Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Learning for Decentralized Control of Multiagent Systems in Large, Partially-Observable Stochastic Environments

Miao Liu
LIDS, MIT
Cambridge, MA
miaoliu@mit.edu

Christopher Amato
CS Dept., Univ. New Hampshire
Durham, NH
camato@cs.unh.edu

Emily P. Anesta, J. Daniel Griffith
Lincoln Laboratory, MIT
Lexington, MA
{eanesta, dan.griffith}@ll.mit.edu

Jonathan P. How
AeroAstro Dept., MIT
Cambridge, MA
jhow@mit.edu

Abstract

Decentralized partially observable Markov decision processes (Dec-POMDPs) provide a general framework for multiagent sequential decision-making under uncertainty. Although Dec-POMDPs are typically intractable to solve for real-world problems, recent research on macro-actions (i.e., temporally-extended actions) has significantly increased the size of problems that can be solved. However, current methods assume the underlying Dec-POMDP model is known a priori or a full simulator is available during planning time. To accommodate more realistic scenarios, when such information is not available, this paper presents a policy-based reinforcement learning approach, which learns the agent policies based solely on trajectories generated by previous interaction with the environment (e.g., demonstrations). We show that our approach is able to generate valid macro-action controllers and develop an expectation-maximization (EM) algorithm (called Policy-based EM or PoEM), which has convergence guarantees for batch learning. Our experiments show PoEM is a scalable learning method that can learn optimal policies and improve upon hand-coded "expert" solutions.

Introduction

Multi-agent and multi-robot systems represent important solutions to many real-world problems. For example, in disaster situations, such as earthquakes, hurricanes, 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 (Grayson 2014). To solve a search and rescue (SAR) problem efficiently, a team of ground 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) (Oliehoek 2012; Amato et al. 2013), where a team of agents must cooperate to optimize some global objective in the presence of uncertainty. Moreover, each agent has to make decisions based on its own local observations when there are no other agents in the effective communication range. Researchers have made significant progress on numerous related applications, including transportation (Amato et al. 2015b), extra-planetary exploration (Bernstein et al. 2001), and traffic control (Wu, Zilberstein, and Jennings 2013). However, current Dec-POMDP methods have limited scalability.

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 (Amato, Konidaris, and Kaelbling 2014; Amato et al. 2015a). However, current MacDec-POMDP methods require knowing domain models a priori. Unfortunately, for many real-world problems, such as SAR, the domain model may not be available.

To solve Dec-POMDPs when domain models are unavailable or access to simulators is limited (due to high cost or security reasons), previous work (Wu, Zilberstein, and Jennings 2013; Liu et al. 2015) adopted model-free reinforcement learning (RL) strategies and EM algorithms (Dempster, Laird, and Rubin 1977) to optimize finite-state controllers (FSCs). However, these methods are designed for solving primitive action based problems; directly applying them to learn policies from macro-action level data does not guarantee that learned controllers are executable because neither the differing execution times nor the initial conditions of the macro-actions are considered.

To learn policies in the macro-action case, we propose a novel policy-based RL approach to generate valid FSCs from the macro-action level trajectories collected by previous interactions with the environment (e.g. demonstration). Our method first constructs a validity map (VM) which defines the valid macro-actions based on histories that have been seen. Then, a policy-based expectation maximization (PoEM) algorithm, is developed that optimizes the policy subject to the validity map constraints. Our methods adopt a special type of FSC, a Mealy machine (Amato, Bonet, and Zilberstein 2010), for policy representations. The Mealy machine offers a natural representation of the macro-action validity information while also being concise.

Our proposed algorithm is linear in the number of agents and at most quadratic in the problem size, making it scalable to large domains. In practice, these trajectories can be generated by a simulator or a set of real-world experiences that can be provided. This batch data scenario is general and realistic, as it is widely adopted in learning from demonstration (Martins and Demiris 2010). Moreover, the proposed algorithm performs off-policy batch learning to improve the policy with guaranteed convergence. Our experiments show PoEM is a scalable method that can learn optimal policies for a benchmark problem and outperform a hand-coded (suboptimal) policy, when given examples from its execution, in a large domain inspired by SAR problems.

Background and Problem Statement

Dec-POMDPs generalize POMDPs to the multiagent, decentralized setting. Multiple agents operate under uncertainty based on partial views of the world, with execution unfolding over a bounded or unbounded number of steps. At each step, every agent chooses an action (in parallel) based on locally observable information and then receives an observation. The agents share a single reward function based on the actions of all agents, making the problem cooperative, but their local views mean that execution is decentralized.

A **Dec-POMDP** can be represented as a tuple $\langle N, A, S, Z, T, \Omega, R, \gamma \rangle$, where N is a finite set of agent indices; $A = \otimes_n A_n$ and $Z = \otimes_n Z_n$ respectively are sets of joint actions and observations, with A_n and Z_n available to agent n. At each step, a joint action $\vec{a} = (a_1, \cdots, a_{|N|}) \in A$ is selected and a joint observation $\vec{z} = (z_1, \cdots, z_{|N|})$ is received; S is a set of finite world states; $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function with $T(s'|s, \vec{a})$ denoting the probability of transitioning to s' after taking joint action $\vec{a}$ in s; $\Omega : S \times A \times Z \rightarrow [0, 1]$ is the observation function with $\Omega(\vec{z}|s', \vec{a})$ the probability of observing $\vec{o}$ after taking joint action $\vec{a}$ and arriving in state s'; $R : S \times A \rightarrow \mathbb{R}$ 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 π_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 state s_0, the objective is to find a joint policy $\pi = \otimes_n \pi_n$, such that the value of π starting from s_0, $V^{\pi}(s_0) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t)|s_0, \pi\big]$, is maximized. Specifically, given $h_t = \{a_{0:t-1}, z_{0:t}\} \in H_n$, the history of actions and observations up to t, the policy π_n probabilistically maps h_t to a_t: $H_n \times A_n \rightarrow [0, 1]$.

