Logical Team Q-learning: An approach towards factored policies in cooperative MARL

Lucas Cassano
École Polytechnique Fédérale de Lausanne

Ali H. Sayed
École Polytechnique Fédérale de Lausanne

Abstract

We address the challenge of learning factored policies in cooperative MARL scenarios. In particular, we consider the situation in which a team of agents collaborates to optimize a common cost. The goal is to obtain factored policies that determine the individual behavior of each agent so that the resulting joint policy is optimal. The main contribution of this work is the introduction of *Logical Team Q-learning (LTQL)*. LTQL does not rely on assumptions about the environment and hence is generally applicable to any collaborative MARL scenario. We derive LTQL as a stochastic approximation to a dynamic programming method we introduce in this work. We conclude the paper by providing experiments (both in the tabular and *deep* settings) that illustrate the claims.

1 INTRODUCTION

Reinforcement Learning (RL) has had considerable success in many domains. In particular, Q-learning [Watkins and Dayan, 1992] and its deep learning extension DQN [Mnih et al., 2013] have shown great performance in challenging domains such as the Atari Learning Environment [Bellemare et al., 2013]. At the core of DQN lie two important features: the ability to use expressive function approximators (in particular, neural networks (NN)) that allow it to estimate complex Q-functions; and the ability to learn off-policy and use replay buffers [Lin, 1992], which allows DQN to be very sample efficient. Traditional RL focuses on the interaction between one agent and an environment. However, in many cases of interest, a multiplicity of agents will need to interact with a unique environment and with each other. This is the object of study of Multi-agent RL (MARL), which goes back to the early work of [Tan, 1993] and has seen renewed interest of late (for an updated survey see [Zhang et al., 2019]). In this paper we consider the particular case of cooperative MARL in which the agents form a *team* and have a shared unique goal. We are interested in tasks where collaboration is fundamental and a high degree of coordination is necessary to achieve good performance. In particular, we consider two scenarios.

In the first scenario, the global state and all actions are visible to all agents. One example of this situation could be a team of robots that collaborate to move a big and heavy object. It is well known that in this scenario the team can be regarded as one single agent where the aggregate action consists of the joint actions by all agents [Littman, 2001]. The fundamental drawback of this approach is that the joint action space grows exponentially in the number of agents and the problem quickly becomes intractable [Kok and Vlassis, 2004, Guestrin et al., 2002b]. One well-known and popular approach to solve these issues, is to consider each agent as an independent learner (IL) [Tan, 1993]. However, this approach has a number of problems. First, from the point of view of each IL, the environment is non-stationary (due to the changing policies of the other agents), which jeopardizes convergence. And second, replay buffers cannot be used due to the changing nature of the environment and therefore even in cases where this approach might work, the data efficiency of the algorithm is negatively affected. Ideally, it is desirable to derive an algorithm with the following features: i) it learns individual policies (and is therefore scalable), ii) local actions chosen greedily with respect to these individual policies result in an optimal team action, iii) can be combined with NN's, iv) works off-policy and can leverage replay buffers (for data efficiency), v) and enjoys theoretical guarantees to team optimal policies at least in the dynamic programming scenario. Indeed, the main contribution of this work is the introduction of *Logical Team*

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

Q-learning (LTQL), an algorithm that has all these properties. We start in the dynamic programing setting and derive equations that characterize the desired solution. We use these equations to define the *Factored Team Optimality Bellman Operator* and provide a theorem that characterizes the convergence properties of this operator. A stochastic approximation of the dynamic programming setting is used to obtain the tabular and deep versions of our algorithm. For the single agent setting, these steps reduce to: the Bellman optimality equation, the Bellman optimality operator and Q-learning (in its tabular form and DQN).

In the second scenario, we consider the centralized training and decentralized execution paradigm under partial observability. Under this scheme, training is done in a centralized manner and therefore we assume global information to be available during training. During execution, agents only have access to their own observations. Therefore, even though during training global information is available, the learned policies must only rely on local observations. An example of this case would be a soccer team that during training can rely on a centralized server where data is aggregated but has to play games in a fully decentralized manner without the aid of such server.

1.1 Relation to Prior Work

Some of the earliest works on MARL are [Tan, 1993, Claus and Boutilier, 1998]. [Tan, 1993] studied *Independent Q-learning (IQL)* and identified that IQL learners in a MARL setting may fail to converge due to the non-stationarity of the perceived environment. [Claus and Boutilier, 1998] compared the performance of IQL and *joint action learners (JAL)* where all agents learn the Q-values for all the joint actions, and identified the problem of coordination during decentralized execution when multiple optimal policies are available. [Littman, 2001] later provided a proof of convergence for JALs. Recently, [Tampuu et al., 2017] did an experimental study of ILs using DQNS in the Atari game Pong. All these mentioned approaches cannot use experience replay due to the non-stationarity of the preceived environment. Following *Hyper Q-learning* [Tesauro, 2004], [Foerster et al., 2017] addressed this issue to some extent using *fingerprints* as proxys to model other agents' strategies.

[Lauer and Riedmiller, 2000] introduced *Distributed Q-learning (DistQ)*, which in the tabular setting has guaranteed convergence to an optimal policy for deterministic MDPs. However, this algorithm performs very poorly in stochastic scenarios and becomes divergent when combined with function approximation. Later *Hysteretic Q-learning (HystQ)* was introduced in [Matignon et al., 2007] to improve these two limitations. HystQ is based on a heuristic and can be thought of as a generalization of DistQ. These works also consider the scenario where agents cannot perceive the actions of other agents. They are related to LTQL (from this work) in that they can be considered approximations to our algorithm in the scenario where agents do not have information about other agents' actions. Recently [Omidshafiei et al., 2017] introduced *Dec-HDRQNs* for multi-task MARL, which combines HystQ with Recurrent NNs and experience replay (which they recognize is important to achieve high sample efficiency) through the use of *Concurrent Experience Replay Trajectories.*

[Wang and Sandholm, 2003] introduced *OAB*, the first algorithm that converges to an optimal Nash equilibrium with probability one in any team Markov game. OAB considers the team scenario where agents observe the full state and joint actions. The main disadvantage of this algorithm is that it requires estimation of the transition kernel and rewards for the joint action state space and also relies on keeping count of state-action visitation, which makes it impractical for MDPs of even moderate size and cannot be combined with function approximators.

[Guestrin et al., 2002a, Guestrin et al., 2002b, Kok and Vlassis, 2004] introduced the idea of factoring the joint Q-function to handle the scalability issue. These papers have the disadvantage that they require coordination graphs that specify how agents affect each other (the graphs require significant domain knowledge). The main shortcoming of these papers is the factoring model they use, in particular they model the optimal Q-function (which depends on the joint actions) as a sum of K local Q-functions (where K is the number of agents, and each Q-function considers only the action of its corresponding agent). The main issue with this factorization model is that the optimal Q-function cannot always be factored in this way, in fact, the tasks for which this model does not hold are typically the ones that require a high degree of coordination, which happen to be the tasks where one is most interested in applying specific MARL approaches as opposed to ILs. The approach we introduce in this paper also considers learning factored Q-functions. However, the fundamental difference is that the factored relations we estimate always exist and the joint action that results from maximizing these individual Q-functions is optimal. *VDN* [Sunehag et al., 2018] and *Qmix* [Rashid et al., 2018] are two recent *deep* methods that also factorize the optimal Q-function assuming additivity and monotonicity, respectively. This factoring is their main limitation since many MARL problems of interest do not satisfy any of these two assumptions. Indeed, [Son et al., 2019] showed that these methods are unable to solve a simple matrix game. Furthermore, the individual policies cannot be used for prediction, since the individual Q values are not estimates of the return. To improve on the representation limitation due to the factoring assumption, [Son et al., 2019] introduced *QTRAN* which factors the Q-function in a more general manner and therefore allows for a wider applicability. The main issue with QTRAN is that although it can approximate a wider class of Q-functions than VDN and Qmix, the algorithm resorts to other approximations, which degrade its performance in complex environments (see [Rashid et al., 2020]).

Recently, actor-critic strategies have been explored. The algorithm introduced in [Zhang et al., 2018] has the disadvantage that it performs poor credit assignment and as a consequence can easily converge to highly suboptimal strategies (see [Cassano et al., 2019]). [Gupta et al., 2017] introduces policy gradient schemes that also have the credit assignment issue. The algorithm presented by [Foerster et al., 2018] addresses this issue, but does so by learning the team's *joint* q-function and hence this approach does not address the exponential scalability issue. These methods have the added inconvenience that they are on-policy and hence do not enjoy the data efficiency that off-policy methods can achieve.

2 PROBLEM FORMULATION

We consider a situation where multiple agents form a team and interact with an environment and with each other. We model this interaction as a *decentralized partially observable Markov decision process* (Dec-POMDP)[Oliehoek et al., 2016], which is defined by the tuple ($\mathcal{S}$,$\mathcal{K}$,o^k,$\mathcal{A}^k$,$\mathcal{P}$,r), where, $\mathcal{S}$ is a set of global states shared by all agents; $\mathcal{K}$ is the set of K of agents; $o^k : \mathcal{S} \to \mathcal{O}^k$ is the observation function for agent k, whose output lies in some set of observations $\mathcal{O}^k$; $\mathcal{A}^k$ is the set of actions available to agent k; $\mathcal{P}(s'|s, a^1, \cdots, a^K)$ specifies the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken joint actions $a^k \in \mathcal{A}^k$; and $r : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^K \times \mathcal{S} \to \mathbb{R}$ is a global reward function. Specifically, $r(s, a^1, \cdots, a^K, s')$ is the reward when the team transitions to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken actions $a^1, \cdots, a^K$. The reward $r(s, a^1, \cdots, a^K, s')$ can be a random variable following some distribution $f_{s,a^1,\cdots,a^K,s'}(r)$. We clarify that from now on we will refer to the collection of all individual actions as the team's action, denoted as $\bar{a}$. Furthermore we will use a^{-k} to refer to the actions of all agents except for action a^k. Therefore we can write $\mathcal{P}(s'|s, a^1, \cdots, a^K) = \mathcal{P}(s'|s, a^k, a^{-k}) = \mathcal{P}(s'|s, \bar{a})$.

The goal of the team is to maximize the team's return:

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi,\mathcal{P},d,f} \left[\boldsymbol{r}(\boldsymbol{s_t}, \boldsymbol{\bar{a}_t}, \boldsymbol{s_{t+1}}) \right] \tag{1}$$

where s_t and $\bar{a}_t$ are the state and actions at time t, respectively, $\pi(\bar{a}|s)$ is the team's policy, d is the distribution of initial states, and $\gamma \in [0, 1)$ is the discount factor. We clarify that we use bold font to denote random variables and the notation $\mathbb{E}_\ell$ makes explicit that the expectation is taken with respect to distribution ℓ. From now on, we will only make the distributions explicit in cases where doing so makes the equations more clear. Accordingly, the team's optimal state-action value function ($q^\dagger$) and optimal policy ($\pi^\dagger$) are given by [Sutton and Barto, 1998]:

$$\pi^\dagger(\bar{a}|s) = \underset{\pi(\bar{a}|s)}{\arg\max}\, \mathbb{E}_{\pi,\mathcal{P}} \big[r(s, \boldsymbol{\bar{a}}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'}, \bar{a}') \big] \tag{2a}$$

$$q^\dagger(s, \bar{a}) = \mathbb{E}_{\mathcal{P}} \big[r(s, \bar{a}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'}, \bar{a}') \big] \tag{2b}$$

where $r(s, \bar{a}) = \mathbb{E}_{\mathcal{P},f} \boldsymbol{r}(s, \bar{a}, \boldsymbol{s'})$. As already mentioned, a team problem of this form can be addressed with any single-agent algorithm. The fundamental inconvenience with this approach is that the joint action space scales exponentially with the number of agents, more specifically $|\bar{\mathcal{A}}| = \prod_{k=1}^{K} |\mathcal{A}^k|$ (where $\bar{\mathcal{A}}$ is the joint action space). Another problem with this approach is that the learned Q-function cannot be executed in a decentralized manner using the agents' observations. For these reasons, in the next sections we concern ourselves with learning factored quantities.

