Version of Record: https://www.sciencedirect.com/science/article/pii/S0925231216000783
3d80fb727768e4af323ddfa474cc39a0

Multi-agent Reinforcement Learning as a Rehearsal for Decentralized Planning

Landon Kraemer and Bikramjit Banerjee
Corresponding Author: Bikramjit Banerjee
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406
Phone: +1-601-266-6287
{Bikramjit.Banerjee, Landon.Kraemer}@usm.edu

Abstract

Decentralized partially-observable Markov decision processes (Dec-POMDPs) are a powerful tool for modeling multi-agent planning and decision-making under uncertainty. Prevalent Dec-POMDP solution techniques require centralized computation given full knowledge of the underlying model. Multi-agent reinforcement learning (MARL) based approaches have been recently proposed for distributed solution of Dec-POMDPs without full prior knowledge of the model, but these methods assume that conditions during learning and policy execution are identical. In some practical scenarios this may not be the case. We propose a novel MARL approach in which agents are allowed to *rehearse* with information that will not be available during policy execution. The key is for the agents to learn policies that do not explicitly rely on these *rehearsal features*. We also establish a weak convergence result for our algorithm, RLaR, demonstrating that RLaR converges *in probability* when certain conditions are met. We show experimentally that incorporating rehearsal features can enhance the learning rate compared to non-rehearsal-based learners, and demonstrate fast, (near) optimal performance on many existing benchmark Dec-POMDP problems. We also compare RLaR against an existing approximate Dec-POMDP solver which, like RLaR, does not assume a priori knowledge of the model. While RLaR's policy representation is not as scalable, we show that RLaR produces higher quality policies for most problems and horizons studied.

Keywords: Multi-agent Reinforcement Learning; Decentalized Planning.

1 Introduction

Decentralized partially observable Markov decision processes (Dec-POMDPs) offer a powerful model of decentralized decision making under uncertainty and incomplete knowledge. Many exact solvers have been developed for Dec-POMDPs [28, 19, 26], but these approaches are not scalable because the underlying problem is provably NEXP-complete [6]. More scalable approximate solvers have been proposed (e.g. [24, 1, 10]); however, most of these approaches are centralized and assume the model is known a priori.

Recently, multi-agent reinforcement learning (MARL) techniques have been applied [30, 4] to overcome the limitations of centralized computation and comprehensive knowledge of model parameters. While MARL distributes the policy computation problem among the agents themselves, it essentially solves a more difficult problem, because it does not assume the model parameters are known a-priori. The hardness of the underlying problem translates to significant sample complexity for RL solvers as well [4].

One common feature of the existing RL solvers is that the learning agents subject themselves to the same constraints that they would encounter when executing the learned policies. In particular, agents assume that the environment states are hidden and the other agents' actions are invisible, in addition to the other agents' observations being hidden too. We argue that in many practical scenarios, it may actually be easy to allow learning

agents to observe some otherwise hidden information *only while they are learning*. We view such learning as a *rehearsal*—a phase where agents are allowed to access information that will not be available when executing their learned policies. While this additional information can facilitate the learning during rehearsal, agents must learn policies that can indeed be executed in the Dec-POMDP (i.e., without relying on this additional information). Thus agents must ultimately wean their policies off of any reliance on this information. This creates a principled incentive for agents to explore actions that will help them achieve this goal. Based on these ideas, we present a new approach to RL for Dec-POMDPs—*R*einforcement *L*earning *a*s a *R*ehearsal or RLaR, including a new exploration strategy. We establish a weak convergence result for RLaR, demonstrating that RLaR's value function converges *in probability* when certain conditions are met, and demonstrate experimentally that RLaR can nearly optimally solve several existing benchmark Dec-POMDP problems with a low sample complexity. We also compare RLaR against an existing approximate Dec-POMDP solver, Dec-RSPI [29], which also does not assume a priori knowledge of the model. Instead, Dec-RSPI assumes full access to a *simulator*, which makes it a planning algorithm. Even so, it is comparable to a learning algorithm such as RLaR due to the shared lack of a priori knowledge of the model, and the access to at least as much information as RLaR assumes. We show that while RLaR's policy representation is not as scalable as Dec-RSPI's, it produces higher quality policies for problems and horizons studied.

2 Background

We use the standard notation of subscript i to denote agent i, and $-i$ to denote all agents except i. Superscript of "*" denotes true, unknown values that are typically going to be estimated. Values with "hats" denote estimates.

2.1 Decentralized POMDPs

A Dec-POMDP is defined as a tuple $\langle n, S, A, P, R, \Omega, O, \beta \rangle$, where:

- n is the number of agents in the domain.
- S is a finite set of (unobservable) environment states.
- $A = \times_i A_i$ is a set of joint actions, where A_i is the set of individual actions that agent i can perform.
- $P(s'|s, \vec{a})$ gives the probability of transitioning to state $s' \in S$ when joint action $\vec{a} \in A$ is taken in state $s \in S$.
- $R(s, \vec{a})$ gives the immediate scalar reward that the agents jointly receive upon executing action $\vec{a} \in A$ in state $s \in S$.
- $\Omega = \times_i \Omega_i$ is the set of joint observations, where Ω_i is the finite set of individual observations that agent i can receive from the environment.
- $O(\vec{\omega}|s', \vec{a})$ gives the probability of the agents jointly observing $\vec{\omega} \in \Omega$ if the current state is $s' \in S$ and the previous joint action was $\vec{a} \in A$.
- $\beta \in \Delta(S)$ is the initial state distribution.

Additionally, for finite horizon problems, a horizon T is given that specifies how many steps of interaction the agents are going to have with the environment and each other. The objective in such problems is to compute a set of decision functions or *policies*—one for each agent—that maps the history of action-observations ($h_i = (a_i^0, \omega_i^1, a_i^1, \omega_i^2, a_i^2, \ldots, \omega_i^k), k \leq T-1$) of each agent to its best next action (i.e., $\pi_i : (A_i \times \Omega_i)^k \mapsto A_i$ for agent i), such that the *joint* behavior over T steps optimizes the total reward obtained by the team. Agents do not share their actions and observations with other agents, hence each agent i's policy is defined on the basis of *private* information—action and observation history, h_i. In principle, this problem can be posed as an optimization problem as follows:

- Binary variables $P(a_i|h_i)$ for every agent i that essentially describe the agents' deterministic policies, i.e. $P(a_i|h_i) = 1 \iff \pi_i(h_i) = a_i$, i.e., agent i will execute action a_i with certainty after observing history h_i.
- Real variables $P_i(s, h_{-i}|h_i)$: likelihood of features hidden to i given its private history h_i, constrained by the following equation

$$P_i(s', (h_{-i}, a_{-i}, \omega_{-i})|(h_i, a_i, \omega_i)) = P(a_{-i}|h_{-i}) \sum_{s \in S} P(s', \omega_{-i}|s, \vec{a}, \omega_i) P_i(s, h_{-i}|h_i), \tag{1}$$

  with the base condition for the recursion being $P_i(s, h_\emptyset|h_\emptyset) = \beta(s)$, where $h_\emptyset$ is the empty history.
- Real variables $Q_i^*(h_i, a_i)$: action values of agent i associated with private history h_i, constrained by the following equation

$$Q_i^*(h_i, a_i) = \sum_{s, a_{-i}, h_{-i}} P(a_{-i}|h_{-i}) P_i(s, h_{-i}|h_i) \cdot \left[R(s, \vec{a}) + \sum_{\omega_i \in \Omega_i} P(\omega_i|s, \vec{a}) \max_{b \in A_i} Q_i^*((h_i, a_i, \omega_i), b) \right]. \tag{2}$$

  Note that s on the right-hand side above is the state of the system before the joint action $\langle a_i, a_{-i} \rangle$ is executed, and the related parameters $P(\omega_i|s, \vec{a})$ are explained later in Equation 3.
- The $P(a_i|h_i)$ variables are constrained to be consistent with the Q^*-values so that $\arg\max_{b \in A_i} Q_i^*(h_i, b) \neq a_i \implies P(a_i|h_i) = 0$, and they are further constrained so that exactly one action must be chosen for each history, i.e. $\sum_{a_i \in A_i} P(a_i|h_i) = 1$, since a deterministic optimal policy always exists in Dec-POMDPs.
- The optimization objective: maximize $\sum_{a_i \in A_i} P(a_i|h_\emptyset) Q_i^*(h_\emptyset, a_i)$ for *any* agent i.

The above optimization problem utilizes individual Q-values, unlike *joint* value functions featured in prior work [20, 2]. We establish the soundness of our novel optimization formulation in Appendix A. This optimization corresponds to a game theoretic problem, and any feasible solution gives a Nash equilibrium policy of the agents. The optimal solution gives a Pareto optimal Nash equilibrium.

Solving the above optimization problem will yield optimal policies for the agents via the $P(a_i|h_i)$ variables, as well as the optimal Q-values corresponding to that joint policy. Unfortunately, this is at least as hard as solving the Dec-POMDP (which is NEXP-complete), and furthermore, this optimization problem may have multiple solution sets corresponding to multiple Pareto optimal (Nash equilibrium) policies. The significance of this formulation, however, is in defining a set of target Q-values (Equation 2) to which reinforcement learning (next section) should ideally converge.

2.2 Reinforcement Learning for Dec-POMDPs

Reinforcement Learning (RL) is a family of techniques applied normally to MDPs [27] $\langle S, A, P, R \rangle$ (i.e., Dec-POMDPs with full observability and single agent). When states are visible, the task of an RL agent in a horizon T problem is to learn a *non-stationary* policy $\pi : S \times t \mapsto A$ that maximizes the sum of current and future rewards from any state s, given by $V^\pi(s^0, t) = E_P[R(s^0, \pi(s^0, t)) + \ldots + R(s^{T-t}, \pi(s^{T-t}, T))]$, where $s^0, s^1, \ldots s^{T-t}$ are successive samplings from the distribution P controlling the Markov chain with policy π. A popular RL algorithm for such problems is Q-learning, which maintains an action-quality value function Q given by $Q(s, t, a) = R(s, a) + \max_\pi \sum_{s'} P(s'|s, a) V^\pi(s', t+1)$. This quality value stands for the sum of rewards obtained when the agent starts from state s at step t, executes action a, and follows the optimal policy thereafter. These Q-values can be learned in model-free or model-based manner, with no prior knowledge of R, P being required.

