Learning for Multiagent Decentralized Control in Large Partially Observable Stochastic Environments

Miao Liu
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
miaoliu@mit.edu

Christopher Amato
Department of Computer Science
University of New Hampshire
Durham, NH 03824
camato@cs.unh.edu

Emily P. Anesta
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
eanesta@ll.mit.edu

J. Daniel Griffith
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
dan.griffith@ll.mit.edu

Jonathan P. How
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
jhow@mit.edu

Abstract

This paper presents a probabilistic framework for learning decentralized control policies for cooperative multiagent systems operating in a large partially observable stochastic environment based on batch data (trajectories). In decentralized domains, because of communication limitations, the agents cannot share their entire belief states, so execution must proceed based on local information. Decentralized partially observable Markov decision processes (Dec-POMDPs) provide a general framework for modeling multiagent sequential decision making processes in the presence of uncertainty. Although Dec-POMDPs are typically intractable to solve for real-world problems, recent research on macro-actions in Dec-POMDPs has significantly increased the size of problems that can be solved. However, existing methods are confined to tree-based policies in finite-horizon problems, and assume the underlying POMDP models are known a priori. To accommodate more realistic scenarios when the full POMDP model is unavailable and the planning horizon is unbounded, this paper presents a policy-based reinforcement learning approach to learn the macro-action policies represented by Mealy machines. Based on trajectories of macro-actions, observations, and rewards generated by interacting with the environment with hand-coded policies (demonstrations) and random exploration, an expectation-maximization (EM) algorithm is proposed to learn the decentralized macro-action policies, leading to a new framework called POEM (Policy-based EM), which has convergence guarantee for bath learning. The performance of POEM is demonstrated on two domains, including a benchmark navigation-among-movable-obstacle problem, and a newly designed large search and rescue problem. Our empirical study shows POEM is a scalable batch learning method that can learn optimal policies and achieve policy improvement over hand-coded (suboptimal) policies for missions in partially observable stochastic environments.

Keywords: Dec-POMDPs, Reinforcement Learning, Multiagent Planning, Mealy Machine, Monte-Carlo Methods

Acknowledgements

This work was supported in part by the ONR under MURI program award #N000141110688. The Lincoln Laboratory portion of this work was sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States government.

1 Introduction

Multi-agent and multi-robot systems are becoming an important solution to many real-world problems. For example, in disaster situations, such as earthquake, hurricane, or terrorist attacks, it is urgent that trapped survivors can be found and rescued within 48 hours. Otherwise, the chance of finding victims alive decreases substantially [7]. To solve a search and rescue (SAR) problem efficiently, a team of ground vehicles and aerial vehicles have to be deployed to locate, rescue and medically stabilize survivors trapped in hazardous spaces. This SAR problem in its most general form can be formulated as a decentralized partially observable Markov decision process (Dec-POMDP) [2, 12], where a team of agents must cooperate to optimize some global objective in the presence of uncertainty. Moreover, because of possibly limited range of communication, each agent has to make its own decisions based on its own local observations when there are no other agents in the effective communication range. To date, researches have achieved significant progress on solving the cooperative multiagent sequential decision-making problems that arise in numerous applications including transportation [3], extra-planetary exploration [5], and traffic control [14]. However, most Dec-POMDP formulations assume low level state-action granularity and operate with primitive actions which last exactly one time step, so large problems remain intractable.

Recent research has addressed the more scalable macro-action based Dec-POMDP (MacDec-POMDP) case where each agent has macro-actions (temporally extended actions), which may require different amounts of time to execute [4]. However, current MacDec-POMDP methods [4] require knowing domain models a priori, and are confined to tree-based policy representations in finite horizon problems. For many real-world problems, such as SAR, the exact domain model may not be directly available, and the planning horizon might be indefinite. To solve long horizon (or infinite-horizon) MacDec-POMDP problems when the domain models are unknown, we propose a policy-based expectation maximization (POEM) algorithm to learn the macro-action based finite-state controllers (FSCs). POEM adopts a special type of FSC, Mealy machine [1] for policy representations, and performs batch off-policy reinforcement learning (RL) based on trajectories collected by executing hand-coded and exploration policies.

