Learning from Multiple Independent Advisors in Multi-agent Reinforcement Learning

Sriram Ganapathi Subramanian
Vector Institute, Toronto, Canada
University of Waterloo, Waterloo, Canada
sriram.subramanian@vectorinstitute.ai

Matthew E. Taylor
University of Alberta, Edmonton, Canada
Alberta Machine Intelligence Institute, Edmonton, Canada
matthew.e.taylor@ualberta.ca

Kate Larson
University of Waterloo
Waterloo, Canada
kate.larson@uwaterloo.ca

Mark Crowley
University of Waterloo
Waterloo, Canada
mcrowley@uwaterloo.ca

ABSTRACT

Multi-agent reinforcement learning typically suffers from the problem of sample inefficiency, where learning suitable policies involves the use of many data samples. Learning from external demonstrators is a possible solution that mitigates this problem. However, most prior approaches in this area assume the presence of a single demonstrator. Leveraging multiple knowledge sources (i.e., *advisors*) with expertise in distinct aspects of the environment could substantially speed up learning in complex environments. This paper considers the problem of simultaneously learning from multiple independent advisors in multi-agent reinforcement learning. The approach leverages a two-level Q-learning architecture, and extends this framework from single-agent to multi-agent settings. We provide principled algorithms that incorporate a set of advisors by both evaluating the advisors at each state and subsequently using the advisors to guide action selection. We also provide theoretical convergence and sample complexity guarantees. Experimentally, we validate our approach in three different test-beds and show that our algorithms give better performances than baselines, can effectively integrate the combined expertise of different advisors, and learn to ignore bad advice.

KEYWORDS

Multi-agent systems; Multi-agent reinforcement learning; Learning from action advising; Reinforcement learning; Sample efficiency

ACM Reference Format:
Sriram Ganapathi Subramanian, Matthew E. Taylor, Kate Larson, and Mark Crowley. 2023. Learning from Multiple Independent Advisors in Multi-agent Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Reinforcement learning (RL) has been successful in obtaining superhuman performances in a wide range of challenges such as Atari games [21], Go [30], and simple robotic tasks like opening doors and learning visuomotor policies [16]. However, it has not been straightforward to replicate these successes in complex real-world problems. One reason is that these problems often have a multi-agent structure, where more than one learning agent participates at the same time, resulting in complicated dynamics. Despite research advances in multi-agent reinforcement learning (MARL) [10], poor sample efficiency in existing algorithms is one issue that still causes significant hurdles in applying MARL to complex problems [28].

Using external sources of knowledge that help in accelerating MARL training is one solution [2], which has extensive support in literature [28]. However, most prior work include two limiting assumptions. First, all demonstrations need to come from a single demonstrator [4]. In complex MARL environments, since agents learn policies that meet the twin goals of responding to changing opponent(s) and environments [18], a learner can likely benefit from multiple knowledge sources that have expertise in different parts of the environment or different aspects of the task. Second, all demonstrations are near-optimal (i.e., from an "expert") [24]. In practice, these knowledge sources are typically sub-optimal, and we broadly refer to them as *advisors* (to differentiate from experts).

In this paper, we provide an approach that simultaneously leverages multiple different (sub-optimal) advisors for MARL training. Since the advisors may provide conflicting advice in different states, an algorithm needs to resolve such conflicts to take advantage of all the advisors effectively. We propose a two-level learning architecture and formulate a Q-learning algorithm for simultaneously incorporating multiple advisors in MARL, improving upon the previous work of Li et al. [17] in single-agent RL. This architecture uses one level to evaluate advisors and the other learns values for actions. Further, we extend our approach to an actor-critic variant that applies to the centralized training and decentralized execution (CTDE) setting [20]. Since RL is a fixed point iterative method [36], we provide convergence results, proving that our Q-learning algorithm converges to a Nash equilibrium [22] (under common assumptions). Additionally, we provide a detailed finite-time analysis of our Q-learning algorithm under two different types of learning rates. Finally, we experimentally study our approach in three different multi-agent test-beds, in relation to standard baselines.

Since we relax the two limiting assumptions regarding learning from demonstrators in MARL, our hope is that this approach will spur successes in real-world applications, such as autonomous driving [9] and fighting wildfires [13], where MARL methods could use existing (sub-optimal) solutions as advisors to accelerate training. Our full paper with appendices is available on arXiv [34].

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

2 RELATED WORK

This work is most related to the approach of *reinforcement learning from expert demonstrations* (RLED) [24]. A well-known RLED technique is *deep Q-learning from demonstrations* (DQfD) [11], which combines a temporal difference (TD) loss, an L2 regularization loss, and a classification loss that encourages actions to be close to that of the demonstrator. Another method, *normalized actor-critic* (NAC) [14], drops the classification loss and is more robust under imperfect demonstrations. However, NAC is prone to weaker performances than DQfD under good demonstrations due to the absence of classification loss. A different approach, *human agent transfer* (HAT) [38], extracts information from limited demonstrations using a classifier, while *confidence-based human-agent transfer* (CHAT) [42] improves HAT by using a confidence measurement to safeguard against sub-optimal demonstrations. A related approach is the teacher-student framework [39], where a pretrained policy (teacher) can be used to provide limited advice to a learning agent (student). Subsequent works expand this framework towards interactive learning [1], however, almost all works in this area assume a moderate level expertise for the teacher. Moreover, these are all independent methods primarily suited for single-agent environments, and may not be directly applicable in MARL context.

Furthermore, external knowledge sources have also been used in MARL [28], where prior works often assume near optimal experts[25, 44] or are only applicable to restrictive settings, such as fully cooperative or zero-sum competitive games [23, 29, 30, 41, 47]. Leno et al. [29] introduced a framework where an agent can learn from its peers in a shared learning environment, in addition to learning from the environmental rewards. Here the peers can be sub-optimal, however this work only applies to cooperative environments. Other works have provided a cooperative teaching framework for hierarchical learning [15, 45]. For multi-agent general-sum environments, *advising multiple intelligent reinforcement agents - decision making* (ADMIRAL-DM) [33] is a Q-learning approach that incorporates real-time information from a single online sub-optimal advisor.

One limitation of many prior works is the assumption of a single source of demonstration. In MARL, it may be possible to obtain advisors from different sources of knowledge that provide conflicting advice. For single-agent settings, Li et al. [17] provides the two-level Q-learning (TLQL) algorithm that incorporates multiple advisors in RL. The TLQL maintains two Q-networks, where the first Q-network (high-level) keeps track of each advisor's performance and the second Q-network (low-level) learns the quality of each action. We improve upon TLQL and make it applicable to MARL settings.

