Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Pruning for Monte Carlo Distributed Reinforcement Learning in Decentralized POMDPs

Bikramjit Banerjee
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39402
Bikramjit.Banerjee@usm.edu
http://www.cs.usm.edu/~banerjee

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*. Recently a Monte Carlo based distributed reinforcement learning approach was proposed, 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. However, this Monte Carlo approach has a large sample complexity, which we address in this paper. In particular, we propose and analyze a modified version of the previous algorithm that adaptively eliminates parts of the experience tree from further exploration, thus requiring fewer samples while ensuring unchanged confidence in the learned value function. Experiments demonstrate significant reduction in sample complexity – the maximum reductions ranging from 61% to 91% over different benchmark Dec-POMDP problems – with the final policies being often better due to more focused exploration.

Introduction

Decentralized partially observable Markov decision processes (Dec-POMDPs) offer a powerful modeling technique for realistic multi-agent coordination and decision making under uncertainty. Given the computational complexity of Dec-POMDPs (Bernstein et al. 2002), exact solution techniques for finite horizon problems require significant time and memory resources (Szer and Charpillet 2006; Oliehoek et al. 2010; Spaan, Oliehoek, and Amato 2011). Additionally, Dec-POMDP solvers are mostly *centralized* (see (Emery-Montemerlo et al. 2004) for an exception) and assume prior knowledge of the *model*. 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. Recently, reinforcement learning techniques have been applied to decentralized planning (Zhang and Lesser 2011; Banerjee et al. 2012), that overcome both of these limitations. That is, instead of a single program computing the optimal joint policy, with the full knowledge of the problem parameters, each agent learns its own policy, without prior knowledge of the problem parameters.

Banerjee et. al. (2012) proposed a reinforcement learning approach – MCQ-ALT –where agents take turns to learn mutual best responses, and theoretically established the sample complexity of each best response learning phase. They also showed experimentally that (near) optimal policies were learned in two benchmark problems. However, this approach has a large sample complexity, owing partly to the fact that a learner devotes equal amounts of resources to all parts of the policy space. We argue that this is wasteful, and that a learner can utilize the knowledge acquired so far to bias its future focus, thus eventually requiring fewer samples to learn. So we propose a modification –Iterative MCQ-ALT, or IMCQ-ALT– to perform this biased exploration. Our approach prunes actions opportunistically from further exploration. Most importantly, we theoretically establish a key parameter of the algorithm that leads to pruning *without affecting the confidence in the learned value function, compared to* MCQ-ALT. This means, even though IMCQ-ALT requires fewer samples, it guarantees the same maximum error and minimum confidence as MCQ-ALT. We evaluate both MCQ-ALT and IMCQ-ALT on 4 benchmark Dec-POMDP problems and experimentally show that while IMCQ-ALT often produces smaller errors in output policy values than MCQ-ALT by virtue of more focused exploration, IMCQ-ALT also uses significantly less samples to do so. The maximum advantage in sample complexity of IMCQ-ALT ranges from 61% to 91% over the 4 benchmark domains studied in our experiments.

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.
- S is a finite set of environment states that are not directly observable.
- $A = \times_i A_i$ is the (product) 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 transition to state $s' \in S$ when joint action $\vec{a} \in A$ is executed 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 (product) 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 (i.e., a *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, i.e., $\pi_i : (A_i \times \Omega_i)^t \mapsto A_i$.

Reinforcement Learning

Reinforcement learning (RL) problems are modeled as *Markov Decision Processes* or MDPs (Sutton and Barto 1998). 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 resulting from 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 $Q(s,a,t) = R(s,a) + \max_{\pi} \sum_{s'} P(s,a,s')V^{\pi}(s',t+1)$. 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. 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 sample-based online dynamic programming, e.g., *Q-learning*. In this paper, we build on a previous algorithm –MCQ-ALT – that is a *semi*-model (explained later) based learning algorithm for Dec-POMDPs.