A **MacDec-POMDP** with (local) macro-action extends the MDP-based options (Sutton, Precup, and Singh 1999) framework to Dec-POMDPs. Formally, a **MacDec-POMDP** is defined as a tuple $\langle N, A, M, S, Z, O, T, \Omega, R, \gamma \rangle$, where N, A, S, Z, T, Ω, R and γ are the same as defined in the Dec-POMDP; $O = \otimes O_n$ are sets of joint macro-action observations which are functions of the state; $M \otimes M_n$ are sets of joint macro-actions, with $M_n = \langle I_n^m, \beta_n^m, \pi_n^m \rangle$, where $I_n^m \subset H_n^M$ is the initiation set that depends on macro-action observation histories, defined as $h_{n,t}^M = \{o_n^0, m_n^1, \cdots, o_n^{t-1}, m_n^t\} \in H_n^M$, $\beta_n^m : S \rightarrow [0, 1]$ is a stochastic termination condition that depends on the underlying states, and $\pi_n^m : H_n \times M_n \rightarrow [0, 1]$ is an option policy for macro-action m (H_n is the space of primitive-action and observation). 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).

MacDec-POMDPs can be thought of as decentralized partially observable *semi*-Markov decision processes (Dec-POSMDPs) (Amato et al. 2015a; Omidshafiei et al. 2015), because it is important to consider the amount of time that may pass before a macro-action is completed. The high level policy for each agent Ψ_n, can be defined for choosing macro-actions that depends on macro-action observation histories. Given a joint policy, the primitive actions at each step is determined by the high-level policy to choose the MA, and the MA policy chooses the primitive action. The joint high level policies and macro-action policies can be evaluated as: $V^{\Psi}(s_0) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t)|s_0, \pi, \Psi\big]$. In this paper, our goal is to optimize a high-level policy based solely on macro-actions and macro-action observations (i.e., the underlying Dec-POMDP is unknown).

An **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 $\Theta_n = \langle Q_n, M_n, O_n, \delta_n, \lambda_n, \mu_n \rangle$, where, Q_n is the set of nodes; M_n and O_n are the output and input alphabets (i.e., the macro-action chosen and the observation seen); $\delta_n : Q_n \times O_n \times Q_n \rightarrow [0, 1]$ is the node transition probability, i.e., $\delta_n(q, o, q') = \Pr(q'|q, o)$; $\lambda_n^0 : Q_n \times M_n \rightarrow [0, 1]$ is the output probability for node q_n^0, such that $m_{n,0} \sim \lambda_n^0(q_{n,0}, m_{n,0}) = \Pr(m_{n,0}|q_{n,0})$; $\lambda_n : Q_n \times O_n \times M_n \rightarrow [0, 1]$ is the output probability for nodes $\neq q_n^0$ that associates output symbols with transitions, i.e. $m_{n,\tau} \sim \lambda_n(q_{n,\tau}, o_{n,\tau}, m_{n,\tau}) = \Pr(m_{n,\tau}|q_{n,\tau}, o_{n,\tau})$; $\mu : Q_n \rightarrow [0, 1]$ is the initial node distribution $q_{n,0} \sim \mu_n = \Pr(q_{n,0})$. This type of FSC is called Mealy machine (Amato, Bonet, and Zilberstein 2010), where an agent's local policy for action selection $\lambda_n(q, o, m)$ depends on both current controller node (an abstraction of history) and immediate observation. By conditioning action selections on immediate observations, a Mealy machine can use this observable information to help ensure a valid macro-action controller is constructed. We will discuss the validity issue in the next section.

A Dec-POMDP problem can be transformed into an **inference problem** and then efficiently solved by an EM algorithm. Previous EM methods (Kumar and Zilberstein 2010; Kumar, Zilberstein, and Toussaint 2015) have achieved success in scaling to larger problems, but these methods require a Dec-POMDP model both to construct a Bayes net and to evaluate policies. When the exact model parameters T, Ω and R are unknown, an RL problem must be solved instead. To this end, EM has been adapted to model-free RL settings to optimize FSCs (Wu, Zilberstein, and Jennings 2013; Liu et al. 2015). Here, we adopt similar strategy to learn control polices for MacDec-POMDPs. Towards this end, we define the following global empirical value function.

Definition 1. *(Global empirical value function) Let* $\mathcal{D}^{(K)} = \{(\vec{o}_0^k \vec{m}_0^k r_0^k \cdots \vec{o}_{T_k}^k \vec{m}_{T_k}^k r_{T_k}^k)\}_{k=1}^K$ *be a set of episodes resulting from* $|N|$ *agents who choose macro-actions according to* $\Psi = \otimes_n \Psi_n$*, a set of arbitrary stochastic policies with* $p^{\Psi_n}(m|h) > 0$, $\forall$ *action* m, $\forall$ *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* $h_{n,t}^k = (m_{n,0:t-1}^k, o_{n,1:t}^k)$*,* $0 \le \gamma < 1$ *is the discount and* R_{min} *is the minimum reward* [1]*.*

Essentially, 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 parameters $\Theta = \{\Theta_i\}_{i=1}^{|N|}$ 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.

