Solving Finite Horizon Decentralized POMDPs by Distributed Reinforcement Learning

Bikramjit Banerjee
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406
Bikramjit.Banerjee@usm.edu

Jeremy Lyle
Dept. of Mathematics
The University of Southern Mississippi
Hattiesburg, MS 39406
Samuel.Lyle@usm.edu

Landon Kraemer
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406
Landon.Kraemer@eagles.usm.edu

Rajesh Yellamraju
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406
Rajesh.Yellamraju@eagles.usm.edu

ABSTRACT

Decentralized partially observable Markov decision processes (Dec-POMDPs) offer a powerful modeling technique for realistic multi-agent coordination problems under uncertainty. Prevalent solution techniques are *centralized* and assume prior knowledge of the *model*. We propose a distributed reinforcement learning approach, where agents take turns to learn best responses to each other's policies. This promotes decentralization of the policy computation problem, and relaxes reliance on the full knowledge of the problem parameters. We derive the relation between the sample complexity of best response learning and error tolerance. Our key contribution is to show that even the "per-leaf" sample complexity could grow exponentially with the problem horizon. We show empirically that even if the sample requirement is set lower than what theory demands, our learning approach can produce (near) optimal policies in some benchmark Dec-POMDP problems. We also propose a slight modification that empirically appears to significantly reduce the learning time with relatively little impact on the quality of learned policies.

1. INTRODUCTION

Decentralized partially observable Markov decision processes (Dec-POMDPs) offer a powerful modeling technique for realistic multi-agent coordination and decision making problems under uncertainty. Because solving Dec-POMDPs is NEXP-complete [4], exact solution techniques for finite horizon problems require significant time and memory resources [20, 15, 18]. However, solution techniques for Dec-POMDPs (exact or approximate) suffer from less acknowledged limitations as well: that most of them are *centralized* and assume prior knowledge of the *model*. That is, a single program computes the optimal joint policy, with the full knowledge of the problem parameters. While these techniques have had success in benchmark problems with comprehensively defined domain parameters, such strict definitions may be difficult and tedious in many real-world problems. In these cases, the problem parameters may need to be first estimated from experience and then exact/approximate solvers may be applied. However, this problem of model estimation can be complex in POMDPs because states are unobservable [6], and additionally in Dec-POMDPs agents are incapable of observing each other's actions and observations. Not surprisingly, model estimation is largely ignored in the Dec-POMDP literature. Furthermore, the task of computing optimal policies in Dec-POMDPs has seldom been decentralized in any meaningful way (e.g., see [8].)

In this paper we propose a distributed reinforcement learning approach to solving finite horizon Dec-POMDPs. When agents *learn* their *own* policies, not only is the task of policy computation distributed, but also the problem parameters do not need to be known a priori. In lieu of the knowledge of problem parameters, access to a simulator, or simply the ability to draw samples from unknown distributions would be sufficient. Effectively, estimation of the problem parameters is built into the learning algorithm. Policy learning in *finite* horizon tasks is justified due to the same reasons as finite horizon reinforcement learning, viz., that agents can learn policies in offline simulations before applying them in the real domain. Furthermore, unlike many exact and approximate solution approaches for Dec-POMDPs, the memory usage of a learning approach is not much larger than the size of a *single* policy per agent at any time, which makes it relatively more memory efficient. Thus we posit distributed reinforcement learning as a more practical alternative to the traditional Dec-POMDP solvers.

Reinforcement learning has been applied in infinite horizon POMDPs in both model based [6, 12, 17] and model free [13] ways. Model based methods learn a model (e.g., hidden Markov models or utile suffix memories for POMDPs) of the environment first and then compute a policy based on the learned model, while model free methods learn a policy directly. Model learning can be more complex in Dec-POMDPs because the actions and the observations of the other agents are unobservable. We use a semi-model based approach, where we do not attempt to estimate the Dec-POMDP parameters owing to their hidden parts, but instead learn intermediate functions that capture the visi-

The Seventh Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (*MSDM-2012*), held in conjunction with *AAMAS*, June 2012, in Valencia, Spain.

ble parts of the dynamics (see equations 4, 5 given later) via Monte Carlo estimation, and compute a policy based on these functions.

Zhang and Lesser [21] recently applied reinforcement learning to a variant of the finite horizon Dec-POMDP problem, where agents are organized in a network, and agents' influence on each other are limited to cliques. This factored structure of the domain is exploited to solve such problems more scalably than regular Dec-POMDPs. Zhang and Lesser also exploited the known communication structure to coordinate the sub-teams via distributed contraint optimization, and produced a more efficient learning-based alternative to the regular solvers. While our goal is similar and we also consider finite horizon problems, we focus on less structured and unfactored Dec-POMDPs that are inherently less scalable. As in our work, [21] does not guarantee optimality unless the agent coordination graph is acyclic.

Our initial experiments with concurrent *independent* reinforcement learning [7] in benchmark Dec-POMDP problems have yielded unsatisfactory results, some of which are included in the experiments section. In this paper we propose a non-concurrent independent learning approach, where agents take turn in learning best response policies to each other via a semi-model based Monte Carlo algorithm, but no agent explicitly attempts to model the other agent. We show theoretically that Monte Carlo reinforcement learning for best response has a complexity of $O(T^3|A|^T|\Omega|^{3T-1})$, where T is the horizon length, $|A|$ and $|\Omega|$ are respectively the number of actions and observations available to an agent. While it is intuitive that this expression could be exponential in T, the reason turns out to be less intuitive. Our analysis shows that it depends on the number of distinct scenarios that the other (non-learning and invisible) agent may encounter. This is a key distinction between POMDPs and Dec-POMDPs. We also show empirically that a "few" alternations of best response learning produce (near) optimal policies in some benchmark problems, although in general alternate best response learning can converge to local optima. Finally, we propose a slight modification of our algorithm, that empirically appears to significantly reduce the learning time with relatively little impact on the quality of the learned policies. A shorter version of this paper appears in [3].