Turn-taking RL for Dec-POMDPs

Banerjee et. al. (2012) presented a Monte Carlo approach, called Monte Carlo Q Alternating (MCQ-Alt), where agents take turns to learn best response to each others' policies, using an R-Max (Brafman and Tennenholtz 2002) like approach to learn the best response Q-values. Although the agents are unaware of functions P, R, O, MCQ-ALT assumes 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. Despite partial observability, the capability of signalling is a reasonable assumption because this may actually be accomplished by a third party present only during learning, but not during the execution of learned policies. Therefore, agents must still learn not to rely on communication, otherwise they will learn policies that cannot be executed in the real Dec-POMDP.

One reason for our focus on MCQ-ALT is that it has a well-defined *stopping criterion*. Since agents alternate in learning best responses, it is necessary for each agent to know precisely when it is done learning, so that the other agent can begin. Alternative best response learning algorithms, such as straightforward Q-learning (Sutton and Barto 1998), or the more recent and highly successful Upper Confidence bound for Trees (UCT) (Kocsis and Szepesvri 2006) do not have well-defined stopping criteria, making them unsuitable for turn taking learners. In fact, our preliminary experiments imposing a stopping criterion on UCT did not yield usable results.

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

$$Q^*_t(h_t,a|\pi) = R^*_t(h_t,a|\pi) + \sum_{\omega} H^*_t(h_t,a,h_{t+1}|\pi) \cdot \max_b Q^*_{t+1}(h_{t+1},b|\pi) \quad (1)$$

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 $\pi_\ell(h_t) = \arg\max_a Q^*_t(h_t,a|\pi)$. The unknown functions R^*_t and H^*_t represent *true* level-t reward and history transition functions for the learner, given by

$$R^*_t(h_t,a|\pi) = \sum_{s,h_-} P(s|h_t,h_-)P(h_-|h_t,\pi)R(s,\vec{a}) \quad (2)$$

$$H^*_t(h_t,a,h_{t+1}|\pi) = \sum_{s,s',h_-} P(s|h_t,h_-)P(h_-|h_t,\pi) \cdot P(s'|s,\vec{a}) \sum_{\omega_-} O(\vec{\omega}|s',\vec{a}) \quad (3)$$

where h_- is the (unobservable) 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

Algorithm 1 MCQ-ALT(N)

1: **repeat**
2: $h \leftarrow \emptyset$
3: $a \leftarrow$SELECTACTION(h)
4: Execute a and receive r, ω
5: **for** $t \leftarrow 1 \ldots T-1$ **do**
6: $b \leftarrow$STEP(h, a, ω, r)
7: $h \leftarrow (h, a, \omega)$
8: Execute b and receive r, ω
9: $a \leftarrow b$
10: **end for**
11: ENDEPISODE(h, N)
12: **until** $Known(\emptyset) = True$

Algorithm 2 SELECTACTION(h)

1: **if** $Known(h) = True$ **then**
2: $a \leftarrow \arg\max_{b \in A} Q(h, b)$
3: **else**
4: $A' \leftarrow A \setminus \{a | \forall \omega,\ Known((h, a, \omega)) = True\}$
5: $a \leftarrow \arg\min_{b \in A'} frequency(h, b)$
6: **end if**
7: $frequency(h, a) \leftarrow frequency(h, a) + 1$
8: Return a

Algorithm 3 STEP(h, a, ω, r)

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

Algorithm 4 ENDEPISODE(h, N)

1: **if** $frequency(h, a) > N,\ \forall a \in A$ **then**
2: $Known(h) \leftarrow True$
3: QUPDATE(h)
4: **end if**
5: **while** $h \neq \emptyset$ **do**
6: Let $h = (h', a, \omega)$
7: $Remaining(h') \leftarrow Remaining(h') \setminus \{(a, \omega)\}$
8: **if** $Remaining(h') = \emptyset$ **then**
9: $Known(h') \leftarrow True$
10: QUPDATE(h')
11: **else**
12: break
13: **end if**
14: $h \leftarrow h'$
15: **end while**

the right hand sides of equations 2, 3, and must estimate R^* and H^* (as $\hat{R}$ and $\hat{H}$) solely from its own experience of executing actions and receiving observations and rewards.