3 BACKGROUND

Stochastic Games: A N-player stochastic game is represented by a tuple $\langle S, A^1, \ldots, A^N, r^1, \ldots, r^N, P, \gamma \rangle$, where S is the state space, A^j is the action space of the agent $j \in \{1, \ldots, N\}$, and $r^j : S \times A^1 \times \cdots \times A^N \rightarrow \mathcal{R}$ is the reward function of j. Also, $P : S \times A^1 \times \cdots A^N \rightarrow \Omega(S)$ is the transition probability that determines the next state given the current state and the joint action of all agents, where Ω is a probability distribution. Finally, $\gamma \in [0, 1)$ is the discount factor. At each time t, all agents observe the global state s and take a local action a^j [27]. The joint action $\boldsymbol{a} = \{a^1, \ldots, a^N\}$ determines the immediate reward r^j for j and the next state of the system s'. Each agent learns a suitable policy that gives the best responses to its opponent(s). The policy is denoted by $\pi^j : S \rightarrow \Omega(A^j)$. Let $\boldsymbol{\pi} \triangleq (\pi^1, \ldots, \pi^N)$ be the joint policy of all agents. At a state s, the value function of j under the joint policy $\boldsymbol{\pi}$ is $v^j_{\boldsymbol{\pi}}(s) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{\pi, P}[r^j_t | s_0 = s, \boldsymbol{\pi}]$. This represents the expected discounted future reward of j, when all agents follow the policy $\boldsymbol{\pi}$ from the state s. Related to the value function, is the action-value function or the Q-function. The Q-function of agent j, under the policy $\boldsymbol{\pi}$, is given by, $Q^j_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a}) + \gamma \, \mathbb{E}_{s' \sim P}[v^j_{\boldsymbol{\pi}}(s')]$.

The setting we consider is general-sum stochastic games, where the reward functions of the different agents can be related in any arbitrary fashion. In this setting, the *Nash equilibrium* is typically considered as the solution concept [12], where the joint policy $\boldsymbol{\pi}_* = [\pi^1_*, \ldots, \pi^N_*]$ for all $s \in S$ and all j satisfies $v^j(s; \pi^j_*, \boldsymbol{\pi}^{-j}_*) \geq v^j(s; \pi^j, \boldsymbol{\pi}^{-j}_*)$. Here, $\boldsymbol{\pi}^{-j}_* \triangleq [\pi^1_*, \ldots, \pi^{j-1}_*, \pi^{j+1}_*, \ldots, \pi^N_*]$ represents the joint policy of all agents except j. In a Nash equilibrium, each agent plays the best response to the other agents and any deviation from this response is guaranteed to be worse off. Further, Hu and Wellman [12] proved that the Q-updates of an agent j, using the Nash payoff at each stage eventually converges to its Nash Q value (Q^j_*), which is the action-value obtained by the agent j when all agents follow the joint Nash equilibrium policy for infinite periods.

Two-level Q-learning: The TLQL algorithm [17] enables single-agent learning under the simultaneous presence of multiple advisors providing conflicting demonstrations. Here, the challenge is to determine which advisor to trust in a given state. In this regard, the TLQL contains two Q-tables, a high-level Q-table (abbreviated as high-Q) and a low-level Q-table (abbreviated as low-Q). The high-Q stores the value of the $\langle s, ad \rangle$ pair, where $ad \in AD$ represents an advisor (with AD representing the set of all advisors). The high-Q also stores the value of following the RL policy in addition to each advisor. The low-Q maintains the value of each state-action pair.

At each time step, the agent observes the state and selects an advisor (or the RL policy) from the high-Q using the ϵ-greedy strategy. If the high-Q returns an advisor, then the advisor's recommended action is performed. If the RL policy is returned, then an action is executed from the low-Q based on the ϵ-greedy strategy [35]. The low-Q is updated using the vanilla Q-learning Bellman update [43]. Subsequently, the high-Q is updated using a **synchronization** step. In this step, when an advisor's action is performed, the value of the advisor in the high-Q is simply assigned the value of that action from the low-Q. Finally, the high-Q of the RL policy is updated using the relation $highQ(s, RL) = \max_a lowQ(s, a)$. This synchronization update of high-Q preserves the convergence guarantees, due to the policy improvement guarantee in single-agent Q-learning [35].

There are two important limitations of TLQL. First, the high-Q that represents the value of the advisors also depends on the RL policy through the synchronization step. This Q value represents the value of taking the action suggested by the advisor at the current state and then following the RL policy from the next state onward. This definition is problematic since at the beginning of training, the RL policy is sub-optimal, and the objective is to accelerate learning by relying on external advisors and avoid using the RL policy at all. As advisors are evaluated at each state using the RL policy, it is likely that the most effective advisor among the set of advisors is not being followed until the RL policy improves.

At this stage, it might be possible to simply follow the RL policy itself, defeating the purpose of learning from advisors. Second, the advisors have not been evaluated at the beginning of learning. Hence, it is impossible to find the most suitable advisor to follow, from the available advisors. While TLQL simply follows an ϵ-greedy exploration strategy, this approach could take many data samples to figure out the right advisor. We address both these limitations.

4 TWO-LEVEL ARCHITECTURE IN MARL

We consider a general-sum stochastic game, where there are a set of agents that are learning a policy with an objective of providing a best response to the other agents as well as the environment. Each agent j can access a finite set of (possibly sub-optimal) independent advisors AD^j. We use ad^j to represent an advisor of j, where $ad^j \in \{ad^j_1, \ldots, ad^j_{|AD^j|}\}$. Each advisor ad^j can be queried by j to obtain an action recommendation at each state of the stochastic game. These online advisors provide real-time action advice to the agent, which helps in learning to dynamically adapt to opponents. We consider a centralized training setting and assume 1) the advisors are fixed and do not learn, 2) the communication between agents and their advisors is free, 3) there is no communication directly between learning agents, 4) the environment is fully observable (i.e., an agent can observe the global state, all actions, and all rewards), and 5) the state and action spaces are discrete. Though we require these assumptions for theoretical guarantees, we will show that it is possible to relax a number of these assumptions in practice.

To make TLQL applicable to multi-agent settings, we parameterize both the Q-functions with the joint actions, as is common in practice [18]. Also, we do not maintain the RL policy in the high-Q table and do not perform a synchronization step. These steps are no longer needed to preserve the convergence results in multi-agent settings, since we do not have a policy improvement guarantee (unlike in single-agent settings) [37]. Instead, we choose to use the probabilistic policy reuse (PPR) technique [6], where a hyperparameter ($\epsilon' \in [0, 1]$) decides the probability of following any advisor(s) (i.e., using the high-Q) or the agents' own policy (i.e., using the low-Q) for action selection, at each time step during training. This hyperparameter starts with a high value (maximum dependence on the available advisor(s)) at the beginning of training and is decayed (linearly) over time. After some finite time step during training, the value of this hyperparameter goes to 0 (no further dependence on any advisor(s)) and the agent only uses its low-Q (own policy) for action selection. This helps in two ways: 1) in the time limit ($t \to \infty$), a learning agent has the possibility of recovering from poor advising (by learning from the environment), and 2) eventually the trained agent can be independently deployed (with no requirement of having access to any advisor(s)).