3 DYNAMIC PROGRAMMING

Similarly to the way that relation (2b) is used to derive Q-learning in the single agent setting, the goal of this section is to derive relations in the dynamic programming setting from which we can derive a cooperative MARL algorithm. The following two propositions take the first steps in this direction.

Proposition 1. *For each deterministic team optimal policy, there exist K factored functions $q^{k,\star} : \mathcal{S} \times \mathcal{A}^k \to \mathbb{R}$ such that:*

$$\max_{\bar{a}} q^\dagger(s, \bar{a}) = \max_{a^k} q^{k,\star}(s, a^k), \qquad \forall 1 \le k \le K \tag{3a}$$

$$\max_{\bar{a}} q^\dagger(s, \bar{a}) = q^\dagger \Big(s, \arg\max_{a^1} q^{1,\star}(s, a^1), \cdots , \arg\max_{a^K} q^{K,\star}(s, a^K) \Big) \tag{3b}$$

$$q^{k,\star}(s, a^k) = \mathcal{B}_E q^{k,\star}(s, a^k) \tag{3c}$$

$$\mathcal{B}_E q^k(s, a^k) = r(s, a^k, a^{-k}) + \gamma \mathbb{E} \max_{a'} q^k(\boldsymbol{s'}, a') \big|_{a^n = \underset{a^n}{\arg\max}\, q^n(s, a^n) \ \forall n \neq k} \tag{3d}$$

where operator $\mathcal{B}_E$ is defined such that if there are multiple arguments that maximize $\arg\max_{a^n} q^n(s, a^n)$ *the actions that jointly maximize* (3d) *are chosen.*

Proof. Assume that we have some deterministic team optimal policy $\pi^\dagger(\bar{a}|s)$. We define $q^{k,\star}(s, a^k)$ as follows:

$$q^{k,\star}(s, a^k) = q^\dagger(s, a^k, a^{-k})|_{\arg\max_{a^{-k}} \pi^\dagger(a^k, a^{-k}|s)} \quad (4)$$

Note that by construction, $q^{k,\star}(s, a^k)$ satisfies relations (3a) and (3b) and also:

$$\arg\max_{a^k} q^{k,\star}(s, a^k) = a^k \sim \pi^\dagger(a^k, a^{-k}|s) \ \ \forall k \quad (5)$$

Relation (3c) is obtained by combining relations (2b), (4) and (5). ■

A simple interpretation of equation (3c) is that $q^{k,\star}(s, a^k)$ is the expected return starting from state s when agent k takes action a^k while the rest of the team acts in an optimal manner.

Proposition 2. *Each deterministic team optimal policy that can be factored into K deterministic policies $\pi^{k,\star}(a^k|s)$. Such factored deterministic policies can be obtained as follows:*

$$\pi^{k,\star}(a^k|s) = \mathbb{I}\big(a^k = \arg\max_{a^k} q^{k,\star}(s, a^k)\big) \quad (6)$$

where $\mathbb{I}$ is the indicator function.

Proof. The proof follows from equations (3a)-(3b). ■

Propositions 1 and 2 are useful because they show that if the agents learn factored functions that satisfy (3) and act greedily with respect to their corresponding $q^{k,\star}$, then the resulting team policy is guaranteed to be optimal and hence they are not subject to the coordination problem identified in [Lauer and Riedmiller, 2000][1] (we show this in section 5.1). Therefore, an algorithm that learns $q^{k,\star}$ would satisfy the first two of the five desired properties that were enumerated in the introduction. As a sanity check, note that for the case where there is only one agent, equation (3c) simplifies to the Bellman optimality equation. Although in the single agent case the Bellman optimality operator can be used to obtain $q^\dagger$ (by repeated application of the operator), we cannot do the same with $\mathcal{B}_E$. The fundamental reason for this is stated in proposition 3.

[1]This problem arises in situations in which the environment has multiple deterministic team optimal policies and the agents learn factored functions of the form $\max_{a^{-k} q^\dagger(s, a^k, a^{-k})}$ (we remark that these are not the same as $q^{k,\star}(s, a^k)$).

Proposition 3. ***Sub-optimal Nash fixed points:*** *There may exist K functions q^k such that* (3c) *is satisfied but* (3b) *is not.*

Proof. See Appendix 6.1.■

Note that proposition 3 implies that relation (3c) is not sufficient to derive a learning algorithm capable of obtaining a team optimal policy because it can converge to sub-optimal team strategies instead. To avoid this inconvenience, it is necessary to find another relation that is only satisfied by $q^\star$. We can obtain one such relation combining (3a) and (3c):

$$\max_{a^k} q^{k,\star}(s, a^k) = \max_{\bar{a}} \big[r(s, \bar{a}) + \gamma \mathbb{E} \max_{a', k} q^{k,\star}(\boldsymbol{s'}, a'^{,k})\big] \quad (7)$$

The sub-optimal Nash fixed points mentioned in proposition 3 do not satisfy relation (7) since by definition the right hand side is equal to $\max_{\bar{a}} q^\dagger(s, \bar{a})$. Intuitively, equation (7) is not satisfied by these sub-optimal strategies because the $\max_{\bar{a}}$ considers all possible team actions (while Nash equilibria only consider unilateral deviations).

Definition 1. *Combining equations* (3c) *and* (7)*, we define the Factored Team Optimality Bellman operator $\mathcal{B}_p$ as follows:*

$$\mathcal{B}_p q^k(s, a^k) = \begin{cases} \mathcal{B}_E q^k(s, a^k) & \text{with probability } p > 0 \\ \mathcal{B}_I q^k(s, a^k) & \text{else} \end{cases} \quad (8)$$

$$\mathcal{B}_I q^k(s, a^k) = \max\big\{q^k(s, a^k), \max_{a^{-k}} \big(r(s, a^k, a^{-k}) + \gamma \mathbb{E} \max_{a'} q^k(\boldsymbol{s'}, a')\big)\big\} \quad (9)$$

LTQL is based on operator $\mathcal{B}_p$, the reason we use relations (3c) and (7) to define this operator instead of just (7), is that using only relation (7) we would derive DistQ that has the shortcomings we discussed in section 1.1. A simple interpretation of operator $\mathcal{B}_p$ is the following. Consider a basketball game, in which player α has the ball and passes the ball to teammate β. If β gets distracted, misses the ball and the opposing team ends up scoring, should α learn from this experience and modify its policy to not pass the ball? The answer is no, since the poor outcome was player β's fault. In plain English, from the point of view of some player k, what the operator $\mathcal{B}_E$ means is *"I will only learn from experiences in which my teammates acted according to what I think is the optimal team strategy"*. It is easy to see why this kind of stubborn rationale cannot escape Nash equilibria (i.e., agents do not learn when the team deviates from its current best strategy, which obviously is a necessary condition to learn better strategies). The interpretation of the full operator $\mathcal{B}_p$ is *"I will learn from experiences in which: a) my* *teammates acted according to what I think is the optimal team strategy; or b) my teammates deviated from what I believe is the optimal strategy and the outcome of such deviation was better than I expected if they had acted according to what I thought was optimal"*, which arguably is what a logical player would do (this is the origin of the algorithm's name). We now proceed to describe the convergence properties of operator $\mathcal{B}_p$.

Lemma 1. *For any $\delta_1 > 0$, after N applications of operator $\mathcal{B}_p$ to any set of K $q^k(s,a^k)$ functions it holds:*

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}^U_{\delta_1}\big) \geq 1 - \sum_{n=0}^{n_o} \binom{N}{n} p^n (1-p)^{N-n} \tag{10}$$

$$\mathcal{C}^U_{\delta_1} = \big\{q^k | q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) + \delta_1 \quad \forall(k,s,a^k)\in(\mathcal{K},\mathcal{S},\mathcal{A}^k)\big\} \tag{11}$$

$$n_o = \left\lfloor \log_\gamma \left(\frac{\delta_1}{q_U - \min_s \max_{\bar{a}} q^\dagger(s,\bar{a})} \right) \right\rfloor \tag{12}$$

$$q_U = \max\{r_{\max}(1-\gamma)^{-1}, \max_{k,s,a^k} q^k(s,a^k)\} \tag{13}$$

where $r_{\max} = \max_{s,\bar{a}} r(s,\bar{a})$. For the special case where $N > n_o/p$ we can lower bound equation (10) as follows:

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}^U_{\delta_1}\big) \geq 1 - e^{-2N\left(p-\frac{n_o}{N}\right)^2} \tag{14}$$

Proof. See Appendix 6.2.■

Lemma 2. *After $N \geq L$ applications of operator $\mathcal{B}_p$ to any set of K $q^k(s,a^k) \in \mathcal{C}^U_0$ functions it holds:*

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\delta_2}\big) \geq 1 - \beta_{N,L} + (1-p)^L \beta_{N-L,L} \tag{15}$$

$$\beta_{N,L} = \sum_{j=0}^{\lfloor N/(L+1) \rfloor} (-1)^j \binom{N-jL}{j} \big(p(1-p)^L\big)^j \tag{16}$$

$$L = \left\lceil \log_\gamma \left(\frac{\delta_2}{\max_s \big| \max_{a^k} q^k(s,a^k) - \max_{\bar{a}} q^\dagger(s,\bar{a}) \big|} \right) \right\rceil \tag{17}$$

$$\mathcal{C}_{\delta_2} = \big\{q^k | q^{k,\star}(s,a^k) - \delta_2 \leq q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) + \delta_2 \forall(k,s,a^k)\in(\mathcal{K},\mathcal{S},\mathcal{A}^k)\big\} \tag{18}$$

for any $\delta_2 > 0$. If $p > 0.5$, probability (15) can be bounded by:

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\delta_2}\big) \geq 1 - \frac{1-(1-p)\xi_1}{p\xi_1(1+L-L\xi_1)} \xi_1^{-N} - \frac{L}{p}(1-p)^{N+2} \tag{19}$$

where $1 < \xi_1 < 1 + L^{-1}$.

Proof. See Appendix 6.3.■

Lemma 1 indicates that if the initial functions $q^k(s,a^k)$ have values that are larger than $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$, after sufficient applications of operator $\mathcal{B}_p$ all overestimations will be reduced such that $q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) + \delta_1$ with high probability. Lemma 2 show that if the operator $\mathcal{B}_p$ is applied sufficient times to functions that do not overestimate $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$, then the obtained function lie in a small neighborhood of the desired solution with high probability. These results give rise to the following important theorem.

Theorem 1. *Repeated application of the operator $\mathcal{B}_p$ to any initial set of K q^k-functions followed by an application of operator $\mathcal{B}_E$ converge to the δ-neighborhood ($\delta > 0$) of some set $q^{k,\star}$ with high probability. For the particular case, where $p > 0/5$ and $N > L + n_o/p$ it holds:*

$$\mathbb{P}\big(|\mathcal{B}_E \mathcal{B}_p^N q^k(s,a^k) - q^{k,\star}(s,a^k)| < \delta\big) \geq 1 - \mathcal{O}(\theta^N) \tag{20}$$

for any $\delta > 0$, where $0 \leq \theta < 1$ is a constant that depends on δ, γ, p, $r(s,\bar{a})$, $\mathcal{P}$ and the initial functions $q^k(s,a^k)$.

