2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Learning for Multi-robot Cooperation in Partially Observable Stochastic Environments with Macro-actions

Miao Liu[1], Kavinayan Sivakumar[2], Shayegan Omidshafiei[3], Christopher Amato[4] and Jonathan P. How[3]

Abstract— **This paper presents a data-driven approach for multi-robot coordination in partially-observable domains based on Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) and macro-actions (MAs). Dec-POMDPs provide a general framework for cooperative sequential decision making under uncertainty and MAs allow temporally extended and asynchronous action execution. To date, most methods assume the underlying Dec-POMDP model is known a priori or a full simulator is available during planning time. Previous methods which aim to address these issues suffer from local optimality and sensitivity to initial conditions. Additionally, few hardware demonstrations involving a large team of heterogeneous robots and with long planning horizons exist. This work addresses these gaps by proposing an iterative sampling based Expectation-Maximization algorithm (iSEM) to learn polices using only trajectory data containing observations, MAs, and rewards. Our experiments show the algorithm is able to achieve better solution quality than the state-of-the-art learning-based methods. We implement two variants of multi-robot Search and Rescue (SAR) domains (with and without obstacles) on hardware to demonstrate the learned policies can effectively control a team of distributed robots to cooperate in a partially observable stochastic environment.**

I. Introduction

There has been significant progress in recent years on developing cooperative multi-robot systems that can operate in real-world environments with uncertainty. Example applications of social and economical interest include search and rescue (SAR) [1], traffic management for smart cities [2], planetary navigation [3], robot soccer [4], e-commerce and transport logistic processes [5]. Planning in such environments must address numerous challenges, including imperfect models and knowledge of the environment, restricted communications between robots, noisy and limited sensors, different viewpoints by each robot, asynchronous calculations, and computational limitations.

These planning problems, in the most general form, can be formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [6], a general framework for cooperative sequential decision making under uncertainty. In Dec-POMDPs, robots make decisions based on local streams of information (i.e., observations), such that the expected value of the team (e.g., number of victims rescued, average customer satisfaction) is maximized. However, representing and solving Dec-POMDPs is often intractable for large domains, because finding the optimal (even $epsilon$-approximate) solution of a Dec-POMDP (even for finite horizon) is NEXP-complete [6]. To combat this issue, recent research has addressed the more scalable macro-action based Dec-POMDP (MacDec-POMDP), where each agent has temporally-extended actions, which may require different amounts of time to complete [7]. Moreover, significant progress has been made on demonstrating the usefulness of MacDec-POMDPs via a range of challenging robotics problems, such as a warehouse domain [8], bartending and beverage service [9], and package delivery [10], [11]. 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 completely available. Recently, researchers started to address this issue via reinforcement learning and proposed a policy-based EM algorithm (PoEM) [12], which can learn valid controllers via only trajectory data containing observations, macro-actions (MAs), and rewards.

Although PoEM has convergence guarantees for the batch learning setting and can recover optimal policies for benchmark problems with sufficient data, it suffers from local optimality and sensitivity to initial conditions for complicated real-word problems. Inevitably, as an EM type algorithm, the results of PoEM can be arbitrarily poor given bad initialization. Additionally, few hardware demonstrations based on challenging tasks such as SAR, which involves a large team of heterogeneous robots (both ground vehicles and aerial vehicles) and with MacDec-POMDP formulation exists. This paper addresses these gaps by proposing an iterative sampling-based Expectation-Maximization algorithm (iSEM) to learn polices. Specifically, this paper extends previous approaches by using concurrent (multi-threaded) EM iterations providing feedback to one another to enable re-sampling of parameters and reallocation of computational resources for threads which are clearly converging to poor values.

The algorithm is tested in batch learning settings, which is commonly used in learning from demonstration. Through theoretical analysis and numerical comparisons on a large multi-robot SAR domain, we demonstrate the new algorithm can better explore the policy space. As a result, iSEM is able to achieve better expected values compared to the state-of-the-art learning-based method, PoEM. Finally, we present an implementation of two variants of multi-robot SAR domains (with and without obstacles) on hardware to demonstrate the learned policies can effectively control

[1]Miao Liu is with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA miao.liu1@ibm.com

[2]Kavinayan Sivakumar is with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA ks16@princeton.edu

[3] Shayegan Omidshafiei and Jonathan P. How are with Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA {shayegan, jhow}@mit.edu

[4]Christopher Amato is with the College of Computer and Information Science, Northeastern University, Boston, MA, USA camato@ccs.neu.edu

a team of distributed robots to cooperate in a partially observable stochastic environment.

II. Background

We first discuss the background on Dec-POMDPs and MacDec-POMDPs and then describe the PoEM algorithm.

A. Dec-POMDPs and MacDec-POMDPs

Decentralized POMDPs (Dec-POMDPs) generalize POMDPs to the multiagent, decentralized setting [6], [13]. 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 a new observation. The agents share a joint reward based on their joint concurrent actions, making the problem cooperative. However, agents' local views mean that execution is decentralized.