Previous work (Wu, Zilberstein, and Jennings 2013; Liu et al. 2015) has solved primitive action-based Dec-POMDP problems. However these methods are not directly applicable for macro-action problems; the controllers generated might be invalid, because the macro-action execution depends on certain initial conditions, which might not be available in RL settings. To combat this issue and ensure a learned macro-action controller is valid and optimal, we learn and add constraints to the policy optimization problem. These constraints encode validity information extracted from trajectories that are given.

Learning Valid Controllers

Assuming a valid macro-action level dataset $\mathcal{D}^{(K)} = \{(\vec{o}_0^k \vec{m}_0^k r_0^k \cdots \vec{o}_{T_k}^k \vec{m}_{T_k}^k r_{T_k}^k)\}_{k=1}^{K}$, including macro-actions, observations, and rewards at fixed time intervals (primitive time-steps) is generated by some random behavior policy, Ψ, the question we want to ask is, without the knowledge of underlying MacDec-POMDP model, can we learn a valid controller from these trajectories? To answer this question, we first give the following definitions.

Definition 2. *A macro-action,* m_t*, is said to be* ***valid*** *with respect to a trajectory* $d_{t-1} = (o_0, m_0 \cdots, o_{t-1})$*, if the probability of observing* $h_t = (d_{t-1}, m_t)$ *is positive.*

Definition 3. *Define* $p^{\Psi}(h)$ *to be probability of generating a trajectory* h *under policy* Ψ*. Let* ϵ_p *be the smallest positive probability associated with a trajectory generated according to* $p^{\Psi}(h)$*.*

Since we assume trajectories generated from Ψ are valid, we can count the number of unique trajectories and estimate the value of the behavior policy $\hat{V}^{\Psi}$. We follow a similar analysis to (Kearns, Mansour, and Ng 1999) to obtain a bound for the number of samples that are necessary to give an accurate estimate of policy value.

Theorem 4. *Let* Ψ *be a valid stochastic policies in an arbitrary MacDec-POMDP,* $\mathcal{M}$*. Let* K_V *trajectories be generated by using a generative model for* $\mathcal{M}$ *following policy* Ψ*, and let* $\hat{V}^{\Psi}(s_0)$ *be the corresponding empirical value defined as* $\hat{V}^{\Psi}(s_0) = \frac{1}{K_V}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\gamma^t r_t^k$*. If* $K_V = \mathcal{O}\big(((R_{max} - R_{min})/(1-\gamma)\epsilon)^2 \log(2/\delta)\big)$*, then with probability at least* $1-\delta$*,* $|V^{\Psi}(s_0) - \hat{V}^{\Psi}(s_0)| \le \epsilon$*.*

[1] R_{min} ensures the summands in $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ are positive, hence applying Jensens inequality to $\log \hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is legal, as it will be seen in the section deriving the PoEM algorithm.

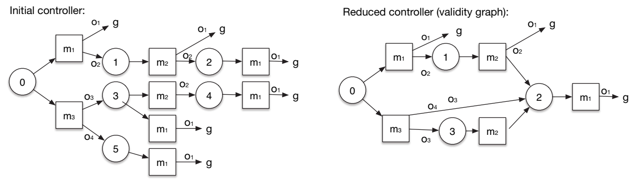

Figure 1: An example of a valid macro-action controller generated from a set of trajectories.

Proof. For any state s_0 and policy Ψ, we have $R_{min}/(1-\gamma) \le V^{\Psi}(s) \le R_{max}/(1-\gamma)$. Given a large enough K, we have $\mathbb{P}(|V^{\Psi}(s_0) - \hat{V}^{\Psi}(s_0)| \le \epsilon) = 1-\delta$. Further, given the independence of trajectories generated by the same policy Ψ, we can apply the Chernoff bound to derive an estimate of $V^{\Psi}(s_0)$ (starting from the same state), $\mathbb{P}(|V^{\psi}(s_0) - \hat{V}^{\Psi}(s_0)| \ge \epsilon) \le 2e^{-\frac{2\epsilon^2 K_v}{(R_{max}-R_{min})^2}}$. Solving this inequality for K gives us the following

$$K_V \le \mathcal{O}\left(\left[\frac{R_{max} - R_{min}}{\epsilon(1-\gamma)}\right]^2 \log(2/\delta)\right) \qquad (1)$$

□

Therefore, we can claim with finite number of samples, the value of a policy Ψ can be accurately estimated.

Moreover, we can estimate the probability of an macro-action m, given a history, h, as $\hat{p}^{\Psi}(m|h)$, which provides validity information of m. Therefore, we can claim the following.