Proof. Combining the results from lemmas 1 and 2 and setting $\delta = \delta_1 = \delta_2$ we get that after $N_1 + N_2 > L + n_o/p > 0$ applications of operator $\mathcal{B}_p$ to any set of K $q^k(s,a^k)$ functions it holds:

$$\begin{aligned}\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_\delta\big) &\geq \max_{\substack{N_1 > \frac{n_o}{p} \\ N_2 \geq L}} \left(1 - e^{-2N_1\left(p-\frac{n_o}{N_1}\right)^2}\right) \\ &\cdot \left(1 - \frac{1-(1-p)\xi_1}{p\xi_1(1+L-L\xi_1)}\xi_1^{-N_2} - \frac{L}{p}(1-p)^{N_2+2}\right) \\ &= 1 - \mathcal{O}(\theta^N)\end{aligned} \tag{21}$$

where $0 \leq \theta < 1$. Now we proceed to analyze $\mathcal{B}_E \mathcal{B}_p^N q^k(s,a^k)$. If $q^k(s,a^k) \in \mathcal{C}_\delta$ and δ satisfies:

$$\delta < \frac{1}{2} \min_s \big(\max_{\bar{a}} q^\dagger(s,\bar{a}) - \max_{\bar{a} \neq \arg\max_{\bar{a}} q^\dagger(s,\bar{a})} q^\dagger(s,\bar{a}) \big) \tag{22}$$

we get:

$$\mathcal{B}_E q^k(s,a^k) = \big(r(s,a^k,a^{-k}) + \gamma \mathbb{E} \max_{a'} q^k(\boldsymbol{s'},a')\big)\big|_{a^n = \arg\max_{a^n} q^n(s,a^n)\ \forall n \neq k} \tag{23}$$

Using the fact that $q^k(s,a^k) \in \mathcal{C}_\delta$ it follows:

$$\max_{a^k} q^{k,\star}(s,\bar{a}) - \delta \leq \max_{a^k} q^k(s,a^k) \tag{24}$$

$$q^k(s,a^{k,\bullet}) \leq \max_{a^{-k}} q^\dagger(s,a^{k,\bullet},a^{-k}) + \delta \overset{(a)}{<} \max_{\bar{a}} q^\dagger(s,\bar{a}) - \delta \overset{(b)}{=} \max_{a^k} q^{k,\star}(s,\bar{a}) - \delta_2 \tag{25}$$

$$a^{k,\bullet} = \mathop{\arg\max}_{a^k \neq \arg\max_{a^k} q^k(s,a^k)} q^k(s,a^k) \tag{26}$$

where in (a) we used condition (22) and in (b) we used equation (3a). Combining equations (23) through (26) we get:

$$\begin{aligned}
\mathcal{B}_E q^k(s,a^k) &= \big(r(s,a^k,a^{-k}) \\
&\quad + \gamma\mathbb{E}\max_{a'} q^k(\boldsymbol{s'},a')\big)\big|_{a^n=\arg\max_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \\
&= \big(r(s,a^k,a^{-k}) + \gamma\mathbb{E}\,\big(\max_{a'} q^{k,\star}(\boldsymbol{s'},a') + \max_{a'} q^k(\boldsymbol{s'},a') \\
&\quad - \max_{a'} q^{k,\star}(\boldsymbol{s'},a')\big)\big)\big|_{a^n=\arg\max_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \\
&= q^{k,\star}(s,a^k) + \gamma\mathbb{E}\,\big(\max_{a'} q^k(\boldsymbol{s'},a') \\
&\quad - \max_{a'} q^{k,\star}(\boldsymbol{s'},a')\big)\big|_{a^n=\arg\max_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \qquad (27)
\end{aligned}$$

Combining equation (27) with the fact that $q^k(s,a^k) \in \mathcal{C}_\delta$ we get:

$$|\mathcal{B}_E q^k(s,a^k) - q^{k,\star}(s,a^k)| \leq \gamma\delta \qquad (28)$$

Combining (28) and (21) completes the proof. ■

Relation (28) shows why we include an application of $\mathcal{B}_E$ at the end in equation (20). The reason is that if we do not, the q-value for *suboptimal actions* oscillates between $q^{k,\star}(s,a^k)$ and $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$, we illustrate this effect in appendix 6.5. We reiterate that $q^{k,\star}(s,a^k)$ and $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ are only equal when optimal actions are chosen (equation (3a)). $q^{k,\star}(s,a^k)$ is the expected return if agent k chooses action a^k and the rest of the team follows an optimal policy, while $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ is the best return that can be achieved if agent k chooses action a^k.

4 REINFORCEMENT LEARNING

In this section we present LTQL (see algorithm 1), which we obtain as a stochastic approximation to the procedure described in theorem 1. Note that the algorithm utilizes two q estimates for each agent k, a biased one parameterized by θ^k (which we denote q_{θ^k}) and an unbiased one parameterized by ω^k (which we denote q_{ω^k}). We clarify that in the listing of LTQL += is the *accumulate and add* operator and that we used a constant step-size, however this can be replaced with decaying step-sizes or other schemes such as *AdaGrad* [Duchi et al., 2011] or *Adam* [Kingma and Ba, 2014]. Note that the target of the unbiased network is used to calculate the target values for both functions; this prevents the bias in the estimates q_{θ^k} (which arises due to the c_2 condition)[2] from propagating through bootstrapping. The target parameters of the biased estimates (θ^k_T) are used solely to evaluate condition c_1. We have found that this stabilizes the training of the networks, as opposed to just using θ^k. Hyperparameter α weights samples that satisfy condition c_2 differently from those who satisfy c_1. As we remarked in the introduction, LTQL reduces to *DQN* for the case where there is a unique agent. In appendix 6.4 we include the tabular version of the algorithm along with a brief discussion.

Algorithm 1 *Logical Team Q-Learning*

```
Initialize: an empty replay buffer $\mathcal{R}$, parameters $\theta^k$ and $\omega^k$ and their corresponding targets $\theta^k_T$ and $\omega^k_T$ for all agents $k \in \mathcal{K}$.
for iterations $e = 0, \dots, E$ do
    Sample $T$ transitions $(s, \bar{a}, r, s')$ by following some behavior policy which guarantees all joint actions are sampled with non-zero probability and store them in $\mathcal{R}$.
    for iterations $i = 0, \dots, I$ do
        Sample a mini-batch of $B$ transitions $(s, \bar{a}, r, s')$ from $\mathcal{R}$.
        Set $\Delta_{\theta^k} = 0$ and $\Delta_{\omega^k} = 0$ for all agents $k$.
        for each transition of the mini-batch $b = 1, \cdots, B$ and each agent $k = 1, \cdots, K$ do
            if $a^n = \arg\max_{a^n} q_{\theta^k_T}(s, a^n)\ \forall n \neq k$ then
                $\Delta_{\theta^k} += \nabla_{\theta^k}\big(r + \max_a q_{\omega^k_T}(s', a) - q_{\theta^k}(s, a^k)\big)$
                $\Delta_{\omega^k} += \nabla_{\omega^k}\big(r + \max_a q_{\omega^k_T}(s', a) - q_{\omega^k}(s, a^k)\big)$
            else if $\big(r + \max_a q_{\theta^k_T}(s', a) > q_{\theta^k}(s, a^k)\big)$ then
                $\Delta_{\theta^k} += \alpha\nabla_{\theta^k}\big(r + \max_a q_{\omega^k_T}(s', a) - q_{\theta^k}(s, a^k)\big)$
            end if
        end for
        $\theta^k += \mu\Delta_{\theta^k}$        $\omega^k += \mu\Delta_{\omega^k}$
    end for
    Update targets $\theta^k_T = \theta^k$ and $\omega^k_T = \omega^k$.
end for
```

Note that LTQL works off-policy and there is no necessity of synchronization for exploration. Therefore in applications where agents have access to the global state and can perceive the actions of all other agents (so that they can evaluate c_1), it can be implemented in a fully decentralized manner. Interestingly, if condition c_1 was omitted (to eliminate the requirement that agents have access to all this information), the resulting algorithm is exactly DistQ [Lauer and Riedmiller, 2000]. However, as the proof of lemma 1 indicates, the resulting algorithm would only converge in situations where it could be guaranteed that during learning, overestimation of the q values is not possible (i.e., the tabular setting applied to deterministic environments; this remark was already made in [Lauer and Riedmiller, 2000]). In the case where

[2]We refer to the condition of the first **if** statement (i.e., $a^n = \arg\max_{a^n} q_{\theta^k_T}(s, a^n)\ \forall n \neq k$) as c_1, and the condition corresponding to the second **if** statement as c_2.

this condition could not be guaranteed (i.e., when using function approximation and/or stochastic environments), some mechanism to decrease overestimated q values would be necessary, as this is the main task of the updates due to c_1. One possible way to do this would be to use all transitions to update the q estimates but use a smaller step-size for the ones that do not satisfy c_2. Notice that the resulting algorithm would be exactly HystQ [Matignon et al., 2007].

Notice that the listing of LTQL relies on global states s as opposed to local agent observations. Therefore in its current form the algorithm is only applicable to the first scenario described in the introduction, in which agents have access to the global state both during training and execution. For the second scenario, in which during execution agents rely on their local observation we make the usual approximation $q^k(\mathcal{H}^k, a^k) \approx q^k(s, a^k)$ where $\mathcal{H}^k$ is the action-observation history of agent k. Hence, to adapt algorithm 1 to this second scenario all that is necessary is to replace $q_{\theta^k}(s, a^k)$ for $q_{\theta^k}(\mathcal{H}^k, a^k)$ (and similarly for ω^k, θ_T^k and ω_T^k), and the observation histories need to be stored in the replay buffer as well. In practice recurrent architectures (like the *Long Short Term Memory* (LSTM) [Hochreiter and Schmidhuber, 1997]) can be used to parameterize $q_{\theta^k}(\mathcal{H}^k, a^k)$ as is done in *Deep recurrent Q-Network (DRQN)* [Hausknecht and Stone, 2015].

5 EXPERIMENTS

5.1 Matrix Game

The first experiment is a simple matrix game (figure 1a shows the payoff structure) with multiple team optimum policies to evaluate the resilience of the algorithm to the coordination issue mentioned in section 3. In this case, we implemented IQL, DistQ, LTQL, Qmix and Qtran (we do not include a curve labeled HystQ because in deterministic environments with tabular representation the optimum value for the small step-size is 0, in which case it becomes exactly equivalent to DistQ). All algorithms are implemented in tabular form.[3] In all cases we used uniform exploratory policies ($\epsilon = 1$) and we did not use replay buffers. IQL fails at this task and oscillates due to the perceived time-varying environment (see figure 2 in appendix 6.5). DistQ converges to (29)-(30), which clearly shows why DistQ has a coordination issue. However, LTQL converges to either of the two possible solutions (31) or (32) (depending on the seed) for which individual greedy policies result in team optimal policies. Qmix fails at identifying an optimum team policy and the resulting joint q-function obtained using the mixing network also fails at predicting the rewards. Qmix converges to (33). The joint q-function is shown in appendix 6.5. And Qtran also oscillates due to the fact that in this matrix game there are two jointly optimal actions. In appendix 6.5 we include the learning curves of all algorithms for the readers reference along with a brief discussion.

$$q^1(a^1) = \max_{a^2} q^\dagger(a^1, a^2) = [2, 2] \tag{29}$$

$$q^2(a^2) = \max_{a^1} q^\dagger(a^1, a^2) = [0, 2, 2] \tag{30}$$

$$q^{1,\star}(a^1) = [2, 1] \quad q^{2,\star}(a^2) = [0, 2, 0] \tag{31}$$

$$q^{1,\star}(a^1) = [0, 2] \quad q^{2,\star}(a^2) = [0, 1, 2] \tag{32}$$

$$q^1(a^1) = [-0.7, 1.1] \quad q^2(a^2) = [-3.5, 1.8, 0.6] \tag{33}$$

5.2 Stochastic Finite Environment

In this experiment we use a tabular representation in an environment that is stochastic and episodic. The environment is a linear grid with 4 positions and 2 agents. At the beginning of the episode, the agents are initialized in the far right. Agent 1 cannot move and has 2 actions (*push button* or *not push*), while agent 2 has 3 actions (*stay*, *move left* or *move right*). If agent 2 is located in the far left and chooses to *stay* while agent 2 chooses *push*, the team receives a +10 reward. If the button is pushed while agent 2 is moving left the team receives a −30 reward. This negative reward is also obtained if agent 2 stays still in the leftmost position and agent 1 does not push the button. All rewards are subject to additive Gaussian noise with mean 0 and standard deviation equal to 1. Furthermore if agent 2 tries to move beyond an edge (left or right), it stays in place and the team receives a Gaussian reward with 0 mean and standard deviation equal to 3. The episode finishes after 5 timesteps or if the team gets the +10 reward (whichever happens first). We ran the simulation 20 times with different seeds. Figure 1b shows the average test return[4] (without the added noise) of IQL, LTQL, HystQ, DistQ, Qmix and Qtran. As can be seen, LTQL and Qtran are the only algorithms capable of learning optimal team policies. In appendix 6.6 we specify the hyperparameters and include the learning curves of the Q-functions along with a discussion on the performance of each algorithm.