Formally, a **Dec-POMDP** is represented as an octuple $\langle N, A, S, Z, T, \Omega, R, \gamma \rangle$, 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; $T : S \times A \times S \to [0, 1]$ 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; $\Omega : S \times A \times Z \to [0, 1]$ 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'; $R : S \times A \to \mathbb{R}$ 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 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: $H_n \times A_n \to [0, 1]$.

A MacDec-POMDP with (local) macro-actions extends the MDP-based options [14] 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 $h_{n,t}^M = \{o_n^0, m_n^1, \cdots, o_n^{t-1}, m_n^t\} \in H_n^M$, $\beta_n^m : S \to [0, 1]$ is a stochastic termination condition that depends on the underlying states, and $\pi_n^m : H_n \times M_n \to [0, 1]$ is an option policy for macro-action m (H_n is the space of history 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) [9], [10], 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 action at each step is determined by the high-level policy that chooses the MA, and the MA policy that chooses the primitive action. The joint high level policies and macro-action policies can be evaluated as: $V^{\Psi}(s_0) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t)|s_0, \pi, \Psi\big]$ [1].

B. Solution Representation

A Finite State Controller (**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[2]; M_n and O_n are the output and input alphabets (i.e., the macro-action chosen and the observation seen); $\delta_n : Q_n \times O_n \times Q_n \to [0, 1]$ is the node transition probability, i.e., $\delta_n(q, o, q') = \Pr(q'|q, o)$; $\lambda_n^0 : Q_n \times M_n \to [0, 1]$ is the output probability for node $q_{n,0}$, such that $m_{n,0} \sim \lambda_n^0(q_{n,0}, m_{n,0}) = \Pr(m_{n,0}|q_{n,0})$; $\lambda_n : Q_n \times O_n \times M_n \to [0, 1]$ is the output probability for nodes $\neq q_{n,0}$ that associates output symbols with transitions, i.e. $m_{n,\tau} \sim \lambda_n(q_{n,\tau}, o_{n,\tau}, m_{n,\tau}) = \Pr(m_{n,\tau}|q_{n,\tau}, o_{n,\tau})$; $\mu : Q_n \to [0, 1]$ is the initial node distribution $q_{n,0} \sim \mu_n = \Pr(q_{n,0})$. This type of FSC is called a Mealy machine [16], 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 [12].

C. Policy Learning Through EM

A Dec-POMDP problem can be transformed into an **inference problem** and then efficiently solved by an EM algorithm. Previous EM methods [17], [18] 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, a Reinforcement Learning (RL) problem must be solved instead. To this end, EM has been adapted to model-free RL settings to optimize FSCs for Dec-POMDPs [19], [20] and MacDec-POMDPs [12].

For both purposes of self-containment and ease of analyzing new algorithm, we first review the policy based EM algorithm (PoEM) developed for the MacDec-POMDP case [12].

[1]Note that MacDec-POMDPs allows asynchronous decision making, so synchronization issues must be dealt with by the solver as part of the optimization. Some temporal constraints (e.g., timeouts) can be encoded into the termination condition of a macro-action.

[2]A controller node can be understood as a decision state (summary of history). They are commonly used for policy representation when solving infinite horizon POMDPs [15] and Dec-POMDPs [6].

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

$$\hat{V}\big(\mathcal{D}^{(K)};\Theta\big) \stackrel{def.}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\gamma^t r_t^k \prod_{n=1}^{N}\frac{p(m_{n,0:t}^k|h_{n,t}^k,\Theta_n)}{p^{\Psi_n}(m_{n,0:t}^k|h_{n,t}^k)} \tag{1}$$

where $h_{n,t}^k = (m_{n,0:t-1}^k, o_{n,1:t}^k)$, $0 \leq \gamma < 1$ is the discount.

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 a 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.

D. PoEM

Direct maximization of $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ is difficult; instead, $\hat{V}\big(\mathcal{D}^{(K)};\Theta\big)$ can be augmented 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):

$$\begin{aligned}\ln \hat{V}\big(\mathcal{D}^{(K)};\Theta\big) &= \ln \sum_{k,t,\vec{q}_{0:t}^k} \frac{f_t^k(\vec{q}_{0:t}^k|\widetilde{\Theta})\tilde{r}_t^k p(\vec{m}_{0:t}^k,\vec{q}_{0:t}^k|\vec{o}_{1:t}^k,\Theta)}{f_t^k(\vec{q}_{0:t}^k|\widetilde{\Theta})} \\ &\geq \sum_{k,t,\vec{q}_{0:t}^k} f_t^k(\vec{q}_{0:t}^k|\widetilde{\Theta}) \ln \frac{\tilde{r}_t^k p(\vec{m}_{0:t}^k,\vec{q}_{0:t}^k|\vec{o}_{1:t}^k,\Theta)}{f_t^k(\vec{q}_{0:t}^k|\widetilde{\Theta})} \stackrel{def.}{=} \mathrm{lb}(\Theta|\widetilde{\Theta}),\end{aligned} \tag{2}$$