2. DECENTRALIZED POMDP

The Decentralized POMDP (Dec-POMDP) formalism is defined as a tuple $\langle n, S, A, P, R, \Omega, O\rangle$, where:

- n is the number of agents playing the game.
- 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 $s \in S$.
- $R : S \times A \rightarrow \mathbb{R}$, where $R(s, \vec{a})$ gives the immediate reward the agents 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 $\vec{a} \in A$.

The reward function R, transition model P, and observation model O are defined over joint actions and/or observations, which forces the agents to coordinate. Additionally, for finite horizon problems, horizon T is also specified. The goal of the Dec-POMDP problem is to find a policy for each agent (*joint policy*) that maximizes the total expected reward over T steps of interaction, given that the agents cannot communicate their observations and actions to each other. A joint policy Π is a set of individual policies, π_i, which maps the histories of action-observation pairs of agent i to actions in A_i.

3. REINFORCEMENT LEARNING

Reinforcement learning (RL) problems are modeled as *Markov Decision Processes* or MDPs [19]. An MDP is given by the tuple $\langle S, A, R, P\rangle$, where S is the set of environmental states that an agent can be in at any given time, A is the set of actions it can choose from at any state, $R : S \times A \mapsto \Re$ is the reward function, i.e., $R(s, a)$ specifies the reward from the environment that the agent gets for executing action $a \in A$ in state $s \in S$; $P : S \times A \times S \mapsto [0, 1]$ is the state transition probability function specifying the probability of the next state in the Markov chain consequential to the agent's selection of an action in a state. In finite horizon problems, the agent's goal 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,

$$V^\pi(s^0, t) = E_P[R(s^0, \pi(s^0, t)) + R(s^1, \pi(s^1, t+1)) + \ldots + R(s^{T-t}, \pi(s^{T-t}, T))]$$

where $s^0, s^1, \ldots s^{T-t}$ are successive samplings from the distribution P following the Markov chain with policy π.

Reinforcement learning algorithms often evaluate an action-quality value function Q given by

$$Q(s, a, t) = R(s, a) + \max_\pi \gamma \sum_{s'} P(s, a, s') V^\pi(s', t+1) \quad (1)$$

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. Action quality functions are preferred over value functions, since the optimal policy can be calculated more easily from the former. Learning algorithms can be model based or model free. Model based methods explicitly estimate $R(s, a)$ and $P(s, a, s')$ functions, and hence estimate $Q(s, a, t)$. Model free methods directly learn $Q(s, a, t)$, often by online dynamic programming, e.g., *Q-learning*. In this paper, we use (semi-) model based learning for Dec-POMDPs which makes for easier analysis of sample complexity, thus establishing a baseline for RL in Dec-POMDPs. Model based reinforcement learning algorithms have been analyzed in many domains before, but to the best of our knowledge such analysis have not been performed for decentralized POMDPs, where partial observability of the learner's environment is compounded by the unobservability of the other agents' observations and actions.

4. RL FOR DEC-POMDPS

Solution techniques for Dec-POMDPs have been mostly centralized [20, 15, 18], in that a single program computes the optimal joint policy, with the full knowledge of the problem parameters, viz., P, R, O. While these techniques have had success in benchmark problems with comprehensively defined P, R, O, such strict definitions may be difficult and tedious in real-world problems. In this paper we address this issue by applying reinforcement learning to the policy computation problem. The main distinguishing characteristics of our approach are

- Instead of a single program computing the optimal joint policy, each agent learns its own policy. In this paper agents learn distributedly, but not concurrently. That is, they share the task of policy learning, by only learning their own policies, but do not update policies concurrently. Concurrent learning will effectively parallelize Dec-POMDP solution, but it is also challenging due to potential oscillation. Our experiments show that concurrent learning is not as efficient as the proposed distributed learning approach. We leave the improvement of concurrent learning in Dec-POMDPs as a future avenue.

- Instead of using knowledge of P, R, O, agents learn on the basis of sampling these unknown functions. This allows our approach to be readily applicable in tasks where these parameters are unknown, or hard to compute. However, for evaluation purposes, we still consider well-defined benchmark problems in this paper.

- Most Dec-POMDP solvers maintain many policies in memory at any time, partly or wholly. Even memory bounded techniques [16] maintain multiple policies, although of a bounded total size. Instead, a learner only needs to effectively maintain sufficient information in memory to construct *one* policy. However, for finite horizon problems, this policy has a size exponential in T.

Although the agents are unaware of P, R, O, we assume that the agents know the size of the problem, i.e., $|A|, |S|, |\Omega|$, the maximum magnitude over all rewards, $R_{\max}$, and that they are capable of signalling to each other so that no two agents are learning at the same time. With > 2 agents, the order of learning phases must also be fixed by prior agreement.