MCQ-Alt: The Algorithm

An MCQ-ALT learner records immediate rewards and history transitions at every history encountered, providing samples of R^* and H^* given in equations 2, 3 respectively. These samples are incorporated into running averages to maintain estimates $\hat{R}$ and $\hat{H}$ respectively. Since it estimates intermediate functions, R^* and H^*, instead of the model parameters, MCQ-ALT is called *semi*-model based. For histories of length $T-1$ (i.e., full length), h_{T-1}, if the pair (h_{T-1}, a) has been encountered

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

times, then the learner sets $Q_{T-1}(h_{T-1}, a)$ to the average of the immediate rewards received (i.e., $\hat{R}_{T-1}(h_{T-1}, a)$), and marks (h_{T-1}, a) as "Known". In the above expression, β measures the amount of memory required by the learner, and ρ measures the likelihood of the rarest history. Equation 4 ensures that the maximal error in the Q values on empty history does not exceed α with a probability at least $1 - \Delta$ (Banerjee et al. 2012). The learner proceeds to check if (h_{T-1}, a) is "Known" for every a, in which case 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 (Algorithm SELECTACTION) is as follows. 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, a, (ties broken randomly) is executed, such that (h, a, ω) is not "Known" for at least one ω. The best response learning algorithm is shown as Algorithm 1 and its subroutines (Algorithms 2– 5). The learner freezes its current policy when the empty history is marked "Known" (line 12, Algorithm 1), and signals to the other agent to start learning, while it executes its current policy without exploration.

Iterative MCQ-Alt

In learning the best reponse, an MCQ-ALT learner attempts to cover every (h, a) encountered equally well, to guarantee arbitrarily small errors in the value function. Figure 1 shows a learner's entire possible experience tree 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.

Apart from rare histories, there are at least two other reasons why the value function may not need to be accurate

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**

to an arbitrary degree: (1) policies usually converge long before value functions, and (2) most importantly, most part of the learner's experience tree does not appear in its optimal policy. In Figure 1, a part of the tree that constitutes a valid policy is shaded. Note that a policy in this tree can be constructed by adopting exactly one action child under an observation node, but all observation children nodes under a selected action node must be adopted. If the policy in Figure 1 was the optimal policy, then there would be little benefit to exploring the leaves outside the shaded area to the extent of N samples each. However, without some exploration of such leaves, it is impossible to determine that they can be discarded with sufficient confidence. The main idea of Iterative MCQ-ALT (IMCQ-ALT) is to strike a balance between these conflicting requirements. In particular, we deduce and utilize a criterion that can adaptively discard actions (at different levels of the experience tree) from further exploration, *without affecting the confidence in the Q-values that yield the best response policy*. Thus the quality of best response policy produced by both MCQ-ALT and IMCQ-ALT are the same, while IMCQ-ALT returns the best response policy faster than MCQ-ALT.

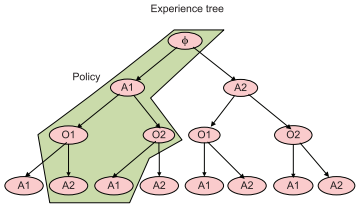

Figure 1: A learner's experience tree.