Theorem 5. *Given a history,* h*, which is valid under policy* Ψ *(i.e.,* $p^{\Psi}(h) > 0$*), let* $\hat{p}^{\Psi}(h) = \frac{1}{K_p}\sum_{k=1}^{K_p}\mathbb{I}\big(h_t^k, h\big)$*, where* $\mathbb{I}(\cdot)$ *is the indicator function, when* K_p *is large enough, then with probability at least* $1-\delta$*,* $|p^{\Psi}(h) - \widehat{p}(h)| \le \eta < \epsilon_p$*.*

Proof. Since we assume Ψ is a valid random policy, the trajectories collected by following Ψ must be valid. If $p^{\Psi}(h) = \epsilon_p > 0$, we can have a large enough K_p, such that $\mathbb{P}(|p^{\Psi}(h) - \hat{p}(h)| \le \eta) = 1-\delta$ with $\eta < \epsilon_p$. Given the independence of trajectories generated by the same policy, applying Chernoff bound, we have $\mathbb{P}(|p^{\Psi}(h) - \hat{p}^{\Psi}(h)| \ge \eta) = \delta \le 2e^{-2K_p\eta^2}$. Solving this inequality for K_p gives us the following bound

$$K_p \le \mathcal{O}\big(\eta^{-2}\log(2/\delta)\big). \qquad (2)$$

Theorem 5 shows that for any valid trajectory, we can bound the number of samples to confirm it is valid. □

Remark 6. *Since there are finitely many unique trajectories generated from policy* Ψ*, we can bound the number of samples for learning a full validity map.*

From trajectories to valid controllers

The mapping encoded by $\hat{p}^{\Psi}(h_n)$ provides the validity information of all the trajectories, $h_n \in H_n^M$. To obtain a more concise representation, we convert the validity map into a set of constraints that can be used for controller parameter optimization. To do so, assuming all the trajectories begin at the same initial state and end when the goal condition has been satisfied. Similar to previous methods (Amato and Zilberstein 2009), we first create a controller by converting all distinct trajectories into a tree, with the root node denoting the initial controller nodes, and the leaf nodes corresponding to the goal, hence a branch of the tree corresponds to a unique trajectory seen in the dataset. Then, starting from the leaf nodes, we combine redundant nodes to obtain a reduced controller. Figure. 1 illustrates the procedure for constructing a validity map for controllers. Denote the validity map by $\mathcal{G}^{\Psi} = \langle \rho_n, \omega_n, \nu_n, \rangle$, with $\rho_n(q, o, m) = 1$ if there is a direct link from node q to macro-action m and an observation over this link, otherwise $\rho_n(q, o, m) = 0$. ω_n encoding nodes transitions and ν_n^{Ψ} indicating initial nodes can be constructed in similar ways. To ensure the learned FSC is valid, we give the following definition.

Definition 7. *Let $\mathcal{G}_{\Psi}$ be the validity map for FSC constructed according to the procedure described in the previous section (from the trajectories generated by policy Ψ), a controller Θ is said to be admissible with respect to $\mathcal{G}_{\Psi}$, if the following conditions are satisfied:*

1. *If $p(m|h, \mathcal{G}_{\Psi}) = 0$, then $(m|h, \Theta) = 0$,*
2. *If $p(m|h, \mathcal{G}_{\Psi}) > 0$, then $(m|h, \Theta) \geq 0$.*

This admissibility condition forces the policy parameterized by Θ to only take actions that belongs to trajectories seen before. By leveraging this definition, we can use any optimization algorithms to generate valid macro-action controllers. Here we developed an EM algorithm to achieve the goal of controller optimization.

MacDec-POMDP Policy Learning by Expectation Maximization

After building the validity graph $\mathcal{G}_{\Psi}$, we aim to improve the policy using the available dataset. Again, the assumption is that neither the model nor simulator are available to provide additional data for evaluation. Combining the validity constraints with the empirical value function given in Definition 1 results a constrained off-policy optimization problem. However, direct maximization of $\hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$ is difficult; instead, we augment $\hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$ with controller node sequences $\{\vec{q}_{0:t}^{\,k} : k = 1 \ldots, K, t = 1 : T_k\}$ and maximize the lower bound of the logarithm of $\hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$ (obtained by Jensen's inequality):

$$\ln \hat{V}\big(\mathcal{D}^{(K)}; \Theta\big) = \ln \sum_{k,t,\vec{q}_{0:t}^{\,k}} \frac{f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta})\tilde{r}_t^k p(\vec{m}_{0:t}^{\,k}, \vec{q}_{0:t}^{\,k}|\vec{o}_{1:t}^{\,k}, \Theta)}{f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta})}$$
$$\geq \sum_{k,t,\vec{q}_{0:t}^{\,k}} f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta}) \ln \frac{\tilde{r}_t{}^k p(\vec{m}_{0:t}^{\,k}, \vec{q}_{0:t}^{\,k}|\vec{o}_{1:t}^{\,k}, \Theta)}{f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta})} \overset{def.}{=} \mathrm{lb}(\Theta|\widetilde{\Theta}), \quad (3)$$

where $f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta}) = \tilde{r}_t^k p(\vec{m}_{0:t}^{\,k}, \vec{q}_{0:t}^{\,k}|\vec{o}_{1:t}^{\,k}, \widetilde{\Theta})/\hat{V}(\mathcal{D}^{(K)}; \widetilde{\Theta})$, and $\{f(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta}) \geq 0\}$ satisfy the normalization constraint $\sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{\vec{q}_{0:t}^{\,k}} f_t^k(\vec{q}_{0:t}^{\,k}|\widetilde{\Theta}) = K$ with $\widetilde{\Theta}$ the most recent estimate of Θ, and $\tilde{r}_t^k = \gamma^t r_t^k / \prod_{\tau=0}^{t} p^{\Psi}(\vec{m}_{\tau}^{\,k}|h_{\tau}^k), \forall t, k$ are reweighted rewards, leading to the following constrained optimization problem