Multi-agent reinforcement learning (MARL) allows multiple agents to perform individual reinforcement learning by simultaneous exploration of a shared environment. There is a large body of work in the field of MARL, but [9] offers a most recent compact survey. The key difficulty of reinforcement learning in a concurrent setting is that it induces a *moving target problem*—the changing behavior of one agent impacts the target model that another

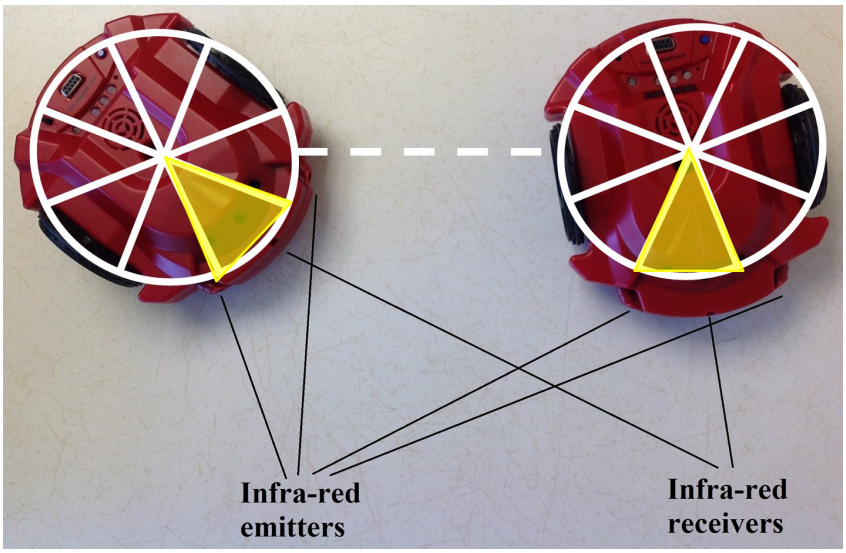

Figure 1: The Scribbler 2 robots featured in the robot alignment problem.

agent wants to learn—and this has been repeatedly acknowledged in the literature [9]. Much of the prior work in MARL addressed fully observable settings where states and joint actions are publicly observed, and the key problem is coordination among the agents [7]. More recently, MARL has been applied to Dec-POMDPs [30, 4], where $Q_i(h_i, a_i)$ values are learned, mapping the history of an agent's own past actions and observations, h_i, and next actions (a_i) to real values. These $Q_i(h_i, a_i)$ values *estimate* the $Q_i^*(h_i, a_i)$ values defined in Equation 2. While [30] assumes perfect communication among teammates in a special class of Dec-POMDPs (called ND-POMDPs), we assume indirect and partial communication among the agents, via a third party observer (but only while learning). Whereas the approach proposed in [4] has a large sample complexity, our results are all based on orders of magnitude fewer episodes in each setting. Furthermore, [4] proposes a turn-taking learning approach where the non-learner's experience is wasted, whereas we propose a concurrent learning algorithm where all samples are used for learning by all agents.

While their assumptions about local information differ—[4] assume that agents only observe their own actions and observations, where [30] assume that agents observe those of their team mates as well—both methods assume that only local information is available during learning. That is, they both assume that the learning must occur under policy execution conditions. However, we argue in the next section that this assumption is not always necessary and relaxing it where possible can improve the learning rate.

3 Reinforcement Learning as a Rehearsal

The goal of this article is to construct, analyze and evaluate a concurrent reinforcement learning algorithm suitable for Dec-POMDPs. That is, the domain of problems is cooperative but partially observable, and the agents must be able to execute learned policies in a decentralized manner without knowing other agents' actions and observations. Thus, this work can be classified as multi-agent reinforcement learning for Dec-POMDPs, which is situated within the broader context of MARL for partially observable stochastic games (POSGs, where agents need not be cooperative). This work specializes MARL for Dec-POMDPs to problems that fit the rehearsal framework introduced next.

There are many scenarios in which the training conditions need not be as strict as execution conditions. Consider, for example, the Robot alignment problem [25] where two robots are required to face each other, based on local observations only. Figure 1 shows an illustration. The relevant actions are the robots' ability to turn, and send and receive infra-red signals. The two possible obsevations are the presence or absence of infra-red reading at a robot's sensor. A robot can only read an infra-red signal if they are aligned and the other is sending a signal.

The problem is to generate decentralized plans that enable the agents to reach alignment (i.e., yellow sectors touching the dotted line simultaneously in Figure 1) and know when it occurs. This problem was introduced in a short paper [15], and more information about this problem can be found at the website `http://www.cs.usm.edu/~banerjee/alignment/`.

Since the Robot alignment problem is a skill learning problem, its dynamics do not depend on the environment. Hence the robots could easily be trained in a laboratory setting, where a computer connected to an overhead camera could relay the states (i.e. orientations) and each others' actions to the robots—information that will be unavailable to the robots in the field when they need to align. Providing robots with this same information in the field would be unwieldy as some sort of third robot with a camera would have to follow the robots around, requiring more coordination than the original problem. Clearly, then, while constraints on execution conditions may be justified, these constraints need not always be applied to the training environment.

To this end, we explicitly break up a problem into distinct "learning" and "execution" phases. We treat the MARL problem as a rehearsal *before* a final stage performance, so to speak. The inspiration comes from real life; while actors train together to produce a coordinated performance on stage, they do not necessarily subject themselves to the same constraints during rehearsal. For instance, actors routinely take breaks, use prompters, receive feedback, practice separately, etc. during rehearsals - things that are not possible during the final stage performance. Similarly in MARL, while the agents must learn distributed policies that can be executed in the target task (the stage of performance), they do not need to be constrained to the exact setting of the target task while learning.

In this article, we assume that agents rehearse under the supervision of a third party observer that can convey to them the hidden state *while they are learning/rehearsing, but not while they are executing the learned policies*. This observer also tells the agents what the other agents' last actions were. However, we assume one-way communication—the agents can receive data from the third party, but not send it any data—so the observer can neither obtain the agents' observations nor can it cross-communicate them at any time. We find such one-way communication in domains such as GPS tracking, where satellites broadcast to GPS units, and multirobotics (for instance, kilobots [12]) where a single overhead controller issues commands and information to all robots simultaneously. We call the extra information available to a learner during rehearsal, $(s \in S, a_{-i} \in A_{-i})$ — i.e, the hidden state and the others' actions — the *rehearsal features*. *The key challenge is that agents still must learn policies that will not rely on these external communications, because the rehearsal features will not be available at the time of executing the policies.* Therefore, the policies returned by our approach must be in the same language as those returned by traditional Dec-POMDP solvers. We call our approach *R*einforcement *L*earning *a*s a *R*ehearsal, or RLaR.

The rehearsal framework induces a novel subclass of MARL where rehearsal features are available for training. It differs from existing work on *assisted* RL. For instance, it differs from *imitation learning* [23] in that the external observer is not a mentor and does not participate in the decision process. This work also differs from the large body of work on *reward shaping* where an agent explicitly engineers the reward function [17, 18] to improve learning speed, or a human feedback is used to construct shaping rewards [13]. RLaR stands alone in solving a novel subclass of general decentralized learning problems.

3.1 A Hybrid Q-function

In this section we define and characterize a *hybrid Q function* for agent i—joint action values given s and its current observable history h_i, i.e., $\widetilde{Q}_i(s, h_i, \vec{a})$—which will be estimated during the rehearsal process. This function relates an agent's observable features (h_i, a_i) with the rehearsal features (s, a_{-i}). Let

$$P_i(\omega_i|s, \vec{a}) = \sum_{s' \in S} O_i(\omega_i|s', \vec{a}) P(s'|s, \vec{a}) \tag{3}$$

be the true probability of agent i observing ω_i after joint action $\vec{a}$ has been executed in state s where $O_i(\omega_i|s', \vec{a}) = \sum_{\omega_{-i}} O(\vec{\omega}|s', \vec{a})$. We define the optimal hybrid Q-function as

$$\widetilde{Q}^*_i(s, h_i, \vec{a}) = R(s, \vec{a}) + \sum_{\omega_i \in \Omega_i} P_i(\omega_i|s, \vec{a}) \max_{a'_i \in A_i} Q^*_i(h'_i, a'_i), \tag{4}$$

where $\vec{a} = \langle a_i, a_{-i} \rangle$, h_i' is the concatenation of h_i with (a_i, ω_i), i.e. (h_i, a_i, ω_i), and $Q_i^*(h_i', a_i')$ is given by Equation 2. $\widetilde{Q}_i^*(s, h_i, \vec{a})$ gives the immediate reward agent i expects the team to receive if joint action $\vec{a}$ is executed in the visible state s, plus the maximum total future reward (i.e., $Q_i^*(h_i', a_i')$) that i expects the team to receive when the next state s' and other agent's next action a'_{-i} *cannot be observed*. Observe that substituting $\widetilde{Q}_i^*(s, h_i, \vec{a})$ in Equation 2, we get

$$\begin{aligned}
Q_i^*(h_i, a_i) &= \sum_{s, a_{-i}, h_{-i}} P^*(a_{-i}|h_{-i}) P_i^*(s, h_{-i}|h_i) \widetilde{Q}_i^*(s, h_i, \vec{a}) \\
&= \sum_{s, a_{-i}, h_{-i}} P_i^*(s, a_{-i}, h_{-i}|h_i) \widetilde{Q}_i^*(s, h_i, \vec{a}) \\
&= \sum_{s, a_{-i}} \widetilde{Q}_i^*(s, h_i, \vec{a}) \sum_{h_{-i}} P_i^*(s, a_{-i}, h_{-i}|h_i) \\
&= \sum_{s, a_{-i}} P_i^*(s, a_{-i}|h_i) \widetilde{Q}_i^*(s, h_i, \vec{a}) \\
&= \sum_{s, a_{-i}} P_i^*(a_{-i}|s, h_i) P_i^*(s|h_i) \widetilde{Q}_i^*(s, h_i, \vec{a}), \qquad (5)
\end{aligned}$$

where the second step utilizes the fact that a_{-i} is conditionally independent of s, h_i given h_{-i}.