5.3 Cowboy Bull Game

In this experiment we use a more complex environment, a challenging predator-prey type game with partial observability, in which 4 cowboys try to catch a

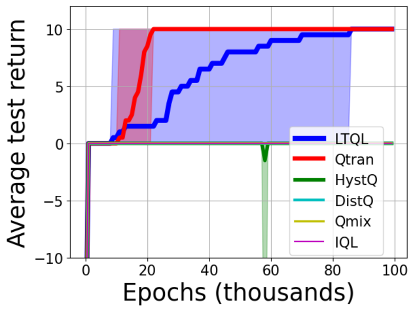
[3]In the case of Qmix we used tabular representations for the individual q functions and a NN for the mixing network.

[4]The average test return is the return following a greedy policy averaged over 50 games.

Agent 2

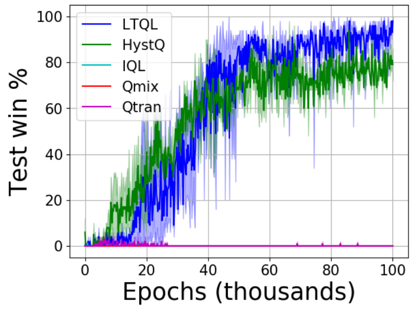
Agent 1		a_1	a_2	a_3
	b_1	0	2	0
	b_2	0	1	2

(a) Experiment 1

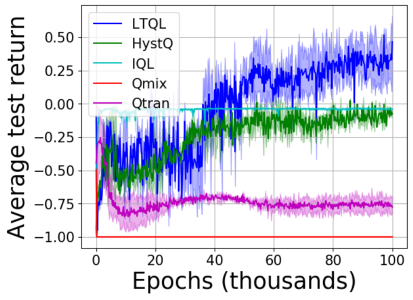


(b) Experiment 2

(c) Experiment 3



(d) Experiment 3



(e) Experiment 3

Figure 1: The dark curves show the mean over all seeds while the shaded region show the minimum and maximum values. We clarify that in figure 1b the curves corresponding to HystQ and DistQ, which are partially occluded, converge to 0.

bull (see figure 1c). The position of all players is a continuous variable (and hence the state space is continuous). The space is unbounded and the bull can move 20% faster than the cowboys. The bull follows a fixed stochastic policy, which is handcrafted to mimic natural behavior and evade capture. Due to the unbounded space and the fact that the bull moves faster than the cowboys, it cannot be captured unless all agents develop a coordinated strategy (the bull can only be caught if the agents first surround it and then evenly close in). The task is episodic and ends after 75 timesteps or when the bull is caught. Each agent has 5 actions (the four moves plus *stay*). When the bull is caught a $+1$ reward is obtained and the team also receives a small penalty $(-1/(4 \times 75))$ for every agent that moves. Note that due to the reward structure there is a very easily attainable Nash equilibrium, which is for every agent to stay still (since in this way they do not incur in the penalties associated with movement). Figure 1d shows the test win percentage[5] and figure 1e shows the average test return for IQL, LTQL, HystQ, Qmix and Qtran. The best performing algorithm is LTQL. HystQ learns a policy that catches the bull 80% of the times, although it fails at obtaining returns higher than zero. IQL fails because the agents quickly converge to the policy of never moving (to avoid incurring in the negative rewards associated with movement). We believe that the poor performance of Qmix in this task is a consequence of its limited representation capacity due to its monotonic factoring model. Qtran fails in this complex scenario, which is in agreement with results reported in [Rashid et al., 2020] where Qtran also shows poor performance in the *Starcraft multi-agent challenge* (SMAC) [Samvelyan et al., 2019]. In the appendix we provide all hyperparameters and implementation details, we detail the bull's policy and the observation function. All code[6], a pre-trained model and a video of the policy learned by LTQL are included as supplementary material.

[5]Percentage of games, out of 50, in which the team succeeds to catch the bull following a greedy policy.

[6]Code is also available at https://github.com/lcassano/Logical-Team-Q-Learning-paper.

6 CONCLUDING REMARKS

In this article we have introduced *Logical Team Q-Learning*, which has the 5 desirable properties mentioned in the introduction. LTQL does not impose constraints on the learned individual Q-functions and hence it can solve environments where state of the art algorithms like Qmix and Qtran fail. The algorithm fits in the centralized training and decentralized execution paradigm. It can also be implemented in a fully distributed manner in situations where all agents have access to each others' observations and actions.

References

[Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[Cassano et al., 2019] Cassano, L., Alghunaim, S. A., and Sayed, A. H. (2019). Team policy learning for multi-agent reinforcement learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3062–3066.

[Claus and Boutilier, 1998] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, pages 2121–2159.

[Foerster et al., 2017] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings International Conference on Machine Learning*, pages 1146–1155.

[Foerster et al., 2018] Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*.

[Guestrin et al., 2002a] Guestrin, C., Koller, D., and Parr, R. (2002a). Multiagent planning with factored MDPs. In *Advances in neural information processing systems*, pages 1523–1530.

[Guestrin et al., 2002b] Guestrin, C., Lagoudakis, M., and Parr, R. (2002b). Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234.

[Gupta et al., 2017] Gupta, J. K., Egorov, M., and Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83, Sao Paulo, Brazil.

[Hausknecht and Stone, 2015] Hausknecht, M. and Stone, P. (2015). Deep recurrent Q-learning for partially observable mdps. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, Virginia, USA.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

[Kok and Vlassis, 2004] Kok, J. R. and Vlassis, N. (2004). Sparse cooperative Q-learning. In *Proceedings International Conference on Machine Learning*, page 61.

[Lauer and Riedmiller, 2000] Lauer, M. and Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. International Conference on Machine Learning* (ICML), pages 535–542, Palo Alto, USA.

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321.

[Littman, 2001] Littman, M. L. (2001). Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66.

[Matignon et al., 2007] Matignon, L., Laurent, G., and Le Fort-Piat, N. (2007). Hysteretic Q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69, San Diego, USA.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv:1312.5602*.

[Oliehoek et al., 2016] Oliehoek, F. A., Amato, C., et al. (2016). *A Concise Introduction to Decentralized POMDPs*, volume 1. Springer.

[Omidshafiei et al., 2017] Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings International Conference on Machine Learning*, pages 2681–2690.

[Rashid et al., 2020] Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

[Rashid et al., 2018] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 4295–4304.

[Samvelyan et al., 2019] Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. (2019). The StarCraft Multi-Agent Challenge. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188.

[Son et al., 2019] Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. (2019). Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896.

[Sunehag et al., 2018] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings International Conference on Autonomous Agents and Multiagent Systems*, pages 2085–2087.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

[Tampuu et al., 2017] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4).

[Tan, 1993] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings International Conference on Machine Learning*, pages 330–337.

[Tesauro, 2004] Tesauro, G. (2004). Extending Q-learning to general adaptive multi-agent systems. In *Proc. Advances in Neural Information Processing Systems*, pages 871–878, Vancouver, Canada.

[Uspensky, 1937] Uspensky, J. V. (1937). *Introduction to mathematical probability.* McGraw-Hill, New York.

[Wang and Sandholm, 2003] Wang, X. and Sandholm, T. (2003). Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Proc. Advances in Neural Information Processing Systems*, pages 1603–1610, Vancouver, Canada.

[Watkins and Dayan, 1992] Watkins, C. and Dayan, P. (1992). *Q*-learning. *Machine Learning*, 8(3-4):279–292.

[Zhang et al., 2019] Zhang, K., Yang, Z., and Başar, T. (2019). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv:1911.10635*.

[Zhang et al., 2018] Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. (2018). Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the International Conference on Machine Learning*, pages 5872–5881.

Appendix

6.1 Proof of proposition 3

Consider the matrix game with two agents, each of which has two actions ($\mathcal{A} = \{\alpha; \beta\}$) and the following reward structure:

Reward structure

		Agent 2 α	Agent 2 β
Agent 1	α	0	-1
Agent 1	β	-1	1

For this case $q^\dagger$, $\pi^\dagger$, $q^{1,\star}$, $q^{2,\star}$, $\pi^{1,\star}$ and $\pi^{2,\star}$ are given by:

$q^\dagger(a^1, a^2)$

	α	β
α	0	-1
β	-1	1

$\pi^\dagger(a^1, a^2)$

	α	β
α	0	0
β	0	1

$q^{1,\star}(a)$

α	β
-1	1

$q^{2,\star}(a)$

α	β
-1	1

$\pi^{1,\star}(a)$

α	β
0	1

$\pi^{2,\star}(a)$

α	β
0	1

Notice that as expected, $q^\star$ satisfies (3c). However, note that (3c) is also satisfied by the following q function which is different from $q^\star$.

$$q(a = \alpha) = 0, \qquad q(a = \beta) = -1 \tag{34}$$

Notice further that the team policy obtained by choosing actions in a greedy fashion with respect to q constitutes a sub-optimal Nash equilibrium.

6.2 Proof of Lemma 1

We start defining:

$$q_U = \max\{r_{\max}(1-\gamma)^{-1}, \max_{k,s,a^k} q^k(s,a^k)\} \tag{35}$$

$$q_U^k(s,a^k) = q_U \tag{36}$$

where $r_{\max} = \max_{s,\bar{a}} r(s,\bar{a})$. We recall that to simplify notation we defined $\mathbb{E}_{\mathcal{P},f}\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) = r(s,a^k,a^{-k})$. The first part of the proof consists in upper bounding any sequence of the form $\mathcal{B}_I^{K_\ell}\mathcal{B}_E^{K_{n-1}}\cdots\mathcal{B}_E^{K_1}\mathcal{B}_I^{K_0}q_U^k(s,a^k)$, where $K_\ell \in \mathbb{N}$ for all ℓ. Applying operator $\mathcal{B}_I$ to $q_U^k(s,a^k)$ we get:

$$\mathcal{B}_I q_U^k(s,a^k) = \max\left\{q_U, \max_{a^{-k}} r(s,a^k,a^{-k}) + \gamma q_U\right\} = q_U \tag{37}$$

Therefore, $\mathcal{B}_I^{K_0}q_U^k(s,a^k) = q_U^k(s,a^k)$ for any $K_0 \in \mathbb{N}$. Applying operator $\mathcal{B}_E$ we get:

$$\mathcal{B}_E q_U^k(s,a^k) = \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\big(r(s,a^k,a^{-k}) + \gamma q_U\big)\big|_{a^n = \arg\max_{a^n} q_U \quad \forall n\neq k} \leq \max_{a^{-k}} r(s,a^k,a^{-k}) + \gamma q_U \tag{38}$$

$$\mathcal{B}_E^2 q_U^k(s,a^k) \leq \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\big(r(s,a^k,a^{-k}) + \gamma \max_{\bar{a}'} r(\boldsymbol{s'},\bar{a}') + \gamma^2 q_U\big)\big|_{a^n = \arg\max_{a^n} \mathcal{B}_E q_U^n(s,a^n) \quad \forall n\neq k}$$
$$\leq \max_{a^{-k}} \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\left(r(s,a^k,a^{-k}) + \gamma \max_{\bar{a}'} r(\boldsymbol{s'},\bar{a}') + \gamma^2 q_U\right) \tag{39}$$

$$\mathcal{B}_E^{K_1} q_U^k(s,a^k) \leq \max_{a_0^{-k},\bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1} q_U \tag{40}$$