Since states are not visible, a reinforcement learning agent can use the policy representation of finite horizon Dec-POMDPs, and learn a mapping from histories of its past actions and observations to actions [21]. For simplicity of notation, we assume two agents only, and identical action and observation sets for both agents. Given the policy of the other agent, π, the quality of a learner's action a at a given level-t history h_t is given by

$$Q_t^*(h_t, a|\pi) = R_t^*(h_t, a|\pi) + \sum_{\omega} H_t^*(h_t, a, h_{t+1}|\pi) \cdot \max_b Q_{t+1}^*(h_{t+1}, b|\pi) \quad (2)$$

where h_{t+1} is a level-$t+1$ history produced by the concatenation of h_t and (a, ω), i.e., $h_{t+1} = (h_t, a, \omega)$. The best response policy of the learner, π_ℓ, to the other agent's policy π is given by

$$\pi_\ell(h_t) = \arg\max_a Q_t^*(h_t, a|\pi). \quad (3)$$

The functions R_t^* and H_t^* represent level-t reward and history transition functions for the learner, given by

$$R_t^*(h_t, a|\pi) = \sum_{s, h_-} P(s|h_t, h_-)P(h_-|h_t, \pi)R(s, \vec{a}) \quad (4)$$

$$H_t^*(h_t, a, h_{t+1}|\pi) = \sum_{s, s', h_-} P(s|h_t, h_-)P(h_-|h_t, \pi) \cdot P(s'|s, \vec{a}) \sum_{\omega_-} O(\vec{\omega}|s', \vec{a}) \quad (5)$$

where h_- is the history of action-observations encountered by the other agent, $\vec{\omega} = \langle \omega, \omega_- \rangle$ and $\vec{a} = \langle a, \pi(h_-) \rangle$ are the joint observation and action respectively. A learning agent is unaware of every factor on the right hand sides of equations 4, 5, and must estimate R^* and H^* solely from its own experience of executing actions and receiving observations and rewards. Note that while the learners do use rewards as "shared observations" in order to estimate the quality values while learning, they do not rely on such observations when executing the policy (i.e., equation 3). Therefore, the crux of a Dec-POMDP is preserved, making our approach an effective alternative to classical Dec-POMDP solvers.

For brevity, we call the following expression β.

$$\beta = |A| \left(\frac{(|A||\Omega|)^T - 1}{|A||\Omega| - 1} \right). \quad (6)$$

β gives the maximum number of (h, a) pairs of all lengths that a learner may encounter. We now give the definition of a key parameter that appears in the complexity expression of our algorithm.

Definition 1. *The minimum reachability over all feasible states at level t for a fixed policy of the other agent, π, is given by*

$$\rho_{t,\pi} = \min_{s, h_t, h_- | \pi} P(s|h_t, h_-)P(h_-|h_t, \pi)$$

Feasibility excludes unreachable states, and therefore ensures that always $\rho_{t,\pi} > 0$.

When a learner takes action a at history h_t (with the other agent executing $\pi(h_-)$) and the resulting joint observation is $\vec{w} = \langle w, w_- \rangle$, then reachability can be propagated as

$$\begin{aligned} & P(s|h_t, h_-)P(h_-|h_t, \pi)P(s'|s, \vec{a})O(\vec{w}|s', \vec{a}) \\ = \; & P(s'|h_{t+1}, h'_-)P(h'_-|h_{t+1}, \pi) \end{aligned}$$

where $h_{t+1} = (h_t, a, \omega)$ and $h'_- = (h_-, \pi(h_-), \omega_-)$. Clearly, the minimum reachability at level $t+1$ is

$$\rho_{t+1,\pi} \leq \rho_{t,\pi},$$

forming a monotonically decreasing sequence with increasing t. Therefore, we refer to the minimum reachability over all steps, $\rho_{T-1,\pi}$, simply as ρ dropping both subscripts when π is clear from the context.

4.1 The Algorithm: MCQ-ALT

In this paper we present an approach where agents take turn to learn best response to each others' policies, using an R-Max [5] like approach to learn the best response Q-values. The main algorithm for any learning agent – called Monte Carlo Q Alternating, or MCQ-ALT–is shown in Algorithm 1, along with its subroutines in Algorithms 2– 5. The learner records immediate rewards and history transitions at every

Algorithm 1 MCQ-ALT(N)

repeat
  $h \leftarrow \emptyset$
  $a \leftarrow$SELECTACTION(h)
  Execute a and receive r, ω
  for $t \leftarrow 1 \ldots T-1$ **do**
    $b \leftarrow$STEP(h, a, ω, r)
    $h \leftarrow (h, a, \omega)$
    Execute b and receive r, ω
    $a \leftarrow b$
  end for
  ENDEPISODE(h, N)
until $Known(\emptyset) = True$

Algorithm 2 SELECTACTION(h)

if $Known(h) = True$ **then**
  $a \leftarrow \arg\max_{b \in A} Q(h, b)$
else
  $a \leftarrow \min_b frequency(h, b)$
end if
$frequency(h, a) \leftarrow frequency(h, a) + 1$
Return a

history encountered, providing samples of R^* and H^* given in equations 4, 5 respectively. These samples are incorporated into running averages to maintain estimates $\hat{R}$ and $\hat{H}$ respectively. For histories of length $T-1$ (i.e., full length), h_{T-1}, if the pair (h_{T-1}, a) has been encountered

$$N = \max(|S|^2|\Omega|^{T-1}, 4/\rho)\frac{(4R_{\max}T|S|)^2|\Omega|^{T+1}}{\alpha^2} \cdot \ln(16|S|^2|\Omega|^T\beta/\delta) \quad (7)$$

times, then the learner sets $Q_{T-1}(h_{T-1}, a)$ to the average of the immediate rewards received (i.e., $\hat{R}_{T-1}(h_{T-1}, a)$), via Algorithms 4, 5. It then marks (h_{T-1}, a) as "Known". If (h_{T-1}, a) is "Known" for every a, then h_{T-1} is marked "Known". For an intermediate length history, h_t, if *every* history, h_{t+1}, produced by concatenating h_t with (a, ω) for all combinations of action-observations encountered is "Known", then h_t is marked "Known". For a "Known" intermediate history h_t, $Q_t(h_t, a)$ is updated for every a as

$$Q_t(h_t, a) = \hat{R}_t(h_t, a) + \sum_{h'} \hat{H}_t(h_t, a, h') \max_b Q_{t+1}(h', b)$$