$$\begin{aligned}
&\max_{\left\{f_t^k\left(\vec{q}_{0:t}^{\,k}; \widetilde{\Theta}\right)\right\}, \Theta} \mathrm{lb}(\Theta|\widetilde{\Theta}) \\
&\text{subject to: } \Theta = \widetilde{\Theta} \odot \mathcal{G}_{\Psi} \\
&p(\vec{m}_{0:t}^{\,k} \vec{q}_{0:t}^{\,k}; \widetilde{\Theta}) = \prod_{n=1}^{|N|} p(m_{n,0:t}^k, q_{n,0:t}^k | o_{n,0:t}^k, \widetilde{\Theta}_n), \\
&\sum_{k=1}^{K} \sum_{t=0}^{T_k} \sum_{q_{n,0:t}^k=1}^{|Q_{1:|N|}|} f_t^k(\vec{q}_{0:t}^{\,k}; \widetilde{\Theta}) = K
\end{aligned} \quad (4)$$

where $\odot$ is the Hadamard product, which encodes the validity constraints from $\mathcal{G}_{\Psi}$ (learned from the previous section).

Based on the previous problem formulation (4), an EM algorithm is derived to learn the macro-action FSCs. Algorithmically, the main steps 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 in the section below.

Computation of Value Function (E-step) In E-step, the controller nodes distribution $p(\vec{q}_{0:t}^{\,k}|\vec{m}_{0:t}^{\,k}, \vec{o}_{1:t}^{\,k}, \widetilde{\Theta})$ and the action selection probability $p(\vec{m}_{0:t}^{\,k}|\vec{o}_{1:t}^{\,k}, \widetilde{\Theta}) \forall t, k$ are updated based on $\widetilde{\Theta}$ obtained in the previous M-step. Hence, we can obtain an updated value function

$$\begin{aligned}
\hat{V}\big(\mathcal{D}^{(K)}; \widetilde{\Theta}\big) &= \tfrac{1}{K} \sum_{k,t} \tilde{r}_t^k \prod_{n=1}^{|N|} p(m_{n,0:t}^k | o_{n,1:t}^k, \widetilde{\Theta}_n) \\
&\overset{def.}{=} \tfrac{1}{K} \sum_{k,t} \sigma_t^k(\widetilde{\Theta}_n)
\end{aligned} \quad (5)$$

which is equivalent to policy-evaluation using all available episodes with the rewards are reweighted by the action selection probability $p(\vec{m}_{0:t}^{\,k}|\vec{o}_{1:t}^{\,k}, \widetilde{\Theta})$ to reflect the improved value of a new policy.

Update of Policy Parameters (M-step) In the M-step, the policy parameters is updated by $\Theta = \arg\max_{\Theta \in \mathcal{F}} \mathrm{lb}(\Theta|\widetilde{\Theta})$, subject to normalization and validity constraints. Specifically, define

$$\begin{aligned}
\xi_{n,k}^{t,\tau}(u_n, v_n) &\overset{def.}{=} p(q_{n,\tau}^k = u_n, q_{n,\tau+1}^k = v_n | m_{n,0:t}^k, o_{n,1:t}^k, \widetilde{\Theta}_n) \\
\phi_{n,k}^{t,\tau}(u_n) &\overset{def.}{=} p(q_{n,\tau}^k = u_n | m_{n,0:t}^k, o_{n,1:t}^k, \widetilde{\Theta}_n).
\end{aligned} \quad (6)$$

Expanding the lower bound of $\ln \hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$ and keeping the terms related to Θ, we have

$$\begin{aligned}
\mathrm{lb}\big(\Theta|\widetilde{\Theta}\big) \propto \sum_{k,t} \sigma_t^k(\widetilde{\Theta}) \sum_{n=1}^{|N|} \Bigg\{ &\sum_{u_n=1}^{|Q_n|} \phi_{n,k}^{t,0}(u_n) \ln \mu_n^{u_n} \\
&+ \sum_{\tau=0}^{t} \Big[\sum_{v_n=1}^{|Q_n|} \phi_{t,\tau}^{n,k}(u_n) \ln \lambda_n(m_{n,\tau}^k, o_{n,\tau}^k, u_n) \\
&+ \sum_{u_n,v_n=1}^{|Q_n|} \xi_{n,k}^{t,\tau}(u_n, v_n) \ln \delta_n(u_n, o_{n,\tau+1}^k, v_n) \Big] \Bigg\}.
\end{aligned}$$

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

$$\begin{aligned}
\lambda_n(u_n, o_n, m_n) \propto &\sum_{k,t,\tau} \sigma_t^k(\widetilde{\Theta}) \phi_{n,k}^{t,\tau}(u_n) \mathbb{I}_{(m_n,o_n)}(m_{n,\tau}^k, o_{n,\tau}^k) \\
&\times \rho_n(u_n, o_n, m_n)), \forall n \in N, u_n, \in Q_n, m_n \in M_n, o_n \in O_n. (7)
\end{aligned}$$

Algorithm 1 PoEM

```
Require: Episodes $\mathcal{D}^{(K)}$ and the number of agents $|N|$,
    lower bound $\text{LB}_0 = -\text{Inf}$, $\Delta\text{LB} = 1$, Iter $= 0$;
 1: Construct a validity map $\mathcal{G}_\Psi$
 2: while $\Delta\text{lb} < 10^{-3}$ do
 3:     for $k = 1$ to $K$, $n = 1$ to $|N|$ do
 4:         Compute $p(m^k_{n,0:t}|o^k_{n,1:t}, \widetilde{\Theta})$ and forward-backward variables $\{\alpha_{n,k}\}$ and $\{\beta_{n,k}\}, \forall t$
 5:     end for
 6:     Iter = Iter + 1
 7:     Compute $\text{lb}_{\text{Iter}}$ using (5)
 8:     $\Delta\text{lb} = (\text{lb}_{\text{Iter}} - \text{lb}_{\text{Iter}-1})/|\text{LB}_{\text{Iter}-1}|$
 9:     for $n = 1$ to $|N|$ do
10:         Compute the $\xi$ and $\phi$ variables,
11:         Update $\Theta_n$ using (7)
12:     end for
13: end while
14: return Controller parameters, $\{\Theta_n\}_{n=1}^{|N|}$.
```

δ, μ are updated in similar ways and their update equations are included in the supplemetary materials (URL) . These updates constitute a policy-improvement step where the reweighted rewards are used to further improve policies.