IMCQ-ALT performs $T+1$ passes of MCQ-ALT learning, where in the τth pass ($\tau = T \ldots 0$) it adds $N/(T+1)$ samples, i.e., accumulates a total of $(T+1-\tau)N/(T+1)$ samples, for each of the (increasingly selective) leaves in the experience tree. In the τth pass ($\tau = T \ldots 1$), IMCQ-ALT considers the actions at the $(\tau-1)$th level of a policy tree, that appear suboptimal by its current estimate, and if an action meets a confidence preserving criterion then that action is removed (lines 5–6, Algorithm 7). This allows actions at lower levels of the experience tree to be removed with relatively fewer samples accumulated, while requiring that a larger number of samples be seen at the leaves to remove actions at a higher level. Removal of an action at a higher level discards an entire subtree from further exploration and focuses learning on more fruitful parts of the experience tree. The last pass ($\tau = 0$) ensures a full collection of N samples for each leaf that survives to this step, but performs no pruning. The new learning algorithm, IMCQ-ALT, is shown in Algorithm 6. All subroutines of MCQ-ALT must also be modified to use an adaptive set of actions, $A(h)$ for history h, instead of the fixed action set A. While $A(h)$ is initialized to A for any observed h, actions can be removed later by a modified version of Algorithm 5, as shown in Algorithm 7. Algorithm IQUPDATE accepts an extra parameter, τ, to ensure that the action it removes is at the proper level (i.e., $\geq \tau - 1$). This would have to be passed to it from IMCQ-ALT, via ENDEPISODE (line 14 in Algorithm 6) which, then, must also accept this extra parameter.

Algorithm 6 IMCQ-ALT(N)

1: **for** $\tau \leftarrow T \ldots 0$ **do**
2: $N_\tau \leftarrow (1 - \tau/(T+1))N$
3: Clear $Remaining$ & $Known$
4: **repeat**
5: $h \leftarrow \emptyset$
6: $a \leftarrow$SELECTACTION(h)
7: Execute a and receive r, ω
8: **for** $t \leftarrow 1 \ldots T-1$ **do**
9: $b \leftarrow$STEP(h, a, ω, r)
10: $h \leftarrow (h, a, \omega)$
11: Execute b and receive r, ω
12: $a \leftarrow b$
13: **end for**
14: ENDEPISODE(h, N_τ, τ)
15: **until** $Known(\emptyset) = True$
16: **end for**

Algorithm 7 IQUPDATE(h, τ)

1: Lines 1–6 from QUPDATE
2: $a^* \leftarrow \arg\max_{a\in A(h)} Q(h,a)$
3: **for** $a \in A(h) \setminus a^*$ **do**
4: $\epsilon \leftarrow Q(h,a^*) - Q(h,a)$
5: **if** $(\epsilon > \alpha(\tau))$ and $(|h| \geq \tau - 1)$ **then**
6: $A(h) \leftarrow A(h) \setminus a$
7: **end if**
8: **end for**

Another difference between IMCQ-ALT and MCQ-ALT is that the greedy choice in SELECTACTION (lines 1–3 in Algorithm 2) is omitted in all but the last pass, in order to keep collecing samples below un"Known" histories to benefit future passes.

Among related techniques, Monte Carlo Tree Search (MCTS) algorithms perform similar planning for fully observable MDPs (Browne et al. 2012), or their multi-agent variants for games. The prevalent pruning techniques applied to MCTS are different from ours, and depend on either domain knowledge or visitation frequencies (Huang et al. 2010).

Analysis

Banerjee et. al. (2012) showed that the sample complexity of each best response learning phase is $O(T^3|A|^T|\Omega|^{3T-1})$ which is consistent with the size of the belief space of alternate best response as argued in (Nair et al. 2003). The belief space consists of not only the unobservable state space, but also the unobservable space of other agents' policies.

We focus on sample complexity analysis of the best response learning process. We denote levels in policy tree as subscripts and the pass number as superscripts. E.g., an estimated Q value at level t in the policy tree computed in pass τ in IMCQ-ALT is Q_t^τ.

