Multiagent Reinforcement Learning: Rollout and Policy Iteration for POMDP With Application to Multirobot Problems

Sushmita Bhattacharya, Siva Kailas, Sahil Badyal, Stephanie Gil, *Member, IEEE*, and Dimitri Bertsekas

***Abstract*—In this article, we consider the computational and communication challenges of partially observable multiagent sequential decision-making problems. We present algorithms that simultaneously or sequentially optimize the agents' controls by using multistep lookahead, truncated rollout with a known base policy, and a terminal cost function approximation. In particular: 1) we consider multiagent rollout algorithms that dramatically reduce required computation while preserving the key policy improvement property of the standard rollout method. We improve our multiagent rollout policy by incorporating it in an offline approximate policy iteration scheme, and we apply an additional "online play" scheme enhancing offline approximation architectures; 2) we consider the imperfect communication case and provide various extensions to our rollout methods to deal with this case; and 3) we demonstrate the performance of our methods in extensive simulations by applying our method to a challenging partially observable multiagent sequential repair problem (state space size 10^{37} and control space size 10^{7}). Our extensive simulations demonstrate that our methods produce better policies for large and complex multiagent problems in comparison with existing methods, including POMCP, MADDPG, and work well where other methods fail to scale up.**

***Index Terms*—Approximate policy iteration (approximate PI), imperfect communication, multiagent reinforcement learning, multiagent rollout, online play policy, partial observation Markovian decision problem (POMDP).**

I. Introduction

We consider the classical partial observation Markovian decision problem (POMDP) with a finite number of states and controls, and discounted additive cost over an infinite horizon. We focus on a version of the problem that has a multiagent character: it involves a control that has multiple components, each corresponding to a different agent. This version of the problem is suitable for multiagent sequential decision-making tasks. We address several concerns that are typical in realistic scenarios, including partial state observation, a large state space, a large control space, and imperfect communication between agents. We showcase the performance of our proposed methods on an important class of multirobot repair problems under partial state observation. Fig. 1 shows an instance of the repair problem where four agents coordinate with each other to repair a set of partially observable damaged locations in a network. An optimal solution to such problems by dynamic programming is typically intractable. In this article, we instead propose a suboptimal solution/reinforcement learning approach, whose principal characteristic is the proper exploitation of the multiagent structure to dramatically reduce the computational requirements of the solution method. We extend and deploy the multiagent rollout and policy iteration (PI) ideas (proposed in [3], [4] for the perfect observation case), to a partially observable multiagent decision-making framework that extends to the imperfect agent communication cases.

The standard form of rollout [5], starts with some easily implementable policy, called the *base policy*, and produces another

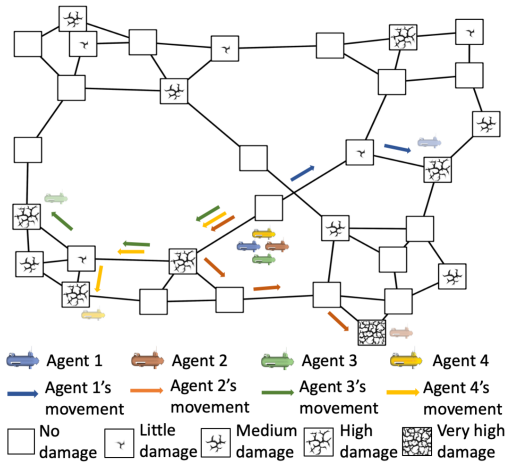


Fig. 1. Instance of multiagent sequential decision-making problem where four agents coordinate in repairing a partially observable network of 32 vertices, each with a different level of damage.

Manuscript received 1 May 2023; accepted 14 October 2023. Date of publication 26 December 2023; date of current version 11 March 2024. This paper was recommended for publication by Associate Editor R. Tron and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported in part by ONR Young Investigators Award under Grant N00014-21-1-2714, in part by Sloan Research Fellowship under Grant FG-2020-13998, in part by NSF CPS award 2114733, and in part by Apple Scholars in AI/ML Ph.D. Fellowship Program. *(Corresponding author: Sushmita Bhattacharya.)*

Sushmita Bhattacharya and Stephanie Gil are with the REACT Lab, Computer Science Department, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02139 USA (e-mail: sushmita_bhattacharya@g.harvard.edu; sgil@seas.harvard.edu).

Siva Kailas and Sahil Badyal are with REACT Lab, Cambridge, MA 02139 USA (e-mail: skailas@cs.cmu.edu; sbadyal@asu.edu).

Dimitri Bertsekas is with the Department of Computer, Information, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281 USA, and also with the Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: dimitrib@mit.edu).

Digital Object Identifier 10.1109/TRO.2023.3347128

policy, the *rollout policy*, using one-step or multistep lookahead optimization. Its key property is *the policy improvement property:* here, the rollout policy improves performance over the base policy for all states. In a multiagent setting, the lookahead optimization portion of the standard rollout algorithm becomes very computationally expensive. By contrast, in our multiagent rollout approach, the computation complexity of the lookahead optimization is dramatically reduced while maintaining the fundamental policy improvement property. Our multiagent rollout policy is implemented approximately using truncated rollout and a terminal cost approximation, where, the policy improvement property applies in an approximate form, which is quantified by an error bound not worse than that of the standard rollout.

We employ our multiagent rollout algorithm in an approximate policy iteration (approximate PI) framework in order to improve the policy in an iterative fashion where successive policies are approximated using neural networks. The performance of a purely offline trained policy may degrade in a complex and dynamic environment. Thus, we study an "online play policy" (first introduced in [6] for the perfect observation case) for much more complex problems due to the partial state observation. The "online play policy" performs an online lookahead optimization with an offline trained policy as the base policy and an optional terminal cost approximation, trained offline. We show in our simulation experiments that the performance of the online play policy is superior than both our multiagent rollout (without any offline training) and our approximate PI algorithms (with offline approximations). This performance improvement can be attributed to the fact that the online play implements a true Newton step that has a superlinear convergence property, making the online play policy converge to the optimal policy much faster than the offline approximations (see [6]).

We demonstrate the implementation of our multiagent rollout methods on a challenging class of multirobot repair problems, where a policy needs to identify and execute critical repairs in minimum time by leveraging coordination among the agents. We apply our method to a complex repair problem involving a network of 500 partially observable, potentially damaged locations, and as many as 50 repair robots/agents in our largest experiment (see Fig. 1, where four agents are employed). In particular, we present favorable comparisons (with four agents) of our proposed method with the state-of-the-art POMDP solvers partially observable Monte-Carlo planning (POMCP [1]), multiagent deep deterministic policy gradient (MADDPG [2]), and POMCP with action prioritization and progressive widening (PA-POMCPOW [7]).

We relax the perfect communication assumption by considering practical extensions where agents cannot communicate their controls to one another at all times; instead, each agent estimates other agents' control using a *signaling policy*. A *signaling policy* applied by an agent guesses the control components for other agents without the exact knowledge of other agents' computed control components. However, the loss of perfect communication between the agents leads to great challenges, including the lack of the policy improvement property of the multiagent rollout. Furthermore, we show that imperfect communication may result in the loss of finite termination for the learned policy in some cases. We show that the incorporation of a simple randomized policy can recover finite termination without communication of controls. In addition, we study a communication scheme where a cloud server is intermittently available, and provides the communication of the computed control components of each agent so that agents can perform the multiagent rollout. When the cloud server is not available, the agents apply the control given by the base policy without performing any optimization. We show that under this scheme, the policy improvement property can be recovered. We present extensive numerical results demonstrating the performance of various imperfect communication architectures for multiagent rollout and a comparison study with a recent method A3C3 [8] in the large and complex POMDP setup for the multirobot repair problem where agents may not always share belief states and computed controls.

This article is most closely related to the multiagent rollout methods developed in [3], online play policy [6], and autonomous repair problems [9]. Our work differs in various important ways from this prior work as follows.

1) We treat the case of the partially observable state, leading to an explosion in the size of the state space, not addressed in [3].
2) We develop a multiagent decision-making framework in this partially observable context, leading to an explosion in the size of the action space, not treated in [9].
3) We employ the online play policy to a partially observable multirobot repair problem (not discussed in [6]).
4) We consider more general repair problems described over arbitrary graphs (in contrast with the linear or strict grid topology in [9]) and treat the significantly more challenging and realistic nonterminating case where previously repaired locations can fall into disrepair (not treated in [9]).
5) We consider practical issues of imperfect communication in the multiagent rollout that are not discussed in [3].