2 Background

A **Dec-POMDP** can be represented as a tuple $\langle \mathcal{N}, \mathcal{A}, \mathcal{S}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma \rangle$, where $\mathcal{N}$ is a finite set of agent indices; $\mathcal{A} = \otimes_n \mathcal{A}_n$ and $\mathcal{O} = \otimes_n \mathcal{O}_n$ respectively are sets of joint actions and observations, with $\mathcal{A}_n$ and $\mathcal{O}_n$ available to agent n. At each step, action $\vec{a} = (a_1, \cdots, a_N) \in \mathcal{A}$ is selected and a joint observation $\vec{o} = (o_1, \cdots, o_N)$ is received; $\mathcal{S}$ is a set of finite world states; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function with $\mathcal{T}(s'|s, \vec{a})$ denoting the probability of transitioning to s' after taking joint action $\vec{a}$ in s; $\Omega : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ is the observation function with $\Omega(\vec{o}|s', \vec{a})$ the probability of observing $\vec{o}$ after taking joint action $\vec{a}$ and arriving in state s'; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function with $r(s, \vec{a})$ the immediate reward received after taking joint action $\vec{a}$ in s; $\gamma \in [0, 1)$ is a discount factor. Because each agent lacks access to other agents' observations, each agent maintains a local policy Ψ_n, defined as a mapping from local observation histories to actions. A joint policy consists of the local policies of all agents. For an infinite-horizon Dec-POMDP with initial belief state b_0, the objective is to find a joint policy $\Psi = \otimes_n \Psi_n$, such that the value of Ψ starting from b_0, $V^{\Psi}(b(s_0)) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t)|b_0, \Psi\big]$, is maximized.

A **MacDec-POMDP** with local macro-action (MA) is defined as a Dec-POMDP where $\mathcal{M}_n$ represents a finite set of MA for each agent n, with $\mathcal{M} = \otimes \mathcal{M}_n$ the set of joint MAs. Here, MAs are represented by options [13]. Given $\mathcal{H}_n$, the observation history of agent n's local option is defined by the tuple: $\mathcal{M}_n = \langle \beta_n^m, \mathcal{I}_n^m, \pi_n^m \rangle$, where $\beta_n^m : \mathcal{H}_n \to [0, 1]$ is a stochastic termination condition, $\mathcal{I}_n^m \in \mathcal{H}_n$ is the initiation set and $\pi_n^m : \mathcal{H}_n \times \mathcal{A}_n \to [0, 1]$ is an option policy for macro-action m. Macro-actions are natural representations for robot or human operation for completing a task (e.g., navigating to a way point or placing an object on a robot).

A **FSC** is a compact way to represent a policy as a mapping from histories to actions. Formally, a stochastic FSC for agent n is defined as a tuple $\Theta_n = \langle \mathcal{A}_n, \mathcal{O}_n, \mathcal{Z}_n, \mu_n, W_n, \pi_n \rangle$, where, $\mathcal{A}_n$ and $\mathcal{O}_n$ are the same as defined in the Dec-POMDP; $\mathcal{Z}_n$ is a finite set of nodes used to represent the policy for agent n; μ_n is the initial node distribution with μ_n^z the probability of agent n initially being in z; W_n is a set of Markov transition matrices with $W_{n,o}^{z,z'}$ denoting the probability of the controller node transiting from z to z' when agent n in z sees observation o; π_n is a set of stochastic policies with $\pi_{n,z}^{a,o}$ the probability of agent n taking action a in z after seeing observation o. This type of FSC is called Mealy machine [1], where an agent's local policy for action selection $\pi_{n,z}^{a,o}$ depends on both current controller node (an abstraction of history) and immediate observation. There is another type of FSCs called Moore machines, where π_n^a only depends on controller node, is also widely used for (Dec-)POMDP policy representations. In contrast to a Moore machine, by conditioning action selections on additional immediate observations, a Mealy machine can use this observable information to initiate a valid macro-action controllers when the MacDec-POMDP model is unknown.