3.2 The RLaR Algorithm

An immediate benefit of providing agents with the state, s, and joint action, $\vec{a}$, is that agents can each maintain their own estimates of the transition function $\hat{P}(s'|s, \vec{a})$, the reward function $\hat{R}(s, \vec{a})$, and the initial distribution over states $\hat{\beta} \in \Delta S$. They can also maintain estimates of *individual* (i.e., different for different agents) observation probability function $\hat{O}_i(\omega_i|s', \vec{a})$. All of these estimates are based on relative frequencies as encountered by the RLaR agents; the frequencies are initialized to 0. The agents learn these model estimates *while* also learning their value functions as described below.

Note that such internal models of the environment are incomplete (e.g., *joint* observation distributions are inestimable), therefore standard solvers cannot be invoked on the learned model. In some scenarios, however, the one-way communication assumption may be relaxed and the joint observations may also be easily shared during learning, allowing the full model to be estimated and then, for standard solvers to be invoked in lieu of RLaR. However, we may not wish to do so because, unlike standard solvers, RLaR's computation time can become transparent to the agents in problems where sample collection is time consuming (e.g., a robot executing an action before taking an observation). This is because the agent can perform RLaR updates while executing an action, and by the time it produces a model of some accuracy it also produces a policy without consuming any additional time. If a traditional solver is invoked on this model, it will cost additional time.

In addition to model estimation, providing agents with rehearsal features enables agents to treat the problem as a fully-observable MDP $\langle S, A, \hat{P}, \hat{R} \rangle$, and learn an MDP policy via action-quality values (same for all agents) given by

$$Q(s, t, \vec{a}) = \hat{R}(s, \vec{a}) + \sum_{s' \in S} \hat{P}(s'|s, \vec{a}) \max_{\vec{a'} \in A} Q(s', t+1, \vec{a'}). \qquad (6)$$

Obviously, this MDP policy will not be useful during policy execution as it assumes full observability; however, this policy could be a useful starting point—an idea studied before as *transfer learning*. In particular, *value function transfer* [5] has been used in reinforcement learning to bootstrap an agent's value function in a related task, by initializing it with learned values from a previous task. The key challenge is to map the value functions between the two different tasks [16], which we accomplish for RLaR by mapping $(s, t, \vec{a})$ to $(s, h_i, \vec{a})$ such that $t = |h_i|/2+1$, i.e., history h_i reflects the same decision step of the agent in the Dec-POMDP as t in the underlying MDP. This way, we use learned $Q(s, t, \vec{a})$ values to initialize the hybrid value function $\widetilde{Q}_i(s, h_i, \vec{a})$. Among other work related to this idea, pre-solved MDP (and POMDP) policies have been used as search heuristics for Dec-POMDP solvers before [21].

We break the rehearsal phase into 2 successive stages, where the MDP policy is learned as above in the first (model based RL) stage. Since the agents perceive identical $s, t, \vec{a}, r, s'$ at all steps of this stage, they learn identical $Q(s, t, \vec{a})$ values. However, agents explore independently by picking their portions of $\vec{a}$ that maximizes some objective based on the $Q(s, t, \vec{a})$ values. If the maximizing $\vec{a}$ is non-unique, the agents' choices may not constitute a maximizing joint action. Thus in the first stage the agents learn the $Q(s, t, \vec{a})$ values of actual $\vec{a}$ that they execute, whether these joint actions are coordinated or not.

In the second (multi-agent learning) stage, each agent i learns the hybrid Q-function as follows: First it uses MDP Q values (from Equation 6) to initialize its $\widetilde{Q}_i(s, h_i, \vec{a})$ values, as $\widetilde{Q}_i(s, h_i, \vec{a}) \leftarrow Q(s, t, \vec{a})$, for $t = |h_i|/2 + 1$. Then the agent updates the $\widetilde{Q}_i(s, h_i, \vec{a})$ values by estimating the right-hand side of Equation 4: $R(s, \vec{a})$ by $\hat{R}(s, \vec{a})$, and $P(\omega_i|s, \vec{a})$ by $\hat{P}(\omega_i|s, \vec{a})$ (derived by substituting $\hat{O}_i$ for O_i and $\hat{P}(s'|s, a)$ for $P(s'|s, a)$ in Equation 3). It leverages Equation 5 to estimate the $Q_i^*(h_i, a_i)$ values needed for the right-hand side of Equation 4 as

$$Q_i(h_i, a_i) = \sum_{s \in S} \sum_{a_{-i} \in A_{-i}} \hat{P}_i(a_{-i}|s, h_i) \hat{P}_i(s|h_i) \widetilde{Q}_i(s, h_i, \vec{a}), \tag{7}$$

where the $\hat{P}_i$'s are i's estimates of the respective probability terms in Equation 5. These estimates essentially stand for i's *belief*s about the rehearsal features after observing h_i. Such prediction of rehearsal features based on observable history is key for the agent to achieve independence from the rehearsal features for the policy execution phase. Note that unlike in the first stage, the agents' Q-values depend on private information, h_i, and hence can vary from agent to agent. Essentially, the agents learn to coordinate with each other in the second stage which is a difficult problem in the presence of private information.

Each agent maintains a sliding window over (s, a_{-i}) samples for each history h_i and uses these samples to estimate $\hat{P}_i(a_{-i}|s, h_i)$ via

$$\hat{P}_i(a_{-i}|s, h_i) = \frac{N_{h_i}(s, a_{-i})}{\sum_{a_{-i}} N_{h_i}(s, a_{-i})}, \tag{8}$$

where $N_{h_i}(s, a_{-i})$ gives the current windowed count of (s, a_{-i}) samples associated with history h_i. By discarding older samples, agents are able to adapt more quickly to changes in other agents' policies during learning. The size of the window for history h_i is set to $\frac{E}{\tau |\Omega_i|^{|h_i|/2}}$, where E is the total number of learning episodes, and τ is the number of times we wish the windows to be completely flushed during the E episodes. Since longer histories are expected to be encountered less often, the variable window size allows the windowed estimates to be roughly equally responsive to changes in the others' policies for different histories.

To make more efficient use of sample information, each agent i *propagates* the belief $\hat{P}_i(s|h_i)$ using Bayesian belief propagation via

$$\hat{P}_i(s'|h_i') = \alpha \cdot \hat{O}_i(\omega_i|s', a_i) \sum_{s \in S} \hat{P}_i(s|h_i) \cdot \sum_{a_{-i} \in A_{-i}} \hat{P}(s'|s, \vec{a}) \hat{P}_i(a_{-i}|s, h_i), \tag{9}$$

where $h_i' = (h_i, a_i, \omega_i)$ and α is a normalization factor. These updates leverage the agent's estimates of the Dec-POMDP model, which can improve belief estimates because while the true $P_i(a_{-i}|s, h_i)$ values change as the agents learn, the underlying Dec-POMDP model is stationary.

The RLaR algorithm is outlined in Algorithm 1. Each agent executes a separate instance of RLaR, but concurrently with other agents. EXECUTEACTION returns the next state s', the last joint action $\vec{a}$, the agent's own observation ω, and the reward r. This feedback includes rehearsal features, and comes from the third party observer. Upon receiving this feedback, each agent updates its model $\langle S, A, \hat{P}, \hat{R}, \Omega_i \rangle$ as well as its model of the other agent's action $\hat{P}_i(a_{-i}|s, h_i)$. In the first stage agents do not require a model of other agent's actions; however, we have found it useful to sample i's estimate of $P_i(a_{-i}|s, t)$ from stage 1 to replace missing $\hat{P}_i(a_{-i}|s, h_i)$ values in stage 2. UPDATEQSTAGEI$(s, t, \vec{a})$ implements Equation 6, while UPDATEQSTAGEII$(s, h_i, \vec{a})$ implements the other equations in the previous paragraphs. Specifically, when an agent i executes UPDATEQSTAGEII$(s, h, \vec{a})$, it recalculates $\widetilde{Q}_i(s, h_i, \vec{a})$ and $Q_i(h_i, .)$ using its updated estimates of the (partial) Dec-POMDP model and $\hat{P}_i(a_i|s, h_i)$ (from the UPDATEMODEL step). In our experiments, we use an upper-confidence bound (UCB) [3] approach for SELECTACTIONMDP; however, other action selection approaches might be applied successfully too. The function SELECTACTION is described in the next section.

Algorithm 1 RLAR($mdp\_episodes, total\_episodes$)

1: **for** $m = 1 \ldots total\_episodes$ **do**
2: $s \leftarrow$ GETINITIALSTATE()
3: $h \leftarrow \emptyset$
4: **for** $t = 1 \ldots T$ **do**
5: **if** $m \leq mdp\_episodes$ **then**
6: $a \leftarrow$ SELECTACTIONMDP(s, t)
7: $(s', \vec{a}, \omega, r) \leftarrow$ EXECUTEACTION(a)
8: UPDATEMODEL$(s, \vec{a}, s', \omega, r)$
9: UPDATEQSTAGEI$(s, t, \vec{a})$
10: **else**
11: $a \leftarrow$ SELECTACTION(h)
12: $(s', \vec{a}, \omega, r) \leftarrow$ EXECUTEACTION(a)
13: UPDATEMODEL$(s, \vec{a}, s', \omega, r)$
14: UPDATEQSTAGEII$(s, h, \vec{a})$
15: **end if**
16: $s \leftarrow s'$
17: $h \leftarrow (h, a, \omega)$
18: **end for**
19: **end for**

In domains where sampling is time consuming (step 12 of RLaR), the UPDATE steps could be performed concurrently with EXECUTEACTION by processing the output of the previous call to EXECUTEACTION. This would introduce a 1-step lag, but is unlikely to impact performance.

Figure 2 shows a high level block diagram of RLaR. One of the emphases of this figure is the distinction between the purposes of the two stages: in stage 1 the agents simply acquire estimates of the invisible model, treating the combination of agents as a (virtually) single agent; by contrast in stage 2 they correlate the (individually) visible features with the hidden features of the environment as well the choices of the other agents. Hence stage 2 captures the aspect of multi-agency of RLaR. An additional distinction is that the agents sometimes select actions that improves their individual knowledge of these correlations, in stage 2.

3.3 Exploration

A common method of action selection in reinforcement learning is the ϵ-greedy approach [27], in which an agent chooses a random action with probability ϵ, and with probability $(1 - \epsilon)$ returns $\arg\max_{a_i} Q_i(h_i, a_i)$. However, since the RLaR agents intend to output policies that are independent of the rehearsal features, there is a principled incentive to explore actions that help predict the rehearsal features. Thus, we propose an additional exploration criterion that values information gain rather than expected reward.