This article evolved from our earlier conference publication [10], and the **new contributions in this article which were not included in [10]** are the following.

1) We discuss the online play policy for multiagent sequential decision-making problems under partial state observation, which has not been explored in previous work to the best of our knowledge.
2) We present in-depth analytical results for the imperfect communication cases. We show that not communicating controls to other agent hinders finite termination and we show how to recover the finite termination by using a randomized approach. Finally, we show that the policy improvement property is recoverable using an intermittent communication architecture.
3) We present extensive numerical simulations on a multirobot repair problem using our approach where damages can propagate to the neighboring locations, additional comparison study with multiagent POMDP planning methods, including PA-POMCPOW [7] (for perfect agent communication) and A3C3 [8] (for imperfect agent communication). We show that the online play significantly improves policy and cost approximations trained using an offline approximate PI which is robust and adaptable with dynamically changing system parameters.

II. Related Work

Several reinforcement learning algorithms for POMDP problems have been proposed in the literature. In particular, [11] describes a general solution method for POMDP, [12] discusses a policy search method using finite state controllers, [5], [13], [14] discuss aggregation-based methods, and [15], [16], [17] consider actor–critic-based policy gradient methods. These methods are fundamentally different from our proposed rollout-based methodologies, as they do not directly rely on policy improvement starting from a base policy.

Among works in the POMDP literature, there are some that like our rollout-based methods, use lookahead minimization, and also try to tradeoff the length of simulated trajectories with variable length lookahead tree and pruning. In particular, POMCP [1] uses multistep lookahead and Monte-Carlo tree search (MCTS) to generate a suboptimal policy, and determinized sparse partially observable tree (DESPOT [18]) similarly reduces the lookahead search tree by adaptive pruning. However, these methods do not use any kind of rollout with a base policy. Furthermore, POMCP and DESPOT do not address multiagent issues.

On the other hand, various multiagent reinforcement learning and policy gradient methods [19], [20] have been proposed. Among them, [21] deals with multiagent cooperative planning under uncertainty in POMDP using decentralized belief sharing and policy auction, done after each agent executes a value iteration. The paper [7] discusses ways to POMCP for multiple agents with PA-POMCPOW. The papers [2], [22] consider an actor–critic policy gradient approach that scales well with multiple agents. However, the per-agent policy networks use only the local observations and do not leverage any extra information when the agents fully or partially communicate between themselves about their controls and observations. By contrast, our methodology uses extra information from other agents and the cloud server whenever available. Papers [8], [23], [24] discuss decentralized action and communication policies for multiagent problems, where agents learns when and how to communicate information to other agents. However, these works assume the availability of a centralized critic that helps training decentralized policies, whereas our methods do not depend on such assumptions. In Section VII, we compare the performance of our methods to several of these state-of-the-art POMDP methods.

The paper [9] proposes rollout and PI methods that can address POMDP, but does not deal with multiagent problems and has difficulty dealing with a large control space. The paper [9] presents simulation results for a partially observable repair problem with two agents, but it does not consider cases with more agents since the rollout method discussed in [9] optimizes over Q-factors corresponding to all combinations of the controls by all agents, which is exponential in the number of agents. In contrast, this article addresses this issue by decomposing the control space. The rollout policy improvement property, given in [9], also holds for the multiagent version of this article. The proof was given in [3] and an associated performance bound was given in the research monograph [4]. The paper [6] interprets the online play policy in the context of rollout as a Newton step to improve offline trained policies with an online optimization and their superlinear convergence to the optimal policy but does not consider partial state observation and its associated challenges. AlphaZero [25] discusses a scheme whereby MCTS-based online lookahead is used to improve offline approximation architectures given by deep neural networks in both policy and value spaces for perfect-state observation cases. In contrast, this article considers online play for partially observable multiagent decision-making problems with a smaller but denser lookahead optimization that improves over offline trained policy and value approximations with shallow networks.

The methods discussed in this article are well suited for related multiagent contexts, such as search and rescue applications [26], [27], [28]. Lauri et al. [29] discussed the problem of decentralized information gathering as a discrete multiagent POMDP using policy graph improvement, where the authors consider cases for convex reward functions, which may not be directly applied to problems, such as our multiagent partially observable repair problem. We do not consider continuous control space POMDPs, including multiagent path finding [30] that are not directly comparable to our approach. Among other challenges that we do not consider is the generation safe multiagent policies, [31] discusses sufficient and necessary conditions for safe policy synthesis for multirobot safety-critical problems using discrete-time barrier functions.

III. Belief Space Problem Formulation for Multiagent POMDP

We introduce the classical belief space formulation of POMDP. We assume that there are n states denoted by $i = 1, \ldots, n$, and that the control u consists of m components, $u = (u_1, u_2, \ldots, u_m)$. Each of the components corresponds to a separate agent. Given a starting state i, and control vector u, there is a known transition probability to reach the next state j, which is denoted by $p_{ij}(u)$. Each component of the control $u_\ell, \ell \in \{1, 2, \ldots, m\}$, must belong to a finite set U_ℓ, so that the control space U is the Cartesian product $U_1 \times U_2 \times \cdots \times U_m$. The cost at each stage is denoted by $g(i, u, j)$ and is discounted with a factor $\alpha \in (0, 1)$. The total cost is the sum of the α-discounted expected costs incurred over an infinite horizon.

We assume that a transition from state i to the next state j under control u, will generate an observation z with a probability $p(z \mid j, u)$, where z belongs to a known finite set Z. However, we assume that the agents share observations, so that all computations are done with full knowledge of the entire history of the observation vectors. Our goal is to determine the control component for each agent at every stage as a function of the current belief state, which minimizes the discounted expected total cost, starting from any initial belief state.

We use the belief space transformation of a POMDP to a problem of perfect state information, similar to the belief space transformation used in [9]. In particular, the belief state is the conditional probability vector $b = (b(1), \ldots, b(n))$, where $b(i)$ is the conditional probability that the state is i, given the control-observation history up to the current time. The belief state

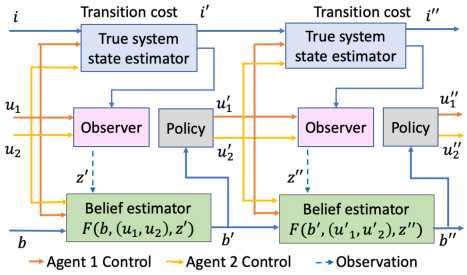


Fig. 2. Composite system simulator for POMDP for a given policy involving two agents (adapted from [9]). The starting state i of a trajectory is generated randomly using the belief state b.

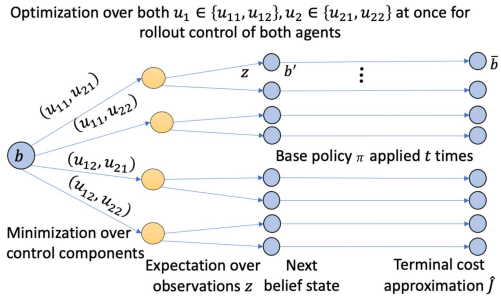


Fig. 3. Standard truncated rollout algorithm for two agents: one-step lookahead followed by t applications of the base policy π, and cost approximation $\hat{J}$.

can be sequentially updated using a belief estimator $F(b,u,z)$, from a given belief state b, control u, and observation z (see Fig. 2). Our formulation considers that the knowledge of the belief, observation, and control components are shared among the agents up until Section VII. Sections VIII-D, VIII-E, VIII-F, and VIII-G consider cases where control is either never shared or imperfectly shared, but belief and observation are shared. Section VIII-H considers cases where control components, observation, and belief are imperfectly shared. The optimal cost function $J^*(b)$ is the unique solution of the Bellman equation

$$J^*(b) = \min_{u\in U}\left[\hat{g}(b,u) + \alpha \sum_{z\in Z} \hat{p}\,(z\,|\,b,u)\,J^*\,(F(b,u,z))\right]$$

where F is the belief estimator, and

$$\hat{g}(b,u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} p_{ij}(u) g(i,u,j)$$

$$\hat{p}(z\,|\,b,u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} p_{ij}(u) p(z\mid j,u).$$

Our suboptimal solution approach is based on approximation in value space, implemented through the use of rollout. In particular, we replace J^* in the Bellman equation with an approximation $\tilde{J}$. The corresponding suboptimal rollout policy $\tilde{\pi}$ is obtained by the one-step lookahead minimization

$$\tilde{\pi}(b) \in \arg\min_{u\in U}\left[\hat{g}(b,u) + \alpha \sum_{z\in Z} \hat{p}(z|b,u)\tilde{J}\,(F(b,u,z))\right]. \tag{1}$$

A more general version involves multistep lookahead minimization. In the pure form of rollout, we use $\tilde{J}$ as the cost function of some policy, referred to as the base policy. In the next section, we define a rollout algorithm, which uses a simplified agent-by-agent lookahead minimization, and approximations $\tilde{J}$ that involve a base policy with trajectory truncation and terminal cost approximation.

