_t)$ for the learning agent and by $v^\pi = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} \tilde{x}_t^T \mathbf{c}(\hat{Q})$ for the observed agent.

**VI. Simulations**

This section presents an example to demonstrate the proposed online learning algorithm for the finite state and action MDPs. Here, the learning agent as well as an optimal observed agent explore a region, which can be partitioned into $n$ disjoint subregions (or bins). We can model the system as an MDP where the states of agents are the bin locations ($n$ states) and the actions are defined by the possible transitions to neighboring bins. Each agent incurs some costs while traversing the area where, due to the stochastic environment, transitions are stochastic (i.e., even if the agent’s command is to go “North”, the environment can send the agent with a small probability to “East”). For simplicity we consider the operational region to be a 3 by 3 grid, so there are essentially $n = 9$ states. Each vehicle has $p = 5$ possible actions: go “North”, “South”, “East”, “West”, and “Stay”, see Fig. 2.

When an agent is at a boundary bin and takes an action to go off the boundary, the environment will make it stay in the same bin. Also note that the action “Stay” is deterministic: if a vehicle decides to stay in a certain bin, the environment will not send it anywhere else. The cost $C(i,a)$ is chosen uniformly at random in the interval $[0,1]$. The unconstrained MDP solution (which is known to give deterministic optimal policies [3]) will send all the vehicles to the bin with lowest cost.

We apply Algorithm 1 with different values of constant $\alpha_t = 0$, $\alpha_t = 1$, or $\alpha_t = 5$ for all $t$. The figure shows the performance of the policy $\hat{\pi} = (Q_1, Q_2, \ldots)$ generated by the proposed online algorithm for different values of $\alpha_t$ as compared to the optimal offline policy $\tilde{\pi} = (Q^*, Q^*, \ldots)$:
The results in Fig. 3 corroborate the theoretical findings in Theorem 1 and show that as \( N \) tends to infinity, the learning agent performance is as well as the observed agent, i.e., the term in (27) is going to zero. Moreover, the figure shows the sensitivity of the online learning algorithm to the design choice \( \alpha_t \). It is worth noting that when \( \alpha_t = 0 \), the algorithm does not use the transition matrices \( G_k \), so the algorithm can still provide good performance even without any knowledge on the uncertainties in the environment, and just using the observed state probability distribution of the optimal agent.

VII. CONCLUSION

In this paper, we consider an MDP framework for a multi-agent system where a learning agent observes the state \( pd \) of other agents. In the presence of a large number of agents, the state \( pd \) of other agents can be estimated by simply considering the number of agents distributed among the states of the MDP. We propose an online learning algorithm where the learning agent updates its decision policy through using a descent gradient technique to learn optimal MDP policies with average expected cost by using these observations. We show that the online-learned policy performs at least as well as the one followed by an observed agent.

As future work, it would be interesting to study a distributed version of the algorithm in which the learning agent has access to only partial information on the state \( pd \), e.g. the density of agents in the neighboring states. The network structure of transitions and the connectivity between the states would then be the key components that impact the performance of the online learning algorithm.

REFERENCES