Further application of $\mathcal{B}_I$ we get:

$$\begin{aligned}
\mathcal{B}_I\mathcal{B}_E^{K_1}q_U^k(s,a^k) &\leq \max\Bigg\{\max_{a_0^{-k},\bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1}\gamma^i r(\boldsymbol{s}_i,a_i^k,a_i^{-k})|\boldsymbol{s}_0=s\right)+\gamma^{K_1}q_U,\\
&\qquad \max_{a_0^{-k},\bar{a}_1,\cdots,\bar{a}_{K_1}} \mathbb{E}\left(\sum_{i=0}^{K_1}\gamma^i r(\boldsymbol{s}_i,a_i^k,a_i^{-k})|\boldsymbol{s}_0=s\right)+\gamma^{K_1+1}q_U\Bigg\}\\
&= \max_{a_0^{-k},\bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1}\gamma^i r(\boldsymbol{s}_i,a_i^k,a_i^{-k})|\boldsymbol{s}_0=s\right)+\gamma^{K_1}q_U
\end{aligned}\tag{41}$$

Therefore, we conclude that $\mathcal{B}_I^{K_2}\mathcal{B}_E^{K_1}\mathcal{B}_I^{K_0}q_U^k(s,a^k)=\mathcal{B}_E^{K_1}q_U^k(s,a^k)$. More generally, we can write:

$$\begin{aligned}
\mathcal{B}_I^{K_\ell}\cdots\mathcal{B}_E^{K_1}\mathcal{B}_I^{K_0}q_U^k(s,a^k)=\mathcal{B}_E^{n}q_U^k(s,a^k) &\leq \max_{a_0^{k},\bar{a}_1,\cdots,\bar{a}_{n-1}} \mathbb{E}\left(\sum_{i=0}^{\boldsymbol{n}-1}\gamma^i r(\boldsymbol{s}_i,a_i^k,a_i^{-k})|\boldsymbol{s}_0=s\right)+\gamma^{\boldsymbol{n}}q_U\\
&\leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k})+\gamma^{\boldsymbol{n}}\big(q_U-\min_s\max_{\bar{a}}q^\dagger(s,\bar{a})\big)
\end{aligned}\tag{42}$$

where we define $\boldsymbol{n}$ to be the total number of times that operator $\mathcal{B}_E$ is applied. Notice that if operator $\mathcal{B}_p$ is applied N times, $\boldsymbol{n}$ is a random variable that follows a binomial distribution with total samples N and probability p. Therefore, we get:

$$\mathcal{B}_p^N q_U^k(s,a^k)\leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k})+\gamma^{\boldsymbol{n}}\big(q_U-\min_s\max_{\bar{a}}q^\dagger(s,\bar{a})\big)\tag{43}$$

To ensure that $\mathcal{B}_p^N q_U^k(s,a^k)\in\mathcal{C}_{\delta_1}^U$ we need:

$$\delta_1\geq\gamma^{\boldsymbol{n}}\big(q_U-\min_s\max_{\bar{a}}q^\dagger(s,\bar{a})\big)\qquad\rightarrow\qquad \boldsymbol{n}\geq\log_\gamma\left(\frac{\delta_1}{q_U-\min_s\max_{\bar{a}}q^\dagger(s,\bar{a})}\right)\tag{44}$$

Since n follows a binomial distribution we get:

$$\mathbb{P}\left(\boldsymbol{n}\geq\underbrace{\left\lfloor\log_\gamma\left(\frac{\delta_1}{q_U-\min_s\max_{\bar{a}}q^\dagger(s,\bar{a})}\right)\right\rfloor}_{\triangleq n_o}\right)=1-\sum_{n=0}^{n_o}\binom{N}{n}p^n(1-p)^{N-n}\tag{45}$$

Therefore we can conclude that:

$$\mathbb{P}\big(\mathcal{B}_p^N q_U^k(s,a^k)\in\mathcal{C}_{\delta_1}^U\big)\geq 1-\sum_{n=0}^{n_o}\binom{N}{n}p^n(1-p)^{N-n}\tag{46}$$

$$\mathcal{C}_{\delta_1}^U=\big\{q^k|q^k(s,a^k)\leq\max_{a^{-k}}q^\dagger(s,a^k,a^{-k})+\delta_1\quad\forall(k,s,a^k)\in(\mathcal{K},\mathcal{S},\mathcal{A}^k)\big\}\tag{47}$$

Noting that by construction $\mathcal{B}_p^N q_U^k(s,a^k)\geq\mathcal{B}_p^N q^k(s,a^k)$ for all $N\geq 1$, we get that:

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k)\in\mathcal{C}_{\delta_1}^U\big)\geq 1-\sum_{n=0}^{n_o}\binom{N}{n}p^n(1-p)^{N-n}\tag{48}$$

For the special case where $n_o<pN$ we can bound the cumulative distribution function of the binomial distribution using Hoeffding's bound:

$$\mathbb{P}\big(\mathcal{B}_p^N q^k(s,a^k)\in\mathcal{C}_{\delta_1}^U\big)\geq 1-e^{-2N\left(p-\frac{n_o}{N}\right)^2}\tag{49}$$

which concludes the proof.

6.3 Proof of Lemma 2

We start stating the following auxiliary lemma.

Lemma 3. *If $q^k(s,a^k) \in \mathcal{C}_0^U$ then it holds that $\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_0^U$ for all $N \geq 0$.*

Proof. We start noting that if $q^k(s,a^k) \in \mathcal{C}_0^U$ then $\mathcal{B}_E q^k(s,a^k) \in \mathcal{C}_0^U$ and $\mathcal{B}_I q^k(s,a^k) \in \mathcal{C}_0^U$.

$$\begin{aligned}\mathcal{B}_I q^k(s,a^k) &= \max\left\{q^k(s,a^k), \max_{a^{-k}} \mathbb{E}\left(r(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a',k} q^k(\boldsymbol{s'},a'^{,k})\right)\right\} \\ &\overset{(a)}{\leq} \max\left\{\max_{a^{-k}} q^\dagger(s,a^k), \max_{a^{-k}} \mathbb{E}\left(r(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'},\bar{a}')\right)\right\} = \max_{a^{-k}} q^\dagger(s,a^k) \quad (50)\\ \mathcal{B}_E q^k(s,a^k) &= \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a'} q^k(\boldsymbol{s'},a')\right)\Big|_{a^n=\arg\max_{a^n} q^n(s,a^n)\ \forall n\neq k} \\ &\leq \max_{a^{-k}} \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a'} q^k(\boldsymbol{s'},a')\right) \\ &\overset{(b)}{\leq} \max_{a^{-k}} \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'},\bar{a}')\right) = \max_{a^{-k}} q^\dagger(s,a^k) \quad (51)\end{aligned}$$

where in (a) and (b) we used the fact that $q^k(s,a^k) \in \mathcal{C}_0^U$. Since it holds that $\mathcal{B}_E q^k(s,a^k) \in \mathcal{C}_0^U$ and $\mathcal{B}_I q^k(s,a^k) \in \mathcal{C}_0^U$ it immediately follows that $\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_0^U$ for all $N \geq 0$. ■

We follow by noting that applying operator $\mathcal{B}_I$ L times to $q^k(s,a^k) < \max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ we get:

$$\begin{aligned}\mathcal{B}_I q^k(s,a^k) &= \max\left\{q^k(s,a^k), \max_{a^{-k}} \mathbb{E}\left(r(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a',k} q^k(\boldsymbol{s'},a'^{,k})\right)\right\} \\ &\geq \max_{a^{-k}} \mathbb{E}\left(r(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a',k} q^k(\boldsymbol{s'},a'^{,k})\right) \quad (52)\\ \mathcal{B}_I^L q^k(s,a^k) &= \max_{a_0^{-k},\bar{a}_1,\cdots,\bar{a}_{L-1}} \mathbb{E}\left(\sum_{i=0}^{L-1} \gamma^i r(\boldsymbol{s}_i,a_i^k,a_i^{-k}) + \gamma^L \max_{a_L^k} q^k(\boldsymbol{s}_L,a_L^k)|\boldsymbol{s}_0=s,a_0^k=a^k\right) \\ \left|\mathcal{B}_I^L q^k(s,a^k) - \max_{a^{-k}} q^\dagger(s,a^k,a^{-k})\right| &\overset{(c)}{\leq} \gamma^L \max_s \left|\max_{a^k} q^k(s,a^k) - \max_{\bar{a}} q^\dagger(s,\bar{a})\right| \triangleq \epsilon(L) \quad (53)\end{aligned}$$

where in (c) we used $\mathcal{B}_I^L q^k(s,a^k) < \max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ from lemma 3. If $\epsilon(L) < \delta_2$, we get:

$$\mathcal{B}_I^L q^k(s,a^k) \in \mathcal{C}_{\delta_2} \quad (54)$$

$$\mathcal{C}_{\delta_2} = \left\{q^k | q^{k,\star}(s,a^k) - \delta_2 \leq q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) + \delta_2 \forall (k,s,a^k) \in (\mathcal{K},\mathcal{S},\mathcal{A}^k)\right\} \quad (55)$$

Lemma 4. *If $q^k(s,a^k) \in \mathcal{C}_{\delta_2}$ and δ_2 is small enough, then it holds that $\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\gamma^N \delta_2}$ for all $N > 0$.*

Proof. Assume δ_2 satisfies the following relation:

$$\delta_2 < 2^{-1} \min_s \left(\max_{\bar{a}} q^\dagger(s,\bar{a}) - \max_{\bar{a} \neq \arg\max_{\bar{a}} q^\dagger(s,\bar{a})} q^\dagger(s,\bar{a})\right) \quad (56)$$

We clarify that the term in between parenthesis in the r.h.s. of relation (56) is the difference between the optimal q-value and the second highest q-value for state s. Note that if $q^k(s,a^k) \in \mathcal{C}_{\delta_2}$ then it trivially follows that $\mathcal{B}_I q^k(s,a^k) \in \mathcal{C}_{\delta_2}$, for the case of $\mathcal{B}_E$ we get:

$$\mathcal{B}_E q^k(s,a^k) = \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a'} q^k(\boldsymbol{s'},a')\right)\Big|_{a^n=\arg\max_{a^n} q^n(s,a^n)\ \forall n\neq k} \quad (57)$$

Using equation (3a) and the fact that $q^k(s,a^k) \in \mathcal{C}_{\delta_2}$ it follows:

$$\max_{a^k} q^{k,\star}(s,\bar{a}) - \delta_2 \leq \max_{a^k} q^k(s,a^k) \quad (58)$$

$$q^k(s,a^{k,\bullet}) \leq \max_{a^{-k}} q^\dagger(s,a^{k,\bullet},a^{-k}) + \delta_2 \overset{(d)}{<} \max_{\bar{a}} q^\dagger(s,\bar{a}) - \delta_2 = \max_{a^k} q^{k,\star}(s,\bar{a}) - \delta_2 \quad (59)$$

$$a^{k,\bullet} = \arg\max_{a^k \neq \arg\max_{a^k} q^k(s,a^k)} q^k(s,a^k) \quad (60)$$

where in (d) we used condition (56). Combining equations (57) through (60) we get:

$$\begin{aligned}\mathcal{B}_E q^k(s,a^k) &= \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a'} q^k(\boldsymbol{s'},a')\right)\big|_{a^n=\arg\max\limits_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \\ &= \mathbb{E}\left(\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a'} q^k(\boldsymbol{s'},a') + \gamma \max_{a'} q^{k,\star}(\boldsymbol{s'},a') - \gamma \max_{a'} q^{k,\star}(\boldsymbol{s'},a')\right)\big|_{a^n=\arg\max\limits_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \\ &= q^{k,\star}(s,a^k) + \gamma \mathbb{E}\left(\max_{a'} q^k(\boldsymbol{s'},a') - \max_{a'} q^{k,\star}(\boldsymbol{s'},a')\right)\big|_{a^n=\arg\max\limits_{a^n} q^{n,\star}(s,a^n)\ \forall n\neq k} \qquad (61)\end{aligned}$$