Both the above steps require $\xi^{t,\tau}_{n,k}(u_n, v_n)$, which are computed based on $\alpha^{\tau}_{n,k} = p\big(q^k_{n,\tau}|m^k_{n,0:\tau}, o^k_{n,1:\tau}, \widetilde{\Theta}_n\big)$ and $\beta^{t,\tau}_{n,k} = \frac{p(m^k_{n,\tau+1:t}|q^k_{n,\tau}, o^k_{n,\tau+1:t}, \widetilde{\Theta}_n)}{\prod^t_{\tau'=\tau} p(m^k_\tau|h^k_{n,\tau'}, \widetilde{\Theta}_n)}, \forall n, k, t, \tau$. The (α, β) are forward-backward messages. The computation details are provided in the supplementary materials (URL).

Convergence Analysis

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 8.

Theorem 8. *Define* $\mathcal{F} = \Big\{\Theta = \{\Theta_n\}_{n=1}^{|N|}$ *with* $\Theta_n = \langle Q_n, M_n, O_n, \delta_n, \lambda_n, \mu_n\rangle : \sum_{u_n=1}^{|Q_n|} \mu_n(u_n) = 1, \sum_{m_n=1}^{|M_n|} \lambda_n(m_n, o_n, , u_n) = 1, \sum_{u_n=1}^{|Q_n|} \delta_n(v_n, o_n, u_n) = 1, v_n = 1 \cdots |Q_n|, m_n = 1 \cdots |M_n|, o_n = 1 \cdots |O_n|\Big\}$, *and* $\Theta^{(n)}$ *be a sequence produced by the iterative update* $\Theta^{(i+1)} = \arg\max_{\Theta\in\mathcal{F}} \text{lb}(\Theta|\Theta^{(i)})$, *where* $\Theta^{(0)}$ *is an arbitrary initialization, then* $\{\Theta^{(i)}\}_{i\geq 0}$ *monotonically increases* $\hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$, *until convergence to a maximum.*

Proof. Note that $\text{lb}(\Theta|\widetilde{\Theta})$ in (3) is maximized when $f^k_t(\vec{q}^k_{0:t}|\widetilde{\Theta}) = \tilde{r}^k_t p(\vec{m}^k_{0:t}, \vec{q}^k_{0:t}|\vec{o}^k_{1:t}, \Theta)/\hat{V}(\mathcal{D}^{(K)}; \Theta)$, which turns the inequality into an equality, that is $\text{lb}(\Theta|\Theta) = \ln \hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$. Hence, according to (3),

$$\text{lb}(\Theta|\widetilde{\Theta}) \leq \text{lb}(\Theta|\Theta) = \ln \hat{V}\big(\mathcal{D}^{(K)}; \Theta\big) \qquad (8)$$

holds for any Θ and $\widetilde{\Theta}$.

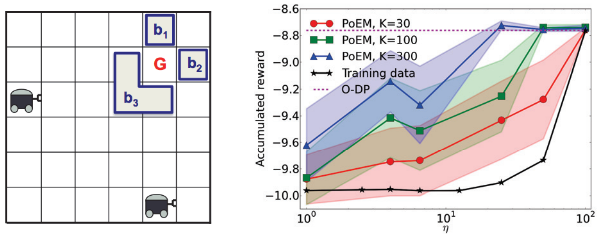

Figure 2: A 6×6 NAMO problem (left); The performance of Algorithm 1 as a function of η% of data generated from an optimal policy using different sizes of training samples K (right).

Then we solve for $\widetilde{\Theta} = \arg\max_{\Theta\in\mathcal{F}} \text{lb}(\Theta|\widetilde{\Theta})$. Since $\widetilde{\Theta}$ increases $\text{lb}(\Theta|\widetilde{\Theta})$, we have

$$\ln \hat{V}\big(\mathcal{D}^{(K)}; \widetilde{\Theta}\big) = \text{lb}(\widetilde{\Theta}|\widetilde{\Theta}) \geq \text{lb}(\Theta|\widetilde{\Theta}) \qquad (9)$$

Combining (8) and (9), we have $\ln \hat{V}\big(\mathcal{D}^{(K)}; \widetilde{\Theta}\big) = \text{lb}(\widetilde{\Theta}|\widetilde{\Theta}) \geq \text{lb}(\Theta|\widetilde{\Theta}) \geq \text{lb}(\Theta|\Theta) = \ln \hat{V}\big(\mathcal{D}^{(K)}; \Theta\big)$. Starting from $\Theta^{(0)}$, we can iterate between these two steps to solve for $\Theta^{(i+1)} = \arg\max_{\Theta\in\mathcal{F}} \text{lb}(\widetilde{\Theta}|\Theta^{(i)})$, which satisfy $\hat{V}\big(\mathcal{D}^{(K)}; \Theta^{(i)}\big) \leq \hat{V}\big(\mathcal{D}^{(K)}; \Theta^{(i+1)}\big) \leq \cdots$. Since $\hat{V}\big(\mathcal{D}^{(K)}; \widetilde{\Theta}^{(i)}\big)$ is upper bounded, by monotone convergence theorem, $\{\Theta^{(i)}\}$ converges to a maximum of $\hat{V}\big(\mathcal{D}^{(K)}, \Theta\big)$. $\square$