The learner's exploration strategy is shown in Algorithm 2. For a "Known" history, action selection is greedy, i.e., the Q-maximizing action is selected. For a history that is not yet marked "Known", the least frequently taken action (ties broken randomly) is executed. The learner freezes its current policy when the empty history is marked "Known", and signals to the other agent to start learning, while it executes its current policy without exploration. MCQ-ALT learns best response directly without modeling the other agents in the environment, and only modeling the visible parts of the environment's dynamics.

The infimum reachability, ρ used in equation 7, may not be known in many problems. Since it decreases geometrically with inceasing T, it may even be hard to determine whether $|S|^2|\Omega|^{T-1}$ dominates $4/\rho$. In such cases, it may be possible to ignore it, but this may be inadequate for higher T. For

Algorithm 3 STEP(h, a, ω, r)

$h' \leftarrow (h, a, \omega)$
$\hat{H}(h, a, h') \leftarrow \hat{H}(h, a, h') + 1$
$\hat{R}(h, a) \leftarrow \hat{R}(h, a) + r$
$Remaining(h) \leftarrow Remaining(h) \cup \{(a, \omega)\}$
Return SELECTACTION(h')

Algorithm 4 ENDEPISODE(h, N)

if $frequency(h, a) > N,\ \forall a \in A$ **then**
  $Known(h) \leftarrow True$
  QUPDATE(h)
end if
while $h \neq \emptyset$ **do**
  Let $h = (h', a, \omega)$
  $Remaining(h') \leftarrow Remaining(h') \setminus \{(a, \omega)\}$
  if $Remaining(h') = \emptyset$ **then**
    $Known(h') \leftarrow True$
    QUPDATE(h')
  else
    break
  end if
  $h \leftarrow h'$
end while

the domains and horizons used for experiments in this paper, ρ does not appear to be a dominating factor, so our analysis focuses on the dependence on T instead.

An important feature of MCQ-ALT is its well-defined stopping criterion, viz., when the empty history becomes "Known", which is controlled by a single parameter, N. In contrast, Q-learning is controlled by multiple parameters, and its stopping criterion can be affected by oscillation or non-convergence.

Note that in learning the best reponse, a learner attempts to cover every (h_t, a) encountered equally well, to guarantee arbitrarily small errors in the value function. However, there are at least two reasons why the value function may not need to be accurate to an arbitrary degree: (1) policies usually converge long before value functions, which we verify in this paper experimentally, and (2) some less likely paths may have little impact on the value function and N could be lowered for these paths; we address this next.

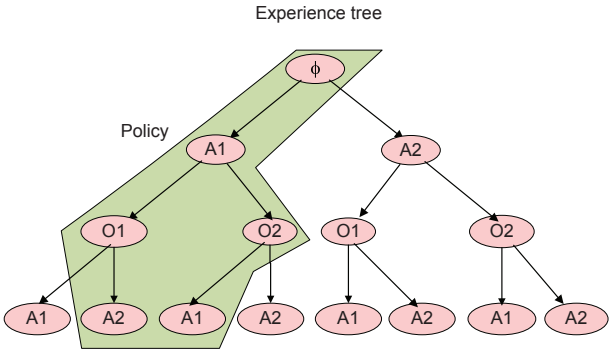


Figure 1: A learner's experience tree.

4.2 Adjustment for Rare Histories

Figure 1 shows a learner's entire possible experience tree

Algorithm 5 QUPDATE(h)

1: **for** $a \in A$ **do**
2: $Q(h,a) \leftarrow \hat{R}(h,a)/frequency(h,a)$
3: **if** h is not full-length history **then**
4: $Q(h,a) \leftarrow Q(h,a) + \frac{1}{H}\sum_{\omega|h'=(h,a,\omega)} \hat{H}(h,a,h') \max_{b\in A} Q(h',b)$
5: **end if**
6: **end for**

in a 2-action, 2-observation, $T = 2$ scenario. MCQ-ALT would invest N samples to *every* leaf node in this experience tree. However, in cases where some histories are rare, this becomes a significant liability, since it requires a vast series of episodes to collect sufficient (i.e., N) samples of such rare histories. Furthermore, such histories possibly contribute little to the value function. A valid policy is a (small) part of the experience tree, as shaded in Figure 1. Clearly, there are many histories that can be allotted fewer samples because they do not partake in the optimal policy.

In this paper, we test a simple modification of MCQ-ALT, called "MCQ-ALT Adjusted for Rare Histories", or MCQ-ALT-ARH. This modification estimates the likelihood of a full length history, h_{T-1}, that has been actually encountered, as $f_{h_{T-1}} = \frac{\hat{H}(h_{T-2},a,h_{T-1})}{\sum_{a,\omega}\hat{H}(\emptyset,a,(a,\omega))}$ (where $h_{T-1} = (h_{T-2},a,\omega)$ for a specific a and ω) and only requires $f_{h_{T-1}} \cdot N$ samples for h_{T-1} instead of N. In the experiments section, we perform this modification only for histories whose $f_{h_{T-1}}$ falls below a threshold, ϵ, given as an external parameter.