IV. Multiagent Truncated Rollout With Cost Function Approximation

In the pure form of rollout with l-step lookahead, to find the rollout control at the current belief state b, we form an l-step lookahead tree using the transition and observation probabilities (see Fig. 3). Starting from each leaf node b' of the tree, we use the cost of the base policy π as the cost approximation in (1) [$\tilde{J}(b') = J_\pi(b')$, where $J_\pi(b')$ is the discounted cost of applying the policy π starting from belief state b' until termination]. In the truncated rollout version, $\tilde{J}(b')$ is the discounted cost of applying a base policy π for a given number of stages t, starting from the leaf node b', followed by a terminal cost function approximation $\hat{J}(\bar{b})$. Here, $\bar{b}$ is the belief state obtained at the end of the t steps of application of the base policy starting from b'. In other words, we truncate the system trajectory at the belief state $\bar{b}$, and we approximate the cost of the remainder of the trajectory with $\hat{J}(\bar{b})$.

The truncated rollout algorithm involves a few parameters: the lookahead length l, the length of the simulated trajectory before truncation t, the choice of the base policy π, and the terminal cost function approximation $\hat{J}$. The parameters l and t are usually chosen based on a tradeoff between implementation complexity and obtained performance. The base policy can be a greedy policy, and the terminal cost function approximation $\hat{J}(\bar{b})$ can be an estimate of the cost function of the base policy or an estimated steady-state cost from the belief state $\bar{b}$, or it may be simply set to 0. The paper [9] (Prop. 1) provides theoretical performance bounds on the policy improvement of the truncated rollout algorithm. These bounds indicate, among others, that increasing the lookahead length l improves the rollout performance bound. Moreover, they state that the performance of the rollout policy $\tilde{\pi}$ improves over the base policy π as the terminal cost function approximation $\hat{J}$ gets closer to the base policy cost J_π.

A. Standard Rollout (All-at-Once)

In the standard form of rollout with multiple agents, at the current belief state b, we construct an l-step lookahead tree where each branch represents a possible control vector $u = (u_1,\ldots,u_m)$, where $u_\ell \in U_\ell$, $\ell = 1,\ldots,m$ (see [3]). The branch corresponding to control u is associated with a Q-factor corresponding to (b,u), which is $\hat{g}(b,u) + \alpha\sum_{z\in Z}\hat{p}(z|b,u)\tilde{J}(F(b,u,z))$, the expression in brackets in (1). The standard rollout algorithm chooses the control that is associated with minimal Q-factor, cf., (1). This rollout algorithm in the pure form, where $\tilde{J}$ is given by J_π in (1)

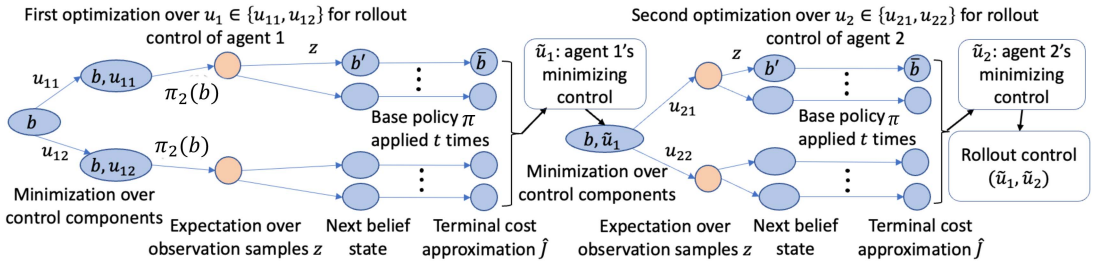


Fig. 4. One-agent-at-a-time truncated rollout algorithm for two agents using base policy π with terminal cost approximation $\hat{J}$ on the reformulated state space $(b), (b, \tilde{u}_1)$.

possesses the policy improvement property in an exact form

$$J_{\tilde{\pi}}(b) \leq J_{\pi}(b)$$

where π is the base policy and $\tilde{\pi}$ is the rollout policy that does not use a terminal cost approximation. Fig. 3 demonstrates standard rollout with two agents, each having two possible control components. The difficulty with this formulation is that the overall rollout algorithm computation with 1-step lookahead is of order $O(C^m)$ at each stage, where $C = \max\{|U_1|, |U_2|, \ldots, |U_m|\}$ is the maximum cardinality of the control component constraint sets. To alleviate this difficulty, we will introduce next a multiagent variant of rollout, where the lookahead minimization is performed one agent at a time, and the computation at each stage is reduced to $O(Cm)$.

B. One-Agent-at-a-Time Rollout (One-at-a-Time)

In order to reduce the algorithmic complexity of the standard rollout algorithm, the minimization over the control branches in the abovementioned formulation needs to be simplified. To achieve improved algorithmic complexity, we introduce an equivalent problem formulation where the control $u = (u_1, u_2, \ldots, u_m)$ is broken down into its m components. Given a belief state b, m intermediate states are generated such that the agents choose their control components sequentially between the current belief state b and the next belief state b'. Thus, the transition sequence from b and b' is $\{b, (b, u_1), (b, u_1, u_2), \ldots, (b, u_1, u_2, \ldots, u_{m-1}), b'\}$ assuming the agents choose their controls sequentially in a fixed order. The last transition from $(b, u_1, u_2, \ldots, u_{m-1})$ to b' involves the choice of the last component u_m and includes the cost $\hat{g}(b, u)$ of choosing control $u = (u_1, u_2, \ldots, u_m)$ at the current belief state b. Every other intermediate transition has 0 cost. In the reformulated problem, at each stage, the rollout algorithm performs m sequential optimizations over Q-factors that involve a single control component. The one-agent-at-a-time rollout control thus produced is denoted by $\tilde{\pi}(b) = (\tilde{\pi}_1(b), \ldots, \tilde{\pi}_m(b))$, where $\tilde{\pi}_\ell(b)$ is the control component for agent ℓ at belief state b, $\ell = \{1, \ldots, m\}$. When optimizing over the Q-factors of the component corresponding to agent ℓ, we set $u_1, \ldots, u_{\ell-1}$ to $\tilde{\pi}_1(b), \ldots, \tilde{\pi}_{\ell-1}(b)$, the optimized values calculated earlier by the rollout algorithm, and we set $u_{\ell+1}, \ldots, u_m$ to the values dictated by the base policy at belief state b

$$\tilde{\pi}_\ell(b) \in \arg\min_{u_\ell \in U_\ell} \left[\hat{g}(b, u') + \alpha \sum_{z \in Z} \hat{p}(z|b, u') \tilde{J}(F(b, u', z)) \right] \tag{2}$$

where $u' = (\tilde{\pi}_1(b), \ldots, \tilde{\pi}_{\ell-1}(b), u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b))$, and $\pi(b) = (\pi_1(b), \ldots, \pi_m(b))$ is the base policy's control at b.

The per-stage complexity of this rollout algorithm is $O(Cm)$, which is a dramatic improvement over the exponential computational complexity of the (all-at-once) standard rollout algorithm. Our numerical experiments are consistent with the theoretical results, namely that this computational economy is often obtained with minimal loss of performance. The algorithm is illustrated in Fig. 4.

In [3] and the research monograph [4], the one-agent-at-a-time rollout method was shown to maintain the policy improvement property of standard rollout in the perfect state observation case. For the case of one-step lookahead, it was also shown that the performance bounds for the standard and the one-agent-at-a-time truncated rollout algorithms are identical (see Prop. 5.2.7 of [4]). These properties were proved for the perfectly observable case where the number of states is finite. However, the arguments of the proof do not depend on the finiteness of the state space and can be extended to our POMDP case with infinite belief space.