There are several features in MacDec-POMDP frameworks that benefit policy learning. Firstly, observations are determined by the termination condition of macro-actions (although a more general model of high-level observations can be included). Therefore, the observation set can be encoded as $\bar{\Omega} = \{Null, \Omega\}$, where Ω is a set indicating what agents see after the corresponding macro-action termination condition is satisfied, and $Null$ indicates the same macro-action is still in execution. Additionally, controllers stay in the same nodes until current macro-actions are completed, hence the diagonal elements of W corresponding to each $o \in \Omega$ can be set to zero, i.e. $W(z_\tau = z|z_{\tau-1} = z, o_\tau \neq Null) = 0$. Moreover, before a macro-action terminates, the agent stays in the same controller node, i.e. $W(z_\tau = z|z_{\tau-1} = z, o_\tau = Null) = 1$.

A Dec-POMDP planning problem can be transformed into an **inference problem** and then efficiently solved by an EM algorithm. Previous EM methods [8] have achieved success in scaling to larger problems, but these methods require using a Dec-POMDP model both to construct a Bayes net and to evaluate policies. When the exact model parameters $\mathcal{T}$, Ω and $\mathcal{R}$ are unknown, an RL problem must be solved instead. To address this important yet less addressed problem in decentralized domains, a global empirical value function is constructed based on the macro-action, observation and reward trajectories, and the product of local policies of all agents.

Definition 1. *(Global empirical value function) Let* $\mathcal{D}^{(K)}=\{(\vec{o}_0^k \vec{m}_0^k r_0^k \cdots \vec{o}_{T_k}^k \vec{a}_{T_k}^k \; r_{T_k}^k)\}_{k=1}^K$ *be a set of episodes resulting from* N *agents who choose actions according to* $\Psi=\otimes_n \Psi_n$*, a set of arbitrary stochastic policies with* $p^{\Psi_n}(a|h) > 0$*,* $\forall$ *action* a*,* $\forall$ *history* h*. The global empirical value function is defined as* $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big) \overset{def.}{=} \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\gamma^t (r_t^k - R_{min}) \prod_{\tau=0}^t \prod_{n=1}^N p(m_{n,\tau}^k | h_{n,\tau}^k, \Theta_n)}{\prod_{\tau=0}^t \prod_{n=1}^N p^{\Psi_n}(m_{n,\tau}^k | h_{n,\tau}^k)}$ *where* $h_{n,t}^k = (m_{n,0:t-1}^k, o_{n,1:t}^k)$*,* $0 \leq \gamma < 1$ *is the discount and* R_{min} *is the minimum reward.*

Definition 1 provides an off-policy learning objective: given data $\mathcal{D}^{(K)}$ generated from a set of behavior policies Ψ, find a set of decentralized policies $\Theta = \{\Theta_i\}_{i=1}^N$ such that $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is maximized. Here, we assume factorized policy representation $p(\vec{m}_{0:\tau}^k | \vec{h}_{1:\tau}, \Theta) = \prod_{n=1}^N p(m_{n,\tau}^k | h_{n,\tau}^k, \Theta_n)$ to accommodate decentralized policy execution.

3 MacDec-POMDP Policy Learning by Expectation Maximization

Direct maximization of $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is difficult; instead, we maximize the lower bound of the logarithm of $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$. To this end, we apply Jensen's inequality to obtain obtain a lower bound of $\ln \hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ augmented by node sequences $\{\vec{z}_t^k\}$:

$$\ln \hat{V}\big(\mathcal{D}^{(K)};\Theta\big) = \ln \sum\nolimits_{k,t,\vec{z}_{0:t}^k} \frac{\tilde{r}_t^k q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}{K} \frac{p(\vec{m}_{0:t}^k, \vec{z}_{0:t}^k | \vec{o}_{1:t}, \Theta)}{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})} \geq \sum\nolimits_{k,t,\vec{z}_t^k} \frac{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})}{K} \ln \frac{\tilde{r}_t^k p(\vec{m}_{0:t}^k, \vec{z}_{0:t}^k | \vec{o}_{1:t}^k, \Theta)}{q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta})} \overset{def.}{=} \mathrm{lb}(\Theta|\widetilde{\Theta}), \quad (1)$$

where $\tilde{r}_t^k = \gamma^t r_t^k / \prod_{\tau=0}^t p^\Psi(\vec{m}_\tau^k | h_\tau^k), \forall t, k$ are reweighted rewards, and $\{q(\vec{z}_{0:t}^k|\widetilde{\Theta}) \geq 0\}$ satisfy the normalization constraint $\sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{\vec{z}_{0:t}^k} q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta}) = K$ with $\widetilde{\Theta}$ the most recent estimate of Θ.