Having observed history h_i, a learner can pick, with some probability, an action that has the highest expected *information gain* about the current rehearsal features. To do this, it uses a distribution over the rehearsal features $\eta_i(s, a_{-i}|h_i, a_i, \omega_i)$, where η_i is i's *posterior* belief in what s, a_{-i} might be at history h_i, *assuming* (hypothetically) that action a_i is executed at this history and observation ω_i is received as evidence. $\eta_i(s, a_{-i}|h_i, a_i, \omega_i)$ is estimated by backward propagation as $\alpha\hat{P}_i(\omega_i|s, \vec{a})\hat{P}_i(s|h_i)\hat{P}_i(a_{-i}|s, h_i)$, where α is a normalization factor. Then, the entropy of the distribution η is given by

$$E_{\omega_i}(h_i, a_i) = -\sum_{s, a_{-i}} \eta_i \log \eta_i.$$

Finally, the exploration action, $a_{explore}$, is picked as

$$a_{explore} = \arg\min_{a_i \in A_i} \sum_{\omega_i \in \Omega_i} P(\omega_i|h_i, a_i) E_{\omega_i}(h_i, a_i), \tag{10}$$

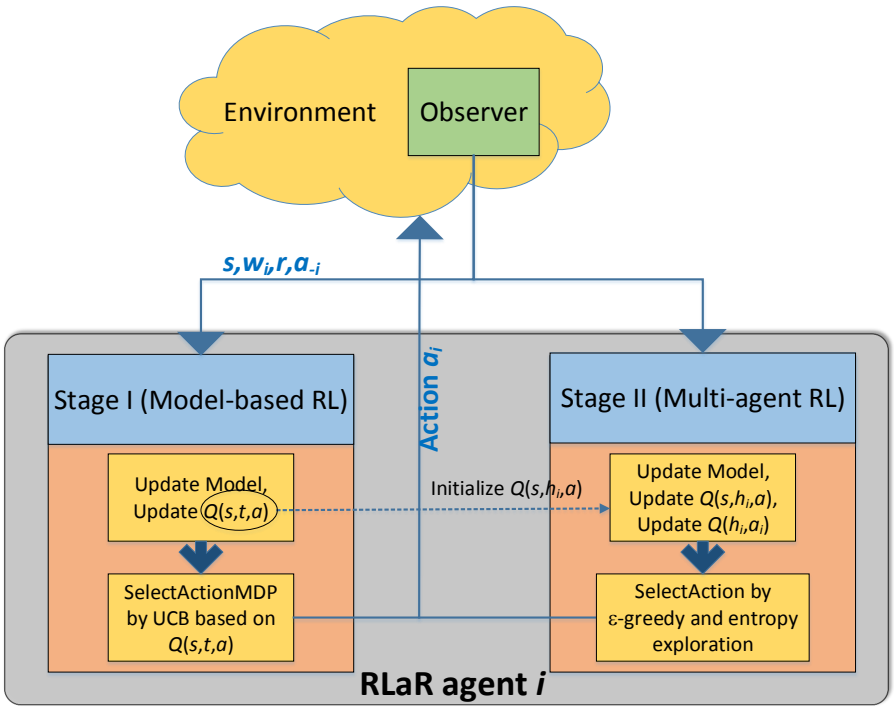

Figure 2: A high level view of the RLaR algorithm.

where $P(\omega_i|h_i, a_i) = \sum_{s,a_{-i}} P(\omega_i|s, \vec{a})P_i(s, a_{-i}|h_i)$ is simply estimated by $\sum_{s,a_{-i}} \hat{P}_i(\omega_i|s, \vec{a})\hat{P}_i(s|h_i)\hat{P}_i(a_{-i}|s, h_i)$ based on estimates of Equations 3, 9, and 8. Equation 10 measures the *expected* entropy over the observation distribution, accounting for the fact that action a_i and observation ω_i are hypothetical. We call this exploration criterion *entropy based exploration*. In each action selection cycle, a RLaR agent picks an action according to the entropy based exploration scheme with some probability ϵ', and with probability $(1 - \epsilon')$ it performs ϵ-greedy exploration.

The motivation behind entropy exploration springs uniquely in our rehearsal based framework, and is not a feature of standard reinforcement learning. A relevant question then, is whether this new kind of exploration might detract from the main objective of reinforcement learning. The most important observation in this context is that the first stage of rehearsal already trains the agents on the relation between the environmental (hidden) dynamics and the various actions that are needed to improve the agents' models and performance. Hence, the entropy exploration in the second stage only serves as a bridge between the agents' observations and the hidden dynamics. Being performed with only some probability, it does not preclude the execution of actions that are needed to improve the agents' models and performance.

3.4 Algorithmic Complexity

Each UPDATEQSTAGEI costs $O(|S||A|)$ time to compute Equation 6, whereas UPDATEMODEL costs a constant amount of time. The time complexity of EXECUTEACTION depends on the task, and is assumed to also be a constant. The update of $\widetilde{Q}_i(s, h_i, \vec{a})$ requires $O(|S||\Omega_i||A_i|)$ time, and the update of $Q_i(h_i, a_i)$ requires $O(|S|^2|A_{-i}|^2)$ time. Therefore, UPDATEQSTAGEII requires $O(|S|^2|A|^2|\Omega_*|)$ time, where $|\Omega_*|$ is the size of the largest observation space among all agents. Finally, while SELECTACTIONMDP costs $O(|A|)$ time, the entropy exploration used in SELECTACTION costs $O(|S||\Omega_i||A|)$ time. This gives an overall algorithmic time complexity of $O(ET|S|^2|A|^2|\Omega_*|)$ for each RLaR agent learning concurrently, where E is the total number of episodes input to Algorithm 1.

Since $|A|$ is the size of the joint action space, the per agent complexity of RLaR is still exponential in the number

of agents, n. The algorithmic complexity gives us an idea of the size of the model learned by the RLaR agents. In the case of sample based reinforcement learners, there is another type of complexity that is also relevant: *sample* complexity, i.e., the number of samples needed to achieve high enough quality of output policies with a high probability. The algorithmic complexity is usually unrelated to the sample complexity. In this article, we only study the sample complexity of RLaR empirically in Section 5.

4 Theoretical Analysis

In this section we establish a weak convergence proof for RLaR, in Q-values. Inspired by the action-gap regularity criterion introduced recently in RL [11], we introduce a similar criterion that aids in the proof of convergence of RLaR. We call the new criterion *minimal action gap* (MAG), and define it in the context of the Q-functions of Equation 2 as below:

Definition 1 (MAG) *Let $P_i^*(h_i)$ be the relative frequency of agent i encountering history h_i when a T-step optimal joint policy, $\vec{\pi}^*$, is executed infinitely often. Then the action-gap function under $\vec{\pi}^*$ for agent i is defined as $G_i(h_i) = |Q_i^*(h_i, 1) - Q_i^*(h_i, 2)|$, where 1 and 2 represent the best and the second best action at h under $\vec{\pi}^*$. The MAG criterion requires that there exists a $\epsilon_0 > 0$ such that $P(G_i > 2\epsilon_0) = \sum_{h_i} P_i^*(h_i) I(G_i(h_i) > 2\epsilon_0) = 1, \forall i$, where I is the indicator function. Note that the criterion also holds $\forall \epsilon \leq \epsilon_0$.*

The original proposal of action-gap regularity [11] required a polynomial upper bound on $P(0 < G_i \leq \epsilon)$ in order to establish a contraction operation in value updates for single agent RL. We do not believe such strong results are possible in a multi-agent RL context due to the moving target problem, hence we set a more modest objective of proving the convergence of RLaR *in probability*. The above MAG criterion is useful for this purpose.

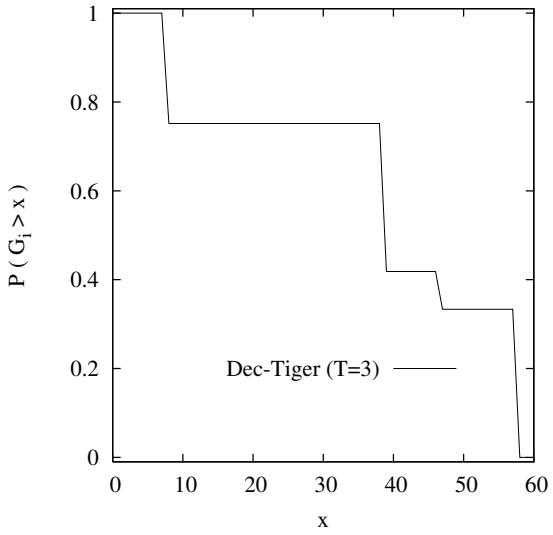

Figure 3: Plot of $P(G_i > x)$ vs. x for Dec-Tiger ($T = 3$) corresponding to the optimal (Nash equilibrium) joint policy.

Figure 3 demonstrates that the MAG criterion is indeed satisfied in Dec-Tiger, for one of the agents for any $\epsilon_0 \in [0, 3.6717]$. Due to symmetry, the same range, and hence the MAG criterion, will also hold for the other agent.

We rely on two supremum norms for our proof of convergence, $\|\mathcal{Q}^t - \mathcal{Q}^*\|$ and $\|\hat{P}^t - P^*\|$ in the learning episode t, where P^* is a marginalization over the optimal (under $\vec{\pi}^*$) $P^*(a_{-i}|h_{-i})P_i^*(s, h_{-i}|h_i)$ values given in Equation 1, i.e.

$$P_i^*(s, a_{-i}|h_i) = \sum_{h_{-i}} P^*(a_{-i}|h_{-i}) P_i^*(s, h_{-i}|h_i).$$

$\hat{P}_i^t(s, a_{-i}|h_i)$ represents agent i's estimate after t updates. Note that in our proof of Theorem 1 (in the Appendix B), we assume $\hat{P}_i^t(s, a_{-i}|h_i)$ is accurate with respect to the current joint policy $\vec{\pi}^t$, thus $\vec{\pi}^t = \vec{\pi}^* \implies \hat{P}_i^t(s, a_{-i}|h_i) = P_i^*(s, a_{-i}|h_i)$. If agents update at sufficiently large intervals, while disabling Q-Updates in the interim, these values will converge to the true values for a given $\vec{\pi}^t$. In practice, however, we do not take this approach (instead using the sliding window estimation approach outlined in Section 3.2) because it may slow the learning rate.