5. ANALYSIS

We focus on sample complexity analysis of MCQ-ALT. Although the fixed policy of the other agent effectively reduces the Dec-POMDP to a POMDP, the fact that the other agent's actions and observations are unobservable to the learner makes the analysis more complex than a POMDP. In particular, the number of scenarios encountered by the other agent (referred to as K) becomes a key parameter in the sample complexity analysis, which would not appear in comparable POMDP analyses.

First we note that the number of episodes needed for the empty history to be "Known" is

$$\geq N|\Omega|^{T-1}|A|^T$$

since the number of distinct (h_{T-1},a) tuples is $|\Omega|^{T-1}|A|^T$. Also, given the backup process of histories becoming "Known" in our algorithm, when all (h_{T-1},a) become "Known" the empty history must also become "Known", and this takes N visitations of each tuple. The actual number of episodes needed is, however, likely to be greater than $N|\Omega|^{T-1}|A|^T$, because only part of the exploration process is under the learner's control, where it can select a but not ω. Thus it can be led to revisit paths that are already "Known".

While it is fairly intuitive that the episode complexity given above should be exponential in T, it is not immediately clear that so could N. This is precisely where the complexity of Monte Carlo reinforcement learning differs between POMDPs [6, 12, 9] and Dec-POMDPs. In order to demonstrate this, we first present a generic sampling process, and use the resulting sample complexity expression to derive N.

5.1 The Basic Sampling Process

Consider the following sampling process, with K classes of random variables, $\{X_{jl}\}_{j=1}^{K}$, such that

$$Y = \frac{\sum_{l=1}^{N_1} X_{1l} + \sum_{l=1}^{N_2} X_{2l} + \ldots \sum_{l=1}^{N_K} X_{Kl}}{N},$$

where $N = \sum_j^K N_j$. The process generates a sample of some X_{jl} at each of the N iterations, where the probability that the sample belongs to class $j \in [1,K]$ is p_j. Therefore, all X_{jl} as well as all N_j are random variables. Suppose that $E[X_{jl}] = M_j$, $\forall l$, and that the maximum magnitude of any X_{jl} is $X_{\max} > 0$. We wish Y to estimate the unknown value $\sum_j p_j M_j$, therefore we call $|Y - \sum p_j M_j|$ the *estimation error*. We claim the following sufficient condition for bounding the estimation error, but give the proof in [2] due to lack of space.

Theorem 1. *If the total number of samples is set* $N \geq \max(K\eta, 4\eta/\min_j p_j)$ *where*

$$\eta = \frac{4X_{\max}^2 K}{\epsilon^2}\ln(8K/\delta),$$

then the estimation error is bounded, i.e., $P(|Y - \sum p_j M_j| > \epsilon) < \delta$.

5.2 Derivation of N

In our analysis we shall use the max norm function, i.e., $\|f - g\|$ represents $\max_x |f(x) - g(x)|$. We first establish the dependence of the error in the Q functions on the errors in our estimates $\hat{H}$ and $\hat{R}$.

Lemma 2. *If* $\|\hat{H}_\tau - H^*_\tau\| \leq \epsilon_1$ *and* $\|\hat{R}_\tau - R^*_\tau\| \leq \epsilon_2$ *for all* τ*, then at any step* t

$$\|Q_t - Q^*_t\| \leq (T-t)\epsilon_2 + (T-t-1)|\Omega|R_{\max}\epsilon_1$$

Proof: By induction. For basis, $t = T-1$. Since T is the last step in an episode, $\|Q_{T-1} - Q^*_{T-1}\| = \|\hat{R}_{T-1} - R^*_{T-1}\| \leq \epsilon_2$, hence true. For the inductive case, we see that $|Q_t(h_t,a) - Q^*_t(h_t,a)|$

$$\begin{aligned}
&= |\hat{R}_t(h_t,a) + \textstyle\sum_{h'} \hat{H}_t(h_t,a,h') \max_b Q_{t+1}(h',b) - \\
&\quad R^*_t(h_t,a) - \textstyle\sum_{h'} H^*_t(h_t,a,h') \max_b Q^*_{t+1}(h',b)| \\
&\leq \|\hat{R}_t - R^*_t\| + |\textstyle\sum(\hat{H}_t \max Q_{t+1} - H^*_t \max Q^*_{t+1})| \\
&= \|\hat{R}_t - R^*_t\| + |\textstyle\sum(\hat{H}_t \max Q_{t+1} - \hat{H}_t \max Q^*_{t+1} \\
&\quad + \hat{H}_t \max Q^*_{t+1} - H^*_t \max Q^*_{t+1})| \\
&\leq \epsilon_2 + |\textstyle\sum(\hat{H}_t \max Q_{t+1} - \hat{H}_t \max Q^*_{t+1})| + \\
&\quad |\textstyle\sum(\hat{H}_t \max Q^*_{t+1} - H^*_t \max Q^*_{t+1})| \\
&\leq \epsilon_2 + \max|Q_{t+1} - Q^*_{t+1}| + |\textstyle\sum(\hat{H}_t - H^*_t)\max Q^*_{t+1}|
\end{aligned}$$

In the last expression, the second term is upper bounded by $(T-t-1)\epsilon_2 + (T-t-2)|\Omega|R_{\max}\epsilon_1$, by the induction hypothesis. In the third term, $\max Q^*_{t+1} \leq R_{\max}$, and the sum is taken over all observations. Therefore the third term is upper bounded by $|\Omega|R_{\max}\epsilon_1$. Adding the bounds of the three terms we get the result. □

Lemma 2 implies that the error bound increases for smaller histories, and therefore is maximum at the empty history. This is why the learner must continue until Q_0 is sufficiently accurate, i.e., the empty history becomes "Known". In the

following analysis, we characterize "sufficiently accurate", to derive a bound on N used by the algorithm (equation 7).