Theorem 2. *Define* $\mathcal{F} = \Big\{\Theta = \{\Theta_n\}_{n=1}^N$ *with* $\Theta_n = (\mu_n, \pi_n, W_n) : \sum_{u_n=1}^{|\mathcal{Z}_n|} \mu_n^{u_n} = 1, \sum_{m_n=1}^{|\mathcal{M}_n|} \pi_{n,u_n}^{m_n,o_n} = 1, \sum_{u_n=1}^{|\mathcal{Z}_n|} W_{n,o_n}^{v_n,u_n} = 1, v_n = 1 \cdots |\mathcal{Z}_n|, m_n = 1 \cdots |\mathcal{M}_n|, o_n = 1 \cdots |\mathcal{O}_n|\Big\}$*, and* $\Theta^{(m)}$ *be a sequence produced by the iterative update* $\Theta^{(m+1)} = \arg\max_{\Theta \in \mathcal{F}} \mathrm{lb}(\Theta|\Theta^{(m)})$*, where* $\Theta^{(0)}$ *is an arbitrary initialization, then* $\{\Theta^{(m)}\}_{m \geq 0}$ *monotonically increases (1), until convergence to a maxima.*

Algorithm 1 POEM

```
Require: Episodes $\mathcal{D}^{(K)}$ and the number of agents $n$,
 1: while $\Delta\mathrm{lb} < \epsilon$ do
 2:     for $k = 1$ to $K$, $n = 1$ to $N$ do
 3:         Compute action selection prob. $p(m_{n,0:t}^k | o_{n,1:t}^k, \widetilde{\Theta}), \forall t$,
            and forward-backward variables $\alpha_{n,k}$ and $\beta_{n,k}$
 4:     end for
 5:     Compute lb using $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big) = \frac{1}{K} \sum_{k=1}^K \sum_{t=1}^{T_k} \sigma_t^k\big(\widetilde{\Theta}\big)$
 6:     for $n = 1$ to $N$ do
 7:         Compute the $\xi$ and $\phi$ variables, update $\Theta_n$ using (2)
 8:     end for
 9: end while
10: return Parameters of the MacDec-POMDP policy, $\{\Theta_n\}_{n=1}^N$
```

Given the empirical value function in Definition 1 and its lower bound (1), the POEM algorithm is derived to learn the macro-action FSCs. The POEM algorithm is guaranteed to monotonically increase the empirical value function over successive iterations and converges to a local maximum. The convergence property is summarized by theorem 2. The proof for Theorem. 2 is an extension of the single agent case [9], which is omitted here because of space limitations. Algorithmically, the main steps of POEM involve alternating between computing the lower bound of the log empirical value function (E-step) and parameter estimation (M-step). The complete algorithm is summarized in Algorithm 1, and computational details are discussed next.