Definition 2 (Max-norm Q-error) *Let $\|\mathcal{Q}^t - \mathcal{Q}^*\|$ be the supremum of the set $\bigcup_i^n \{|\widetilde{Q}_i^t(s, h_i, \vec{a}) - \widetilde{Q}_i^*(s, h_i, \vec{a})| : \forall s, h_i, \vec{a}\} \cup \{|Q_i^t(h_i, a_i) - Q_i^*(h_i, a_i)| : \forall h_i, a_i\}$.*

Definition 3 (Max-norm Belief Error) *Let $\|\hat{P}^t - P^*\|$ be the supremum of the set $\bigcup_i^n \{|\hat{P}_i^t(a_{-i}, s|h_i) - P_i^*(a_{-i}, s|h_i)| : \forall h_i, s, a_{-i}\}$.*

Theorem 1 *Under the MAG assumption, as well as the following assumptions:*

1. *The Dec-POMDP only has a unique Nash equilibrium joint policy, which is then also the optimal joint policy;*
2. *t is large enough that the model estimates are nearly accurate, i.e., $\hat{R} \approx R$ and $\hat{P}(\omega_i|s, \vec{a}) \approx P(\omega_i|s, \vec{a}) \forall i$, and any error in these estimates is no larger than δ and $\frac{\delta}{(\max_{s,\vec{a}} R(s,\vec{a}) - \min_{s,\vec{a}} R(s,\vec{a}))|\Omega_i|T}$ respectively, where δ is negligibly small;*
3. *because t is large, the agents have ceased exploration;*
4. *$P(\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon) < 1$,*

we have $P(\|\mathcal{Q}^{t+1} - \mathcal{Q}^*\| \leq \epsilon) > P(\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon)$, for any $\epsilon \leq \epsilon_0$ (proof is given in Appendix B).

The theorem implies that the likelihood of the Q-errors falling below a threshold strictly increases with more episodes. In other words, the Q-values are guaranteed to become sufficiently (relative to a chosen ϵ) accurate over time, *in probability*. While assumptions 2–4 are used in the technical proof, without assumption 1 there would not be a unique target Q^*, and the theorem could simply mean the Q-errors move back and forth among multiple targets without approaching any in the long term.

The theorem does not guarantee that the Q values will converge to the unique optimal values. It only says that the likelihood of that event will strictly improve with increasing experience. Note that this does not mean that the likelihood will become arbitrarily close to 1 (or even approach it), even in the limit. To see this, consider some point $t = \tau$ (say 20), when the probability (p_τ, say 0.3) starts increasing at the rate $\Delta_t = 1/2^t$. Then even in the limit, the probability will not exceed $p_\tau + 1/2^{\tau+1}$ (here 0.300000477). The remark below illustrates why it is difficult to provide stronger guarantees in these domains.

According to Equation 5, the $Q(h, a)$ values of an agent depends on the policies of other agents. Therefore, it is possible for the $Q(h, a)$ values to have multiple (local) optima even when the Dec-POMDP has a unique Nash equilibrium joint policy. However, note that an agent is unlikely to get stuck at such a local optimum Q value because these values are defined by the *current policy* of the other agents; if the agents have not reached the Nash equilibrium, then some agent will very likely change its policy, redefining the Q value function for the said agent. Thus local optima of $Q(h, a)$, if they exist, are volatile. Only in the (highly improbable) event that there exists a joint policy profile which defines a local optimum of every agent's $Q(h, a)$ function simultaneously, and that every agent simultaneously reaches this policy, will they be unable to converge to the unique (globally optimal) Nash equilibrium policy. Note that the probabilistic guarantee of the theorem does not contradict the above scenario, as discussed in the previous paragraph.

Problem	Horizon	Time (secs)	Known Optimum	RLaR	RLaR Without Stage 1	RLaR Without Entropy Exploration
DecTiger	3	7.38	5.19	5.19	2.07	5.19
	4	14.13	4.8	4.46	2.95	3.999
	5	33	7.03	6.65	3.84	5.97
Box Pushing	2	331	17.6	17.6	1.04	17.6
	3	5656	66.08	66.08	13.90	66.08
	4	38841	98.59	98.59	49.98	98.59
Alignment	3	75.74	9.62	6.468	-3	6.468
	4	146	21.25	19.36	-3.37	17.46
	5	263.34	32.87	28.24	-4.36	29.09

Table 1: Average policy value for RLaR after 60,000 episodes, along with those of two variants of RLaR: one without stage 1, and another without the entropy exploration. Known optimal values [25] are shown alongside for reference. Runtimes (in secs) are also shown for 60,000 episodes.

5 Experiments

We evaluated the performance of RLaR on three benchmark problems: DecTiger, Cooperative Box Pushing, and Robot Alignment [25]. The DecTiger and Cooperative Box Pushing problems are extremely challenging, and have only been exactly solved up to horizons 6 and 4 respectively [1]. The Robot Alignment problem is a particularly challenging problem from the perspective of explorative learning, such as reinforcement learning, because it features a *combination lock* problem—the robots must execute a series of coordinated actions in the proper sequence to just achieve the goal reward in order to learn that it is worthwhile. Note that combination-lock kind of problems are not necessarily challenging from the perspective of planning which has access to the comprehensive model.

For each domain and horizon value T, we evaluated the performance of RLaR over 20 runs of 100,000 episodes each, using a combination of entropy-based exploration (equation 10) with a probability of 0.25 and ϵ-greedy exploration with $\epsilon = 0.005$. For the Cooperative Box Pushing domain we allocated 10,000 of the episodes to stage 1, and for the others we used 5000 stage 1 episodes. Note that we chose to use a larger number of stage 1 episodes for Cooperative Box Pushing because the state space is larger and requires more episodes to explore. The impact of these stage 1 episodes can be seen in Table 1, which depicts the average policy values after 60,000 episodes with and without stage 1. In all cases, using stage 1 produces better policies on average than not using it at all. It should be noted, however, that for domains where information gathering is important, such as DecTiger, the MDP policy, which assumes full observability, may be at odds with the optimal Dec-POMDP policy and thus transfer may have a negative impact. Indeed, our experiments have indicated that RLaR performs slightly better for DecTiger with fewer stage 1 episodes (~ 500) than we have used here; however, in cases such as these, entropy exploration serves to offset to some extent the negative impact by encouraging agents to take actions which improve their state belief. Table 1 also shows RLaR without the entropy exploration, i.e., using only ϵ-greedy exploration. Although the averages of RLaR are slightly better in many settings, the differences are not statistically significant in any of the settings. Furthermore, Table 1 shows the runtime (in seconds) of RLaR in these benchmark problems, run on a cluster with 12 Dell M610 Blades, each having 8 Intel Xeon 3.00GHz processors with 12GB RAM shared amongst them. Note that the concurrency of the cluster was utilized only to achieve multiple runs on multiple settings simultaneously, but not to simulate concurrent learning by the agents. Therefore the reported times will effectively be halved (since all experiments were performed on 2-agent benchmarks) to get the individual agent learning times.

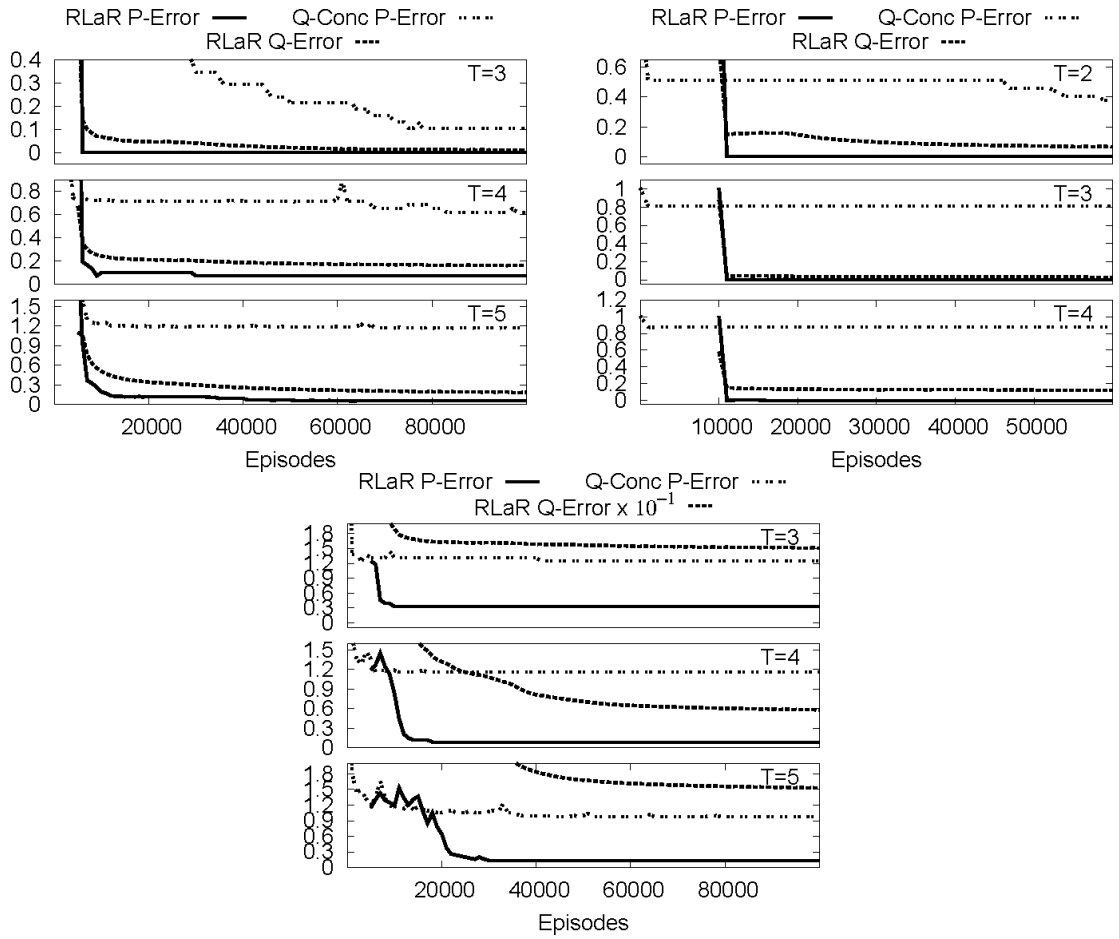

Figure 4: Policy and Q errors vs. episodes, given as "P-Errors" and "Q-Errors" respectively. Left: DecTiger; Middle: Cooperative Box Pushing; Right: Robot Alignment.