where $f_t^k(\vec{q}_{0:t}^k|\widetilde{\Theta}) \stackrel{def.}{=} \tilde{r}_t^k p(\vec{m}_{0:t}^k,\vec{q}_{0:t}^k|\vec{o}_{1:t}^k,\widetilde{\Theta})/\hat{V}(\mathcal{D}^{(K)};\widetilde{\Theta})$, and $\{f(\vec{q}_{0:t}^k|\Theta) \geq 0\}$ 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 $\tilde{r}_t^k \stackrel{def.}{=} \gamma^t(r_t^k - r_{min})/\prod_{\tau=0}^t p^{\Psi}(\vec{m}_\tau^k|h_\tau^k), \forall t,k$ are reweighted rewards with r_{min} denoting the minimum reward, leading to the following constrained optimization problem

$$\begin{aligned}&\max_{\left\{f_t^k\left(\vec{q}_{0:t}^k;\widetilde{\Theta}\right)\right\},\Theta} \mathrm{lb}(\Theta|\widetilde{\Theta}) \\ &\text{subject to: } \textstyle\sum_{k=1}^K\sum_{t=0}^{T_k}\sum_{q_{n,0:t}^k=1}^{|Q_{1:|N|}|} f_t^k(\vec{q}_{0:t}^k;\widetilde{\Theta}) = K, \\ &p(\vec{m}_{0:t}^k\vec{q}_{0:t}^k;\widetilde{\Theta}) = \textstyle\prod_{n=1}^{|N|} p(m_{n,0:t}^k, q_{n,0:t}^k|o_{n,0:t}^k,\widetilde{\Theta}_n).\end{aligned} \tag{3}$$

Based on the problem formulation (3), an EM algorithm can be derived to learn the macro-action FSCs. Algorithmically, the main steps involve alternating between computing the lower bound of the log empirical value function (2) (E-step) and parameter estimation (M-step). This optimization algorithm is called policy based expectation maximization (PoEM), the details of which is referred to [12].

III. Related Work

The use of multi-robot teams has recently become viable for large-scale operations due to ever-decreasing cost and increasing accessibility of robotics platforms, allowing robots to replace humans in team-based decision-making settings including, but not limited to, search and rescue [1]. Use of multiple robots allows dissemination of heterogeneous capabilities across the team, increasing fault-tolerance and decreasing risk associated with losing or damaging a single all-encompassing vehicle [21].

The large body of work on multi-robot task allocation (MRTA) comes in decentralized, centralized, and distributed/hybrid flavors. Centralized architectures [22], [23] rely on full information sharing between all robots. However, in settings such as SAR, communication infrastructure may be unavailable, requiring the use of alternative frameworks. Distributed frameworks, such as those used in auction-based algorithms [24], use local communication for consensus on robot policies. This enables robustness against communication failures in hazardous, real-world settings. However, in settings such as SAR, it can be unreasonable or impossible for robots to communicate with one another during task execution. Decentralized frameworks, such as Dec-POMDPs [13] and the approach proposed in this paper, target this setting, allowing a spectrum of policies ranging from communication-free to explicitly communication-enabled. The flexibility offered by decentralized planners makes them suitable candidates for multi-robot operation in hazardous or uncertain domains, such as SAR.

Finally, note that unlike the majority of the existing MRTA literature, the work presented here exploits the strengths of the MacDec-POMDP framework [8] to develop a unifying framework which considers sources of uncertainty, task-level learning and planning, temporal constraints, and non-deterministic action durations.

IV. Iterative Sampling Based Expectation Maximization Algorithm

The PoEM algorithm [12] is the first attempt to address policy learning for MacDec-POMDPs with batch data. However, one of the biggest challenge for PoEM is that it only grantees convergence to a local solution, a problem often encountered when optimizing mixture models, such as the empirical value function (1) [3]. Moreover, PoEM is a deterministic algorithm for approximate optimization, meaning that it converges to the same stationary point if initialized repeatedly from the same starting value. Hence, PoEM can be prone to poor local solution for more complicated real-world problems (as it will be shown in a later numerical experiment). To address these issues, we propose a concurrently (multi-threaded) randomized method called iterative sampling based Expectation Maximization (iSEM). The iSEM algorithm is designed to run multiple instances of PoEM with randomly initialized FSC parameters in parallel to minimize the probability of converging to a sub-optimal solution due to poor initialization. Furthermore, to exploit information and computational efforts on runs of PoEM which are clearly converging to poor values, iSEM allows re-sampling of parameters once convergence of $V(D_{test})$ is detected, increasing the chance of overcoming poor local optima. Because of the re-sampling step, which involves random reinitialization for threads converging to poor local value, iSEM can be deemed as a randomized version of

[3]Note that the empirical value function (1) can be interpreted as a likelihood function for FSCs with the number of mixture components equal to the total number of subepisodes $\sum_{k=1}^K T_k$ [25].