Computation of Lower Bounds (E-step) Theorem 2 and inequality (1) yield the lower bound $\mathrm{lb}\big(\Theta|\widetilde{\Theta}\big) \geq \ln \hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$, where $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ is the value of the policy parameterized by $\widetilde{\Theta}$ (computed on line 5 in Algorithm 1). Computing $\hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ is equivalent to policy-evaluation, which uses all available episodes with the rewards are reweighted by the action selection probability $p(m_{n,0:t}^k | h_{o,0:t}^k, \widetilde{\Theta})$ to reflect the improved value of the new policy updated in the previous M-step. Note that $\mathrm{lb}(\Theta|\widetilde{\Theta})$ in (1) is maximized when $q_t^k(\vec{z}_{0:t}^k|\widetilde{\Theta}) = \tilde{r}_t^k p(\vec{m}_{0:t}^k, \vec{z}_{0:t}^k | \vec{o}_{1:t}^k, \widetilde{\Theta}) / \hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$, which is equal to $\tilde{r}_t^k p(\vec{a}_{0:t}^k | \vec{o}_{1:t}^k, \widetilde{\Theta}) p(\vec{z}_{0:t}^k | \vec{m}_{0:t}^k, \vec{o}_{1:t}^k, \widetilde{\Theta}) / \hat{V}\big(\mathcal{D}^{(K)};\widetilde{\Theta}\big)$ where $p(\vec{z}_{0:t}^k | \vec{m}_{0:t}^k, \vec{o}_{1:t}^k, \widetilde{\Theta})$ is the joint distribution of controller nodes for all agents, and $\sigma_t^k(\widetilde{\Theta}) \overset{\mathrm{def.}}{=} \tilde{r}_t^k p(\vec{m}_{0:t}^k | \vec{o}_{1:t}^k, \widetilde{\Theta}) = \Pi_{n=1}^N \tilde{r}_t^k p\big(m_{n,0:t}^k | o_{n,1:t}^k, \widetilde{\Theta}\big)$ is a reweighted reward.

Update of Policy Parameters (M-step) After computing the reweighted rewards $\{\sigma_t^k\}$ and the posterior distribution of controller nodes $p(\vec{z}_{0:t}^k | \vec{m}_{0:t}^k, \vec{o}_{1:t}^k, \widetilde{\Theta}), \forall t, \forall k$, the policy parameters is updated by $\Theta = \arg\max_{\Theta \in \mathcal{F}} \mathrm{lb}(\Theta|\widetilde{\Theta})$, subject to normalization constraints. Specifically, let $\xi_{n,k}^{t,\tau}(u_n, v_n) \overset{def.}{=} p(z_{n,\tau}^k = u_n, z_{n,\tau+1}^k = v_n | m_{n,0:t}^k, o_{n,1:t}^k, \widetilde{\Theta}_n)$ and $\phi_{n,k}^{t,\tau}(u_n) \overset{def.}{=} p(z_{n,\tau}^k = u_n | m_{n,0:t}^k, o_{n,1:t}^k, \widetilde{\Theta}_n)$. Expanding the lower bound of $\ln \hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ and keeping the terms related to Θ, we have

$$\mathrm{lb}\big(\Theta|\widetilde{\Theta}\big) \propto \sum\nolimits_{k,t} \sigma_t^k(\widetilde{\Theta}) \sum\nolimits_{n=1}^N \Big\{ \sum\nolimits_{u_n=1}^{|\mathcal{Z}_n|} \phi_{n,k}^{t,0}(u_n) \ln \mu_n^{u_n} + \sum\nolimits_{\tau=0}^t \big[\sum\nolimits_{v_n=1}^{|\mathcal{Z}_n|} \phi_{t,\tau}^{n,k}(u_n) \ln \pi_{n,u_n}^{a_{n,\tau}^k, o_{n,\tau}^k} + \sum\nolimits_{u_n,v_n=1}^{|\mathcal{Z}_n|} \xi_{n,k}^{t,\tau}(u_n, v_n) \ln W_{n,o_{n,\tau+1}^k}^{u_n,v_n} \big] \Big\}.$$

Therefore, an analytic solution to problem $\Theta = \arg\max_{\Theta \in \mathcal{F}} \mathrm{lb}(\Theta|\widetilde{\Theta})$ can be obtained as

$$W_{n,o_n}^{u_n,v_n} \propto \sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{\tau=0}^{t} \sigma_t^k \xi_{n,k}^{t,\tau}(u_n, v_n) \mathbb{I}(o_{n,\tau+1}^k = o_n), \forall n \in \mathcal{N}, u_n, v_n \in \mathcal{Z}_n, a_n \in \mathcal{A}_n, o_n \in \mathcal{O}_n. \quad (2)$$

where is $\mathbb{I}(\cdot)$ is the indicator function. π, μ are updated in similar ways. These updates constitute a policy-improvement step where the reweighted rewards are used to further improve policies.