5.1 Convergence of Q-values (Q-Error)

Theorem 1 assumes that exactly one set of Q^* (i.e. a unique optimum) policy exists. However, problems such as Robot Alignment have multiple optimal policies, so we also evaluate RLaR empirically by comparing the learned Q-Values $Q_i(h_i, a_i)$ to the ideal $Q_i^*(h_i, a_i)$ values described in Equation 2. Despite the existence of multiple solution sets, in general, to the optimization problem outlined there, there *is a unique* set of Q_i^* values if we are given an optimal policy $\{\pi_i^*\}$ in the form of optimal $P(a_i|h_i)$ values for all i. These values can be substituted in Equations 2, 1, to get the unique set of Q^* values. Since we evaluate RLaR in problems for which optimal policies are already known from exact solution methods, (some of) these $P(a_i|h_i)$ values (and hence Q^* values) are readily available, without solving any optimization problem.

However, these $P(a_i|h_i)$ values are only available for those histories that are consistent with known optimal policies. For example, if the optimal policy π_i^* dictates that agent i executes a particular action a at the first step, then all histories beginning with another action, say b, will never occur during execution of π_i^*. If a given history h_i is inconsistent with π_i^*, $Q_i^*(h_i, \cdot)$ will be undefined. Hence, we compare the set of Q-values only for those histories which are consistent with the known optimal policy. Some domains, such as the Robot alignment and GridSmall have multiple optimal policies, which necessarily have different sets of consistent histories, and so, in order to measure convergence more accurately, we compared to the optimal policy that produced the lowest Q-error. In Figure 4, we report this value as $Q\text{-}Error = \sum_{h_i \in H^C} \frac{|Q_i^*(h_i, a_i^*) - Q_i(h_i, a_i^*)|}{|H^C|}$, where H^C is the set of histories consistent with the comparison policy π_i^* and a_i^* for a given history h_i is $\pi_i^*(h_i)$.

[1]However, bounded approximations are currently available up to horizon 10 for both problems.

From the RLaR Q-Error plots in Figure 4, we can see that RLaR converges rapidly (usually in fewer than 20000 episodes) resulting in low Q-error for most domains and settings (excepting Robot Alignment), suggesting that the learned Q-Values indeed converged to values close to the ideal Q^* values. Robot Alignment Q-errors were relatively high, but rather than being the result of divergence it is the result of convergence to suboptimal policies. Even when the expected value of a suboptimal policy is relatively close to the expected value of the optimal policy, the corresponding sets of consistent histories may be radically different between the two policies, resulting in larger Q-errors.

5.2 Quality of Learned Policies (P-Error)

$Q\text{-}Error$s only indicate the quality of the learner's value function for the histories in H^C. However, a learner potentially learns Q-values for a much larger set of histories, H^L, and poor learning on $H^L - H^C$ can produce poor policies even when $Q\text{-}Error$s are low. Therefore, it is important to also evaluate the quality of the *policies* actually produced by RLaR, relative to the optimal policies that are known. In order to measure the policy quality for RLaR after each episode, we found the error of agents' current policy value relative to the known optimal policy value (the comparison policy), i.e. $\frac{|v_{pol} - v_{opt}|}{|v_{opt}|}$. We refer to this measure in Figure 4 as the "Policy error" or "P-Error".

For comparison purposes, we also report the performance of concurrent Q-Learning without rehearsal, i.e. agents that learn $Q_i(h_i, a_i)$ using only local information. We refer to this setting as "Q-Conc". No stage 1 episodes were allocated for Q-Conc because an MDP policy cannot be learned without access to the rehearsal features. Furthermore, while an initialization phase for Q-Conc was studied in [14], it was unclear that this initialization improved results, thus we used default initialization. We use ϵ-greedy exploration for Q-Conc with $\epsilon = 0.05$ which gave the best results for Q-Conc.

From the RLaR P-Error and Q-Conc P-Error plots in Figure 4 we can see that RLaR converges to near optimal (< 0.1 relative policy error) for most domains and horizons studied. Q-Conc performs much worse than RLaR in all domains, which highlights the impact of incorporating non-local information into the learning process, i.e., the efficacy of rehearsal based learning.

5.3 Comparison with Dec-RSPI

Here we compare RLaR with the decentralized rollout sampling policy iteration (Dec-RSPI) algorithm proposed by Wu et al. [29]. Dec-RSPI is an approximate Dec-POMDP solver capable of scaling to large horizons (100+). Other scalable approximate methods for Dec-POMDP solution exist (e.g. PBIP-IPG [1]); however, we find Dec-RSPI to be a more appropriate target for comparison because it, like RLaR, does not assume full a priori knowledge of the model. While Dec-RSPI does not assume the model is known a priori, it assumes access to a simulator that can be used to generate samples. Since RLaR agents learn Dec-POMDP policies by essentially *sampling* from the environment (and from the third party observer), let us distinguish this from querying a simulator (as Dec-RSPI does). When agents must interact with the environment to learn, they only receive samples that correspond to the actual hidden states visited and joint actions executed at those states. Dec-RSPI uses rollout evaluation to estimate the value of joint policies, and this requires that a given state s and joint action $\vec{a}$ can be sampled as many times as desired. Agents in RLaR, on the other hand, must actually reach s and execute $\vec{a}$ to obtain a sample for that pair. This is an important distinction because some states may require significant coordination to reach. Where RLaR agents must actually coordinate to sample these states, Dec-RSPI assumes it is possible to sample them with no coordination. Furthermore, Dec-RSPI assumes that the observations and policies of other agents are known during policy computation, where agents in RLaR only have knowledge of other agents' actions during learning. Thus, RLaR solves a more difficult problem than Dec-RSPI.

For Dec-RSPI, we used the code provided by the authors, and essentially used the same configuration from [29] but evaluated over an increasing set of values for the sampling constant K. As K increases, the total number of samples increases. The smallest value of K evaluated was 20, which was the value of K used by Wu et al., so our results include the exact configuration studied there.

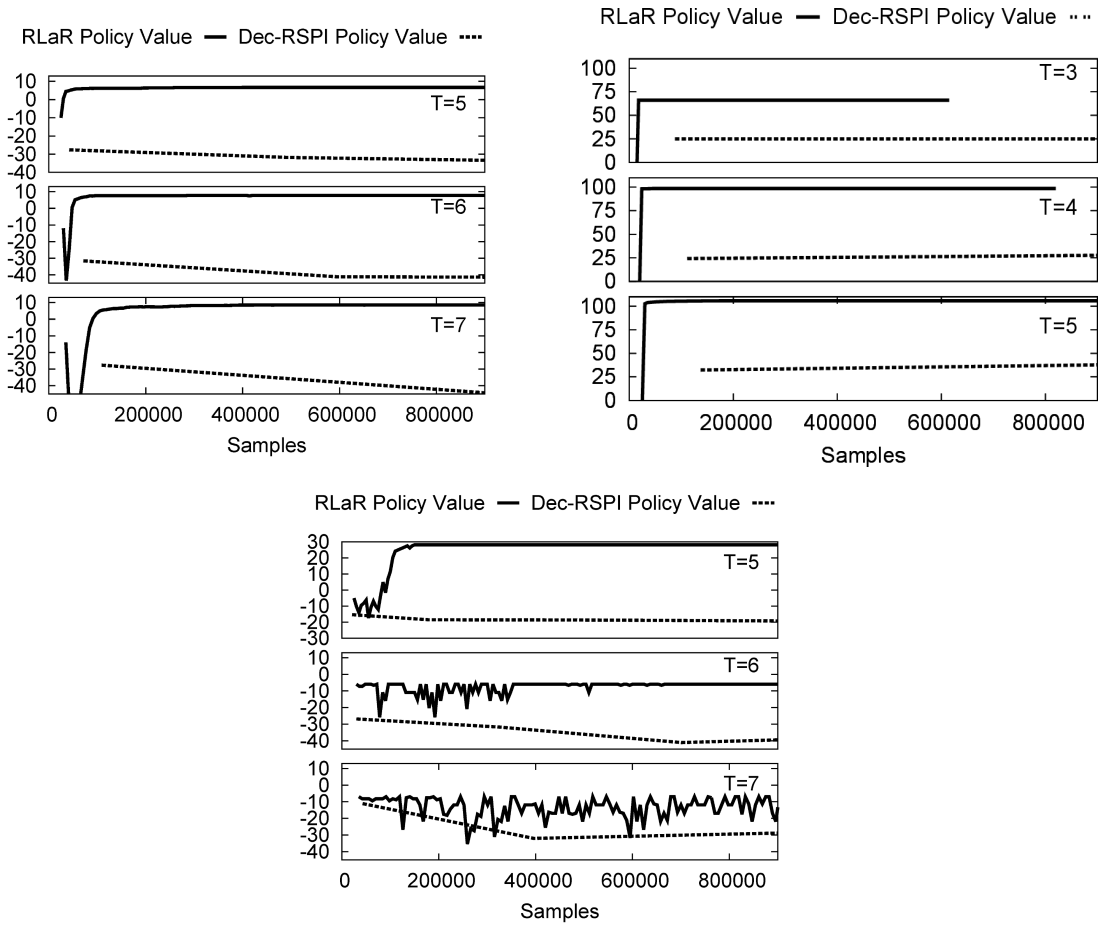

Figure 5: Policy values of RLaR and Dec-RSPI vs. number of samples. Left: DecTiger; Middle: Cooperative Box Pushing; Right: Robot Alignment.

Figure 5 gives the average policy values vs the total number of *samples*. Note that the decrease in Dec-RSPI's average policy value as samples increase in DecTiger is likely due to the fact that agents perform alternating maximization—each agent's policy is optimized while holding other agents' policies fixed—with random initial policies. There are many poor policies in the space of DecTiger policies (for example, a policy in which an agent opens a door at every step), and alternating maximization often leads to convergence to poor local optima. Those runs which use fewer samples have a greater chance to avoid converging to this local optima because agents may fail to compute the best response to these poor policies.