Computational complexity

The time complexity of Algorithm 1 in each iteration is summarized in Table 1, assuming the averaged episode length is T and averaged nodes number is $|Q|$, which are the same for all agents. In Table 1, the worst case refers to the case when there is a nonzero reward at every step in an episode (dense reward), while the best case is that nonzero reward is received only at the terminal step in each episode. Hence in general the algorithm scales linearly with the number of episodes and the number of agents. The time dependency on T is between linear and quadratic.

Table 1: Computational Complexity of Algorithm 1.

VARIABLES	BEST CASE	WORST CASE
α/β	$\Omega(\|N\|\|Q\|^2KT)$	$\mathcal{O}(\|N\|\|Q\|^2KT)$
ξ/ϕ	$\Omega(\|N\|\|Q\|^2KT)$	$\mathcal{O}(\|N\|\|Q\|^2KT^2)$
σ^k_t	$\Omega(K)$	$\mathcal{O}(KT)$
Θ	$\Omega(\|N\|\|Q\|^2KT)$	$\mathcal{O}(\|N\|\|Q\|^2KT^2)$

Experiments

We evaluate Algorithm 1 on a multi-robot navigation benchmark and a large domain motivated by SAR problems.

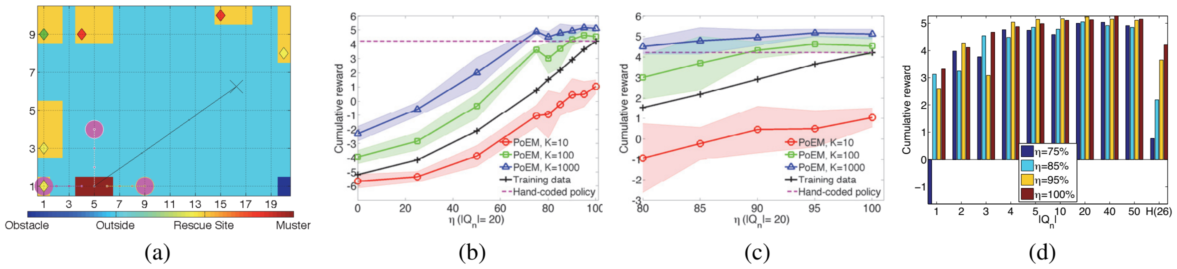

Figure 3: (a) A SAR domain (diamonds represent victims with colors indicate health, circles and cross represent UGV and UAV respectively); (b) Testing performance using different number of training sample K and percentage of hand-coded policies η; (c) Testing performance of policies with different FSC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

A multi-robot benchmark

We first consider a multi-robot navigation among movable obstacles (NAMO) problem, introduced in (Amato, Konidaris, and Kaelbling 2014). Here as shown in Figure 2, both agents are trying to reach a goal location "G", but have to move obstacles in order to reach it.

For this problem, a batch of episodes is collected by a combination of random action selection and actions provided by a MaDec-POMDP algorithm (Option-based dynamic programming (O-DP)) (Amato, Konidaris, and Kaelbling 2014). 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 access to a domain expert, but by adding random actions, we can observe performance as our expert data becomes more noisy. For each run of algorithm 1, $K = 30, 100, 300$ episodes of 10 steps are used to learn the FSCs with $|Q_n| = 20, \forall n \in N$, and the learned policies are evaluated by the accumulated reward averaged over 100 test episodes of 10 steps. For $\eta = [1, 4, 12.5, 25, 50, 100]$, the corresponding testing results are plotted in Figure 2. It can be seen that our method can learn high-quality policies without needing much domain experts information. In fact, our experiments already show higher-valued policies can be learned with a large amount of noisy trajectories (as K increases). Figure 2 also shows the actual smallest setting for η when PoEM learns the optimal policy depends on the amount of episodes. With a small percentage of expert knowledge ($\eta = 25$) and large enough amount of data ($K = 300$), PoEM is able to recover the optimal policy.

A Search and Rescue (SAR) Problem

To further demonstrate the scalability and learning efficiency of the proposed algorithm, we designed a SAR problem involving a team of heterogeneous agents: one unmanned aerial vehicle (UAV) and three unmanned ground vehicles (UGVs). The scenario begins after a natural disaster strikes the simulated world. These agents operate in a 20×10 gridworld (shown in figure 3(a)), where there are 6 rescue sites with different distance to a muster. There are six victims, with varying initial levels of criticality. Each victim's health degrades linearly with time. The speed of UAV is three times faster than UGV, but only the UGVs can pick-up victims. 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 current location; OV is the state of the victim at OL, the other location (from communication), and there are 8 macro-actions, including go to one of the six sites, retrieve a victim at one site (retrieve) and go to muster (and drop off victim). For the UAV, there are 560 observations (assuming the UAV cannot hold victims) and 7 macro-actions, including going to the muster and going to one of the six sites. All vehicles begin in the muster and communication can happen bidirectionally when two vehicles are within a range (which is larger for UAV-UGV communication than for UGV-UGV communication). Noise exists in all observations, communications and actions (all of which are assumed to succeed with probability 0.95). The reward is $+1$ for each victim brought back to muster alive and -1 for each victim who dies.