Derivation of $\alpha(\tau)$

In the following analysis we will use the max norm function, i.e., $\|f-g\|$ to represent $\max_x |f(x)-g(x)|$. In (Banerjee et al. 2012) the dependence of the error in Q functions on the errors in estimates $\hat{H}$ and $\hat{R}$ was established. In our case, this error depends, additionally, on the pruning error accumulated on all estimates that eventually back up into a certain Q value. We first establish this dependence of the error in the Q functions on the errors in our estimates $\hat{H}$ and $\hat{R}$, as well as on the pruning error. We refer to the maximum pruning error at level t at the end of pass τ as

$$\epsilon_{A,t}^\tau \triangleq \begin{cases} \max_h |\max_{a\in A(h)} Q_t^*(h,a) - \max_{a\in A} Q_t^*(h,a)|, & t \geq \tau-1 \\ 0, & 0 \leq t \leq \tau-2 \end{cases}$$

where $|h| = t$. Note that passes τ lie in the range $T \ldots 0$, while t (the length of histories) lie in the range $0 \ldots T-1$. Therefore, in pass τ, no pruning could have happened at levels $t \leq \tau-2$.

Lemma 1. *Suppose $\|\hat{H}_t^\tau - H_t^*\| \leq \epsilon_H^\tau$, and $\|\hat{R}_t^\tau - R_t^*\| \leq \epsilon_R^\tau$ for all t in pass τ, then at any step $0 \leq t \leq \tau-2$*

$$\|Q_t^\tau - Q_t^*\| \leq (T-t)\epsilon_R^\tau + (T-t-1)|\Omega|R_{\max}\epsilon_H^\tau + \sum_{t'=\tau-1}^{T-1} \epsilon_{A,t'}^\tau$$

The most important intuition delivered by Lemma 1 is that although the number of occurrences of pruning error may multiply exponentially below a node, the total contribution of this error is linear bounded (i.e., $\sum_{t'=\tau-1}^{T-1} \epsilon_{A,t'}^\tau$). It is also worth noting that pruning error at a lower level can become costless if it falls in the subtree under a suboptimal action at a higher level. Thus our estimate of the total pruning error is truly an upper bound. Note that the errors ϵ_R^τ and ϵ_H^τ can be reduced with increased sampling, but $|\max_{A(h)} Q_t^*(h,.) - \max_A Q_t^*(h,.)|$ cannot be "reduced" the same way because it is not directly based on Q estimates. Instead, we seek to bound the *likelihood* of this quantity being non-zero. We will show that ultimately, this probability can, in fact, be bounded by errors in Q estimates.

Note that $\max_A Q_t^*(h,.) - \max_{A(h)} Q_t^*(h,.) \geq 0$, and the only case when it is non-zero is when $a^* = \arg\max_A Q_t^*(h,.)$ is wrongly removed from $A(h)$. In other words, $Q_t^\tau(h,a) - Q_t^\tau(h,a^*) > \alpha(\tau)$ for some $a \in A$, as well as $t \geq \tau-1$, according to line 5 of Algorithm 7. We want to ensure that when an action a^* is removed by the above test, it is highly unlikely that it is the optimal action. We see that $Q_t^*(h,a^*) - Q_t^*(h,a)$

$$\begin{aligned} &= Q_t^*(h,a^*) - Q_t^\tau(h,a^*) + Q_t^\tau(h,a^*) - Q_t^\tau(h,a) + Q_t^\tau(h,a) - Q_t^*(h,a) \\ &< (Q_t^*(h,a^*) - Q_t^\tau(h,a^*)) - \alpha(\tau) + (Q_t^\tau(h,a) - Q_t^*(h,a)) \\ &< |Q_t^\tau(h,a^*) - Q_t^*(h,a^*)| - \alpha(\tau) + |Q_t^\tau(h,a) - Q_\tau^*(h,a)| \end{aligned} \quad (5)$$