The general structure of our proposed *Multi-Agent Two-Level Q-Learning* (MA-TLQL) algorithm is given in Figure 1. Since we are in a fully observable setting, like [12], we specify that each agent maintains copies of the Q-tables of other agents from which it can obtain the joint actions of other agents for the current state. If such copies cannot be maintained, agents could use the previously observed actions of other agents for the joint action as done in prior works [33, 46]. We use the two-level architecture, where each agent will maintain a high-Q as well as a low-Q. The

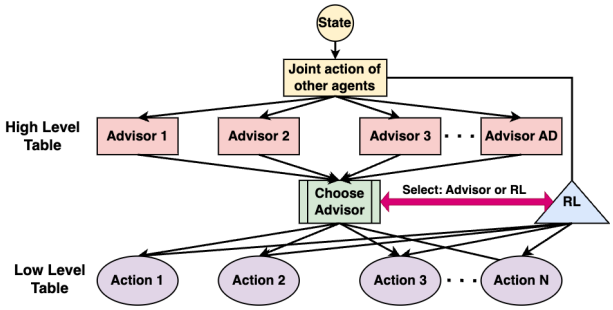


Figure 1: Structure of MA-TLQL, for a representative agent having access to a set of AD advisor(s)

high-Q provides a value for the $\langle s, \boldsymbol{a}^{-j}, ad^j \rangle$ tuples, where $\boldsymbol{a}^{-j} = \{a^1, \ldots, a^{j-1}, a^{j+1}, \ldots, a^N\}$ is the joint action of all agents except the agent j. This high-Q is a value estimate for the advisor ad^j as estimated by the agent j at the state s and joint action $\boldsymbol{a}^{-j}$. The high-Q estimates are updated with an evaluation update given by

$$highQ^j_{t+1}(s, \boldsymbol{a}^{-j}, ad^j) = highQ^j_t(s, \boldsymbol{a}^{-j}, ad^j) + \alpha\Big((r^j_t + \gamma highQ^j_t(s', \boldsymbol{a'}^{-j}, ad^j) - highQ^j_t(s, \boldsymbol{a}^{-j}, ad^j)\Big), \quad (1)$$

where s and s' are the states at t and $t+1$, and α is the learning rate. Also, $\boldsymbol{a}^{-j}$ and $\boldsymbol{a'}^{-j}$ are joint actions at s and s', respectively.

As described previously, a hyperparameter is used to decide between choosing to follow an advisor or the RL policy. If the agent follows an advisor, the high-Q is used to select an advisor using an ensemble selection technique. Let us denote, $\boldsymbol{Q}^j = \{highQ^j(s, \boldsymbol{a}^{-j}, ad^j_1), \ldots, highQ^j(s, \boldsymbol{a}^{-j}, ad^j_m)\}$, to represent the high-$Q$ estimates of a set of M advisors (with $|M| = m$) advising the same action a^j to an agent j. Here, ad^j_i represents an advisor $i \in \{1, \ldots, m\}$ of j. Then the value of vote for action a^j, at the state s and the joint action $\boldsymbol{a}^{-j}$, denoted by $\mathcal{V}^j(s, \boldsymbol{a}^{-j}, a^j)$, is calculated as

$$\mathcal{V}^j(s, \boldsymbol{a}^{-j}, a^j) = \max \boldsymbol{Q}^j + \sum^m_{i=1, i \neq \arg\max_i highQ^j(s, \boldsymbol{a}^{-j}, ad^j_i)} \frac{1}{\mu(s)} highQ^j(s, \boldsymbol{a}^{-j}, ad^j_i). \quad (2)$$

Here, $\mu(s)$ is the number of times the agent has visited the state s. In Eq. 2, if an action is recommended by more than one advisor, the value of its vote is a weighted sum of all high-Q estimates of advisors recommending that action. Each high-Q estimate (except the best high-Q estimate) is weighted by the reciprocal of the number of times the respective state is visited. In this way, when a state is visited many times, the advisor with the best high-Q estimate is likely to be followed (wisdom of individual). When a state is visited only a few times, then the action suggested by a majority of advisors is likely to be selected (wisdom of crowd). From Eq. 2, if an action a^j is recommended by only one advisor, then the value of vote for a^j will be equal to the high-Q estimate of that advisor. After the value of votes for all actions are calculated, the action with the maximum value of vote is executed, and the high-Q estimate of the advisor recommending this action is updated by the agent j using Eq. 1.

If the agent decides to use its RL policy, it uses its low-Q, which contains a value for the $\langle s, \boldsymbol{a}^{-j}, a^j \rangle$ tuples (value for each action). At each step, the low-Q is updated using a control update as follows:

$$lowQ^j_{t+1}(s, \boldsymbol{a}^{-j}, a^j) = lowQ^j_t(s, \boldsymbol{a}^{-j}, a^j) \\ +\alpha\big(r^j_t + \gamma \max_{a'^j} lowQ^j_t(s', \boldsymbol{a'}^{-j}, a'^j) - lowQ^j_t(s, \boldsymbol{a}^{-j}, a^j)\big). \quad (3)$$

Now we describe how MA-TLQL addresses the two limitations of TLQL. The first was the dependence of high-Q on the RL policy in TLQL. Note, the high-Q in MA-TLQL maintains the Q values of the advisor themselves, i.e., the value of following the advisor's policy from the current state onward (see Eq. 1). Thus, the coupling between the advisor values and the RL policy is removed (no synchronization). The second was the difficulty in picking the right advisor in TLQL. MA-TLQL uses an ensemble technique to choose the advisor during the early stages of learning. In later stages, it switches to following the best advisor according to the high-Q estimates, which addresses this limitation of TLQL. In Appendix J, we present a toy example that illustrates the limitations of TLQL.

We provide the complete pseudocode for a tabular implementation of the MA-TLQL algorithm in Appendix A (Algorithm 1). Further, we extend this approach to large state-action environments using a neural network based implementation (Algorithm 2), which uses a target network and a replay buffer, as in the Deep Q-learning (DQN) algorithm [21]. We also provide an actor-critic implementation (Algorithm 3) which is suitable for CTDE [20]. We will refer to this algorithm as *multi-agent two-level actor-critic* (MA-TLAC). In MA-TLAC, each agent has two actors and two critics (high-level and low-level), where the respective Q-functions serve as the critic and the corresponding policies serve as the actors. In this CTDE method, agents can obtain global information (including actions and rewards of other agents) during training, however, the agents only require access to its local observation during execution. This makes our method applicable to partially observable environments as in Lowe et al. [20]. MA-TLAC applies to continuous state space environments as well (refer to Appendix A for more details).

5 THEORETICAL RESULTS

We present a convergence guarantee for tabular MA-TLQL and characterize the convergence rate. For these results, we build on some prior works that provide several fundamental results on the nature of stochastic iterative functions [3, 5]. We apply these to MA-TLQL in general-sum stochastic games using three assumptions from Hu and Wellman [12], where the first two are standard [36].

Assumption 1. *Every $s \in \mathcal{S}$ and $a^j \in A^j$, for every agent j are visited infinitely often, and the reward function ($\forall j$) stays bounded.*

Assumption 2. *For all s, t, and $\boldsymbol{a}$, $0 \leq \alpha_t(s, \boldsymbol{a}) < 1$, $\sum_{t=0}^{\infty} \alpha_t(s, \boldsymbol{a}) = \infty$, $\sum_{t=0}^{\infty}[\alpha_t(s, \boldsymbol{a})]^2 < \infty$.*

Assumption 3. *The Nash equilibrium is a global optimum or saddle point in every stage game of the stochastic game.*

The third assumption is a restriction on the nature of the stochastic game. Several prior works note that this assumption is restrictive but needed to theoretically prove the convergence of Q-learning methods in general-sum stochastic games with two or more agents. In practice, however, it is still possible to observe convergence of Q-learning methods when this assumption is violated [12, 33, 46].