Theorem 3. *To ensure that $\|Q_0 - Q_0^*\| \leq \alpha$ w.p. $\geq 1-\delta$, it is sufficient to set*

$$N \geq \max(|S|^2|\Omega|^{T-1}, 4/\rho)\frac{(4R_{\max}T|S|)^2|\Omega|^{T+1}}{\alpha^2} \cdot \ln(16|S|^2|\Omega|^T\beta/\delta)$$

in our algorithm, where β is given in equation 6 and ρ results from Definition 1.

Proof: By Lemma 2,

$$\|Q_0 - Q_0^*\| \leq T\epsilon_2 + (T-1)|\Omega|R_{\max}\epsilon_1.$$

To achieve the α bound on the error, it is sufficient to set $\epsilon_1 \leq \alpha/2(T-1)|\Omega|R_{\max}$, and $\epsilon_2 \leq \alpha/2T$.

Now the number of $\hat{H}$ and $\hat{R}$ entries that need to be learned are $|\Omega|\beta$ and β respectively, where β is given in equation 6. Therefore, it is sufficient to require both of the following for any t:

$$P(\|\hat{R}_t - R_t^*\| > \alpha/2T) < \delta/2\beta \quad (8)$$

$$P(\|\hat{H}_t - H_t^*\| > \alpha/2(T-1)|\Omega|R_{\max}) < \delta/2\beta|\Omega|$$

First consider $\|\hat{R}_t - R_t^*\|$ and equation 4. The estimation of any $\hat{R}_t$ in our algorithm matches the description of the sampling process, with $E[X_{jl}] = R(s,\vec{a})$ and $p_j = P(s|h_t,h_-)\cdot P(h_-|h_t,\delta)$; the last quantity being the reachability, of which the infimum is ρ (Definition 1). Note that p_j cannot be set to $\sum_{h_-} P(s|h_t,h_-)P(h_-|h_t,\delta)$, since each sample X_{jl} received corresponds to a specific history h_- encountered by the other agent. Therefore in this case, the number of variable classes in the sampling process is $K = |S||\Omega|^t \leq |S||\Omega|^{T-1}$. This is the maximum possible number of terms in the summation of equation 4 which corresponds to full length histories that can be encountered by the other agent, given its (fixed) policy. Making the substitutions for K, $X_{\max}$, and using $\alpha/2T$ for α and $\delta/2\beta$ for δ in Theorem 1, we see that to ensure equation 8, it is sufficient to set

$$N \geq \max(|S||\Omega|^{T-1}, 4/\rho)\frac{(4R_{\max}T)^2|S||\Omega|^{T-1}}{\alpha^2} \cdot \ln(16|S||\Omega|^{T-1}\beta/\delta)$$

Similarly, for $\|\hat{H}_t - H_t^*\|$, the sampling process is characterized by $E[X_{jl}] = 1$ and

$$p_j = P(s|h_t,h_-)P(h_-|h_t,\delta)P(s'|s,\vec{a})P(\vec{\omega}|s',\vec{a}).$$

The last quantity is the propagated reachability, of which the infimum is also ρ. Since $t \leq T-2$, this yields $K = |S|^2|\Omega|^{t+1} \leq |S|^2|\Omega|^{T-1}$, and we have

$$N \geq \max(|S|^2|\Omega|^{T-1}, 4/\rho)\frac{(4R_{\max}(T-1)|S|)^2|\Omega|^{T+1}}{\alpha^2} \cdot \ln(16|S|^2|\Omega|^T\beta/\delta)$$

Combining the two, we get the result. □

It is interesting to note that N is polynomial in most problem parameters, except that it is logarithmic in $|A|$, and exponential in T. Although Theorem 3 suggests that $N = O(T^3|\Omega|^{2T})$, our experiments suggest that it does not require to grow in some domains due to simpler structure. Even in the domains where it does need to grow, the rate of growth could be lower.

6. EVALUATION

6.1 Initial Policy

If agents alternate in learning best responses, the agent that does not learn initially must play some previously specified fixed policy. Our experiments show that if this policy is random then the final outcome is unpredictably poor. Instead, we simply let the two agents perform concurrent reinforcement learning to learn initial policies, on a slightly simpler Dec-POMDP. This Dec-POMDP reduces the set of observations to one dummy observation. In other words, the agents simply ignore the observations, and learn a mapping from their own past action histories to actions ($\pi(a_1,a_2,\ldots,a_t) = a$). This policy can be readily translated to the regular policy language, by setting $\pi(a_1,\omega_1,a_2,\omega_2,\ldots a_t,\omega_t) = a$ for all possible chains of observations $(\omega_1,\omega_2,\ldots\omega_t)$. This is the policy used by the initially non-learning agent. More details on the generation of this initial policy, and comparison with alternatives, can be found in [10, 11].

6.2 Policy Computation and Evaluation

Although a learner computes its best response policy as given in equation 3, its executable policy can be given more compactly, since not all histories encountered during learning will be encountered when executing its best response policy. This executable policy can be constructed by *only* considering histories of the form

$$h_t = (h_{t-1}, \pi_\ell(h_{t-1}), \omega)$$

for all possible observations ω, starting at $h_0 = \emptyset$. If an observation was never encountered at some $(h_{t-1}, \pi_\ell(h_{t-1}))$, then the learner can output a random action. In this case however, all histories that contains this history as a prefix will also be unseen, and will constitute an unlearned part of the policy tree.