C. Order-Optimized Rollout

We extend a variant of one-agent-at-a-time rollout called order-optimized rollout (proposed in [4] for perfect state information) in the case of partial state observation. The one-agent-at-a-time rollout algorithm assumes a fixed order chosen a priori in which the agent control components are optimized. However, the algorithm also works with any agent order, and in fact, it also works if the order is changed at each stage. Appendix A shows that the policy improvement property of rollout holds if the order of agents changes at each round of rollout. This motivates algorithmic variants where the agent order is approximately optimized at each stage. An effective and relatively inexpensive way to do this is to first optimize over all single agent Q-factors, by solving the m minimization problems that correspond to each of the agents $\ell = 1, \ldots, m$ being first in the one-agent-at-a-time rollout order. If ℓ_1 is the agent that produces the minimal Q-factor, we fix ℓ_1 to be the first agent in the one-agent-at-a-time rollout order. Then we optimize

over all single agent Q-factors, by solving the $m-1$ Q-factor minimization problems that correspond to each of the agents $\ell \neq \ell_1$ being second in the one-agent-at-a-time rollout order. If ℓ_2 is the agent that produces the minimal Q-factor, we fix ℓ_2 to be the second agent in the one-agent-at-a-time rollout order, and continue in this manner. In the end, after $m(m+1)/2$ minimizations, we obtain an agent order $\ell_1, \ldots, \ell_m$ that produces a potentially reduced Q-factor value, as well as the corresponding rollout control component selections. Based on our experimental results, agent order optimization produces modest, but consistent performance improvement over the case of a fixed agent order.

V. Multiagent Approximate PI

We will now discuss the approximate PI method as an extension to the rollout algorithm. The truncated rollout policy can be considered as the base policy in PI. The policy evaluation is done in an online fashion by 1-step lookahead minimization over the simulated trajectories (using t times base policy π application followed by an optional terminal cost approximation $\tilde{J}$) at each stage. The subsequent iterations can be expedited by replacing the online evaluation of the rollout policy with an approximation architecture (namely, a neural network). See [5], Sections 2.1.5 and 5.7.2. Here, the newly trained approximation architecture for the rollout policy serves as the subsequent base policy for the next iteration. Using the multiagent truncated rollout as a basis, we now describe corresponding approximate PI algorithms.

A. Approximate PI With Truncated Rollout

This algorithm uses standard rollout to generate the belief state–rollout control pairs to train the policy network in each iteration. We define a parametric policy approximation $\hat{\pi}(b, \bar{r})$, that produces a control given a belief state b, where $\bar{r}$ is the parameter of the approximation architecture. For example, a neural network can be trained with a large set consisting of q belief state–control pairs $(b^s, \tilde{u}^s)$, $s = 1, \ldots, q$, in a supervised learning fashion, where $\bar{r}$ may include the weights of each layer. We can estimate the rollout control $\tilde{u}^s$ from a belief state b^s and add it to the training set. The training process solves an optimization/classification problem using the training set and generates a neural network-based approximation $\hat{\pi}(., \bar{r})$ for the rollout policy, which in turn is used as the base policy for the next iteration. This was proposed in the context of PI for the perfect observation case in the paper [32], and is also described in the book [5], Section 3.5. The corresponding computation is expensive, especially for a large number of agents, using C^m Q-factors. Instead, we extend this idea for the one-at-a-time case discussed next.

B. Approximate PI With Truncated One-Agent-at-a-Time Rollout (One-at-a-Time API)

This algorithm uses the one-agent-at-a-time rollout scheme to train the parametric architecture for policy space approximation. Given a belief state b^s, the one-agent-at-a-time rollout algorithm produces one agent's control component $\tilde{u}^s_\ell$, $\ell \in \{1, \ldots, m\}$ at a time. Each component $\tilde{u}^s_\ell$ of the rollout policy, starting from

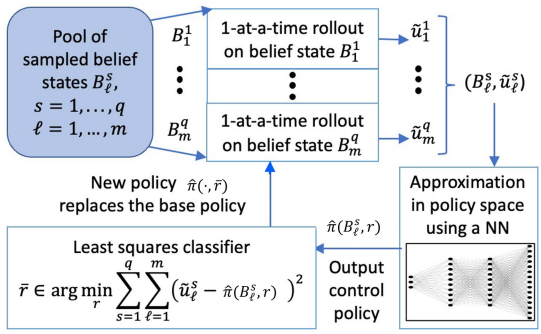


Fig. 5. Approximate policy iteration based on multiagent rollout and approximation in policy space. This figure is adapted from [9], which we extend by applying the one-at-a-time rollout in the policy improvement phase for tackling multiple agents (not given in [9]).

$\ell = \{1, \ldots, m\}$ is given by the following equation at each stage:

$$\tilde{u}^s_\ell \in \arg\min_{u_\ell \in U_\ell} \left[\hat{g}(b^s, u'^s) + \alpha \sum_{z \in Z} \hat{p}(z \mid b^s, u'^s) \tilde{J}\left(F(b^s, u'^s, z)\right)\right]$$

where $u'^s = (\tilde{u}^s_1, \ldots, \tilde{u}^s_{\ell-1}, u_\ell, \pi_{\ell+1}(b^s), \ldots, \pi_m(b^s))$. Here, $\pi(b^s) = (\pi_1(b^s), \ldots, \pi_m(b^s))$ denotes the base policy's control at belief state b^s. At the end of m such optimizations, we construct the entire rollout control $\tilde{u}^s = (\tilde{u}^s_1, \ldots, \tilde{u}^s_m)$. All pairs of $(B^s_\ell, \tilde{u}^s_\ell)$ for $\ell = \{1, \ldots, m\}, s = \{1, \ldots, q\}$, where $B^s_\ell = (b^s, \ell, \tilde{u}^s_1, \ldots, \tilde{u}^s_{\ell-1}, \pi_{\ell+1}(b^s), \ldots, \pi_m(b^s))$ are used to train the approximation architecture and obtain the policy network $\hat{\pi}(., \bar{r})$. The policy network is trained with $q \cdot m$ samples in total. To construct the entire control vector from the approximation architecture in an iteration, we need to invoke the policy network m times. For example, when estimating the first component $\tilde{u}_1$ of the rollout control $\tilde{u}$ for a belief state b, we need to construct tuple $B_1 = (b, 1, \pi_2(b), \ldots, \pi_m(b))$. Invoking the policy network $\hat{\pi}(B_1, \bar{r})$ will produce the first rollout control component $\tilde{u}_1$. The second component $\tilde{u}_2$ of the rollout control is similarly estimated by first constructing the tuple $B_2 = (b, 2, \tilde{u}_1, \pi_3(b), \ldots, \pi_m(b))$ and invoking the policy network $\hat{\pi}(B_2, \bar{r})$. Repeating this process will produce the entire rollout control $\tilde{u} = \{\tilde{u}_1, \ldots, \tilde{u}_m\}$ (see Fig. 5). We discuss the approximate bound of convergence for the approximate PI with the one-at-a-time rollout as the policy improvement step in Appendix C.

C. Approximate PI With Truncated Order-Optimized Rollout

Similar to the previous variant, one can use the order-optimized rollout scheme to generate the belief state–rollout control pairs and train the approximate policy network. Apart from that, this variant of approximate PI will exactly follow the previous approximate PI method. Note that one-agent-at-a-time rollout (as opposed to standard rollout) is used in all of our approximate PI experiments.