Now we prove our theoretical results. All theorem statements are provided here, while the proofs can be found in Appendices B – D. First, we provide the convergence guarantee for the low-Q. Recall, the PPR technique guarantees that the MA-TLQL dependence on high-Q is only until a finite time step during training. After this step, the agent only uses its low-Q for action selection. As the convergence result in Theorem 1 is provided in the time limit ($t \to \infty$), the influence of high-Q can be neglected for this result.

Theorem 1. *Given Assumptions 1, 2, 3, the low-Q values of an agent j converges to its Nash Q value in the limit ($t \to \infty$).*

Next, we provide sample complexity bounds for the MA-TLQL algorithm. Instead of explicitly considering the high-Q values, we specify that the underlying joint policy has a covering time of L. The covering time specifies an upper bound on the number of time steps needed for all state-joint action pairs to be visited at least once starting from any state-joint action pair. Further, since the action selection is only based on the low-Q values in the limit ($t \to \infty$), we are most interested in the sample complexity of low-Q, where the dependence on the high-Q is effectively represented by L.

Regarding sample complexity, as is done in [5], we distinguish between two kinds of learning rates. Consider the following equation for the low-Q (rewriting Eq. 3 and dropping *low* for simplicity),

$$Q^j_{t+1}(s_t, \boldsymbol{a}_t) = \big(1 - \alpha^{\omega}_t(s_t, \boldsymbol{a}_t)\big)\big(Q^j_t(s_t, \boldsymbol{a}_t)\big) \\ +\alpha^{\omega}_t(s_t, \boldsymbol{a}_t)\big(r^j_t + \gamma \max_{a^j} Q^j_t(s_{t+1}, \boldsymbol{a}_{t+1})\big). \quad (4)$$

The value of $\alpha^{\omega}_t(s, \boldsymbol{a}) = \frac{1}{[\#(s, \boldsymbol{a}, t)^{\omega}]}$, where $\#(s, \boldsymbol{a}, t)$ is the number of times until t that the joint action $\boldsymbol{a}$ is performed at s. Here, we consider $\omega \in (1/2, 1]$. The learning rate is linear if $\omega = 1$, and the learning rate is polynomial if $\omega \in (1/2, 1)$.

The next theorem provides a lower bound on the number of time steps needed for convergence in the case of a polynomial learning rate. From Assumption 1, let us specify that all rewards for the agent j are bounded by $R^j_{\max}$. We consider a variable $Q^j_{\max}$, which denotes the maximum possible low-Q value for the agent j, which is bounded by $Q^j_{\max} = R^j_{\max}/(1-\gamma)$. Additionally, we also use another variable $\beta = (1-\gamma)/2$ to present our upcoming results concisely.

Theorem 2. *Let us specify that with probability at least $1 - \delta$, for an agent j, $||Q^j_T - Q^j_*||_{\infty} \leq \epsilon$. The bound on the rate of convergence of low-Q, Q^j_T, with a polynomial learning rate of factor ω is given by (with Q^j_* as the Nash Q-value of the agent j)*

$$T = \Omega\Big(\Big(\frac{L^{1+3\omega} Q^{2,j}_{\max} \ln(\frac{|S|\Pi_i |A_i| Q^j_{\max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2}\Big)^{1-\omega}/L \\ +\Big((\frac{L}{\beta}\ln\frac{Q^j_{\max}}{\epsilon} + 1)/2\Big)^{\frac{1}{1-\omega}}\Big). \quad (5)$$

Assuming the same action spaces for all agents (i.e. $|A_1| = |A_2| = \cdots = |A_N| = |A|$), we note that the dependence on the number of agents is $\ln |A|^N = N \ln |A|$. Overall this results in a sub-linear dependence on the number of agents based on the value of ω, which is far superior to recent works that report an exponential dependence on the number of agents when learning in general-sum stochastic game environments (with an arbitrary number of agents) for convergence to a Nash equilibrium [19, 31]. Further, the dependence on the state space and action space in Theorem 2 is sub-linear ($\ln |S|$), and the dependence on the covering time is $\Omega(L^{2\omega - 3\omega^2} + L^{1/1-\omega})$, which is a polynomial dependence.

The next theorem considers the linear learning rate case.

Theorem 3. *Let us specify that with probability at least* $1-\delta$*, for an agent* j*,* $||Q_T^j - Q_*^j||_\infty \leq \epsilon$*. The bound on the rate of convergence of low-Q,* Q_T^j*, with a linear learning rate is given by*

$$T = \Omega\Big((L+\psi L+1)^{\frac{1}{\beta}} \ln \tfrac{Q^j_{\max}}{\epsilon} \frac{Q^{2,j}_{\max} \ln(\frac{|S||\Pi_i||A_i|Q^j_{\max}}{\delta\beta\epsilon\psi})}{\beta^2\epsilon^2\psi^2}\Big), \quad (6)$$

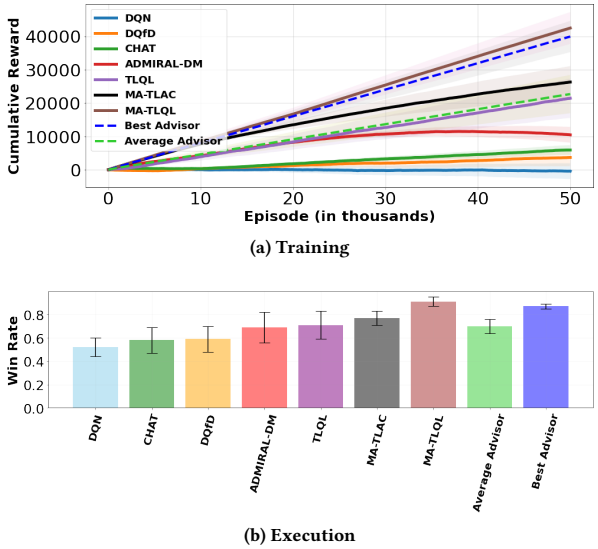
where ψ *is a small arbitrary positive constant satisfying* $\psi \leq 0.712$*.*

Theorem 3 shows that the bound is linear in the number of agents and sub-linear in the state and action spaces. This linear dependence on the number of agents is also superior to prior results [19, 31]. Note, the dependence on the covering time in Theorem 3 could be much worse than that of Theorem 2, depending on the value of $Q^j_{\max}$ and ϵ. Since the value of ϵ is small, the dependence is certainly worse than that obtained for the polynomial learning rate case. Also, the dependence on $Q^j_{\max}$ is exponential as opposed to a polynomial dependence for Theorem 2. The last two theorems illustrate the performance benefit in using a polynomial learning rate as opposed to a linear learning rate in our algorithm.