Combining equation (61) with the fact that $q^k(s,a^k) \in \mathcal{C}_{\delta_2}$ we get:

$$q^{k,\star}(s,a^k) - \gamma\delta_2 \leq \mathcal{B}_E q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) - \gamma\delta_2 \qquad (62)$$

which completes the proof. ∎

Lemma 5. *For any $q^k(s,a^k) \in \mathcal{C}_0^U$, $\delta_2 > 0$ and $N \geq$, it holds that $\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\delta_2}$ as long as the sequence of N operators $\mathcal{B}_p$ includes at least L consecutive $\mathcal{B}_I$'s. Where L is given by:*

$$L = \left\lceil \log_\gamma \left(\frac{\delta_2}{\max_s \left| \max_{a^k} q^k(s,a^k) - \max_{\bar{a}} q^\dagger(s,\bar{a}) \right|} \right) \right\rceil \qquad (63)$$

Proof. The statement is an immediate consequence of lemmas 3 and 4 and relation (53). Relation (63) follows from combining equation (53) and $\epsilon(L) < \delta_2$. ∎

Notice that the probability of applying operator $\mathcal{B}_I$ at least L consecutive times when operator $\mathcal{B}_p$ is applied $N \geq L$ times, is the same as the probability of obtaining at least L consecutive heads when a biased coin (with probability of head $1-p$) is tossed N times. This problem has been extensively studied and the result is available in the literature. We state the following useful result from [Uspensky, 1937]:

Lemma 6. *[Uspensky, 1937]: If a biased coin (with probability of head being $1-p$) is tossed $N \geq L$ times, the probability of having a sequence of at least L consecutive heads is given by:*

$$\mathbb{P}(L) = 1 - \beta_{N,L} + (1-p)^L \beta_{N-L,L} \qquad (64)$$

$$\beta_{N,L} = \sum_{j=0}^{\lfloor N/(L+1) \rfloor} (-1)^j \binom{N-jL}{j} \left(p(1-p)^L\right)^j \qquad (65)$$

Furthermore, if $p > 0.5$ the probability can be lower bounded as follows:

$$\mathbb{P}(L) \geq 1 - \frac{1-(1-p)\xi_1}{p\xi_1(1+L-L\xi_1)}\xi_1^{-N} - \frac{L}{p}(1-p)^{N+2} \qquad (66)$$

where $1 < \xi_1 < 1 + L^{-1}$.

Combining lemmas 6 and 5 we can conclude that after $N \geq L > 0$ applications of operator $\mathcal{B}_p$ to any set of K $q^k(s,a^k) \in \mathcal{C}_0^U$ functions it holds:

$$\mathbb{P}\left(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\delta_2}\right) \geq 1 - \beta_{N,L} + (1-p)^L \beta_{N-L,L} \qquad (67)$$

$$L = \left\lceil \log_\gamma \left(\frac{\delta_2}{\max_s \left| \max_{a^k} q^k(s,a^k) - \max_{\bar{a}} q^\dagger(s,\bar{a}) \right|} \right) \right\rceil \qquad (68)$$

$$\mathcal{C}_{\delta_2} = \left\{ q^k \middle| q^{k,\star}(s,a^k) - \delta_2 \leq q^k(s,a^k) \leq \max_{a^{-k}} q^\dagger(s,a^k,a^{-k}) + \delta_2 \forall (k,s,a^k) \in (\mathcal{K},\mathcal{S},\mathcal{A}^k) \right\} \qquad (69)$$

If $p > 0.5$ we can lower bound probability (67) by:

$$\mathbb{P}\left(\mathcal{B}_p^N q^k(s,a^k) \in \mathcal{C}_{\delta_2}\right) \geq 1 - \frac{1-(1-p)\xi_1}{p\xi_1(1+L-L\xi_1)}\xi_1^{-N} - \frac{L}{p}(1-p)^{N+2} \qquad (70)$$

6.4 Tabular *Logical Team Q-Learning*

In the particular case where the MDP is deterministic (and hence $\mathbb{E}\left(\boldsymbol{r}(s,\bar{a},\boldsymbol{s'})+\gamma\max_{a'}q^k(s',a')\right)=r(s,\bar{a},s')+\gamma\max_{a'}q^k(s',a')$) the tabular version of *Logical Team Q-learning* is given by algorithm 2.

Algorithm 2 Tabular *Logical Team Q-Learning* for deterministic MDPs

```
Initialize: an empty replay buffer $\mathcal{R}$ and estimates $\widehat{q}_B^k$ and $\widehat{q}_U^k$.
for iterations $e = 0, \dots, E$ do
  Sample $T$ transitions $(s, \bar{a}, r, s')$ by following some behavior policy which guarantees all joint actions are sampled with non-zero probability and store them in $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ do
    Sample a transition $(s, \bar{a}, r, s')$ from $\mathcal{R}$.
    for agent $k = 1, \cdots, K$ do
      if $\big(a^n = \arg\max_{a^n} \widehat{q}^n(s, a^n)\ \forall n \neq k\big)$ then
        $\widehat{q}^k(s,a^k) = \widehat{q}^k(s,a^k) + \mu\big(r + \max_a \widehat{q}^k(s',a) - \widehat{q}^k(s,a^k)\big)$
      else if $\big(r + \max_a \widehat{q}^k(s',a) > \widehat{q}^k(s,a^k)\big)$ then
        $\widehat{q}^k(s,a^k) = \widehat{q}^k(s,a^k) + \mu\alpha\big(r + \max_a \widehat{q}^k(s',a) - \widehat{q}^k(s,a^k)\big)$
      end if
    end for
  end for
end for
```

Algorithm 3 Tabular *Logical Team Q-Learning*

```
Initialize: an empty replay buffer $\mathcal{R}$ and estimates $\widehat{q}_B^k$ and $\widehat{q}_U^k$.
for iterations $e = 0, \dots, E$ do
  Sample $T$ transitions $(s, \bar{a}, r, s')$ by following some behavior policy and store them in $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ do
    Sample a transition $(s, \bar{a}, r, s')$ from $\mathcal{R}$.
    for agent $k = 1, \cdots, K$ do
      if $a^n = \arg\max_{a^n} \widehat{q}_B^n(s, a^n)\ \forall n \neq k$ then
        $\widehat{q}_B^k(s,a^k) = \widehat{q}_B^k(s,a^k) + \mu\big(r + \max_a \widehat{q}_U^k(s',a) - \widehat{q}_B^k(s,a^k)\big)$
        $\widehat{q}_U^k(s,a^k) = \widehat{q}_U^k(s,a^k) + \mu\big(r + \max_a \widehat{q}_U^k(s',a) - \widehat{q}_U^k(s,a^k)\big)$
      end if
      if $\big(r + \max_a \widehat{q}_U^k(s',a) > \widehat{q}_B^k(s,a^k)\big)$ then
        $\widehat{q}_B^k(s,a^k) = \widehat{q}_B^k(s,a^k) + \mu\alpha\big(r + \max_a \widehat{q}_U^k(s',a) - \widehat{q}_B^k(s,a^k)\big)$
      end if
    end for
  end for
end for
```

If algorithm 2 were applied to a stochastic MDP, due to condition c_2 $\left(r+\max_a \widehat{q}^k(s',a) > \widehat{q}^k(s,a^k)\right)$, it would be subject to bias, which would propagate through bootstrapping and hence could compromise its performance. This can be solved by having a second unbiased estimate q_U that is updated only when c_1 is satisfied and use this unbiased estimate to bootstrap. The resulting algorithm is shown in algorithm 3.

6.5 Matrix game additional results

We start specifying the hyperparameters. For IQL, DistQ, LTQL and Qtran we used a step-size equal to 0.1. The α parameter for LTQL is equal to 1. The mixing network in Qmix has 2 hidden layers with 5 units each, the nonlinearity used was the *ELu* and the step-size used was 0.05 (we had to make it smaller than the others to make the SGD optimizer converge). We finally remark that due to the use of a NN in Qmix we had to train this algorithm with 100 times more games (notice the x-axis in figure 2).

In figure 2 we show the convergence curves for IQL 2a, DistQ 2b, LTQL 2c-2f, Qmix 2g and Qtran 2h. Figures 2c and 2d correspond to the deterministic (algorithm 2) and general version (algorithm 3) of LTQL, respectively. Figures 2e and 2f also show curves for LTQL but use different seeds and show that this algorithm can converge to either of the two following set of factored q-functions:

$$q^{1,\star}(a^1) = [2,1] \quad q^{2,\star}(a^2) = [0,2,0] \qquad \textbf{or} \qquad q^{1,\star}(a^1) = [0,2] \quad q^{2,\star}(a^2) = [0,1,2] \tag{71}$$

One interesting fact to note is that the suboptimal values of q^k_B (in figures 2c and 2e) do not converge while the same values do converge in the case of q^k_U. The reason for this is that there is a parallel between including the unbiased estimate q^k_U in the RL algorithm and including an application of operator $\mathcal{B}_E$ at the end of the dynamic programming procedure described in theorem 1. The proof of the theorem shows that if such operator is not included, only the optimal values of the estimates generated by *Logical Team Q-learning* converge to $q^{k,\star}$, while the values corresponding to suboptimal actions oscillate in the region between $q^{k,\star}(s,a^k)$ and $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ (which is what happens when the unbiased estimate q^k_U is not used in algorithm 2). Note that in the case of algorithm 3 where the unbiased estimate q^k_U is included all values q^k converges to $q^{k,\star}$ for all actions, not just the optimal ones (this is equivalent to including the operator $\mathcal{B}_E$ after $\mathcal{B}^N_p$ in the dynamic programming setting). Below we show the joint q values generated by Qmix's mixing network.

		Agent 2		
		a_1 (-3.49)	a_2 (1.83)	a_3 (0.62)
Agent 1	b_1 (-0.74)	-4.78×10^{-2}	1.17	6.86×10^{-1}
	b_2 (1.09)	1.51×10^{-3}	1.57	1.09

Table 1: Qmix full results

Code is available at https://github.com/lcassano/Logical-Team-Q-Learning-paper.

6.6 Stochastic TMDP additional results

In this environment agents rely on the following observations. The observation corresponding to agent 1 is a vector with two binary elements: the first one indicates whether or not agent 2 is in the leftmost position, and the second element indicates whether or not there is enough time for agent 2 to reach the leftmost position. The observation corresponding to agent 2 is a vector with two elements: the first one is the number of the position it occupies and the second one is the same as agent 1 (whether there is enough time to reach the leftmost position).

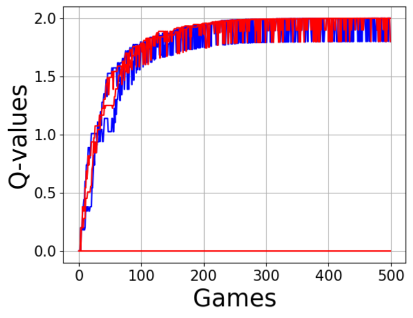
All algorithms are implemented in an on-line manner with no replay buffer. ϵ-greedy exploration with a decaying schedule is used in all cases ($\epsilon = \max[0.05, 1 - \textit{epoch}/2 \times 10^5]$). The step-size used is $\mu = 0.025$ and the smaller step-size for HystQ is $\mu_{\text{small}} = 10^{-2}$, in the case of Qmix we used $\mu = 10^{-3}$ to guarantee stability. The α parameter for LTQL is equal to 1.

Figures 3 show the estimated q-values for LTQL corresponding to 4 different observations at the 4 positions. Note that in all figures the optimum action has the highest value and correctly estimates the return corresponding to the optimal team policy (+10).