In contrast with [21], we use the exact method (instead of simulation) to evaluate a joint policy, since we limit our evaluations to precisely defined benchmark problems. For more practical problems, the simulation approach to policy evaluation would be the only option. To evaluate a joint policy $\vec{\pi}$, we find

$$V^{\vec{\pi}}(\vec{h}_0) = \sum_{s\in S} b_0(s)V^{\vec{\pi}}(\vec{h}_0, s) \quad (9)$$

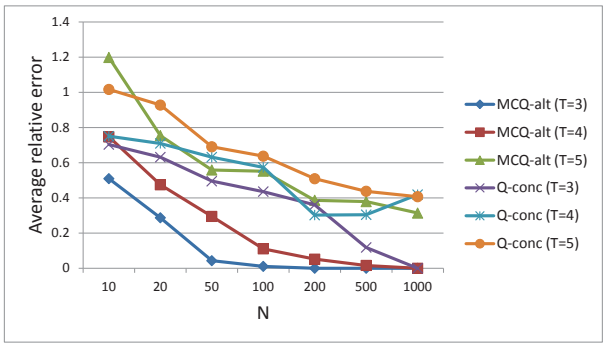
where $b_0 \in \Delta(S)$ is the initial state distribution. $V^{\vec{\pi}}(\vec{h}_t, s)$ for a given joint history $\vec{h}_t$ and state s is given by

$$V^{\vec{\pi}}(\vec{h}_t, s) = R(s,\vec{\pi}(\vec{h}_t)) + \textstyle\sum_{s'\in S} P(s'|s,\vec{\pi}(\vec{h}_t)) \cdot \sum_{\vec{\omega}\in\Omega} O(\vec{\omega}|s',\vec{\pi}(\vec{h}_t))V^{\vec{\pi}}((\vec{h}_t,\vec{\pi}(\vec{h}_t),\vec{\omega}_t),s')$$

where $\vec{\pi}(\vec{h}_t) = \langle\pi_1(h_{t,1}),\ldots,\pi_n(h_{t,n})\rangle$.

6.3 Experimental Results

We present experimental results from two benchmark domains: DEC-TIGER [14] and RECYCLING-ROBOTS [1]. We used the initial policy learned by concurrent reinforcement learning (as described above) over 200000 episodes to perform alternating Monte-Carlo Q learning as described in this paper, for values of N ranging from 10 to 1000. There were 2 alternations, i.e., each agent learned best response to the other's policy exactly once. In each experiment, the resulting joint policy after 2 alternations was evaluated to yield



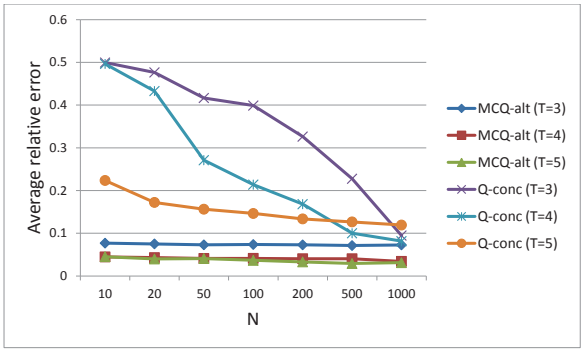


Figure 2: Plots of average relative errors against N in Dec-Tiger (left) and Recycling-Robots (right).

$|v_{jpol} - v_{opt}|/|v_{opt}|$, i.e., the relative error based on known optimal values for horizons 3, 4 and 5. The plots in Figure 2 show these relative errors averaged over 50 runs. For comparison, we also show the result from concurrent Q-learning (referred to as "Q-conc"), with $\alpha = 0.01$, $\epsilon = 0.005$, which were found to produce best results in the selected settings. Table 1 also shows the average relative error rates with the initial policy (derived by concurrent learning), to verify that MCQ-Alt does indeed improve these policies.

Each setting of N and T makes MCQ-Alt finish in a certain number of episodes, say $e_{N,T}$, i.e., until the empty history becomes "Known" in each alternation. The average of these numbers of episodes over all agents and all runs is used to determine the length of the Q-conc runs. The average relative error of Q-conc for a given N and T is reported at the end of (average) $e_{N,T}$ episodes.

In Figure 2 (left) for Dec-Tiger, first we note that horizons 3 and 4 are solved accurately with $N \geq 200$ and 1000 respectively, by MCQ-alt. Q-conc solves horizon 3 accurately with a number of episodes corresponding to $N = 1000$, but is unable to solve horizon 4. Neither achieves 0 error for horizon 5. MCQ-Alt is also clearly more efficient than Q-conc. More importantly, we see that for a given N, the relative error increases with increasing horizon. This is clear with MCQ-alt, but not so clear with Q-conc with even a hint of non-convergence (error increases for $T = 4$). For MCQ-alt, this implies that N needs to increase to produce the same error on increasing horizons. This is direct evidence for the claim made earlier in this paper, although the rate at which N *needs* to increase falls short of the $O(T^3|\Omega|^{2T})$ rate established in this paper. This is explained by the fact that $O(T^3|\Omega|^{2T})$ is a *sufficient* rate for *value convergence*; it is not *necessary*, and policies can converge sooner.

	Dec-Tiger			Recycling-Robots		
	T=3	T=4	T=5	T=3	T=4	T=5
Initial policy relative error	2.16	2.67	2.42	0.37	0.39	0.33
% unreachable histories	0	0	0	27.9	39.8	47.8

Table 1: Relative errors of initial policies, and the proportion of unreachable histories.