VI. Online Play Policy

Given an offline trained policy, obtained through the use of multiple approximate PIs, we can improve substantially its performance with an online play algorithm that is based on one-step or multistep lookahead minimization. In the context

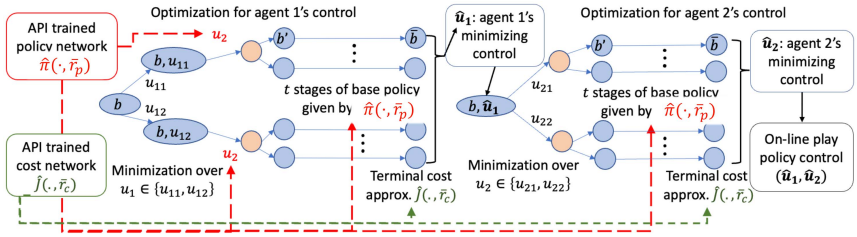


Fig. 6. Online play with the one-agent-at-a-time rollout using a base policy $\hat{\pi}(,.\bar{r}_p)$ and a terminal cost approximation $\hat{J}(,.\bar{r}_c)$ for two agents. The control component u_2 used in the first optimization (left-hand side) is obtained after calling the network $\hat{\pi}(B_2,\bar{r}_p)$ using feature B_2, constructed from belief state b (see Section V-B for feature construction for policy network).

of our problem, this can be done by implementing online a truncated rollout algorithm with the offline trained policy used as the base policy. The recent book [6] has focused on the substantial beneficial effects of online play built on top of offline training, and has interpreted the online play algorithm as a fast/superlinearly convergent Newton step for solving the Bellman equation of the problem. The starting point for the Newton step is provided by the cost function of the offline trained policy, or more accurately, its neural network representation.

An additional benefit of policy improvement by online play is that it works well with changing problem parameters and online replanning, similar to classical methods of indirect adaptive control. When the problem parameters change, the policy obtained offline by an approximate PI algorithm is degraded, since it was trained with the old parameters. However, in the online play algorithm, the Bellman equation is perturbed due to the parameter changes, but approximation in value space still operates as a powerful Newton step. An essential requirement here is that a system model is estimated online through some parameter identification method, and is used during the one-step or multistep lookahead minimization process. It is not unreasonable to assume that such a method is available within our context.

In our implementation, we have incorporated an online play policy where controls are computed by employing an online one-at-a-time truncated rollout using a policy $\hat{\pi}(.,\bar{r}_p)$ as base policy and a terminal cost approximation network $\hat{J}(.,\bar{r}_c)$. The parametric policy approximation with parameter $\bar{r}_p$, and parametric cost approximation with parameter $\bar{r}_c$, are given by training an offline one-at-a-time approximate PI algorithm.

Starting from a belief state b, we perform m online sequential optimizations, one agent at a time, $\ell = \{1,\dots,m\}$, in that order. First, we construct the base policy's control $\pi(b) = (\pi_1(b),\dots,\pi_m(b))$ by calling the policy network $\hat{\pi}(.,\bar{r}_p)$ given by one-at-a-time approximate PI method m times from belief state b (as described in Section V-B). The online play policy's control is denoted by $\hat{u} = (\hat{u}_1,\dots,\hat{u}_m)$ and the control component of agent ℓ is computed as

$$\hat{u}_\ell \in \arg\min_{u_\ell \in U_\ell} \left[\hat{g}(b,u') + \alpha \sum_{z\in Z} \hat{p}(z \mid b,u')\tilde{J}\left(F(b,u',z)\right)\right]$$

where $u' = (\hat{u}_1,\dots,\hat{u}_{\ell-1},u_\ell,\pi_{\ell+1}(b),\dots,\pi_m(b))$. The cost approximation $\tilde{J}(b')$ at belief state b' is the discounted cost of t applications of the control given by the offline trained policy, starting from belief state b', followed by the terminal cost approximation $\hat{J}(\bar{b},\bar{r}_c)$, where $\bar{b}$ is the belief state after t applications of the policy $\hat{\pi}(.,\bar{r}_p)$ starting from b'. Note that the control at each of the t stages is given by calling the policy network $\hat{\pi}(.,\bar{r}_p)$, m times. Fig. 6 shows an online play algorithm with two agents. We present an extensive simulation study of the online play policy on a large and complex POMDP problem. In addition, we show that the online play policy largely overcomes the difficulties of offline training in cases with dynamically changing system parameters.

TABLE I
METHODS USED IN THE COMPARISON STUDY FOR SOLVING THE PARTIALLY OBSERVABLE MULTIROBOT REPAIR PROBLEM

Acronym	Name
All-at-once	Standard rollout
One-at-a-time	One-agent-at-a-time rollout
Order optimized	Order-optimized rollout
One-at-a-time API	One-agent-at-a-time approximate PI
On-line play policy	Online play policy with offline trained approximations
POMCP [1]	POMCP
DESPOT [18]	DESPOT
MADDPG [2]	MADDPG

VII. Simulation Studies and Comparative Results on a Multirobot Repair Problem

In this section, we provide computational results and a comparative study with existing POMDP methods applied to a partially observable multirobot repair problem. Our computational results demonstrate that:

1) our one-agent-at-a-time rollout and order-optimized rollout result in substantial computational savings with comparable performance versus the standard rollout;
2) our one-agent-at-a-time approximate PI method improves the policy over several iterations;
3) our methodologies applied to a complex multirobot repair problem significantly outperform existing methods, and work well where other methods fail to scale up (e.g., with ten agents);
4) an online play policy that utilizes the offline trained policy and cost approximations during online operation further improves performance beyond our rollout methods and our approximate PI.

Table I summarizes various methods used in the simulation study in this section to solve the partially observable multirobot repair problem.

A. Multirobot Repair Problem

Here, we are interested in solving a challenging multirobot repair problem on a partially observable network with several damaged vertices where agents need to physically visit and carry out the repair tasks. Our objective is to learn a coordinated policy that determines the vertices that need to be repaired before other vertices. Damaged locations in this problem can be a proxy for many applications, from pipeline damage locations, to forest fire threats, to damaged equipment sites in a grid. The network is represented as an undirected graph with vertex set V denoting locations, and each location $v \in V$ has one of ν damage levels $(0, 1, \ldots, \nu - 1)$ that evolve over time according to a known Markov decision process (MDP) with ν states, as shown in Fig. 8. The transition probability γ_λ denotes the probability of damage level λ evolving to damage level $\lambda + 1$, $\lambda = \{0, \ldots, \nu - 2\}$. A nonzero transition probability from state 0 to state 1 ($\gamma_0 > 0$) represents that a repaired location (with damage level 0) can become damaged over time. This problem is a generalized extension of the autonomous repair problem discussed in paper [9]. The generalized graph topology and possible decay of a repaired location make the problem in this article a significantly more difficult infinite horizon problem than that described in [9]. We assume perfect observations of the agents' current locations and of the damage levels at the agents' current locations. The damage distribution for each location v can be represented as $d^v = (d^v_0, \ldots, d^v_{\nu-1})$, consisting of the conditional probabilities of the damage level given the prior initial belief and a control-observation history for all agents. The shared belief state b consists of the locations of all m agents $\beta_1, \ldots, \beta_m$ and damage distributions of all locations in the graph $d^v, \forall v \in V$. At each time step, once an agent at location v has made an observation, it can either choose to stay in v and repair the location (if damaged) or move to one of its neighboring locations. The agents incur a cost per unit time whenever there is nonzero damage in the network. The cost at stage k is given by $\sum_{v\in V} d^{v,k} \cdot c$, where $d^{v,k}$ is the damage distribution at vertex v at stage k and c is a cost vector ($c \in \mathbb{R}^\nu_{0+}$) that maps each damage level to a cost (we use $c = [0, 0.1, 1, 10, 100]$ in our experiments). The policy needs to minimize the discounted sum of costs over an infinite horizon ($\sum_{k=0}^{\infty} \alpha^k \sum_{v\in V} d^{v,k} \cdot c$). This is a POMDP with $|V|^m \nu^{|V|}$ states since each of the m agents can be located at any of the $|V|$ locations, and each of the $|V|$ locations can take any one of the ν damage levels. This POMDP has a variable control space depending on the location. The size of the control space is upper bounded by $(\max_{v\in V} \delta_v)^m$, where δ_v is the degree of vertex v. As a terminal cost approximation in our multiagent rollout and approximate PI, we use a steady-state value at the time of truncation at stage k', which is the discounted cost sum over an infinite horizon, assuming that no further control is applied beyond stage k', $\hat{J} = (1/(1-\alpha)) \sum_{v\in V} d^{v,k'} \cdot c$. This type of terminal cost works well when the lookahead tree has a high branching factor or when simulating the trajectories is fairly expensive.