Algorithm 1 iSEM

Require: Episodes $\mathcal{D}_{train}^{(K)}$,$\mathcal{D}_{eval}^{(K)}$, number of MC samples M, maximum iteration number T_{max}, threshold ϵ, $J = \emptyset$, Iter $= 0$

1: **while** $I \neq \emptyset$ **or** Iter $\leq$ T$_{\max}$ **do**
2: $I = \{1, \cdots, M\} \setminus J$, Iter $=$ Iter $+ 1$
3: **for** $i \in I$ **do**
4: Sample $\{\Theta_i\} \sim Dirichlet(1)$
5: Θ_i^{∞} = PoEM(Θ_i, D_{train})
6: Compute $V(D_{eval}, \Theta_i^{\infty})$ using (1)
7: **end for**
8: Compute $\Theta^* = \arg\max_{i \in \{1,\cdots,M\}} V(D_{eval}, \Theta_i^{\infty})$
9: $J = \emptyset$
10: **for** $i = 1$ to M **do**
11: **if** $V(D_{eval}, \Theta^*) - V(D_{eval}, \Theta_i^{\infty}) < \epsilon$ **then**
12: $J = J \cup \{i\}$
13: **end if**
14: **end for**
15: **end while**
16: **return** Controller parameters Θ^*.

the PoEM algorithm. This is essential for convergence to well-performing policies, since it widely known that global optimization paradigms are often based on the principal of stochasticity [26].

iSEM is outlined in Algorithm 1. Domain experience data is first partitioned into training and evaluation sets, $\mathcal{D}_{train}^{(K)}$ and $\mathcal{D}_{eval}^{(K)}$. iSEM takes the partitioned data, the number of Monte Carlo samples (threads) M and parameters controlling convergence as input, and maintains two sets, I and J: I records the indices of threads whose evaluation values are ϵ lower than the best value, and J records the remaining thread indices (and is initialized as empty). iSEM iteratively applies four steps: 1) update I (line 2); 2) for the threads in I, randomly initialize FSC parameters by drawing samples from Dirichlet distributions with concentration parameter 1, run the PoEM algorithm [12] and evaluate the resulting policy $\{\Theta_i^{\infty}\}_{i \in I}$[4] (line 4-6); 3) update the best policy and its evaluation value obtained in current iteration (line 8); 4) update J by recording the indices of threads whose converged policy values are ϵ close to the best policy (line 9-13). Critically, the final step (update of J) enables distinguishing threads that clearly converge to poor local solutions and "good" local solutions. In the subsequent iteration, threads with poor local solutions are reinitialized and re-executed until the policy values from all the threads are ϵ close to the best solution learned so far. The iSEM algorithm is guaranteed to monotonically increase the lower bound of empirical value function over successive iterations and the convergence property is summarized by the following theorem.

Theorem 2: Algorithm 1 monotonically increases $\hat{V}\left(\mathcal{D}^{(K)}; \Theta\right)$, until convergence to a maximum.

Proof: Assume that $\Theta^*(t)$ is a policy with the highest evaluation value among the policies learned by all the threads at iteration t, and the set J_t records the thread indices with corresponding policy value ϵ close to $V(D_{eval}, \Theta^*(t))$. In the iteration $t+1$, the set I_{t+1} contains the thread indices with corresponding policy values satisfy $V(D_{eval}, \Theta^*(t)) - V(D_{eval}, \Theta_i(t)) > \epsilon, \forall i \in I_{t+1} = \{1, \cdots, M\} \setminus J_t$. Starting from $t = 0$, we have $V(D_{eval}, \Theta^*(0)) \geq V(D_{eval}, \Theta_i^{\infty}(0)), \forall i \in I_0 = \{1, \cdots, M\}$. In the next iteration (i.e., $t = 1$), we have $|I_1| = |\{1, \cdots, M\} \setminus J_0| \leq |I_0|$. The steps 4-6 allow the threads in I_1 to rerun with randomly reinitialized parameters. According to step 8 (Algorithm 1), we can obtain $V(D_{eval}, \Theta^*(1)) \geq V(D_{eval}, \Theta^*(0))$. Following the same analysis for $t > 1$, we can obtain $V(D_{eval}, \Theta^*(t)) \geq V(D_{eval}, \Theta^*(t-1))$. Since $\{V(D_{eval}, \Theta^*(t))\}_{t=0}^{\infty}$ is a monotone sequence and it is upper bounded by $\frac{R_{max}}{1-\gamma}$, according to Monotone convergence theorem, $V(D_{eval}, \Theta^*(t))$ has a finite limit, which completes the proof. ■

Note that the convergence of iSEM is different from that of PoEM in the sense that iSEM updates a global parameter estimate based on feedbacks from several local optima (obtained from random initialization). It is also worth mentioning that with finite number of threads, iSEM might still converge to a local maximum. However, we can show that on average, iSEM has higher probability of convergence to better solutions than PoEM. Moreover, the iSEM algorithm can be considered a special case of evolutionary programming (EP) [27], which maintains a population of solutions (i.e., the set of policy parameters in J). Yet, there are obvious differences between iSEM and PE. Notably, instead of mutating from existing solutions, iSEM resamples completely new initializations for parameters and optimizes them using PoEM. In additional, iSEM is highly parallelizable due to its use of concurrent threads.