Therefore, to limit the likelihood of $Q_t^*(h,a^*) - Q_t^*(h,a) > 0$ for an action a^* that is being removed, it is sufficient to limit the following probabilities

$$P(|Q_t^\tau(h,a^*) - Q_t^*(h,a^*)|) > \alpha(\tau)/2)$$
$$P(|Q_t^\tau(h,a) - Q_t^*(h,a)|) > \alpha(\tau)/2)$$

for $t \geq \tau-1$. In general, it is sufficient to require that this holds for every $Q_t^\tau|_{t\geq\tau-1}$, i.e., the following probability is limited

$$P(\|Q_t^\tau - Q_t^*\| > \alpha(\tau)/2) \text{ for } t \geq \tau-1.$$

(Banerjee et al. 2012) established equation 4 as a sufficient input to MCQ-ALT to ensure $P(\|Q_0 - Q_0^*\| \leq \alpha) \geq 1-\Delta$. We assume that this value of N is input to IMCQ-ALT as well. In this paper we establish how to set $\alpha(\tau)$ in IMCQ-ALT such that $P(\|Q_0^0 - Q_0^*\| \leq \alpha) \geq 1-\Delta$ is preserved after the final pass ($\tau = 0$). We first define the function

$$\delta(t,\tau) \triangleq \left(\frac{\tau+\gamma}{\gamma^{t+1}}\right)\Delta \quad (6)$$

for some $\gamma > 1$. Note that $\delta(0,\tau) = \left(\frac{\tau+\gamma}{\gamma}\right)\Delta$ and $\delta(0,0) = \Delta$. Using this function, we wish to set

$$P(\|Q_t^\tau - Q_t^*\| > \alpha(\tau)/2) < \delta(t,\tau), \quad (7)$$

so that ultimately, $P(\|Q_0^0 - Q_0^*\| > \alpha) < \delta(0,0) = \Delta$.

Theorem 2. *To ensure that $P(\|Q_0^0 - Q_0^*\| \leq \alpha) \geq 1-\Delta$, it is sufficient to set*

$$\alpha(\tau) = 2\alpha\sqrt{\left(\frac{T+1}{T+1-\tau}\right)\left(\frac{\ln(16|S|^2|\Omega|^T\beta/\delta(0,\tau)(1-\lambda_\tau))}{\ln(16|S|^2|\Omega|^T\beta/\Delta)}\right)}$$

for some $0 < \lambda_\tau < 1$, given the choice of γ in equation 6. β is the same as defined in (Banerjee et al. 2012).

Proof (sketch): From equation 7, $P(\|Q_0^\tau - Q_0^*\| > \alpha(\tau)/2) < \delta(0,\tau)$. To achieve this $\alpha(\tau)/2$ Q-error, we need to bound the likelihoods $P(\|\hat{R}_t^\tau - R_t^*\| > \alpha(\tau)/4T)$, $P(\|\hat{H}_t^\tau - H_t^*\| > \alpha(\tau)/4(T-1)|\Omega|R_{\max})$ and $P(\sum_{t=\tau-1}^{T-1} \epsilon_{A,t}^\tau > 0)$. Based on equation 5, it can be shown that $P(\epsilon_{A,t}^\tau > 0) \leq P(\|Q_t^\tau - Q_t^*\| > \alpha(\tau)/2)$. Consequently, $P(\sum_{t=\tau-1}^{T-1} \epsilon_{A,t}^\tau > 0) < \lambda_\tau\delta(0,\tau)$, where $\lambda_\tau = \frac{1}{\gamma^{\tau-1}} - \frac{1}{\gamma^T}$. This allows a total probability mass of $(1-\lambda_\tau)\delta(0,\tau)$ to bound the likelihoods of $\hat{R}$ and $\hat{H}$ errors. Then following the proof strategy in (Banerjee et al. 2012), the result can be obtained. □

As $\alpha(\tau)$ decreases, pruning becomes more likely (line 5 in Algorithm 7). Therefore, one would expect that $\alpha(\tau)$ should decrease from one pass to the next, because the next pass uses a greater number of accumulated samples at the leaves, and can hence prune with greater confidence. This is borne out from the expression of $\alpha(\tau)$ in the above theorem.