Both the above steps require $\xi_{n,k}^{t,\tau}(u_n, v_n)$, which are computed based on $\alpha_{n,k}^{\tau} = p(z_{n,\tau}^k | m_{n,0:\tau}^k, o_{n,1:\tau}^k, \widetilde{\Theta}_n)$ and $\beta_{n,k}^{t,\tau} = \frac{p(a_{n,\tau+1:t}^k | z_{n,\tau}^k, o_{n,\tau+1:t}^k, \widetilde{\Theta}_n)}{\prod_{\tau'=\tau}^{t} p(m_\tau^k | h_{n,\tau'}^k, \widetilde{\Theta}_n)}$, $\forall\, n, k, t, \tau$. The (α, β) are forward-backward messages.

4 Experiments

We evaluate the performance of Algorithm 1 on both a benchmark domain of robot navigation and a large domain motivated by SAR.

4.1 A Navigation Among Movable Obstacles (NAMO) Problem

We first consider the NAMO problem, introduced in [4]. Here as shown in Figure 1, both agents are trying to reach a goal location "G" and have to move obstacles in the way. The primitive actions for each robot include move in four directions (up, down, left and right) and a "push" action to attempt to move a box to a specific location. The push action fails and the robot stays in place when the robot is not in front of the box. The small boxes (b_1 and b_2) can be moved by a single robot, and the large box (b_3) requires two robots to push together. Observations are an agent's own location and whether the large box or the same numbered box has been moved. There is noise in both navigation and in box movement: movement is successful with probability 0.9 and pushing the small and large boxes is successful with probability 0.9 and 0.8, respectively. To encourage the robots to reach the goal as quickly as possible, there is a negative reward (-1) when any agent is not in the goal region.

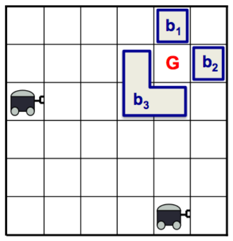

Figure 1: A 6×6 NAMO problem.

There were four macro-actions (option) defined for each agent, including 1) moving to a designated location to push the big box, 2) attempting to push the large box, 3) pushing the designated small box to the corner square, and 4) moving to the goal. The option of moving to the goal is only valid when at least one box has been moved and movement of any box is only valid if the large box and agent's designed box has not yet been moved. Movement options and pushing options terminate with the box successfully or unsuccessfully moved. These options provide high-level choices for the agents to coordinate on this problem, while abstracting away the navigation tasks to option execution.

Here, we want to examine the effect of the behavior policy Π on Algorithm 1's performance. A semi-random policy is applied to collect samples. Specifically, the learning agent is allowed access to episodes collected by taking actions according to a MaDec-POMDP algorithm (Option-based dynamic programming (O-DP)) [4]. Let $\eta\%$ be the probability that the agents follow the O-DP policy and $1 - \eta$ be the probability that the agents take random actions. The use of an O-DP policy is similar to the meta-queries used in [6], where a meta-query consults a domain expert for the optimal action at a particular time step. For each run of algorithm 1, $K = 100$ episodes of 10 steps are used to learn the FSCs with $|\mathcal{Z}_n| = 20, \forall n \in \mathcal{N}$, and the learned policies are evaluated by the discounted ($\gamma = 0.9$) accumulated reward averaged over 100 test episodes of 1000 steps. The results with $\eta = [0, 25, 50, 75, 100]$ are reported in Figure 2, which shows that with a small amount of expert knowledge ($\eta \geq 25\%$), our algorithm is able to recover the optimal policy.

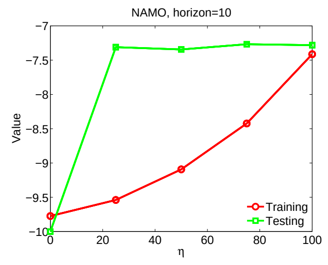

Figure 2: The performance of Algorithm 1 as a function of $\eta\%$ of data generated from an optimal policy.