6 EXPERIMENTS AND RESULTS

We consider three different experimental domains, one each for competitive, cooperative, and mixed settings, where each agent has access to a set of four advisors. We use neural network implementations of MA-TLQL and MA-TLAC, along with 5 other baselines: DQN [21], DQfD [11], CHAT [42], ADMIRAL-DM [33], and TLQL [17]. In Appendix F, we tabulate the characteristics of these baselines and provide further details regarding our choices. Since CHAT and ADMIRAL-DM assume the presence of a single advisor, we use a weighted random policy approach for implementing these two algorithms in the multiple-advisor setting, as in Li et al. [17]. If different advisors provide different actions at the same state, each action is weighted based on the number of advisors suggesting that action. For DQfD, during pre-training [11], we populate the replay buffer using advisor demonstrations from all the available advisors. For all our experiments, we will describe the critical details here, while the complete description is in Appendix K. All the experiments are repeated 30 times, with averages and standard deviations reported. For statistical significance we use the unpaired 2-sided t-test and report p-values, where $p < 0.05$ is considered significant. The tests compare the highest performing algorithm (typically MA-TLQL) with the second-best baseline and best/average advisor performance. We conduct a total of seven experiments. The code for all experiments is open-sourced [32]. Appendix K tabulates all our experimental settings. Appendix L provides the hyperparameter details and Appendix M contains the wall clock times.

Experiments 1–4 use the competitive, two-agent version of Pommerman [26]. The environment is complex, with each state containing roughly 200 elements related to agent position and special features (e.g., bombs). The reward function is sparse: agents only receive a terminal reward of $\{-1, 0, +1\}$. Experiments are conducted in two phases. In the first phase (training), our algorithms and the baselines train against a standard DQN opponent for 50,000 episodes, where we plot the cumulative rewards. During this phase, algorithms can use advisors to accelerate training. In the second phase (execution), we test the performance of the trained policies against DQN for 1000 episodes, where we plot the win rate (fraction of games won) for each algorithm. During this phase, agents cannot access advisors, take no exploratory actions, and do not learn. All advisors pertaining to these four experiments are rule-based agents.



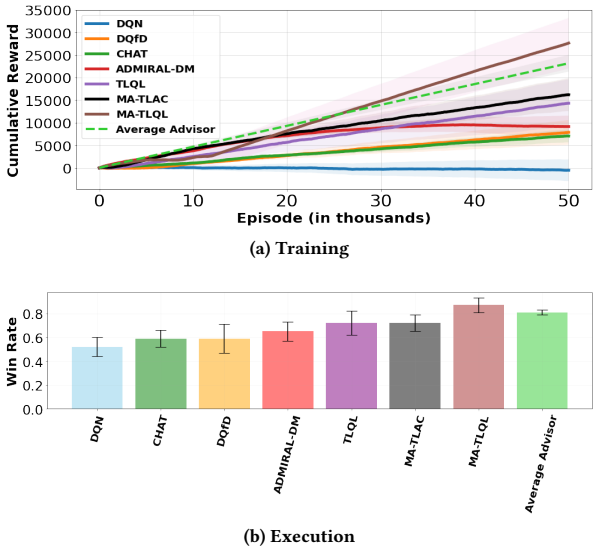
(a) Training

(b) Execution

Figure 2: Two agent Pommerman with four sufficient advisors of different quality (Experiment 1)

Experiment 1: Our first experiment uses a set of four advisors ranked in terms of quality from Advisor 1 to Advisor 4. Here, Advisor 1 is the best advisor, capable of teaching the agent all skills needed to win the game of Pommerman, and Advisor 4 only suggests random actions. In Pommerman, there is a fixed set of six skills that an agent needs to master to be able to win [26]. Since this set of advisors can teach all these skills, we say the agent has access to a *sufficient set* of advisors. We plot the training and execution performances in Figure 2(a) and (b) respectively, including the performance of the best and average advisors (average of all Advisors 1–4) against DQN. MA-TLQL gives the best performance ($p < 0.01$) and is the only algorithm providing a better performance than the best advisor ($p < 0.11$) in both training and execution. MA-TLAC performs better than the average advisor ($p < 0.04$). None of the others show better performances than the average advisor. CHAT and ADMIRAL-DM are not capable of leveraging and distinguishing amongst a set of advisors. DQfD uses pre-training, which is not very effective in the non-stationary multi-agent context. Learning from online advising is preferable in MARL. Also, DQfD and CHAT are independent techniques that are not actively tracking the opponent's performance. While TLQL is capable of learning from multiple advisors, its independent nature in addition to coupling of advisor values with the RL policy reduces its effectiveness in multi-agent environments. MA-TLQL gives a better performance than MA-TLAC in both training and execution ($p < 0.01$). As noted previously, the Q-learning family of algorithms tends to induce a positive bias while using the maximum action value, which leads to

providing the best possible response [40]. This explains the superior performance of MA-TLQL. We conclude that MA-TLQL is capable of leveraging a set of good and bad advisors. Further, the training results in Figure 2(a) show that MA-TLQL is able to learn a better policy faster than the baselines by using advisors ($p < 0.01$). The evaluation results in Figure 2(b) show that amongst all algorithms trained for the same number of episodes, MA-TLQL provides the best performance, when deployed without any advisors ($p < 0.01$). Both observations point to better sample efficiency in MA-TLQL. Supplementary experiments in Appendix E show that MA-TLQL comes to relying more on good advisors than poor advisors, as compared to the baselines, illustrating its superiority.



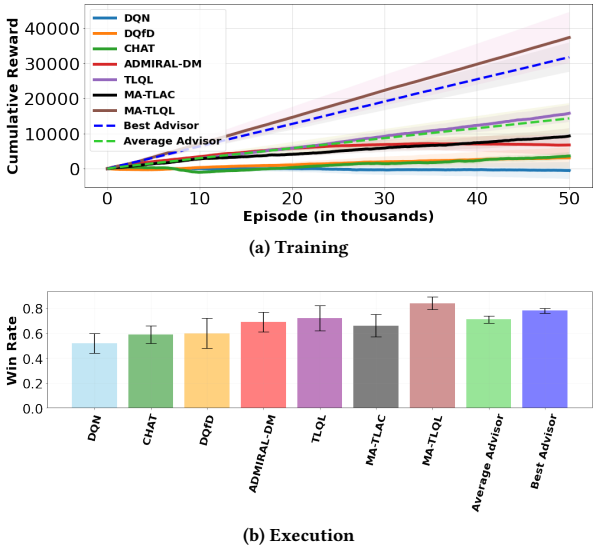
(a) Training

(b) Execution

Figure 3: Two-agent Pommerman with four sufficient advisors of similar quality (Experiment 2)

Experiment 2: We use the same domain as in Experiment 1, but with a different set of advisors. Now, all four advisors can teach strictly different Pommerman skills. For example, Advisor 1 can teach how to escape the enemy (and nothing else), and Advisor 2 can teach how to obtain necessary power-ups (and nothing else — full details are in Appendix K). These advisors provide psuedo-random action advice in states outside their expertise. This set of advisors is also a sufficient set. Now, learning agents must decide what advisor to listen to in the current state. From the training and execution results in Figure 3(a) and (b), we see that MA-TLQL gives the best overall performance ($p < 0.02$), exceeding the average performance of the four advisors ($p < 0.05$). Since all four advisors have similar quality, we only choose to use the average performance of the four advisors in this experiment for comparison. We conclude that MA-TLQL is capable of leveraging the combined knowledge of a set of advisors with different individual expertise, during learning.