1) Simulation Setup: We implement the multiagent rollout methods on a graph topology (shown in Fig. 1) with 32 vertices and four, eight, and ten agents (state space size $10^{28}, 10^{34}, 10^{37}$, and control space size $625, 10^{5.6}, 10^7$, respectively). We use the transition probability values $\gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.05$, $\gamma_0 = 0.01$ and a discount factor $\alpha = 0.95$ for all experiments with eight and ten agents. After performing experiments with different transition probabilities, we observe that if the dynamics of the Markov chain are too fast, the optimal policy behaves like the greedy policy (ignoring a mildly damaged location can result in a quickly degraded damage level), and if the dynamics of the Markov chain are too slow, the optimal policy prefers to *fix the most damaged location first*. However, the optimal policy produced with these transition probabilities shows nontrivial behavior, e.g., it might choose to ignore a few nearby locations to fix a highly damaged moderately distant location before wandering off too far. A variant of the problem where a repaired location remains repaired ($\gamma_0 = 0$) is significantly easier for the agents to solve, and this is used for comparative studies with other existing methods. We use $\gamma_0 = 0, \gamma_1 = 0.01, \gamma_2 = 0.02, \gamma_3 = 0.03$, and $\alpha = 0.99$ for all experiments with four agents. The base policy for each agent is chosen to be a relatively simple "greedy policy" that does not require any problem-specific tailoring, whereby it chooses to repair the current location (if damaged) and otherwise takes one step toward the nearest damaged location. We use Dijkstra's shortest path algorithm to determine the nearest damaged location and the next hop from each location. We train the approximate PI architectures using the Harvard FASRC cluster with Intel Xeon Cascade Lake CPUs (196 cores). All costs reported are aggregated over 1000 random initial states.

TABLE II
COST COMPARISON OF THE BASE POLICY, AND ONE-AT-A-TIME ROLLOUT POLICY ON THE MULTIROBOT REPAIR PROBLEM

Number of vertices	Agent	Base policy	One-at-a-time rollout
32	8	5347	992
32	10	4667	799
500	50	31090	24414 (scalable implementation)

2) Performance of Multiagent Rollout: Table II shows the cost comparison between the base policy and the one-agent-at-a-time rollout policy (with $t = 10$, and steady-state terminal cost approximation). The results show that the one-agent-at-a-time rollout performed significantly better than its base policy, which is consistent with the rollout policy improvement property. The right-hand side of Fig. 7 shows a sample trajectory of our rollout policy, where agents coordinate by splitting their efforts to tackle the repair problem most efficiently. This is in contrast to the base policy, where agents duplicate repair efforts by moving through the graph in concert (Fig. 7 left-hand side). An alternative scenario involves initiating two agents in two different graph sections, with one more severely damaged than the other. In this case, a base policy keeps the agents in their corresponding initial sections leading to longer repair times. In contrast, when using our one-at-a-time rollout, the agent starting in a mildly damaged section moves to more damaged section to assist other agents.

3) Performance of Our Approach on Larger Problems (Scalable Implementation): Notably, we apply the one-at-a-time rollout to a much larger partially observable repair problem with 500 vertices and 50 agents. We exploit the decomposition

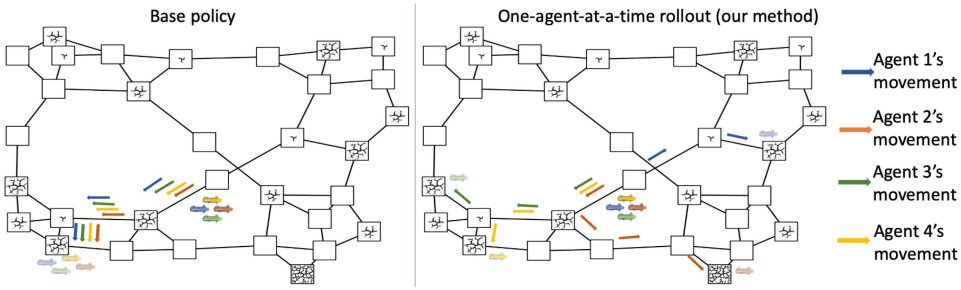


Fig. 7. Trajectories generated by a base policy (left) versus our one-agent-at-a-time rollout policy (right) on a network with several damaged locations. Note the coordinated splitting behavior of the one-agent-at-a-time rollout policy in contrast to the base policy.

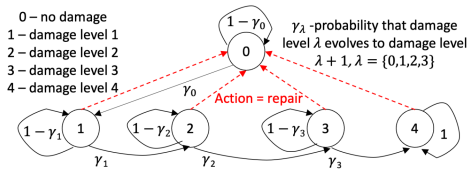


Fig. 8. Markov chain for the damage level of each network location. We use $\gamma_0 > 0$ for eight and ten agents, which is a generalization from [9] for a graph topology where repaired locations can fall into disrepair. We use $\gamma_0 = 0$ for all experiments with four agents.

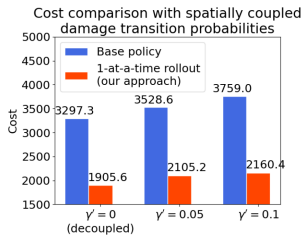


Fig. 9. Cost comparison of the base policy, and the one-at-a-time rollout policy on the spatially coupled multirobot repair problem. The parameter γ' represents how aggressively the damage levels are coupled.

of the underlying graph structure to help scale our approach. We consider a regular graph, with ten vertices and degree two, where each vertex corresponds to a subgraph. We consider each subgraph is a regular graph of degree three. We do not directly apply one-at-a-time rollout to this case. Since each subgraph imposes a partition of the graph, we exploit it to design a highly parallel form of the one-at-a-time rollout. Here, each agent computes the one-at-a-time rollout control by using the computed controls of its predecessors that belong to the same partition as itself. The rollout controls of agents belonging to each partition are computed in parallel, independent of agents from other partitions. A coordinated policy is produced by our one-at-a-time rollout as opposed to a myopic greedy base policy in this case (see Table II).

4) Performance of Coupling Transition Probabilities: So far, we consider the transition probabilities to be spatially decoupled, where the damage level of a location in the network is only affected by the location's previous damage level (given by Markov chain in Fig. 8). Now, we consider that the damage level of one location can affect the damage level of its neighboring locations. In particular, we consider a Markov chain, where $\gamma_0^v = \gamma_0 + \gamma_{\mathcal{N}}^v, \forall v \in V$. Here, γ_0 is the original probability of degrading the damage level from 0 to 1, and $\gamma_{\mathcal{N}}^v = \sum_{u \in \mathcal{N}(v)} \frac{\gamma' d_{\nu-1}^u}{|\mathcal{N}(u)|}$. This simply means that the probability of damage of a location v increases according to the highest level of damage of a neighboring location $u \in \mathcal{N}(v)$. γ' is a parameter that gives how aggressively the damage levels are coupled. The denominator of the expression is chosen to normalize the probability according to the location's (vertex's) degree. We perform experiments with $\gamma' = \{0, 0.05, 0.1\}$ with four agents (see Fig. 9). A value of $\gamma' = 0$ is the case where the damage level of a location is not affected by its neighbors (our original experiments). We observe that one-agent-at-a-time works relatively better than the base policy as the spatial coupling of the damage level increases (higher value of γ'). This means that our approach, with the lookahead optimization, makes better decisions in choosing the right locations to repair sooner for a complicated scenario where the damage level is not localized, and certain damage levels may have devastating effect on the decision-making process.

5) Performance of Multiagent Approximate PI: The neural network used for policy space approximation in the approximate PI method has two hidden layers (256 and 64 ReLU units, respectively) followed by a batch-norm layer. The output layer is a softmax layer which provides the probability distribution over the control components for an agent. The size of the output layer is $|V| + 1$ (one control component is to repair the current location, and others represent the likelihood of traveling to each vertex, one likelihood value for each $v \in V$). We use RMSProp optimizer (learning rate $= 0.001$). We use a one-agent-at-a-time rollout (with $t = 10$) for policy improvement at each iteration. We use 500,000 training samples to train the policy network in each iteration. The training samples were generated by choosing a random set of belief states, followed by sampling from a memory buffer. Note that exploration issues are one of the main challenges in this context, and various solutions have been proposed to resolve this issue; see [32], [33]. To this effect, our memory buffers consist of states generated by taking a few steps from the initial state pool using one of the previous policies and a randomized policy; see [5], Ch. 5.