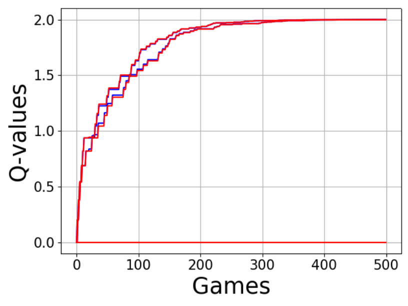
Figures 4 show the learning curves for IQL. Note that IQL fails at this environment because it has no mechanism to discard the -30 penalty incurred due to moving to the left when agent 1 presses the button due to exploration.

Figures 5 show the learning curves for DistQ. The reason that this algorithm cannot solve this environment is that it severely overestimates the value of choosing to move to the right whilst on the rightmost position. It is well known that this is a consequence of the fact that DistQ only performs updates that increase the estimates of the Q-values combined with the stochastic reward received when agent 2 "stumbles" against the right edge.

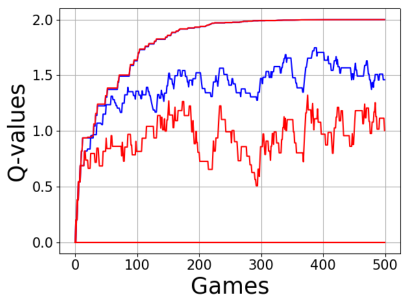
Figures 6 show the learning curves for HystQ. This algorithm cannot solve this environment because it has two issues and the way to solve one makes the other worse. More specifically, one can be solved by increasing the smaller step-size, while the other needs to decrease it. The first issue is the same one that affects DistQ, i.e.,



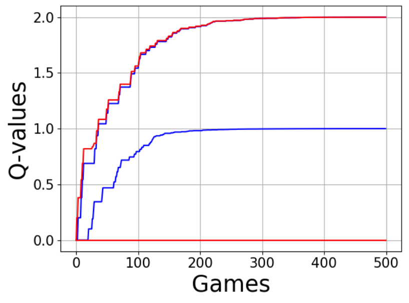
(a) IQL



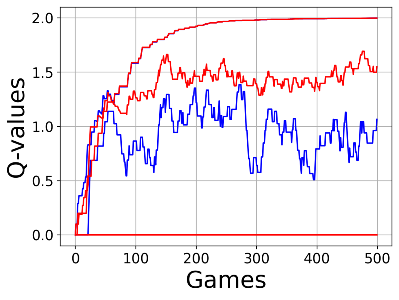
(b) DistQ



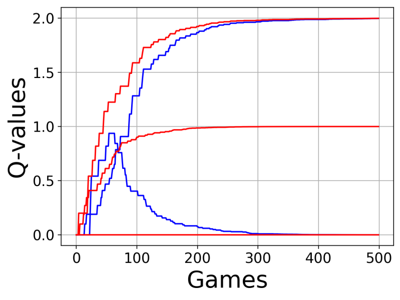
(c) q_B (seed=0)



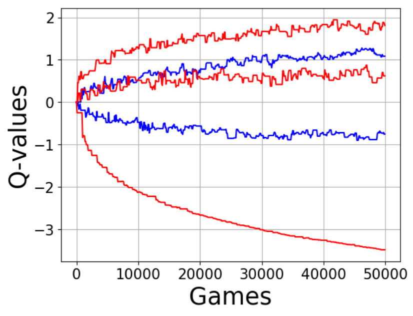
(d) q_U (seed=0)



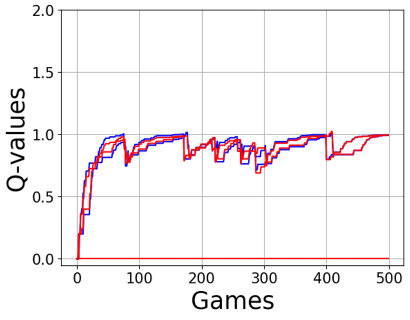
(e) q_B (seed=1)



(f) q_U (seed=1)



(g) Qmix



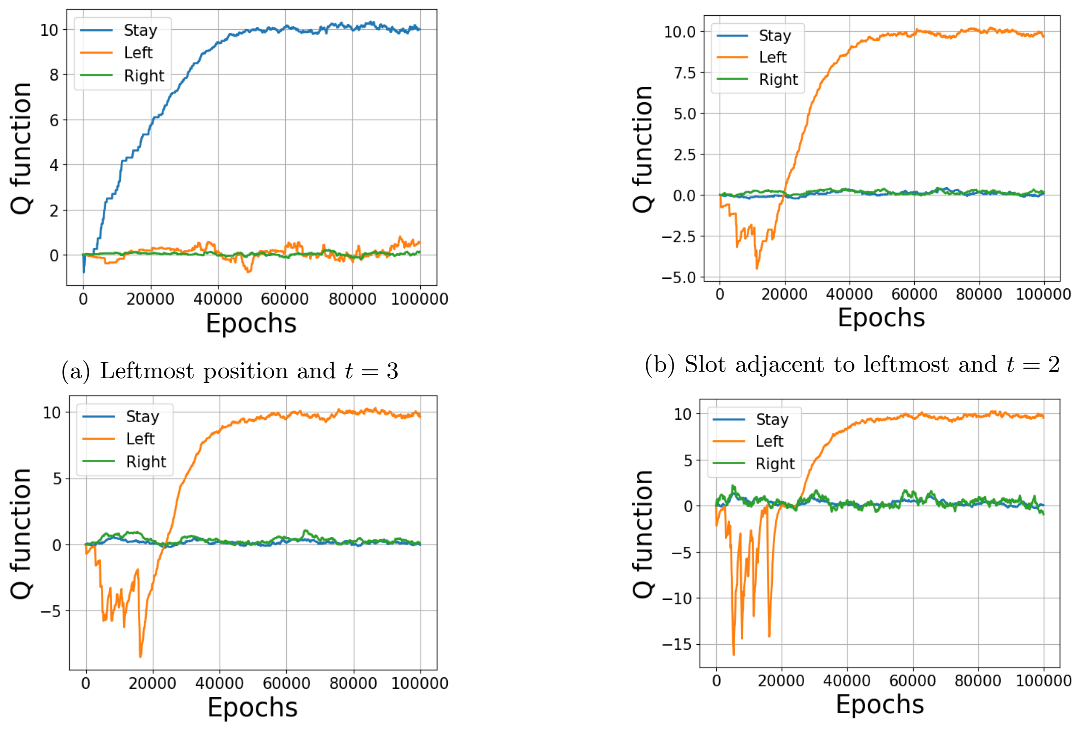
(h) Qtran

Figure 2: Matrix game. In all figures the red curves correspond to the three actions of agent 2, while the two blue curves correspond to the two actions from agent 1.

the overestimation of the *move right* action in the rightmost position. Note that this can be ameliorated by increasing the small step-size. The second issue is the penalty incurred due to moving to the left when agent 1 presses the button. This can be ameliorated by decreasing the small step-size. The fact that there is no intermediate value for the small step-size to solve both issues is the reason that this algorithm cannot solve this environment.

Figures 7 show the learning curves for Qmix. Qmix fails at this task due to the fact that its monotonic factoring assumption is not satisfied at this task. The architecture used is as follows: we used tabular representation for the individual q functions, and for the mixing and hypernetworks we used the architecture specified in [Rashid et al., 2020]. More specifically, the mixing network is composed of two hidden layers (with 10 units each) with *ELu* nonlinearities in the first layer while the second layer is linear. The hypernetworks that output the weights of the mixing network consist of two layers with *ReLU* nonlinearities followed by an activation function that takes the absolute value to ensure that the mixing network weights are non-negative. The bias of the first mixing layer is produced by a network with a unique linear layer and the other bias is produced by a two layer hypernetwork with a *ReLU* nonlinearity. All hypernetwork layers are fully connected and have 5 units.

Figures 8 show the learning curves for Qtran. Qtran succeeds at this task. However, it is important to remark



(a) Leftmost position and $t = 3$

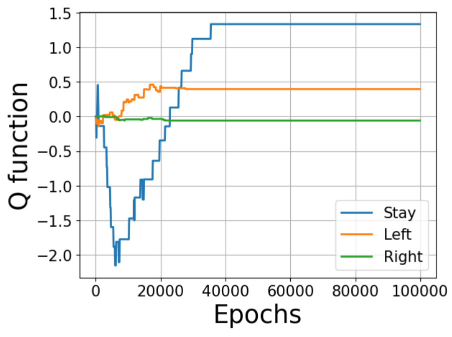
(b) Slot adjacent to leftmost and $t = 2$

(c) Slot adjacent to rightmost and $t = 1$

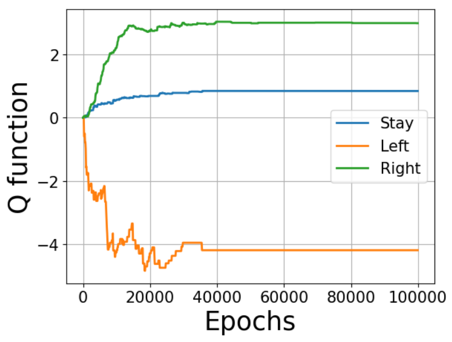
(d) Rightmost position and $t = 0$

Figure 3: Learning curves for agent 2 of *Logical Team Q-learning* for a random seed.

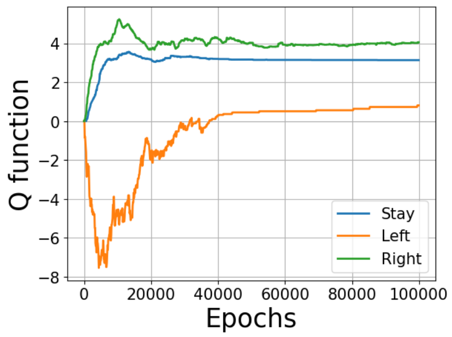
that this is a tabular implementation of Qtran (an algorithm designed to be used in conjuntion with NNs in complex environment), where the algorithm estimates the full joint q-function in tabular form, which is not scalable and defeats the purpose of learning factored q-functions.



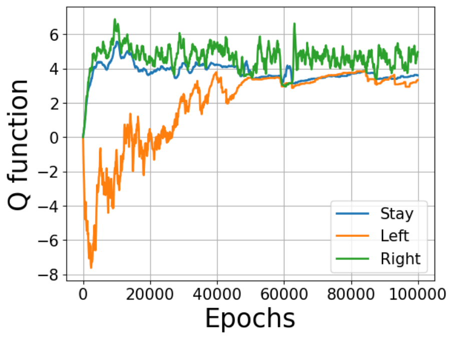
(a) Leftmost position and $t = 3$



(b) Slot adjacent to leftmost and $t = 2$

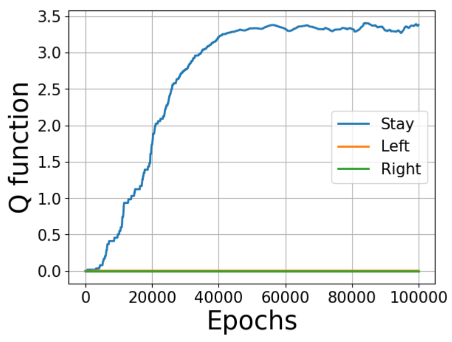


(c) Slot adjacent to rightmost and $t = 1$

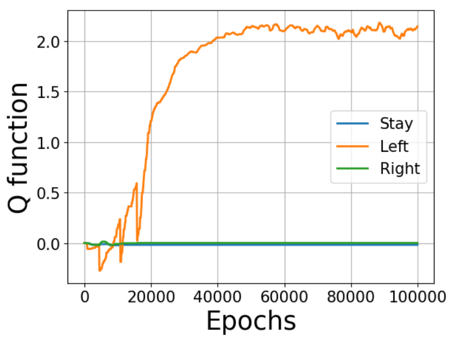


(d) Rightmost position and $t = 0$

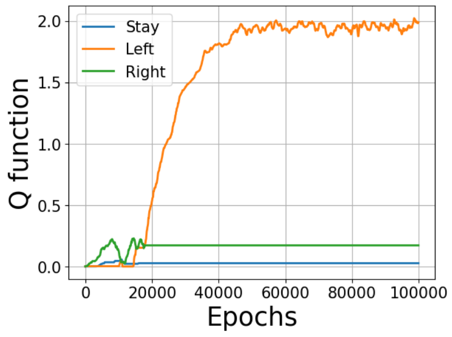
Figure 4: Learning curves for agent 2 of IQL for a random seed.