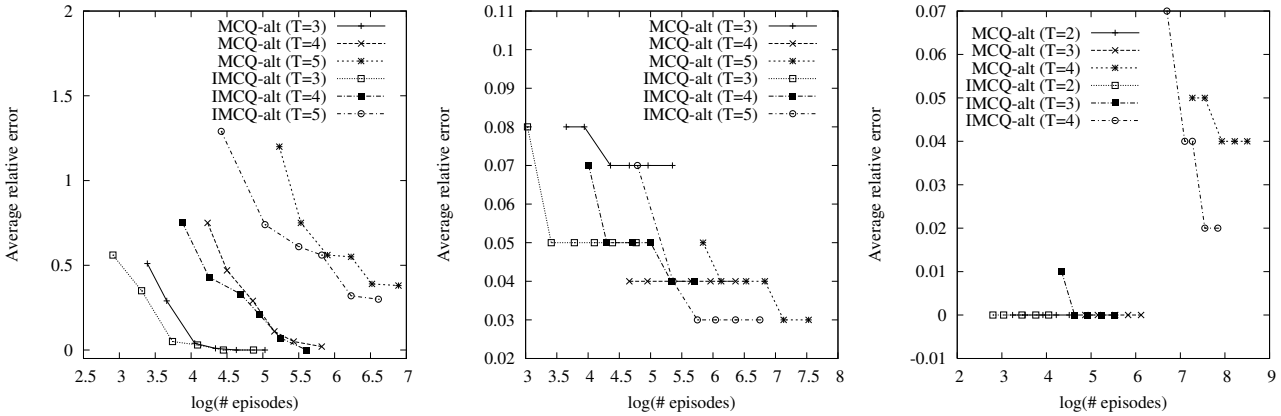

Figure 2: Mean relative error plots for DEC-TIGER (left),RECYCLING-ROBOTS (middle) and BOX-PUSHING (right).

Experimental Evaluation

In order to establish the validity and usefulness of the pruning criterion of IMCQ-ALT, we experimented in 4 benchmark Dec-POMDP tasks, DEC-TIGER (Nair et al. 2003), RECYCLING-ROBOTS (Amato, Bernstein, and Zilberstein 2007), BOX-PUSHING (Seuken and Zilberstein 2007) and MARS-ROVERS (Amato and Zilberstein 2009). As in (Banerjee et al. 2012), we let the agents first learn initial policies by concurrent reinforcement learning ignoring their observations. Then the agents took turns to learn best response to the others' policies using MCQ-ALT and IMCQ-ALT. Since the same initial policies were used for both versions, we exclude the number of episodes devoted to the initial policy learning ($\leq$ 500,000 episodes in all cases) in the plots.

We used $N = 10,\ 20,\ 50,\ 100,\ 200,\ 500$. We noted the number of episodes ($\geq N|\Omega|^{T-1}|A|^T$) taken by MCQ-ALT and IMCQ-ALT, and the relative value error compared to the known optimal policies, calculated as $\frac{|val(policy_{optimal}) - val(policy_{learned})|}{|val(policy_{optimal})|}$ for each choice of N. We used half of the standard deviation of the rewards as defined in the respective .dpomdp files as the value of α, to calculate $\alpha(\tau)$ according to Theorem 2. Finally, all plot points were averaged over 20 runs each.

Figure 2 (left) shows the result from DEC-TIGER for horizons $T = 3,\ 4,\ 5$. We see that IMCQ-ALT requires significantly fewer samples to return policy errors that are comparable to MCQ-ALT, verifying the main claim of this paper. The saving in number of episodes ranges from 31% to 84% over all $T,\ N$ pairs used. These numbers are calculated as $\frac{(\#episodes_{MCQ-Alt} - \#episodes_{IMCQ-Alt}) * 100\%}{\#episodes_{MCQ-Alt}}$. Furthermore, the errors in general increase with increasing horizon, as reported in (Banerjee et al. 2012). Interestingly, for horizon 4 and especially horizon 5 the final error is lower for IMCQ-ALT which is a benefit of focusing on more promising parts of the experience tree. We call this the "pruning bonus". This bonus would be absent whenever both algorithms converge to the same policy, e.g., for $T = 3$ both converge to the optimal policy.