Experiment 3: We use the same domain as in Experiment 1 but with a different set of four advisors. These advisors are similar to the set of advisors in our first experiment, where Advisor 1 gives the best advice throughout the domain, and Advisor 4 is random.



(a) Training

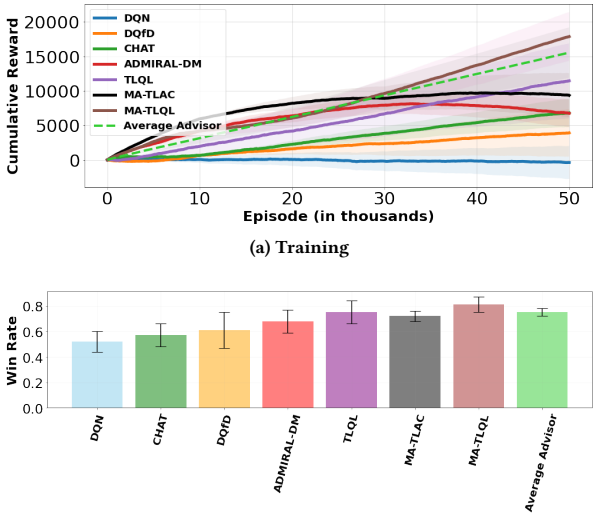
(b) Execution

Figure 4: Two-agent Pommerman with four insufficient advisors of different quality (Experiment 3)

However, this set of advisors is *not* capable of teaching all the strategies (i.e, Pommerman skills) needed to win in Pommerman, and compose an insufficient set (more details in Appendix K). It is critical for agents to learn from the environment in addition to the advisors. Training and execution results in Figure 4 shows the superior performance of MA-TLQL, the only algorithm that outperforms the best advisor ($p < 0.05$) and all baselines ($p < 0.02$). Surprisingly, TLQL performs better than MA-TLAC ($p < 0.02$), likely due to the positive bias of Q-learning. This experiment reinforces the observation that MA-TLQL is capable of learning from good advisors and avoids bad advisors (also see Appendix E). Since MA-TLQL outperforms the best advisor, this experiment demonstrates that MA-TLQL can learn from both, advisors and through direct interactions with the environment, hence having a much improved sample efficiency as compared to other algorithms that learn only from the environment. This is observed during both training and execution.

Experiment 4: This is similar to the Experiment 2: four advisors have similar quality, but each understands a different Pommerman skill. However, our set of advisors in this experiment are insufficient to teach all the skills in Pommerman, and the agent must also learn from the environment. The results in Figure 5 shows that MA-TLQL is capable of leveraging the combined expertise of the advisors and learning from the environment to obtain the best performance, as compared to the baselines ($p < 0.04$) and advisors ($p < 0.05$). This makes MA-TLQL more sample efficient than the prior algorithms.

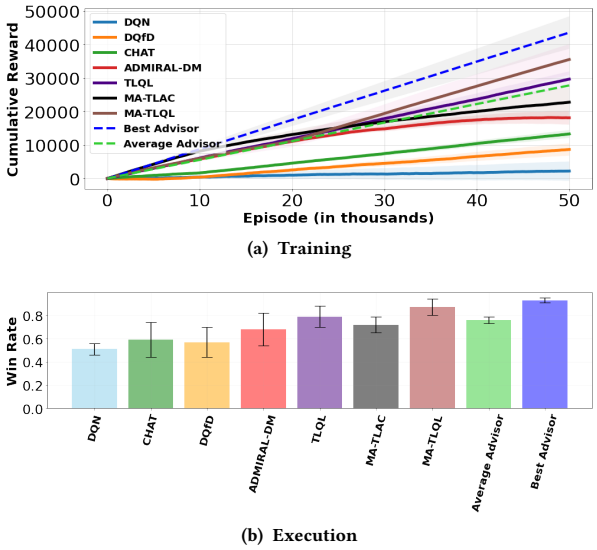
Experiment 5: We now switch to a four-agent version of Pommerman, which is two vs. two. This is a mixed setting as agents need to learn cooperative as well as competitive skills. Overall, this is a more complex domain with a larger state space. We consider four sufficient advisors of different quality, similar to Experiment 1. We conduct two phases — training (for 50,000 episodes) and execution (for 1000 episodes). The training and execution results in Figure 6



(a) Training

(b) Execution

Figure 5: Two-agent Pommerman with four insufficient advisors of similar quality (Experiment 4)

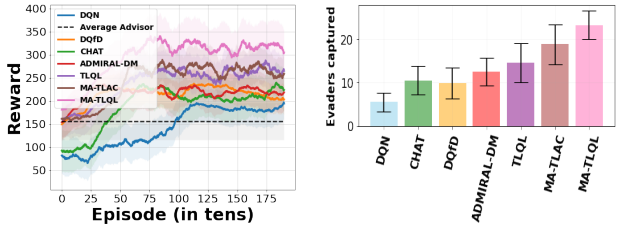


(a) Training

(b) Execution

Figure 6: Team (mixed) Pommerman (Experiment 5)

show that MA-TLQL provides the best performance compared to the baselines ($p < 0.04$) but does not perform better than the best available advisor. Since this is a more complex domain, MA-TLQL needs a larger training period for learning good policies. However, MA-TLQL still performs better than the average performance of the four advisors ($p < 0.03$). We conclude that although MA-TLQL's performance suffers in the more difficult mixed setting, it still outperforms all the other baselines and is capable of distinguishing between good and bad advisors (see also Appendix E). From both training and execution results in Figure 6, we note that MA-TLQL has a superior sample efficiency as compared to the other baselines.

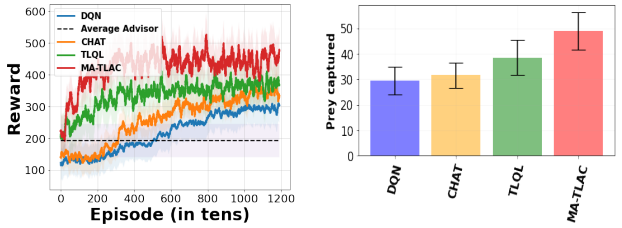


(a) Training

(b) Execution

Figure 7: Cooperative Pursuit setting (Experiment 6)

Experiment 6: This experiment switches to the cooperative Pursuit domain [7]. There are eight pursuer learning agents that learn to capture a set of 30 randomly moving targets (evaders) (details in Appendix K). We use four pre-trained DQN networks as the advisors, learning for 500, 1000, 1500, and 2000 episodes, respectively. We again have two phases — training and execution. During training, all algorithms are trained for 2000 episodes. The trained networks are then used in the execution phase for 100 episodes with no further training or influence from advisors. Figure 7(a) plots the episodic rewards obtained during training and the Figure 7(b) plots the number of targets captured in the execution phase, where MA-TLQL shows the best performance ($p < 0.03$). Hence, MA-TLQL can outperform all baselines in a cooperative environment as well.