In Figure 2 (right) for Recycling-Robots, we see something more interesting. Although MCQ-Alt is still more efficient that Q-conc, the relative errors now *decrease* with increasing horizon, for a given N. This is true with both MCQ-Alt and Q-conc. This is partly due to the fact that there are many unreachable histories in this domain. Table 1 shows the increasing proportion of unreachable histories in Recycling-Robots with increasing T, which suggests that K (and hence the error) grows much slower than Theorem 3 assumes. As it turns out, N does not need to grow with increasing horizons for small errors in this domain. Instead, the increasing values of $e_{N,T}$ even for a fixed N are sufficient to actually *reduce* the average errors with increasing T, as seen in Figure 2 (right). However, it must be noted that alternating best response learning also almost always converges to local optima in this domain, so 0 average error was never observed.

		Dec-Tiger		Recycling-Robots	
	ϵ	$T=3$	$T=4$	$T=3$	$T=4$
Episode ratio	0.01	0.975	0.0138	0.44	0.056
	0.02	0.864	0.0039	0.284	0.0069
	0.03	0.189	0.0039	0.184	0.0023
	0.04	0.055	0.0039	0.11	0.0018
	0.05	0.025	0.0039	0.082	0.0017
Δ (relative error)	0.01	0	2.077	0.01	0
	0.02	0	2.17	0.01	0
	0.03	0	2.17	0.01	0.01
	0.04	0.098	2.17	0.01	0.01
	0.05	0.137	2.17	0	0.02

Table 2: Comparison of MCQ-Alt-ARH and MCQ-Alt.

In Table 2 we show the results from MCQ-Alt-ARH, in the two domains for $T = 3, 4$ only. Here "Episode ratio" stands for the ratio of the average number of episodes (over all agents and all runs) needed by MCQ-Alt-ARH, to that needed by MCQ-Alt, both for $N = 1000$. Δ (relative error) stands for the absolute difference between the average relative errors of the two algorithms for the same settings. As one would expect, the relative number of episodes needed to terminate MCQ-Alt-ARH falls with increasing ϵ, while the error in the policy value increases. Interestingly, the impact on the quality of policy is mostly small except in Dec-Tiger with $T = 4$. In the other settings, it appears that the adjustment for rare histories does indeed lead to significant savings in learning time with relatively little impact on the policy quality. More experiments need to be conducted to ascertain the beneficial impact of MCQ-Alt-ARH on a broad range of problems, and its relation to domain characteristics.

7. CONCLUSION

We have presented a distributed Monte Carlo based reinforcement learning algorithm for solving decentralized POMDPs approximately. Agents alternate in learning best responses which, if accurate enough, is guaranteed to lead to a locally optimal Nash equilibrium. We have derived the sample complexity that guarantees arbitrarily accurate best response policies, and shown empirically that 2 alternations of best response learning can produce (near) optimal joint policies in some benchmark problems. A slight modification of the algorithm was also proposed to account for rare histories, and it appears to significantly reduce the learning time, with relatively little impact on policy quality in most settings. In the future, more judicious use of samples with a variable N will be explored, and a more elaborate investigation into the convergence behavior of concurrent learning will be undertaken. An important future goal is to apply our approach to real-world Dec-POMDP problems where models are unavailable, e.g., two inexpensive (limited sensing and no communication capability) robots carrying an object together. The main challenge in such an application is the fact that observations will be continuous valued, for which naive (discretization) as well as sophisticated (function approximation) techniques will be investigated.

8. ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for helpful feedback. This work was supported in part by the U.S. Army under grant #W911NF-11-1-0124.

9. REFERENCES

[1] C. Amato, D. Bernstein, and S. Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *Proc. UAI*, 2007.

[2] Anonymous. Blind review. Technical report. Available at http://tinyurl.com/86um3on.

[3] B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju. Sample bounded distributed reinforcement learning for decentralized pomdps. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Canada, July 2012. To appear.

[4] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

[5] R. I. Brafman and M. Tennenholtz. R-max - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213 – 231, 2002.

[6] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Articial Intelligence*, pages 183–188, San Jose, CA, 1992. AAAI Press.

[7] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 746–752, Menlo Park, CA, 1998. AAAI Press/MIT Press.

[8] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, 1:136–143, 2004.

[9] S. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

[10] L. Kraemer and B. Banerjee. Informed initial policies for learning in dec-pomdps. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence Student Abstract and Poster Program*, Toronto, Canada, July 2012. To appear.

[11] L. Kraemer and B. Banerjee. Informed initial policies for learning in dec-pomdps. In *Proceedings of the AAMAS-12 Workshop on Adaptive Learning Agents (ALA-12)*, Valencia, Spain, June 2012. To appear.

[12] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

[13] N. Meuleau, L. Peshkin, K. Kim, and L. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proc. UAI*, pages 427–436, 1999.

[14] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 705–711, Acapulco, Mexico, 2003.

[15] F. A. Oliehoek, M. T. J. Spaan, J. S. Dibangoye, and C. 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.

[16] S. Seuken. 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.

[17] G. Shani, R. Brafman, and S. Shimony. Model-based online learning of POMDPs. In *Proceedings of the European Conference on Machine Learning (ECML)*, volume Lecture Notes in Computer Science 3720, pages 353–364. Springer, 2005.

[18] M. T. J. Spaan, F. A. Oliehoek, and C. 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.

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

[20] D. Szer and F. Charpillet. Point-based dynamic programming for dec-pomdps. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1233–1238, Boston, MA, 2006.

[21] C. Zhang and V. Lesser. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proc. AAAI-11*, San Francisco, CA, 2011.