V. Experiments

This section presents simulation and hardware experiments for evaluating the proposed policy learning algorithm. First a simulator for a large problem motivated by SAR is introduced. Then, the performance of iSEM is compared to previous work based on the simulated SAR problem. Finally, a multi-robot hardware implementation is presented to demonstrate a working real-world system.

A. Search and Rescue Problem

The SAR problem involves a heterogeneous set of robots searching for victims and rescuing survivors after a disaster (e.g., bringing them to a location where medical attention can be provided). Each robot has to make decisions using information gathered from observations and limited communications with teammates. Robots must decide how to explore the environment and how to prioritize rescue operations for the various victims discovered.

The scenario begins after a natural disaster strikes the simulated world. The search and rescue domain considered is a 20×10 unit grid with $s = 6$ designated sites: 1 muster site and 5 victim sites. All robots are initialized at the muster site. Victim sites are randomly populated with victims (6 victims total). Each victim has a randomly-initialized health state.

[4] ∞ sign indicates run the PoEM algorithm until convergence.

While the locations of the sites are known, the number of victims and their health at each site is unknown to the robots. The maximum victim capacity of each site also varies based on the site size. Each victim's health degrades with time.

An unmanned aerial vehicle (UAV) surveys the disaster from above. A set of 3 unmanned ground vehicles (UGVs) can search the space or retrieve victims and deliver them to the muster site, where medical attention is provided. The objective of the team is to maximize the number of victims returned to the muster site while they are still alive. This is a challenging domain due to its sequential decision-making nature, large size (4 agents), and both transition and observation process uncertainty, including stochasticity in communication. Moreover, as communication only happens within a limited radius, synchronization and sharing of global information are prohibited, making this a highly-realistic and challenging domain.

B. Simulator Description

All simulation is conducted within the Robot Operating System (ROS) [28]. The simulator executes a time-stepped model of the scenario, where scenario parameters define the map of the world, number of each type of robot, and locations and initial states of victims.

Each robot's macro-controller policy is executed by a lower-level controller which checks the initiation and termination conditions for the macro-action and generates sequences of primitive actions.

1) Primitive Actions: The simulator models primitive actions, each of which take one time-step to execute. The primitive actions for the robots include: (a) move vehicle, (b) pick-up victim (UGVs only), (c) drop-off victim (UGVs only) and (d) do nothing. Observations and communication occur automatically whenever possible and do not take any additional time to execute.

Macro-action policies, built from these primitive actions, may take any arbitrary amount of time in multiples of the time-steps of the simulator. Macro-action durations are also non-deterministic, as they are a function of the scenario parameters, world state, and inter-robot interactions (e.g., collision avoidance).

2) The World: While the underlying robotics simulators utilized are three-dimensional, the world representation is in two dimensions. This allows increased computational efficiency while not detracting from policy fidelity, as the sites for ground vehicles are ultimately located on a 2D plane. The world is modeled as a 2D plane divided into an evenly-spaced grid within a rectangular boundary of arbitrary size. Each rescue site is a discrete rectangle of grid spaces of arbitrary size within the world.

Some number of victims are initially located in each rescue site. Victim health is represented as a value from 0 to 1, with 1 being perfectly healthy and 0 being deceased. Each victim may start at any level of health, and its health degrades linearly with time. If a victim is brought to the muster location, its health goes to 1 and no longer degrades. One victim at a time may be transported by a UGV to the muster, although this can be generalized to larger settings by allowing the vehicle to carry multiple victims simultaneously.

3) Movement: Simulated dynamical models are used to represent the motion of the air and ground vehicles within ROS. The vehicles can move within the rectangular boundaries of the world defined in the scenario.

UGV motion is modeled using a Dubins car model. Real-time multi-robot collision avoidance is conducted using the reciprocal velocity obstacles (RVO) formulation [29]. State estimates are obtained using a motion capture system, and processed within RVO to compute safe velocity trajectories for the vehicles.

UAV dynamics are modeled using a linearization of a quadrotor around hover state, as detailed in [30]. Since the UAV operates at a higher altitude than UGVs and obstacles, there are no restrictions to the air vehicle's movement.These dynamics correspond to the transition model T specified in the (Mac)Dec-POMDP frameworks discussed in the section II-A.

4) Communication: Communication is range-limited. When robots are within range (which is larger for UAV-UGV communication than for UGV-UGV communication), they will automatically share their observations with two-way communication. Communication is imperfect, and has a .05 probability of failing to occur even when robots are in range. For the scenarios used to generate the results in this study, a UGV can communicate its observation with any other UGV within 3 grid spaces in any direction; the UAV can communicate with any UGV within 6 grid spaces in any direction.

C. MacDec-POMDP Representation

We now describe the MacDec-POMDP represention that is used for learning. Note that the reprentation in Section V-B is not observable to the robots and is only used for constructing the simulator.

1) Rewards: The joint reward is $+1$ for each victim brought back to muster alive and -1 for each victim who dies.

2) Observations: In the SAR domain, a UAV can observe victim locations when over a rescue site. However, victim health status is not observable by air. A UGV that is in a rescue site can observe all victims (location and health status) within that site. Robots are always able to observe their own location and whether they are holding a victim at a given moment.