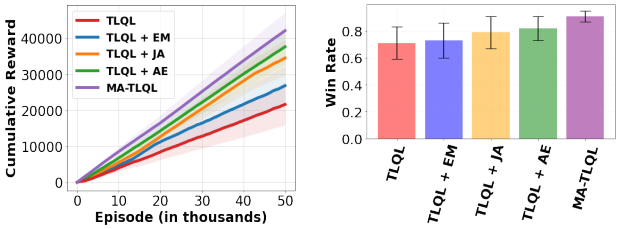


(a) Training

(b) Execution

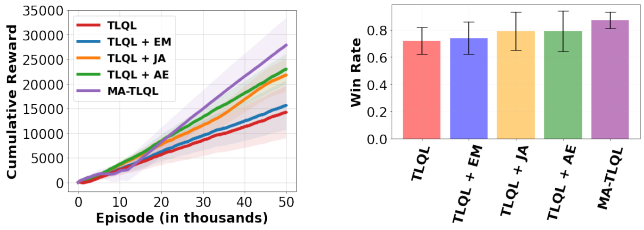
Figure 8: Mixed Predator-Prey setting (Experiment 7)

Experiment 7: This final experiment considers a mixed cooperative competitive Predator-Prey environment which is a part of the Multi Particle Environment (MPE) suite [20]. Our implementation uses a discrete action space and a continuous state space (more details in Appendix K). There are a total of eight predators trying to capture eight prey (prey are not removed, but respawned upon capture). In our experiment, each algorithm trains the predators while the prey is trained using a standard DQN opponent. The experiments have two phases of training and execution, which is modelled as a CTDE setting. Here each agent obtains information about the actions and rewards of all other agents during training, but only has local observation during execution. Since this environment requires decentralization during execution, we omit the fully centralized MA-TLQL and ADMIRAL-DM. We also omit DQfD



(a) Training (b) Execution

Figure 9: Ablation results using Experiment 1



(a) Training (b) Execution

Figure 10: Ablation results using Experiment 2

since it gave poor performances previously. As in Experiment 6, we use four pretrained DQN (predator) networks as advisors (trained for 1000, 2000, 7000, and 12000 episodes). Training is conducted for 12000 episodes and execution is conducted for 100 episodes. The training results in Figure 8(a) (plot of episodic rewards) show that MA-TLAC is the most sample efficient compared to other algorithms as it is able to leverage the available advisors better than others, thus outperforming them ($p < 0.04$). The execution results in Figure 8(b) plots the average prey captured by each algorithm. MA-TLAC outperforms others during execution as well ($p < 0.03$).

From all the p-values across the seven experiments, we note that most of our observations are statistically significant. Despite observing MA-TLQL outperforming the best advisor in many of the experiments, some of these comparisons are not statistically significant (i.e., $p \geq 0.05$). While the main experiments of the paper consider fixed advisors, our algorithms can also be implemented with learning/changing advisors (see Appendix G). In Appendix I we study performances under different numbers of advisors. Also, our algorithms can be used along with opponent modelling techniques as done by prior works [8] (more details in Appendix H).

7 ABLATION STUDY

In this section, we run an ablation study on the three components of MA-TLQL that differ from the previously introduced TLQL algorithm by Li et al. [17]. To recall these three components are: i) joint action (JA) updates, ii) ensemble method (EM), and iii) advisor evaluation (AE). For this ablation study we will consider the two-agent version of Pommerman with four sufficient advisors having different (Experiment 1) and similar quality (Experiment 2).

The ablation results corresponding to Experiment 1 are given in Figure 9, where we plot the performances of TLQL and MA-TLQL in addition to TLQL with each of the three components. In Figure 9(a) and (b), the performance of TLQL with each of the three components is better than vanilla TLQL. TLQL using the ensemble method (i.e., TLQL+EM) is able to perform better than vanilla TLQL, since at the beginning of training the Q-values of the advisors are not accurate, and the ensemble technique chooses the advisor action that is agreed upon by most advisors in the given set (in line with our discussions in Section 4). Recall that the set of four different advisors had four advisors of decreasing quality, with the first three advisors capable of teaching some useful Pommerman skills and the last advisor being just random (see Appendix K). Using the ensemble prevents the use of the random advisor, as the first three advisors are more likely to agree upon an action, increasing the possibility of the agent choosing that action. Further, we see that TLQL highly benefits from using the joint action update (i.e., TLQL+JA) instead of an independent update seen in vanilla TLQL. The joint action update explicitly considers the strategies of other agent(s) and helps in providing stronger best responses as compared to an independent update in the multi-agent environments. Finally, TLQL using advisor evaluation in the high-Q table (i.e., TLQL+AE) provides the best benefit compared to the other components. As discussed in Section 3 and Section 4, the high-Q definition in vanilla TLQL is limiting since the advisor evaluation through the high-Q is coupled with the inaccurate RL policy (and AE addresses this limitation). Further, from Figure 9, we see that MA-TLQL (integrating all the three components) shows the best performance as compared to vanilla TLQL and individual TLQL implementations with each of the three components ($p < 0.05$). Thus, MA-TLQL is able to seamlessly integrate the advantages of each of the individual components of TLQL, demonstrating its superiority.

We also consider a similar ablation study using Experiment 2 (see Figure 10). As in Figure 9, we see that TLQL with each of the three components performs better than vanilla TLQL. Since we have four advisors of similar quality where each advisor is good at a different Pommerman skill, their agreement on an action is expected to be small. Hence, the ensemble technique (i.e., TLQL+EM) provides only a small improvement over vanilla TLQL. However, the other two components (i.e., TLQL+JA and TLQL+AE) provides a good performance benefit over TLQL. Finally, MA-TLQL, that integrates all the three components, provides the best performance ($p < 0.03$).

8 CONCLUSION

This paper provided a principled approach for learning from multiple independent advisors in MARL. Inspired by Li et al. [17], we present a two-level architecture for multi-agent environments. We discuss two limitations in TLQL and address these limitations in our approach. Also, we provide a fixed point guarantee and sample complexity bounds regarding the learning of MA-TLQL. Additionally, we provided an actor-critic implementation that can work in the CTDE paradigm. Further, we performed an extensive experimental analysis of MA-TLQL and MA-TLAC in cooperative, competitive, and mixed settings, where we show that these algorithms are capable of suitably leveraging a set of advisors, and perform better than baselines. As future work, we would like to consider human advisors and further explore some avenues in the real-world context.

ACKNOWLEDGEMENTS

Resources used in preparing this research were provided by the province of Ontario and the government of Canada through CIFAR, NSERC and companies sponsoring the Vector Institute. Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

REFERENCES

[1] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara J. Grosz. 2016. Interactive Teaching Strategies for Agent Training. In *IJCAI*. IJCAI Press, New York, NY, USA, 9-15 July 2016, 804–811.

[2] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* 242 (2017), 132–171.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-dynamic programming*. Optimization and neural computation series, Vol. 3. Athena Scientific, Chestnut Street, USA.

[4] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. 2015. Reinforcement Learning from Demonstration through Shaping. In *IJCAI, July 25-31, 2015*. AAAI Press, Buenos Aires, Argentina, 3352–3358.

[5] Eyal Even-Dar and Yishay Mansour. 2003. Learning Rates for Q-learning. *Journal of Machine Learning Research* 5 (2003), 1–25.

[6] Fernando Fernández and Manuela M. Veloso. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), May 8-12, 2006*. ACM, Hakodate, Japan, 720–727.

[7] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*. Springer, IFAAMAS, Sao Paulo, Brazil, 66–83.