Fig. 10 shows the performance of neural network policies generated by approximate PI with eight and ten agents. The

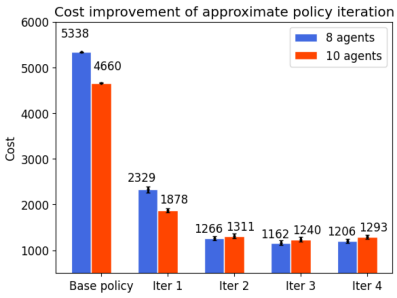


Fig. 10. Cost of base policy and several iterations of approximate PI with one-at-a-time rollout on the multirobot repair problem.

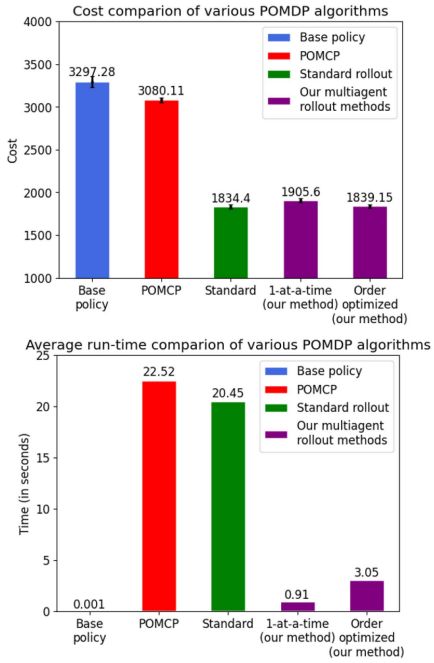


Fig. 11. Comparison (cost at top and run-time at bottom) of POMCP, base policy, rollout on the multirobot repair problem with four agents.

results show that even with a large state and control space, approximate PI with our one-agent-at-a-time rollout retains its policy improvement property over several iterations. We note that cost of policy after iteration four increases. This is typical according to the theoretical bounds on approximate PI with standard rollout and one-at-a-time rollout (Proposition 4.5.3 [5], Appendix C, respectively) indicate that policies are improved during the first few iterations before it enters an error bound and then starts oscillating between the error bound.

6) Performance Comparison to Existing Methods: a) Base policy, standard rollout: This section presents cost comparisons of several existing methods with our one-agent-at-a-time rollout, and our order-optimized rollout. Note that here we cap the number of agents at 4 due to scalability issues (explosion in the number of Q-factor evaluations) for several methods, including standard rollout and POMCP [1]. In fact, due to scalability issues, DESPOT [18] was unsuccessful for the four-agent problem within a reasonable time limit (1 s per stage). For the same time limit, POMCP was able to deal with the four agent case, but not a larger number of agents due to a combination of scalability problems involving computation time and memory requirements. Fig. 11 (top) shows that standard rollout outperforms (cost) all other approaches, and our one-agent-at-a-time rollout methods performed comparably well as the standard rollout with dramatically less computation (Fig. 11 bottom shows the average run-time of end-to-end simulations). The per-stage computational complexity of our one-agent-at-a-time rollout is only $O(4\,C)$, as opposed to $O(C^4)$ of the standard rollout, where $C = \max_{v\in V} \delta_v$ is given by the maximum degree of the network topology. This performance behavior was observed on a broad range of tests involving up to four agents. At the same time, the standard rollout method could not solve the problem with eight and ten agents due to the scalability issue. Furthermore, as expected, our order-optimized rollout [with per-stage $O(4^2\,C)$ computations] outperforms the one-agent-at-a-time rollout. Notably, all of our rollout algorithms outperform the base policy significantly.

b) Comparison with SOTA (POMCP, MADDPG, PA-POMCPOW): We compare our methods with three existing learning methods, POMCP [1], MADDPG [2], and PA-POMCPOW [7]. POMCP uses MCTS-based lookahead. For the implementation, we use default parameters for POMCP (given in [18]), and we modify the code to use a closed form of the belief update governed by the Markov chain in Fig. 8 (with $\gamma_0 = 0$). A single particle with weight $= 1$ is used to represent the belief. Fig. 11 shows the cost comparison of our methods with POMCP, which outperforms the base policy but performs worse than our multiagent rollout methods. One of the reasons is that using a long and sparse lookahead tree results in poor Q-factor estimation in problems with a long planning horizon. In contrast, our rollout methods use shorter lookahead and more precise Q-factor estimation by simulations. We use both the public source code provided by the MADDPG authors and the Berkeley Ray RLLib implementation of MADDPG and conducted an extensive hyperparameter search to tune its parameters. However, MADDPG was consistently outperformed by the base policy for this multiagent repair problem and produced a cost of 5520 (compared with 3277 for a base policy; see Fig. 11). Our approach outperforms a method suitable for problems with large action-space PA-POMCPOW [7] that applies an MCTS with a widening window for search that considers a subset of actions to explore at each stage based on the expected value and the information gain of the action. We use the authors' implementation of PA-POMCPOW and obtain a cost of 3251 with a thorough hyperparameter search. We choose the action-value function for information gain associated with an action at a belief state as the dot product of the damaged levels and their distance from their nearest agent obtained after applying the action. The result is consistent with the POMCP policy since PA-POMCPOW applies an approximate version of POMCP.

7) Online Play Policy: We use the policy network training using our one-at-a-time approximate PI after iteration four in our online play policy (see the details of the policy network in Paragraph 5). The cost network used in our online play policy

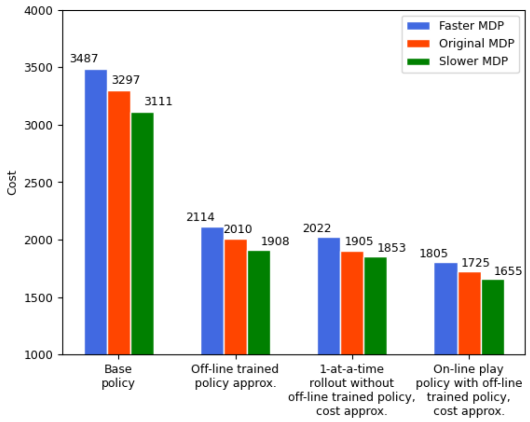


Fig. 12. Performance and robustness study of the online play policy on the multirobot repair problem with four agents, where the damage levels evolve differently with different parameters of the Markov chain (shown in Fig. 8). The original MDP uses $\gamma_0 = 0, \gamma_1 = 0.01, \gamma_2 = 0.02, \gamma_3 = 0.03$. The slow MDP uses the parameters $\gamma_0 = 0, \gamma_1 = 0.005, \gamma_2 = 0.01, \gamma_3 = 0.02$, and the fast MDP uses $\gamma_0 = 0, \gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.04$. Offline policy and cost approximations used in the online play policy are trained using approximate PI with the original MDP parameters.

has three hidden layers (256, 256, and 128 ReLU units, respectively) followed by a batch-norm layer and uses the RMSProp optimizer (learning rate $= 0.001$). The cost network is trained using 500,000 belief state-cost pairs, where the belief states are the same as those used for the policy network training. The cost samples are obtained by averaging the costs of several simulated trajectories using the policy network starting from the corresponding belief states. During online play, we compute the control by employing a one-at-a-time rollout where we estimate the Q-factors by 20 simulated trajectories, each by applying the control given by the policy network $t = 10$ times, followed by the cost network.

Fig. 12 show the performance of the base policy, the one-agent-at-a-time rollout (without offline trained approximations), the offline trained policy network using approximate PI, and the online play policy that uses the offline trained policy and cost networks in the multirobot repair problem with four agents. The red bars of Fig. 12 use the original Markov chain shown in Fig. 8 with parameters $\gamma_0 = 0, \gamma_1 = 0.01, \gamma_2 = 0.02, \gamma_3 = 0.03$. We note that the offline trained policy network, by itself, is not as good as the online rollout with an initial base policy. This behavior can be attributed to several approximation errors in the approximate PI setting. However, using these offline trained networks as the base policy (policy network) and the terminal cost approximation (cost network), the online play policy for the original Markov chain produces a cost of 1725. The online play policy thus outperforms all of our rollout methods and other existing POMDP methods (including 1905.6 for our one-agent-at-a-time rollout and 3080.1 for POMCP; see Fig. 11) in the multirobot repair problem with four agents.