This SAR domain significantly extends previous benchmarks (mostly have two agents) not only in terms of agent numbers, but also state/action/observation numbers. Given its large problem size, unknown model and the stochasticity in observations and communication, it is difficult to generate an optimal policy with existing solvers. Instead, a domain expert (from a large government laboratory) created a heuristic controller, which is complex and works well (a visualization of the graph describing the heuristic policy finite state machine is attached in the supplementary materials (URL). 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(b)).

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 $|Q_n| = 20$ and $K = 1000$, the training time is less than 15min on average. The corresponding testing results are plotted in figure 3(b-c), from which we can see, a) with a sufficient amount of samples ($K > 100$), PoEM is able to achieve policy improvement over policies generating training data; 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 hand-coded policy. Hence the experiments demonstrate PoEM can achieve policy improvement and is scalable to large problems. In addition, nine settings of $|Q_n|$ are used to examine 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 $|Q_n|$ (from 4 to 50), indicating robustness of PoEM to the size of controller nodes. However, when $|Q_n|$ 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 (Liu, Liao, and Carin 2011) and stick-breaking processes (Liu et al. 2015), which is left for future work.

Conclusions

This paper presents a reinforcement learning method for coordinating multiple agents in macro-action level Dec-POMDPs, an important problem that has not been adequately addressed before. Our method solely uses data from demonstrations in the form of previously executed macro-action and observation histories as well as rewards to improve future decision making without explicitly requiring mission models. We showed that our approach is able to generate valid macro-action controllers and developed an algorithm called PoEM for batch learning with guaranteed convergence. Our experimental results show that PoEM can scale to large problems and outperform hand coded methods even with limited and noisy data. Theoretical analysis and empirical results show the proposed method is a promising tool for solving RL in large MacDec-POMDPs domains.

Acknowledgments

This work is supported in part by the ONR under MURI program award #N000141110688 and NSF award #1463945. The Lincoln Laboratory portion of this work is 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.

References

Amato, C., and Zilberstein, S. 2009. Achieving goals in decentralized POMDPs. In *Proc. of the 8th Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS-09)*.

Amato, C.; Chowdhary, G.; Geramifard, A.; Ure, N. K.; and Kochenderfer, M. J. 2013. Decentralized control of partially observable Markov decision processes. In *Proc. of the Conf. on Decision and Control (CDC-13)*.

Amato, C.; Konidaris, G.; Anders, A.; Cruz, G.; How, J. P.; and Kaelbling, L. P. 2015a. Policy Search for Multi-Robot Coordination under Uncertainty. In *Proc. of the 2015 Robotics: Science and Systems Conference (RSS-15)*.

Amato, C.; Konidaris, G. D.; Cruz, G.; Maynor, C. A.; How, J. P.; and Kaelbling, L. P. 2015b. Planning for decentralized control of multiple robots under uncertainty. In *Int'l Conf. on Robotics and Automation (ICRA-15)*. IEEE.

Amato, C.; Bonet, B.; and Zilberstein, S. 2010. Finite-state controllers based on mealy machines for centralized and decentralized POMDPs.

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized POMDPs. In *Proc. of the int'l Conf. on Autonomous agents and multi-agent systems (AAMAS-14)*.

Bernstein, D.; Zilberstein, S.; Washington, R.; and Bresina, J. 2001. Planetary rover control as a Markov decision process. In *Int'l Symp. on AI, Robot. & Automation in Space*.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society B* 39:1–38.

Grayson, S. 2014. Search & Rescue using Multi-Robot Systems. http://www.maths.tcd.ie/~graysons/documents/COMP47130_SurveyPaper.pdf.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 1999. Approximate planning in large POMDPs via reusable trajectories. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS-99)*.

Kumar, A., and Zilberstein, S. 2010. Anytime planning for decentralized POMDPs using expectation maximization. In *Proc. of the 26th Conf. on Uncertainty in Artificial Intelligence (UAI-10)*.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2015. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research* 53(1):223–270.

Liu, M.; Amato, C.; Liao, X.; How, J. P.; and Carin, L. 2015. Stick-Breaking Policy Learning in DEC-POMDPs. In *Proc. of the 24th Intĺ Joint Conf. on Artificial Intelligence (IJCAI-15)*.

Liu, M.; Liao, X.; and Carin, L. 2011. Infinite regionalized policy representation. In *Proc. of the 28th Int'l Conf. on Machine Learning (ICML-11)*, 769–776.

Martins, M. F., and Demiris, Y. 2010. Learning multirobot joint action plans from simultaneous task execution demonstrations. In *Proceedings of the 9th Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS-10)*.

Oliehoek, F. A. 2012. Decentralized POMDPs. In *Reinforcement Learning*. Springer. 471–503.

Omidshafiei, S.; akbar Agha-mohammadi, A.; Amato, C.; and How, J. P. 2015. Decentralized control of partially observable markov decision processes using belief space macro-actions. In *Int'l Conf. on Robotics and Automation (ICRA-15)*. IEEE.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Supplementary. http://www.mit.edu/~miaoliu/publications/AAAI2016_supplementary.pdf.

Wu, F.; Zilberstein, S.; and Jennings, N. R. 2013. Monte-carlo expectation maximization for decentralized POMDPs. In *Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI-13)*.