[8] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. 2016. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*. PMLR, New York City, US, 1804–1813.

[9] Bassam Helou, Aditya Dusi, Anne Collin, Noushin Mehdipour, Zhiliang Chen, Cristhian Lizarazo, Calin Belta, Tichakorn Wongpiromsarn, Radboud Duintjer Tebbens, and Oscar Beijbom. 2021. The Reasonable Crowd: Towards evidence-based and interpretable models of driving behavior. In *IROS*. IEEE, Prague, Czech Republic, 6708–6715.

[10] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (01 Nov 2019), 750–797. https://doi.org/10.1007/s10458-019-09421-1

[11] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2018. Deep Q-learning From Demonstrations. In *AAAI, February 2-7, 2018*. AAAI Press, New Orleans, Louisiana, USA.

[12] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *JMLR* 4, Nov (2003), 1039–1069.

[13] Piyush Jain, Sean CP Coogan, Sriram Ganapathi Subramanian, Mark Crowley, Steve Taylor, and Mike D Flannigan. 2020. A review of machine learning applications in wildfire science and management. *Environmental Reviews* 28, 4 (2020), 478–505.

[14] Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu. 2020. Reinforcement Learning from Imperfect Demonstrations under Soft Expert Guidance. In *AAAI, February 7-12, 2020*. AAAI Press, New York, NY, USA, 5109–5116.

[15] Dong-Ki Kim, Miao Liu, Shayegan Omidshafiei, Sebastian Lopez-Cot, Matthew Riemer, Golnaz Habibi, Gerald Tesauro, Sami Mourad, Murray Campbell, and Jonathan P. How. 2020. Learning Hierarchical Teaching Policies for Cooperative Agents. In *AAMAS, May 9-13, 2020*. IFAAMAS, Auckland, New Zealand, 620–628.

[16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *JMLR* 17, 1 (2016), 1334–1373.

[17] Mao Li, Yi Wei, and Daniel Kudenko. 2019. Two-level Q-learning: learning from conflict demonstrations. *The Knowledge Engineering Review* 34 (2019).

[18] Michael L. Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *ICML, July 10-13, 1994*. Morgan Kaufmann, New Brunswick, NJ, USA, 157–163.

[19] Qinghua Liu, Tiancheng Yu, Yu Bai, and Chi Jin. 2021. A Sharp Analysis of Model-based Reinforcement Learning with Self-Play. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, Virtual Event, 7001–7010.

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NeurIPS, December 4-9, 2017*. Morgan Kaufmann Publishers, Long Beach, CA, USA, 6379–6390.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[22] John Nash. 1951. Non-cooperative games. *Annals of mathematics* (1951), 286–295.

[23] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. 2019. Learning to Teach in Cooperative Multiagent Reinforcement Learning. In *AAAI, January 27 - February 1, 2019*. AAAI Press, Honolulu, Hawaii, USA, 6128–6136.

[24] Bilal Piot, Matthieu Geist, and Olivier Pietquin. 2014. Boosted Bellman Residual Minimization Handling Expert Demonstrations. In *ECML-PKDD, September 15-19, 2014*, Vol. 8725. Springer, Nancy, France, 549–564.

[25] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely V. Zaruba, and Manfred Huber. 2012. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2012), October 14-17, 2012*. IEEE, Seoul, Korea (South), 1930–1935.

[26] Cinjon Resnick, Wes Eldridge, David Ha, Denny Britz, Jakob Foerster, Julian Togelius, Kyunghyun Cho, and Joan Bruna. 2018. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124* (2018).

[27] Lloyd S Shapley. 1953. Stochastic games. *Proceedings of the national academy of sciences* 39, 10 (1953), 1095–1100.

[28] Felipe Leno Da Silva and Anna Helena Reali Costa. 2019. A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems. *JAIR* 64 (2019), 645–703.

[29] Felipe Leno Da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *AAMAS, May 8-12, 2017*. ACM, Sao Paulo, Brazil, 1100–1108.

[30] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.

[31] Ziang Song, Song Mei, and Yu Bai. 2021. When Can We Learn General-Sum Markov Games with a Large Number of Players Sample-Efficiently? *arXiv preprint arXiv:2110.04184* (2021).

[32] Sriram Ganapathi Subramanian. 2023. Learning from Multiple Independent Advisors in Multi-agent Reinforcement Learning. https://github.com/Sriram94/matlql

[33] Sriram Ganapathi Subramanian, Kate Larson, Matthew Taylor, and Mark Crowley. 2022. Multi-Agent Advisor Q-Learning. *Journal of Artificial Intelligence Research* 74 (2022), 1–74.

[34] Sriram Ganapathi Subramanian, Matthew E. Taylor, Kate Larson, and Mark Crowley. 2023. Learning from Multiple Independent Advisors in Multi-agent Reinforcement Learning. https://doi.org/10.48550/ARXIV.2301.11153

[35] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press, Cambridge.

[36] Csaba Szepesvari and Michael L Littman. 1999. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation* 11, 8 (1999), 2017–2060.

[37] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*. Cambridge University Press, Amherst, MA, USA, 330–337.

[38] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *AAMAS, May 2-6, 2011*. IFAAMAS, Taipei, Taiwan, 617–624.

[39] Lisa Torrey and Matthew E. Taylor. 2013. Teaching on a budget: agents advising agents in reinforcement learning. In *AAMAS, May 6-10, 2013*. IFAAMAS, Saint Paul, MN, USA, 1053–1060.

[40] Hado van Hasselt. 2010. Double Q-learning. In *NeurIPS*. Curran Associates, Inc., Vancouver, British Columbia, Canada, 2613–2621.

[41] Yixi Wang, Wenhuan Lu, Jianye Hao, Jianguo Wei, and Ho-fung Leung. 2018. Efficient Convention Emergence through Decoupled Reinforcement Social Learning with Teacher-Student Mechanism. In *AAMAS, July 10-15, 2018*. IFAAMAS / ACM, Stockholm, Sweden, 795–803.

[42] Zhaodong Wang and Matthew E. Taylor. 2017. Improving Reinforcement Learning with Confidence-Based Demonstrations. In *IJCAI, August 19-25, 2017*. ICJAI, Melbourne, Australia, 3027–3033.

[43] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292.

[44] Kevin Waugh, Brian D. Ziebart, and Drew Bagnell. 2011. Computational Rationalization: The Inverse Equilibrium Problem. In *ICML, June 28 - July 2, 2011*. Omnipress, Bellevue, Washington, USA, 1169–1176.

[45] Tianpei Yang, Weixun Wang, Hongyao Tang, Jianye Hao, Zhaopeng Meng, Hangyu Mao, Dong Li, Wulong Liu, Yingfeng Chen, Yujing Hu, et al. 2021. An Efficient Transfer Learning Framework for Multiagent Reinforcement Learning.

NeurIPS, Virtual Event 34 (2021).

[46] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. In *ICML*, Vol. 80. PMLR, Stockholm Sweden, 5571–5580.

[47] Dayong Ye, Tianqing Zhu, Zishuo Cheng, Wanlei Zhou, and S Yu Philip. 2020. Differential Advising in Multi-Agent Reinforcement Learning. *IEEE Transactions on Cybernetics* (2020).