The observation vector O on which the macro-controller makes decisions is a subset of the raw observations each robot may have accumulated through the execution of the prior macro-action. The robots report the state of their current location and one other location (which could be directly observed or received via communication while completing the macro action). The second location reported is the most urgent state with the most recent new observation. If there are no new observations other than the robot's own location, the second location observation is equivalent to the self location.

The observation vector is as follows,

$$O = [\text{self state}, \text{self location}, \text{location state}, \text{second location}, \text{second location state}] \quad (4)$$

where, self state $\in \{1/0$= is/not holding victim$\}$, self location $\in$ {site 1, site 2, ..., site s}, location state $\in \{0 =$ no victims needing help, $1 =$ victims needing help (not critical), $2 =$ victims needing help (critical)$\}$, second location $\in$ {site 1, site 2, ..., site s}, and second location state $\in \{0 =$ no victims needing help, $1 =$ victims needing help (not critical), $2 =$ victims needing help (critical)$\}$. There are $18s^2$ possible observation vectors, making the observation space substantially larger than previous macro-action based domains [10], [8].

3) Macro-Actions: The macro-actions utilized in this problem are as follows:

- **Go to Muster** (available to both UAV and UGV): Robot attempts to go to the muster point from anywhere else, but only if it is holding a live victim. If a victim is on-board, victim will always disembark at the muster.
- **Pick up Victim** (available only to UGV): Robot (UGV only) attempts to go to a victim's location from a starting point within the site. Terminates when the robot reaches the victim; also may terminate if there is no longer a victim needing help at the site (i.e., another robot picked the victim up first or the victim died). If victim and robot are located in the same grid cell, the victim can be "picked up".
- **Go to Site** $i \in \{1, \ldots, s\}$ (available to both UAV and UGV): Robot goes to a specified disaster site i. Terminates when the robot is in the site. Robot can receive observations of the victims at the site.

D. Simulations and Numerical Results

The SAR domain extends previous benchmarks for MacDec-POMDPs both in terms of the number of robots and the number of states/actions/observations. Notably, due to the very large observation space cardinality of the SAR domain, it is difficult to generate an optimal solution with existing solvers such as [10], [8] in a reasonable amount of time. Hence, due to the lack of a known global optima, the RL algorithms (iSEM and PoEM) are compared over the same datasets. The dataset is collected through the simulator by using a behavior policy combining a hand-coded expert policy (the same used in [12]) and a random policy, with ρ denoting the percentage of expert policy.

To compare iSEM and PoEM on the SAR domain, experiments are conducted with $\rho = [50, 75, 85]$ and $|Q_n| = [1, 3, 10]$ (varying controller sizes)[5]. Corresponding test (holdout) set results are plotted in Figure 1. Several conclusions can be drawn from the results. First, as the amount of training data (K) increases, the cumulative reward increases for both PoEM and iSEM (under the same η, as shown in Fig.1b). Second, with the same K, iSEM achieves better performance than PoEM, which validates that iSEM is better at overcoming the local optimality limitation suffered by PoEM. In addition, as the number of threads M increases, iSEM converges to higher average values and smaller variance (as indicated by the error-bar, compared to PoEM), according to Figure 1c, which empirically justifies the discussion under Theorem 2. Moreover, as shown in Fig.1d, under three settings of $|Q_n|$, the FSCs learned by iSEM render higher value than the PoEM policy. As $|Q_n|$ increases, the difference between PoEM and iSEM (with fixed M) tends to decrease, which indicates we should increase M as iSEM is exploring higher dimensional parameter spaces. Finally, even in cases where the mean of iSEM is only slightly higher than PoEM, the variance of iSEM is is consistently lower than PoEM – a critical performance difference given the uncertainty involved in the underlying domain tested.

[5] $|Q_n| = 1$ corresponds to reactive policies (based only on current observations).

(a) UAV is a DJI F330 quadrotor with onboard Jetson TX1 flight controller.

(b) UGVs are custom-build ground robots with onboard Raspberry Pi 2.

Fig. 2: Robots in used Hardware Implementation.

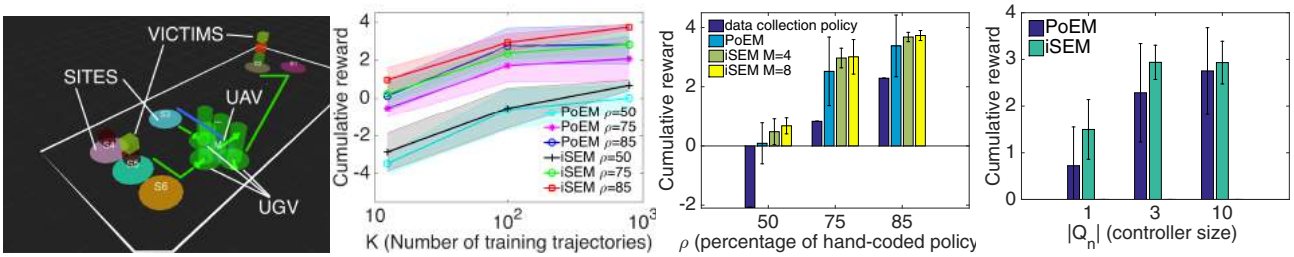
RVIZ [31] was used in conjunction with ROS to visualize the simulations. Fig.1a shows the start of one trial with the different colored circles being sites, the stacked cubes positioned at sites as victims with colors indicating their health values, and the 4 green cylinders indicating the 3 UGVs and the UAV. The sites are as follows (from furthest to closest): site 1 (red circle), site 2 (green), site 3 (sky blue), site 4 (pink), site 5 (turquoise), site 6 (orange). Note that the normal gridworld model used in the POMDP formulation usually assumes discrete state and discrete primitive actions, whereas the simulation models are based on macro-actions which comprised low-level controllers that can deal with both discrete and continuous primitive action and states.