RLaR clearly outperforms Dec-RSPI for problems and horizons shown. Note that while RLaR outperforms Dec-RSPI for these horizons, Dec-RSPI is able to scale to much larger horizons due to its scalable policy representation based on finite state controllers which only grow linearly with the horizon. However, the results of this section suggest that Dec-RSPI's scalability comes at a potentially significant cost in terms of policy quality.

6 Conclusions and Future Direction

We have presented a novel reinforcement learning approach, RLaR with a new exploration strategy suitable for partially observable domains, for learning Dec-POMDP policies when agents are able to rehearse using information that will not be available during execution. We have shown that RLaR can learn near-optimal policies for some existing benchmark problems, with a low sample complexity, and that the Q-Values learned by RLaR tend to converge to ideal Q^*-Values. We have also established a weak convergence proof for RLaR, demonstrating that

RLaR converges *in probability* when certain conditions are met. Stronger convergence results, such as *probably approximately correct* (PAC) may be possible in the future. While we have developed the framework of rehearsal based RL in Dec-POMDPs, it can also be applied to POMDPs (i.e., Dec-POMDPs with $n = 1$ agent).

The goal of this article was to establish the efficacy of our rehearsal based framework, not to assess its impact on practical, large sized problems. This is why we limited the evaluation to small, fully solved benchmark problems, particularly our new Robot Alignment benchmark problem designed to be challenging to RL based approaches. This enabled us to assess (and compare with existing approaches) how close to optimal can our approach get, and how fast, in different kinds of benchmark problems. The insights thus obtained, would not have been possible if we evaluated our approach on a large realistic problem domain—such as the multi-robotic rescue domain [8] with large state, action and observation sets—where it would have been infeasible to compute the optima. However, even in the absence of optimal solutions, it is realistic to expect that the comparative advantages of RLaR over other algorithms would carry forward to such domains. Evaluating RLaR's performance in such large realistic problems would be an important future direction. Another interesting extension of the present work could consider a third party observer that is constrained by partial observability itself.

The assumption of a third party observer may be difficult to contemplate in some domains, e.g., the DecTiger domain. The tiger's location is the crux of this problem, and its narrative is built around the unobservability of this feature. Therefore, assuming an omniscient observer, even if only for the rehearsal phase, appears to defeat the narrative. We note that this fairly abstract domain was chosen because of its hardness, and we do not claim that an observer may be injected into *every* Dec-POMDP. We believe, however, that there are many practically interesting Dec-POMDPs, such as the robot alignment problem and other skill learning tasks, where such an observer can be constructed in a laboratory setting for training robots to interoperate. In some cases, a simulator may even be available for training a team of robots, in which case the simulator itself serves as the omniscient observer. RLaR provides a foundational mechanism for concurrent reinforcement learning in these kinds of applications.

While RLaR performs reasonably well for the benchmark horizons explored here, it lacks the scalability of approximate algorithms such as Dec-RSPI. This is due to RLaR's dependence upon learning $Q_i(h_t, a)$ and $Q_i(s, h_t, \vec{a})$ values (the set of individual action-observation histories is potentially exponential in the horizon T). Exact solution methods [22] have dealt with this exponential growth by *clustering* histories (but require model knowledge and centralized computation to do so). Developing a clustering scheme compatible with the RLaR agents' limited information sets, or a scalable policy representation, may also be useful avenues for future work.

7 Acknowledgments

We thank the anonymous reviewers for detailed and helpful comments and suggestions. This work was supported in part by U. S. Army grant #W911NF-11-1-0124, and National Science Foundation grant #IIS-1526813.

References

[1] Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon Dec-POMDPs. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS), Thessaloniki, Greece, September 19-23*. AAAI, 2009.

[2] Raghav Aras and Alain Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *JAIR*, 37:329–396, 2010.

[3] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[4] Bikramjit Banerjee, Jeremy Lyle, Landon Kraemer, and Rajesh Yellamraju. Sample bounded distributed reinforcement learning for decentralized POMDPs. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, pages 1256–1262, Toronto, Canada, July 2012.

[5] Bikramjit Banerjee and Peter Stone. General game learning using knowledge transfer. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 672–677, Hyderabad, India, 2007.

[6] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

[7] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of 6th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210, 1996.

[8] Lucian Busoniu. The MARL Toolbox version 1.3, 2010. `http://busoniu.net/repository.php`.

[9] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Systems, Man and Cybernetics, Part C*, 38(2):156–172, 2008.

[10] Jilles S. Dibangoye, Abdel-Illah Mouaddib, and Brahim Chai-draa. Point-based incremental pruning heuristic for solving finite-horizon Dec-POMDPs. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-09)*, pages 569–576, Budapest, Hungary, 2009.

[11] Amir Massoud Farahmand. Action-gap phenomenon in reinforcement learning. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 172–180, 2011.

[12] K-Team. Mobile robotics: Kilobots. `http://www.k-team.com/mobile-robotics-products/kilobot`.

[13] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.

[14] Landon Kraemer and Bikramjit Banerjee. Informed initial policies for learning in Dec-POMDPs. In *Proceedings of the AAMAS-12 Workshop on Adaptive Learning Agents (ALA-12)*, pages 135–143, Valencia, Spain, June 2012.

[15] Landon Kraemer and Bikramjit Banerjee. Concurrent reinforcement learning as a rehearsal for decentralized planning under uncertainty (extended abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS-13)*, pages 1291–1292, St. Paul, MN, May 2013.

[16] Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–420, July 2006.

[17] Maja .J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4:73–83, 1997.

[18] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. 16th International Conf. on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

[19] Frans A. Oliehoek, Matthijs T. J. Spaan, Jilles S. Dibangoye, and Christopher Amato. Heuristic search for identical payoff Bayesian games. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 1115–1122, Toronto, Canada, 2010.

[20] Frans A. Oliehoek, Matthijs T.J. Spaan, and Nikos Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *JAIR*, 32:289–353, 2008.

[21] Frans A. Oliehoek and Nikos Vlassis. Q-value heuristics for approximate solutions of Dec-POMDPs. In *Proc. of the AAAI spring symposium on Game Theoretic and Decision Theoretic Agents*, pages 31–37, March 2007.

[22] Frans A. Oliehoek, Shimon Whiteson, and Matthijs T. J. Spaan. Lossless clustering of histories in decentralized POMDPs. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-09)*, pages 577–584, Budapest, Hungary, 2009.

[23] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Articial Intelligence Researchsm4P*, 19:569–629, 2003.

[24] Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for Dec-POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2009–2015, Hyderabad, India, 2007.

[25] Matthijs Spaan. Dec-POMDP problem domains and format. `http://masplan.org/`.

[26] Matthijs T. J. Spaan, Frans A. Oliehoek, and Christopher Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 2027–2032, Barcelona, Spain, 2011.

[27] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[28] Daniel Szer and François Charpillet. Point-based dynamic programming for Dec-POMDPs. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1233–1238, Boston, MA, 2006.

[29] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Rollout sampling policy iteration for decentralized POMDPs. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 666–673, 2010.

[30] Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, San Francisco, CA, 2011.

APPENDIX A

Soundness of Optimization Formulation (Section 2.1)

In [20], optimal joint Q-value functions, say J, are introduced

$$J^*(\vec{h},\vec{a}) = R(\vec{h},\vec{a}) + \sum_{\vec{\omega}} P(\vec{\omega}|\vec{h},\vec{a}) J^*(\vec{h'},\pi^*(\vec{h'})),$$

where π^* is the optimal joint policy and joint-history $\vec{h'}$ is the concatenation of $\vec{a},\vec{\omega}$ to $\vec{h}$. Here, $R(\vec{h},\vec{a}) = \sum_s R(s,\vec{a})P(s|\vec{h})$ and $P(\vec{\omega}|\vec{h},\vec{a}) = \sum_{s'} P(\vec{\omega}|s',\vec{a}) \sum_s P(s'|s,\vec{a})P(s|\vec{h})$. Based on these, it can be shown that

$$Q_i^*(h_i,a_i) = \sum_{a_{-i},h_{-i}} J^*(\vec{h},\vec{a})P(a_{-i}|h_{-i})P(h_{-i}|h_i). \tag{11}$$

In words, the individual Q-values used in this article are really the marginalization of the joint Q-values used in prior work.

Instead of establishing the above claim rigorously, we focus on the intuition, illustrated in Figure 6. $J^*(\vec{h},\vec{a})$ gives the value of the optimal joint-subpolicy, $\langle \pi_i(h_i), \pi_{-i}(h_{-i})\rangle$, when the agents have experienced joint history $\vec{h} = \langle h_i, h_{-i}\rangle$, executed joint action $\vec{a} = \langle a_i, a_{-i}\rangle$ and followed the optimal joint policy thereafter. However, from the perspective of agent i who does not know h_{-i} or a_{-i}, the value of his own subpolicy $\pi_i(h_i)$ is also well-defined, and it is given by the *expectation* of the joint subpolicy value J^* over all possible values of h_{-i} and a_{-i} (as shown in Figure 6), as given in Equation 11.

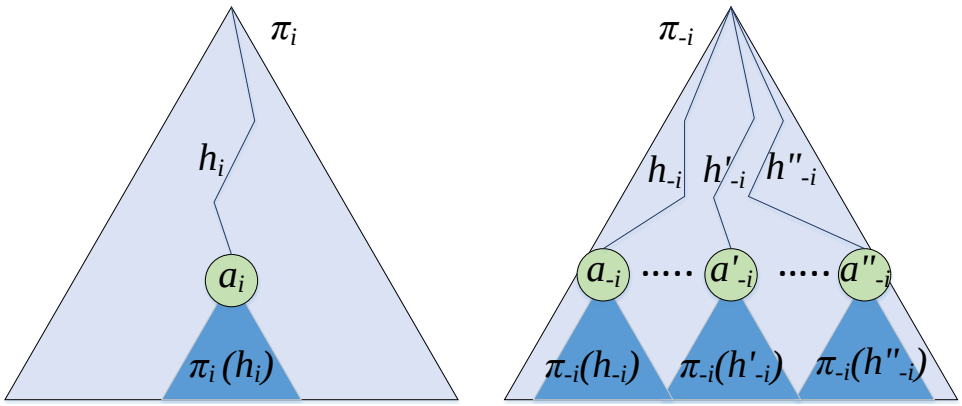

Figure 6: $Q_i(h_i, a_i)$ gives the expected value of agent i's subpolicy at history h_i, averaged over all subpolicies of $-i$ at all histories h_{-i} of the same length as h_i.