4.2 A Search and Rescue (SAR) Problem

To further demonstrate the scalability and learning efficiency of the proposed algorithm, we designed a SAR problem involving four heterogeneous agents: an unmanned aerial vehicle (UAV) and three unmanned ground vehicles (UGVs). These agents operate in a 20×10 gridworld (shown in figure 3 (a)), where there are 6 rescue sites with different distance to the muster, which is the assembly area for the SAR agents. There are six victims, each of which is located in a different rescue site. There is a set of initial health states for each victim (not equal to each other) which do not change, though the initial position of each victim does change. Health trajectories are linear. There is a 5% noise in the observations, communication, and UAV's path. The speed of UAV is three times faster than UGV. However, only the UGVs can pick-up victims. The agents receive a positive reward (+1), if a victim transported to the muster is still alive (i.e., has health greater than zero), and receive a negative reward (-1) when one victim dies. For a UGV, there are 1120 observations encoded by the set $\Omega_{UGV} = SL \times SS \times SV \times OL \times OV$, where $SL = \{$site 1,..., site 6, muster$\}$ is a set of self location; $SS = \{$holding victim, not holding victim$\}$ is a set of self states; $SV = \{$has victims needing help, no victims needing help, unknown, critical victims$\}$ is the set of states of the victim at an agent's

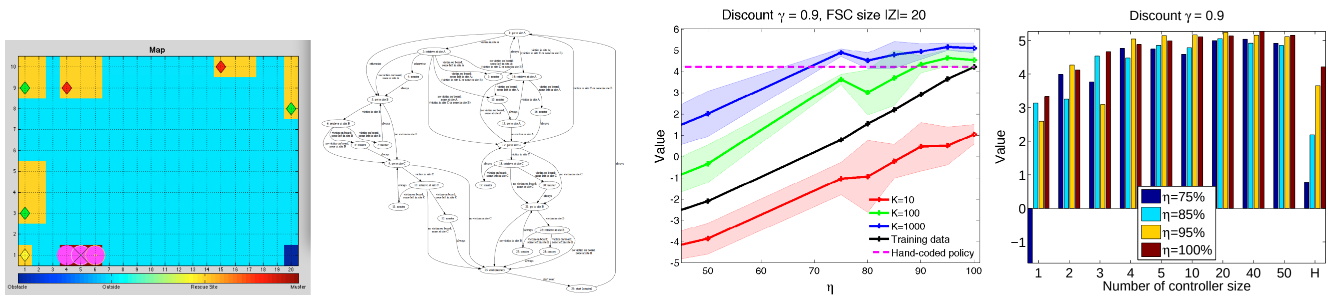

Figure 3: (a) A SAR domain (diamonds represent victims with colors indicate health, circles and cross represent UGV and UAV respectively); (b) A heuristic policy finite state machine for UGV constructed by domain experts; (c) Testing performance using different number of training sample K and percentage of hand-coded policies η; (d) Testing performance of policies with different FSC sizes.

current location; OV is the state of the victim at OL, the other location (from communication), and there are 8 macro-actions, including $\{m_1, \cdots, m_6\}$: go to one of the six sites, m_7: retrieve a victim at one site (retrieve) and m_8: go to muster and drop off victim (muster). For the UAV, there are 560 observations (assuming the UAV cannot hold victims) and 7 macro-actions, including go to muster and go to one of the six sites. All vehicles begin in the muster.

Given the large problem size, unknown POMDP model and the stochasticity in observations and communication, it is difficult to generate an optimal policy with existing solvers. Instead, a domain expert's knowledge is used to create a heuristic controller for exploration. Figure 3(b) provides a visualization of the graph describing the heuristic policy finite state machine. This is the policy for one ground vehicle, where sites A, B, and C represent a subset of the possible sites. The agent-to-site mappings include: a) agent 1: A=6, B=5, C=4; b) agent 2: A=3, B=1, C=2; c) agent 3: A=2, B=1, C=3. The policy for the air vehicle is to cycle linearly through the following sequence: site 1, site 2, muster, site 3, site 4, site 5, site 6, muster. The value of hand-coded policy is estimated based on 1000 runs with randomly placed victims and the mean reward is 4.22 (pink dotted line in Figure 3(c)).