E. Hardware Implementation

While simulation results validate that the proposed MacDec-POMDP search algorithm achieves better performance than state-of-the-art solvers, we also verify the approach on a SAR mission with real robots. This allows further learning from realworld experiences. A video demo is made available online[6]. Learning from simulation allows robots operate in a reasonable (safe) way, whereas real robots experiments can potentially provide "realworld" experiences that are not fully captured by the simulators, hence allowing the robots to improve their baseline policy (learned from simulators). The video essentially demonstrates this potential, assuming the training data is collected from the "realworld".

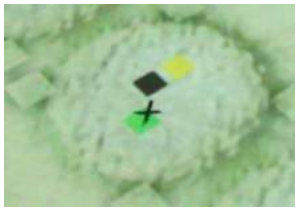
A DJI F330 quadrotor is used as the UAV for hardware experiments, with a custom autopilot for low-level control and an NVIDIA Jetson TX1 for high-level planning and task allocation (Fig. 2a). The UGVs are custom-made ground robots with an onboard Raspberry Pi 2 for computation (Fig. 2b). Experiments were conducted in a 40 ft. $\times$ 20 ft. flight space with a ceiling-mounted projection system used to visualize site locations, obstacles, and victims. As discussed

[6] Video URL: https://youtu.be/B3b60VqWMIE

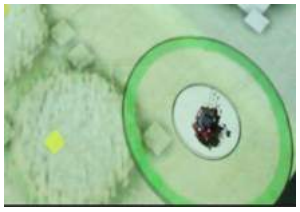


(a) (b) (c) (d)

Fig. 1: (a) RVIZ simulation of the experiment; Testing performance using (b) different number of training samples (with Q_n=10), (c) threads (with Q_n=10, K=500), and (d) controller sizes (with $M = 8, K = 100, \rho = 85$).



(a) Zoomed in view of 3 victims (indicated as squares) at a particular site.

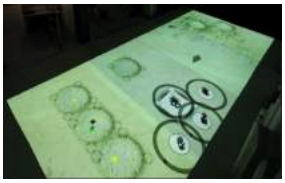


(b) UGV observes the victim at the site with high health.

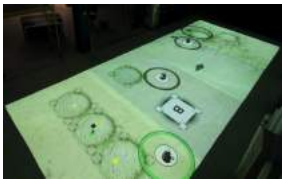
(c) UGV outer ring color is black, as there are no other victims at this site.

(d) UAV can only observe, but not carry, victims. Thus, it only has an outer ring indicating observations.

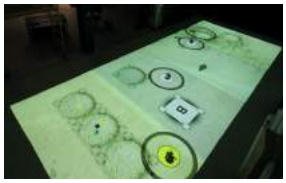
Fig. 3: Overview of hardware domain with 1 UAV and 3 UGVs. Projection system used to visualize sites and victim locations/health state. Victims shown as squares with colors representing health (green: high health, yellow: low health, red: critical health, black: deceased). For all robots, outer ring color indicates its noisy observation of the health of one of the victims present. For UGVs, inner circle color indicates health of the victim it is carrying.



(a) Start of experiment

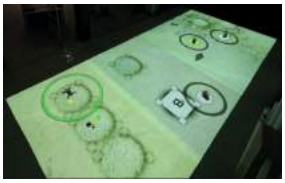


(b) UGV observes victim at site 6



(c) UGV picks up a victim, observes no others at site 6

(d) UGV drops off victim at muster



(e) UAV observes victim at site

(f) UGV picks up victim from site 5, observes another healthy victim at site

(g) UGV picks up a victim from site 5, observes no more healthy victims at site 5

(h) All healthy victims have been saved. End of experiment

Fig. 4: Overview of hardware domain, with 1 UAV and 3 UGVs. Ceiling-mounted projection system used to visualize sites and victim locations/health state.

earlier, limited communication occurs between robots, with a motion capture system used to ensure adherence to maximal inter-robot communication distances.

The hardware experiments conducted demonstrated that the policy generated from iSEM (with $\rho > 75$, $K > 100$, $Q_n > 3$) was able to save all victims consistently well, despite robots having to adhere to collision avoidance constraints. In some instances, the robots were not able to save all 6 victims. However, in these scenarios, only 1 victim was lost, with the cause of loss due to an extremely low starting health for multiple victims. In such cases, an early victim death would occur before any robot could respond.