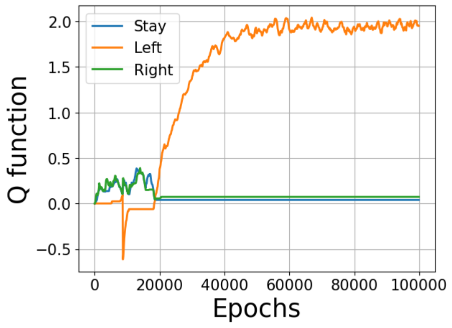
(a) Leftmost position and $t = 3$



(b) Slot adjacent to leftmost and $t = 2$



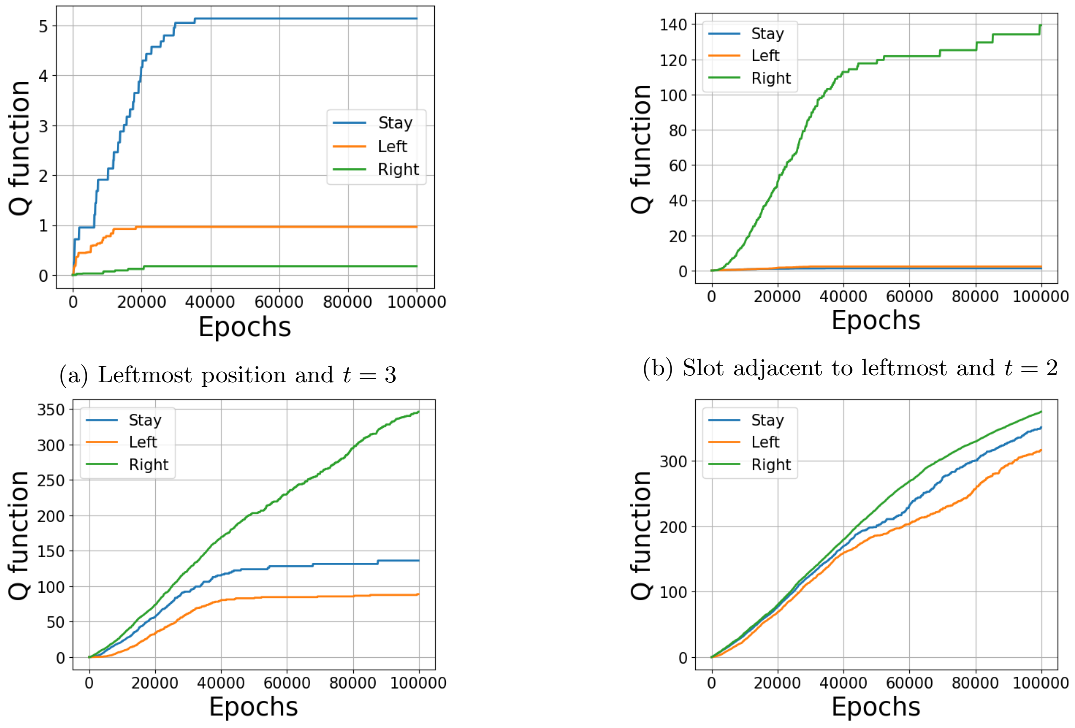
(c) Slot adjacent to rightmost and $t = 1$



(d) Rightmost position and $t = 0$

Figure 8: Learning curves for agent 2 of Qtran for a random seed.

All code is available at https://github.com/lcassano/Logical-Team-Q-Learning-paper.



(a) Leftmost position and $t = 3$

(b) Slot adjacent to leftmost and $t = 2$

(c) Slot adjacent to rightmost and $t = 1$

(d) Rightmost position and $t = 0$

Figure 5: Learning curves for agent 2 of DistQ for a random seed.

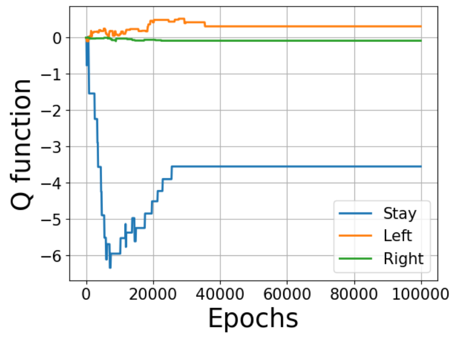
6.7 Cowboy bull game additional results

The bull's policy is given by the pseudocode shown in algorithm 4.

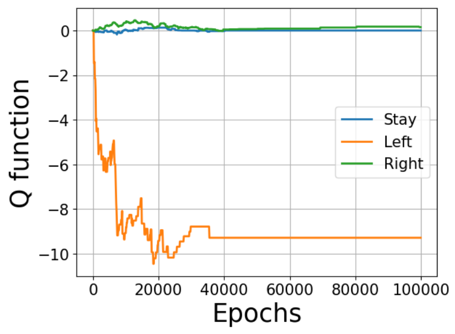
Algorithm 4 Bull's policy.

```
if distance to all predators > 10 (this circumference is depicted by the blue line in figure 1c) then
    Natural foraging behavior: Stay still with 90% probability, otherwise make a small move in a random direction.
else
    if the maximum angle formed by two predators is > $108^o$ then
        There's a hole to escape: Escape through the direction in between these two predators.
    else if distance to farthest predator - distance to closest predator > 5 then
        There's no hole, but one predator is much closer than the others so run in the direction opposite to this predator.
    else
        No way out (scared): Stay still with 70% probability, otherwise make a fast move in a random direction.
    end if
end if
```

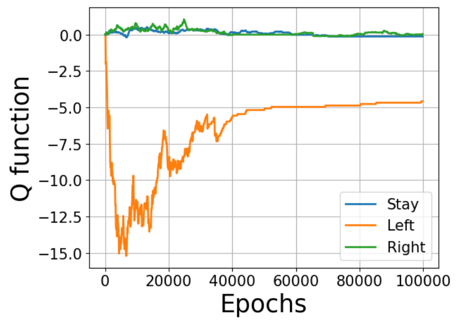
We now specify the hyperparameters for *Logical Team Q-learning*. All NN's have two hidden layers with 50 units and ReLu nonlinearities. However, for each Q-network, instead of having one network with 5 outputs, we have 5 networks each with 1 output (one for each action). At every epoch the agent collects data by playing 32 full games and then performs 50 gradient backpropagation steps. Half of the 32 games are played greedily and the other half use a Boltzmann policy with temperature b_T that decays according to the following schedule $b_T = \max[0.05, 0.5 \times (1 - \mathit{epoch}/15 \times 10^3)]$. We use this behavior policy to ensure that there are sufficient transitions that satisfy condition c_1 and that also there are transitions that satisfy c_2. The target networks are updated every 50 backprop steps. The capacity of the replay buffer is 2.10^5 transitions, the mini-batch size is



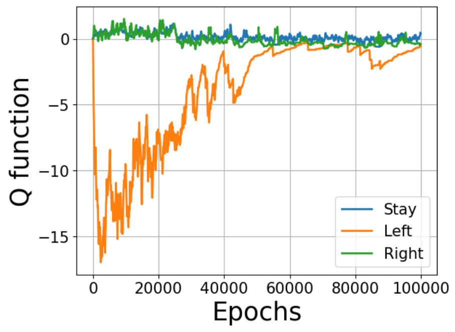
(a) Leftmost position and $t = 3$



(b) Slot adjacent to leftmost and $t = 2$



(c) Slot adjacent to rightmost and $t = 1$



(d) Rightmost position and $t = 0$

Figure 6: Learning curves for agent 2 of HystQ for a random seed.

1024, $\alpha = 1$, we use a discount factor equal to 0.99 and optimize the networks using the Adam optimizer with initial step-size 10^{-5}.

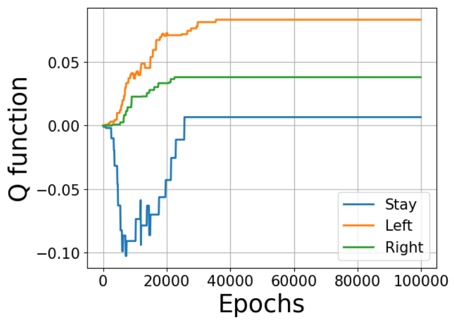
The hyperparameters of the HystQ implementation are the same as those of LTQL, the ratio of the two step-sizes used by HystQ is 0.1. To run IQL we used the implementation of HystQ with the ratio of the two step-sizes set to 0.

The architecture used by Qmix is the one suggested in [Rashid et al., 2020] with the exception that, for fairness, the individual Q-networks used the same architecture as the ones used by the other algorithms (i.e., 5 networks with a unique output as opposed to 1 network with 5 outputs). All hidden layers of the hypernetworks as well the mixing network have 10 units. In this case we did 5 backprop iterations per epoch and the target network update period is 15. We use a batch size of 256, a discount factor equal to 0.98 and optimize the networks with the Adam optimizer with initial step-size 10^{-6}. In this case the behavior policy is always Boltzmann with the following annealing schedule for the temperature parameter $b_T = \max[0.005, 0.05 \times (1 - epoch/25 \times 10^3)]$.

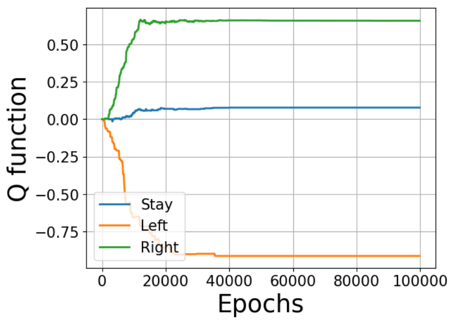
The Qtran-base variant was implemented. The individual Q-networks have the same architecture used by the other algorithms. The joint Q-network has two hidden layers with 60 units each, the input of this network is the global state concatenated with the agents' actions with one hot encoding. The value function network has only one hidden layer with 25 units and its input is the global state. The target networks are updated every 50 backprop steps. The capacity of the replay buffer is 2.10^5 transitions, the mini-batch size is 1024, we use a discount factor equal to 0.99 and optimize the networks using the Adam optimizer with initial step-size 10^{-5}.

The batch size, Boltzmann temperature value, learning step-size and target update period were chosen by grid search.

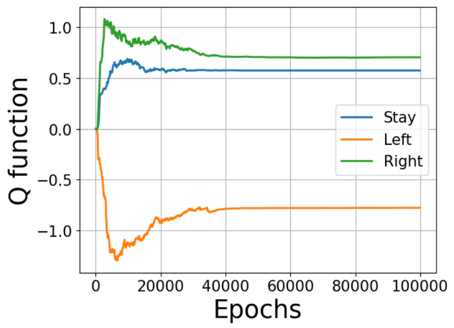
All implementations use TensorFlow 2. The code is available at https://github.com/lcassano/Logical-Team-Q-Learning-paper. Running one seed for one agent takes approximately 12 hours in our hardware (2017 iMac with 3.8 GHz Intel Core i5 and 16GB of RAM).



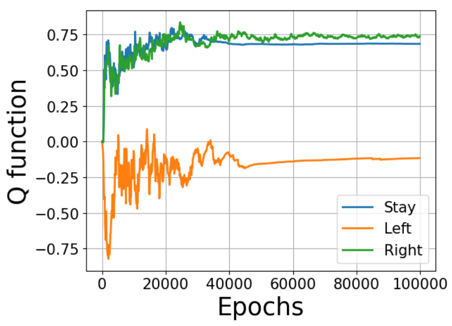
(a) Leftmost position and $t = 3$



(b) Slot adjacent to leftmost and $t = 2$



(c) Slot adjacent to rightmost and $t = 1$



(d) Rightmost position and $t = 0$

Figure 7: Learning curves for agent 2 of Qmix for a random seed.