Figure 2 (middle) shows the result from RECYCLING-ROBOTS for horizons $T = 3,\ 4,\ 5$. Again we see significant pruning, with the number of episodes saved ranging from 70% to 91% over all $T,\ N$ pairs used. However, except for horizon 3, the "pruning bonus" is essentially absent. This is probably because in the other cases there exist suboptimal equilibria very close to the optimal, that attract the alternate hill climbers. In other words, the learners are probably converging to suboptimal equilibria, consequently there is no pruning bonus to be had. Also in this case the plots for the different horizons are not as well separated as in Figure 2 (left). This is a difficulty shared with the next two domains as well, and the reason is small errors across the board.

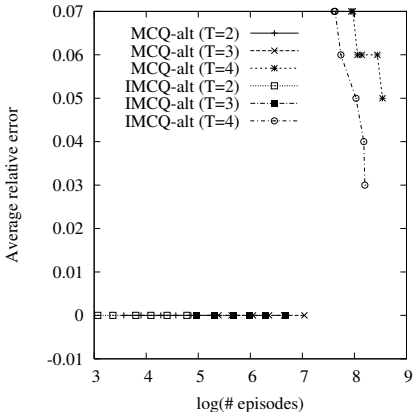

Figure 3: Mean relative error plots for MARS-ROVERS.

Figure 2 (right) shows the result from BOX-PUSHING for horizons $T = 2,\ 3,\ 4$. Both MCQ-ALT and IMCQ-ALT achieve 0 error (i.e., converge to the optimal policy) for $T = 2,\ 3$, and we do see a pruning bonus for $T = 4$. Here IMCQ-ALT saves between 65% and 78% of episodes compared to MCQ-ALT. Similar results are seen in Figure 3, where the number of episodes saved by IMCQ-ALT ranges from 22% to 61%, indicating that MARS-ROVERS is a relatively difficult problem to solve.

Conclusion

In this paper we have proposed an iterative pruning version of a recent distributed Monte Carlo based reinforcement learning algorithm for solving decentralized POMDPs, to yield improved sample complexity without affecting the confidence in the learned policies. We have theoretically established a key parameter of this new algorithm, that guaran-

tees no worse (probabilistic) error rate as the previous algorithm, despite pruning. Experimental results in 4 benchmark domains show (near) optimal policies are learned with significant pruning in most domains, saving upto 91% of the episodes used by the previous algorithm. In the future, more judicious use of samples will be explored, particularly addressing rare histories.

Acknowledgments: We thank the anonoymous reviewers for helpful comments and suggestions. This work was supported in part by the U.S. Army under grant #W911NF-11-1-0124.

References

Amato, C., and Zilberstein, S. 2009. Achieving goals in decentralized POMDPs. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, 593–600.

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

Banerjee, B.; Lyle, J.; Kraemer, L.; and Yellamraju, R. 2012. Sample bounded distributed reinforcement learning for decentralized POMDPs. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 1256–1262.

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

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

Browne, C.; Powley, E.; Whitehouse, D.; Lucas, S.; Cowling, P.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4:1–43.

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

Huang, J.; Liu, Z.; Lu, B.; and Xiao, F. 2010. Pruning in UCT algorithm. In *Proc. Int. Conf. Tech. Applicat. Artif. Intell.*, 177–181.

Kocsis, L., and Szepesvri, C. 2006. Bandit based monte-carlo planning. In *Proceedings of ECML-06*, 282–293.

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

Oliehoek, F. A.; Spaan, M. T. J.; Dibangoye, J. S.; and Amato, C. 2010. Heuristic search for identical payoff bayesian games. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, 1115–1122.

Seuken, S., and Zilberstein, S. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 344–351.

Spaan, M. T. J.; Oliehoek, F. A.; and Amato, C. 2011. 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)*, 2027–2032.

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

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

Zhang, C., and Lesser, V. 2011. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proc. AAAI-11*.