To evaluate POEM's performance on the SAR domain, we test $\eta = [0, 25, 50, 75, 80, 85, 90, 95, 100]$. For each setting of η, 2000 training trajectories (with horizon upper bound set to 200) are generated. When setting $|\mathcal{Z}_n| = 20$ and $K = 1000$, the training time is less than 15min on average. The corresponding testing results are plotted in figure 3(c), from which we can see, a) POEM is able to achieve policy improvement with a sufficient amount of samples ($K > 100$); b) By using 1000 trajectories generated from heuristic policy, POEM is able to achieve a mean value greater than 5, which is higher than the value of hand-coded policy; c) Adding a small amount of noise (5%) to the heuristic policy can help getting a slightly better performance than purely using handed coded policy. Hence the experiments demonstrate POEM can achieve policy improvement and is scalable to large problems. In addition, nine settings of $|\mathcal{Z}_n|$ are used to investigate the influence of the controller size on policy quality. As shown in Figure 3(d), the FSCs learned by POEM render much higher value than the hand-coded policies (with x-axis labeled with H) over a wide range of choices for $|\mathcal{Z}_n|$ (from 5 to 50), indicating robustness of POEM to the size of controller nodes. However, when $|\mathcal{Z}_n|$ is too small, POEM cannot accommodate a good policy. Automatic inference of the necessary size of controllers can be performed using nonparametric methods, such as the hierarchical Dirichlet process [11] and stick-breaking processes [10], which is left for future work.

5 Conclusions

This paper presents an RL method for learning to coordinate multiple agents in macro-action level Dec-POMDPs, an important problem that has not been adequately addressed before. Our method uses previously executed macro-action, observation histories and rewards to improve future decision making without explicitly requiring mission models. An algorithm called POEM is presented for batch learning with convergence guarantee (local optimal). Theoretical analysis and empirical results show the proposed method is a promising tool for solving RL in MacDec-POMDPs.

References

[1] C. Amato, B. Bonet, and S. Zilberstein. Finite-state controllers based on mealy machines for centralized and decentralized pomdps. In *AAAI*, 2010.
[2] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer. Decentralized control of partially observable Markov decision processes. In *CDC*, 2013.
[3] C. Amato, G. D. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *ICRA*, 2015.
[4] C. Amato, G. D. Konidaris, and L. P. Kaelbling. Planning with macro-actions in decentralized pomdps. In *AAMAS*, 2014.
[5] D. Bernstein, S. Zilberstein, R. Washington, and J. Bresina. Planetary rover control as a Markov decision process. In *Int'l Symp. on AI, Robot. & Automation in Space*, 2001.
[6] F. Doshi-Velez, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Artificial Intelligence*, 187:115–132, 2012.
[7] S. Grayson. Search & Rescue using Multi-Robot Systems. `http://www.maths.tcd.ie/~graysons/documents/COMP47130_SurveyPaper.pdf`.
[8] A. Kumar and S. Zilberstein. Anytime planning for decentralized POMDPs using expectation. In *UAI*, pages 2140–2146, 2010.
[9] H. Li, X. Liao, and L. Carin. Multi-task reinforcement learning in partially observable stochastic environments. *JMLR*, 10:1131–1186, 2009.
[10] M. Liu, C. Amato, X. Liao, J. P. How, and L. Carin. Stick-Breaking Policy Learning in DEC-POMDPs. In *IJCAI (to appear)*, 2015.
[11] M. Liu, X. Liao, and L. C. and. Infinite regionalized policy representation. In *Proc. of the 28th Int'l Conf. on Machine Learning*, pages 769–776, 2011.
[12] F. A. Oliehoek. Decentralized POMDPs. In *Reinforcement Learning: State of the Art, Adaptation, Learning, and Optimization*, pages 471–503. Springer, 2012.
[13] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.
[14] F. Wu, S. Zilberstein, and N. R. Jennings. Monte-carlo expectation maximization for decentralized POMDPs. In *IJCAI*, pages 317–403, 2013.