Fig. 4 shows the progression of one hardware trial. Sites are randomly populated with 6 victims total. All robots initiate at the muster site (Fig. 4a). As the UGVs navigate towards sites (dictated by their policy), they simultaneously begin observing their surroundings. When they do, the outer ring surrounding them turns into the color of the latest victim observed (Fig. 4b). A UGV can only pick up a new victim if it is not currently carrying a victim. Its inner circle then

indicates the health of the victim it is carrying, while its outer ring indicates the health of a randomly-selected victim still present at the site (if any). Fig. 4c illustrates a situation where no more victims are present at site 6, thereby causing the UGV's outer ring to turn black (no victims to save at latest encountered site). Note that an observed deceased victim also falls under this category. After a UGV picks up a victim, it drops it off at the muster (Fig. 4d). The victim returns to full health, indicating a successful rescue. When a UAV visits a site, its outer ring also turns into the color of the victim observed at the site (Fig. 4e). The UAV has no inner circle because it cannot pick up victims. Fig. 4f and 4g show two more instances of a UGV picking up a victim from site 5. As mentioned before, a deceased victim results in a observation color of black in Fig. 4g. Fig. 4h shows the end of the hardware trial, where all healthy victims have been rescued.

VI. Conclusion

This paper presents iSEM, an efficient algorithm which improves the state-of-the-art learning-based methods for coordinating multiple robots operating in partially observed environments. iSEM enables cooperative sequential decision making under uncertainty by modeling the problem as a MacDec-POMDP and using iterative sampling based Expectation Maximization trials to automatically learn macro-action FSCs. The proposed algorithm is demonstrated to address local convergence issues of the state-of-the-art macro-action based reinforcement learning approach, PoEM. Moreover, simulation results showed that iSEM is able to generate higher-quality solutions with fewer demonstrations than PoEM. The iSEM policy is then applied to a hardware-based multi-robot search and rescue domain, where we demonstrate effective control of a team of distributed robots to cooperate in this partially observable stochastic environment. In the future, we will make our demonstration even closer to real world scenarios by modeling observations and communications as actions and assigning costs. We will also experiment with other methods other than random sampling, such as active sampling for the resampling step in iSEM, to accommodate restrictions of computational resources (i.e., number of threads).

References

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

[2] K. Dresner and P. Stone, "Multiagent traffic management: Opportunities for multiagent learning," in *Learning and Adaption in Multi-Agent Systems*. Springer, 2006, pp. 129–138.

[3] D. Bernstein, S. Zilberstein, R. Washington, and J. Bresina, "Planetary rover control as a markov decision process," in *Sixth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2001.

[4] K. Jolly, K. Ravindran, R. Vijayakumar, and R. S. Kumar, "Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 589–596, 2007.

[5] M. Gath, *Optimizing transport logistics processes with multiagent planning and control*. Springer, 2016.

[6] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

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

[8] C. Amato, G. Konidaris, G. Cruz, C. Maynor, J. How, and L. Kaelbling, "Planning for decentralized control of multiple robots under uncertainty," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, 2015.

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

[10] S. Omidshafiei, A. akbar Agha-mohammadi, C. Amato, and J. P. How, "Decentralized control of partially observable Markov decision processes using belief space macro-actions." in *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[11] S. Omidshafiei, A.-a. Agha-mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Graph-based cross entropy method for solving multi-robot decentralized pomdps," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5395–5402.

[12] M. Liu, C. Amato, E. Anesta, J. Griffith, and J. How, "Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments," in *AAAI Conference on Artificial Intelligence*, 2016.

[13] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.

[14] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[15] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[16] C. Amato, B. Bonet, and S. Zilberstein, "Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs." 2010.

[17] A. Kumar, S. Zilberstein, and M. Toussaint, "Probabilistic inference techniques for scalable multiagent decision making," *Journal of Artificial Intelligence Research*, vol. 53, no. 1, pp. 223–270, 2015.

[18] Z. Song, X. Liao, and L. Carin, "Solving DEC-POMDPs by expectation maximization of value functions," 2016.

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

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

[21] C. Y. Wong, G. Seet, and S. K. Sim, "Multiple-robot systems for USAR: key design attributes and deployment issues," *International Journal of Advanced Robotic Systems*, vol. 8, no. 1, pp. 85–101, 2011.

[22] Y. Jin, A. A. Minai, and M. M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, 2003, pp. 7–12.

[23] D. Turra, L. Pollini, and M. Innocenti, "Fast unmanned vehicles task allocation with moving targets," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, 2004, pp. 4280–4285.

[24] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

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

[26] R. Horst and P. M. Pardalos, *Handbook of global optimization*. Springer Science & Business Media, 2013, vol. 2.

[27] D. Simon, "Evolutionary optimization algorithms: biologically-inspired and population-based approaches to computer intelligence. hoboken," 2013.

[28] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[29] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928–1935.

[30] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 664–674, 2012.

[31] D. Gossow and W. Woodall. (2016, nov) RVIZ. http://wiki.ros.org/rviz.