By maximizing $Q(h_i, a_i)$ over a_i at each history h_i, as in Equation 2, each agent plays a *best response in expectation* to the joint policy of the other agents. Therefore, the solution joint policy is guaranteed to be a Nash equilibrium. The objective of optimization ensures that the solution is the optimal Nash equilibrium.

A quick verification of the above explanation can be conducted as follows: Since the optimal joint policy, π^*, has a unique value irrespective of which agent calculates it, $Q_i^*(h_\emptyset, a_i^*)$ (where $a_i^* = \arg\max_b Q_i^*(h_\emptyset, b)$) should have a fixed value irrespective of i. In Figure 6, $h_i = h_\emptyset$, so h_{-i} has only one possible value, $h_\emptyset$. Indeed, since $P(h_\emptyset|h_\emptyset) = 1$ (i.e., both agents are guaranteed to experience the empty history at the same time), Equation 11 gives $Q_i^*(h_\emptyset, a_i^*) = \sum_{a_{-i}} J^*(\vec{h}_\emptyset, \vec{a}) P(a_{-i}|h_\emptyset) = J^*(\vec{h}_\emptyset, \vec{a}^*)$ which is the optimal joint policy value, independent of i. This is why the objective function in our optimization formulation can be specified for an arbitrary agent.

APPENDIX B

Proof of Theorem 1 (Section 4)

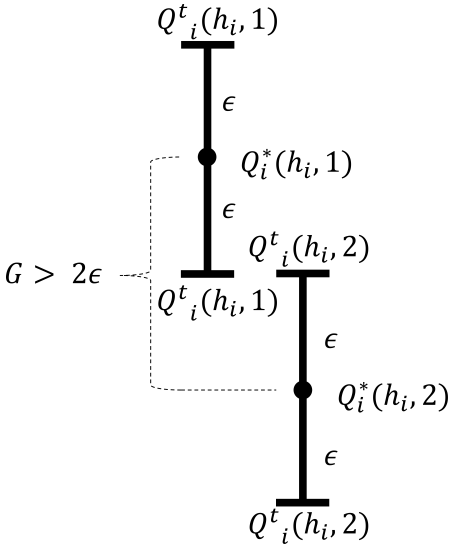

Figure 7: Relation between action gap and $\|Q_i^t - Q_i^*\|$.

First, we shall establish an upper-bound on the $|Q_i^{t+1}(h_i, a_i) - Q_i^*(h_i, a_i)|$ values. From Equation 7,

$$|Q_i^{t+1}(h_i, a_i) - Q_i^*(h_i, a_i)| = \sum_{s, a_{-i}} |\hat{P}_i^{t+1}(a_{-i}, s|h_i)\widetilde{Q}_i^t(s, h_i, \vec{a}) - P_i^*(a_{-i}, s|h_i)\widetilde{Q}_i^*(s, h_i, \vec{a})|.$$

For brevity, we shall write the inner summand as $|\hat{P}_i^{t+1}\widetilde{Q}_i^t - P_i^*\widetilde{Q}_i^*|$, when there is no scope for confusion over the parameters of the functions.

$$\begin{aligned}
|\hat{P}_i^{t+1}\widetilde{Q}_i^t - P_i^*\widetilde{Q}_i^*| &= |(\hat{P}_i^{t+1}\widetilde{Q}_i^t - P_i^*\widetilde{Q}_i^t) + (P_i^*\widetilde{Q}_i^t - P_i^*\widetilde{Q}_i^*)| \\
&= |\widetilde{Q}_i^t(\hat{P}_i^{t+1} - P_i^*) + P_i^*(\widetilde{Q}_i^t - \widetilde{Q}_i^*)| \\
&\leq |\widetilde{Q}_i^t(\hat{P}_i^{t+1} - P_i^*)| + |P_i^*(\widetilde{Q}_i^t - \widetilde{Q}_i^*)| \\
&\leq |\widetilde{Q}_{max}| \cdot \|\hat{P}_i^{t+1} - P_i^*\| + P_i^*\|\widetilde{Q}_i^t - \widetilde{Q}_i^*\|,
\end{aligned}$$

where $\widetilde{Q}_{max}$ is the Q-value from the set of $\widetilde{Q}_i^t(s, h_i, a_i)$ values with the largest absolute value.

Substituting this upper bound for the summand, we find

$$\begin{aligned}
|Q_i^{t+1}(h_i, a_i) - Q_i^*(h_i, a_i)| &\leq \sum_{s, a_{-i}} |\widetilde{Q}_{max}| \cdot \|\hat{P}_i^{t+1} - P_i^*\| + P_i^*\|\widetilde{Q}_i^t - \widetilde{Q}_i^*\| \\
&\leq X_m\|\hat{P}_i^{t+1} - P_i^*\| + \|\widetilde{Q}_i^t - \widetilde{Q}_i^*\| \\
&\leq X_m\|\hat{P}^{t+1} - P^*\| + \|\mathcal{Q}^t - \mathcal{Q}^*\|,
\end{aligned}$$

where X_m is a constant such that $X_m > |\widetilde{Q}_{max}||A_i||S|$, and since $\sum_{s, a_{-i}} P_i^* = 1$ and $\|\mathcal{Q}^t - \mathcal{Q}^*\|$ is constant.

Having upper-bounded $|Q_i^t(h_i, a_i) - Q_i^*(h_i, a_i)|$, we will now establish an upper-bound on $|\widetilde{Q}_i^{t+1}(s, h_i, \vec{a}) - \widetilde{Q}_i^*(s, h_i, \vec{a})|$. Due to assumption (2),

$$\begin{aligned}
|\widetilde{Q}_i^{t+1}(s, h_i, \vec{a}) - \widetilde{Q}_i^*(s, h_i, \vec{a})| &\approx \sum_{\omega_i} P(\omega_i|s, \vec{a})|\max_{b_i} Q_i^t(h', b_i) - \max_{b_i^*} Q_i^*(h', b_i^*)| \\
&\leq \sum_{\omega_i} P(\omega_i|s, \vec{a}) \max_{b_i} |Q_i^t(h', b_i) - Q_i^*(h', b_i)| \\
&\leq \|Q_i^t - Q_i^*\| \\
&\leq \|\mathcal{Q}^t - \mathcal{Q}^*\|.
\end{aligned}$$

Note that this upper bound is dominated by the previously established upper bound on $|Q_i^{t+1}(h_i, a_i) - Q_i^*(h_i, a_i)|$, thus,

$$\|\mathcal{Q}^{t+1} - \mathcal{Q}^*\| \leq X_m\|\hat{P}^{t+1} - P^*\| + \|\mathcal{Q}^t - \mathcal{Q}^*\|.$$

Given this upper-bound, we can write

$$\begin{aligned}
P(\|\mathcal{Q}^{t+1} - \mathcal{Q}^*\| \leq \epsilon) &> P(X_m\|\hat{P}^{t+1} - P^*\| + \|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon) \\
&> P(X_m\|\hat{P}^{t+1} - P^*\| = 0 \wedge \|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon) \\
&\geq P(X_m\|\hat{P}^{t+1} - P^*\| = 0 \wedge \|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon \wedge G > 2\epsilon) \\
&= P(X_m\|\hat{P}^{t+1} - P^*\| = 0 \mid \|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon \wedge G > 2\epsilon)P(\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon \wedge G > 2\epsilon).
\end{aligned}$$

In the first two steps above, we have used the fact that if $A \Rightarrow B$ but $B \not\Rightarrow A$ then $P(A) < P(B)$. In the second step, we have also used assumption (4). In the third step we have assumed $P(G > 2\epsilon) > 0$.

Figure 7 depicts the error bars for a given history h and two actions, "1" and "2" when $G > 2\epsilon$ and $\|Q_i^t - Q_i^*\| < \epsilon$. Since $\|Q_i^t - Q_i^*\| \leq \epsilon$, the Q-value estimate for a given history h_i and action a_i is bounded by

$$Q_i^*(h_i, a_i) - \epsilon \leq Q_i^t(h_i, a_i) \leq Q_i^*(h_i, a_i) + \epsilon.$$

When $G > 2\epsilon$, the lower bound for the optimal action (in the figure, this is action "1") exceeds the upper bound for all suboptimal actions. This means that $\arg\max_{a_i} Q_i^t(h_i, a_i) = \arg\max_{a_i} Q_i^*(h_i, a_i) = \pi_i^*(h_i)$ for all h_i (i.e. agent i's policy is optimal). $\hat{P}_i^{t+1}(s, a_{-i}, |h_i)$ is a marginalization over all other agents' histories, i.e.

$$\hat{P}_i^{t+1}(s, a_{-i}|h_i) = \sum_{h_{-i}} \hat{P}^{t+1}(a_{-i}|h_{-i})\hat{P}_i^{t+1}(s, h_{-i}|h_i).$$

Note that if $\arg\max_{a_i} Q_i^t(h_i, a_i) = \pi_i^*(h_i)$ for all agents i (as we have established is the case when $G > 2\epsilon$ and $\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon$), then $\hat{P}^{t+1}(a_i|h_i) = P^*(a_i|h_i)$ for all agents due to assumption (3), and therefore by induction on Equation 2, $\hat{P}_i^{t+1}(s, h_{-i}|h_i) = P_i^*(s, h_{-i}|h_i)$. It follows then that $\hat{P}_i^{t+1}(s, a_{-i}|h_i) = P_i^*(s, a_{-i}, |h_i)$ and, as a result,

$$\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon \wedge G > 2\epsilon \implies X_m\|\hat{P}^{t+1} - P^*\| = 0.$$

Therefore,

$$\begin{aligned} P(\|\mathcal{Q}^{t+1} - \mathcal{Q}^*\| \leq \epsilon) &> 1 \cdot P(\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon \wedge G > 2\epsilon) \\ &= P(\|\mathcal{Q}^t - \mathcal{Q}^*\| \leq \epsilon), \end{aligned}$$

for any $\epsilon \leq \epsilon_0$ of the MAG assumption.