We next demonstrate the robustness of the online play policy that provides adaptive controls when system parameters differ during the policy evaluation time from the parameters used during the offline architecture training. In particular, once we train the value and policy approximations using the original Markov chain, we apply the trained architectures to two different multirobot repair problems (four agents) with different transition probabilities (denoted by fast Markov chain and slow Markov chain). The slow Markov chain follows the Markov chain shown in Fig. 8 with parameters $\gamma_0 = 0, \gamma_1 = 0.005, \gamma_2 = 0.01, \gamma_3 = 0.02$, which means that any location with a damage level has a smaller chance of evolving to the next damage level than the original Markov chain. The fast Markov chain follows the Markov chain shown in Fig. 8 with parameters $\gamma_0 = 0, \gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.04$, which means that any location with a damage level has a higher chance of evolving to the next damage level than the original Markov chain. Fig. 12 showcases the robustness of the online play policy, which is evaluated using new parameters and compares the relative performance with the original Markov chain. The policy approximation performs poorly during online evaluation using the new parameters since the controls come from the policy trained with the original parameters. The online play policy outperforms the base policy and our one-at-a-time rollout in the multirobot repair problem (both of which do not use offline trained architectures and are evaluated online with new parameters). The results show that the online play policy adapts to the new parameter changes with online replanning while taking full advantage of offline architectures.

VIII. Considerations for Imperfect Communication

The methods discussed so far address the computational challenge of the multiagent sequential decision-making problems. In this section, we focus our attention on another major challenge: imperfect communication between agents. We discuss methods where agents do not need explicit communication of the computed control components from the other agents. Instead, agents use estimates of other agents' controls by a "signaling" policy that can be precomputed, along with their state estimation. We define a signaling policy as a policy that predicts the control components for other agents using heuristics or approximation architectures without the exact knowledge of other agents' computed control components. By doing this, the agents can obtain a significant computational speedup through parallelization. The methods discussed in this section are called approximate multiagent rollout (AMR) since agents select their control components based on an imperfect estimate of other agents' control computations. We consider several modes of communication. In the first case, we assume that agents share the belief state but never share the computed controls. This case is suitable when agents can access belief states that change more slowly (once per time step) than controls that need to be shared at a tighter timescale (several times per time step). In the second case, we assume that agents share the belief and intermittently share the computed controls.

A. AMR when control is never shared

In this strategy, we do not consider any communication of controls between the agents. Here, m independent minimizations are performed, once over each of the agent's control components, with a signaling policy to estimate the control components of

the other agents coming before the corresponding agent in the agent order. Such an AMR with a signaling policy gives a control $\bar{u} = (\bar{u}_1, \ldots, \bar{u}_m)$ at belief state b. Agent ℓ's control component is

$$\bar{u}_\ell \in \arg\min_{u_\ell \in U_\ell} \left[\hat{g}(b, u') + \alpha \sum_{z \in Z} \hat{p}(z \mid b, u') \tilde{J}\left(F(b, u', z)\right)\right] \quad (3)$$

where $u' = (\bar{\bar{u}}_1, \ldots, \bar{\bar{u}}_{\ell-1}, u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b))$. Here, control components $\bar{\bar{u}}_1, \ldots, \bar{\bar{u}}_{\ell-1}$ are given by the signaling policy, which agent ℓ uses as a proxy to the computed control components of agents $\{1, \ldots, \ell-1\}$. The control $\pi(b) = (\pi_1(b), \ldots, \pi_m(b))$ denotes the base policy's control at belief state b.

B. AMR when control is intermittently shared

In this strategy, we consider a centralized cloud server having access to the global control information with an intermittent connection probability of $\rho \in (0, 1)$. We denote this hybrid policy as π_{hybrid}, and it works in the following manner. Each agent can access the computed control components of the predecessor agents and compute the one-agent-at-a-time rollout's control given by $\tilde{\pi}$ [using (2)] with a probability ρ when the cloud is accessible. The hybrid policy uses another policy π_{nocloud} with a probability $1 - \rho$

$$\pi_{\text{hybrid}}(b) = \begin{cases} \tilde{\pi}(b) & \text{with probability } \rho \\ \pi_{\text{nocloud}}(b) & \text{with probability } 1 - \rho. \end{cases} \quad (4)$$

Control $\pi_{\text{nocloud}}(b)$ can be given by a policy that does not require the exact computed control components of the other agents. Examples of such a control $\pi_{\text{nocloud}}(b)$ include the base policy's control $\pi(b)$, and the AMR with a signaling policy (3). This hybrid communication method is similar to the one analyzed in [34].

C. Challenges in Imperfect Communication Cases

Imperfect communication among the agents poses some unique challenges, including the *loss of the policy improvement property* and the issue of *failure to terminate*. In the multirobot repair problem, we consider the state of termination when there are no more locations left to repair.

A nontermination case is more perturbing than the loss of policy improvement property since the cost of a nonterminating policy can be very large, even for a properly discounted problem. We note that reaching termination is significantly easier for the case where a repaired location cannot fall into disrepair (indicated by the Markov chain in Fig. 8, with $\gamma_0 = 0$). Alternatively, termination is hard to achieve in the case where a repaired location may evolve to a higher damage level over time (indicated by the Markov chain in Fig. 8, with $\gamma_0 > 0$). For the analytical arguments for the finite termination case, we will restrict ourselves to the case where a repaired location does not fall into disrepair.

In Section VIII-D, we show that termination is not guaranteed when agents never share their controls, and each agent uses the base policy as signaling to estimate other agents' controls. In Section VIII-E, we show that termination is possible for the multirobot repair problem in the case of no control communication by exploiting randomization of the control policy, in addition to the rollout with base policy signaling. Finally, in Section VIII-F, we consider a centralized, intermittently available cloud server that retains the computed control components of all agents. We show that the policy improvement property is recoverable using a hybrid policy that uses the one-at-a-time rollout with one-step lookahead optimization when the controls are communicated and uses the base policy (without any signaling policy) when the controls are not communicated. We provide an extensive simulation study on imperfect communication applied to the multiagent repair problem in Section VIII-G, where belief states are shared, but the computed controls are not perfectly shared, and in Section VIII-H, where both belief states and the computed controls are imperfectly shared.

D. Failure to Terminate With No Communication of Controls

The AMR method faces a major challenge, where agents may not attain termination for certain initial belief states in the case of no communication of computed controls. In particular, we discuss the AMR where agents estimate other agents' controls using the base policy. The control $\bar{u} = (\bar{u}_1, \ldots, \bar{u}_m)$ given by this policy at belief state b, is obtained using (3), where $u' = (\pi_1(b), \ldots, \pi_{\ell-1}(b), u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b))$. Here, $\pi(b) = (\pi_1(b), \ldots, \pi_m(b))$ is the base policy's control at belief state b. The future cost is $\tilde{J}(F(b, u', z))$ is the cost of applying base policy π until termination, starting at belief state $F(b, u', z)$.

We will use a counterexample to show that an agent executing the AMR without any communication of controls and estimates other agents' controls using the base policy as signaling may not attain termination in a finite time. In this counter example, we consider a group of agents wanting to visit several partially damaged locations in a discrete space. In addition, we consider the damage levels of the locations are independent of other locations, and once repaired, do not fall into disrepair (see the Markov chain in Fig. 8 with $\gamma_0 = 0$). An agent can visit one of its adjacent locations in a single time step. The terminal state is reached when all damaged locations are visited.

This problem is a simplified instance of several real-world multiagent sequential decision problems with spatial controls, including multirobot repair, search and rescue, and taxicab pickup problems. In the AMR, the agents do not share their computed control components but estimate the other agents' controls using a signaling policy. A bad signaling policy may fail to estimate other agents' controls correctly. Examples of such bad signaling policies in this problem instance include a greedy base policy that directs an agent to its nearest partially observable damaged location. In this case, each agent that performs the control optimization with base policy signaling may choose a control that drives it away from its nearest partially observable damaged location, when it observes the presence of another agent equidistant from its nearest partially observable damaged location. Each agent (equidistant from their nearest partially observable damaged locations) will assume that the other agent(s) will visit that location, and as a result, none of

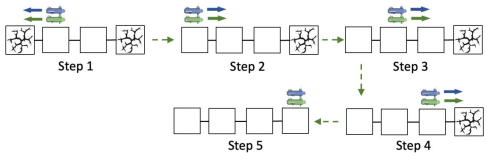


Fig. 13. Trajectory of the base policy that moves an agent toward its nearest partially visible damaged location with no communication with other agents (takes four steps to visit all damaged locations).

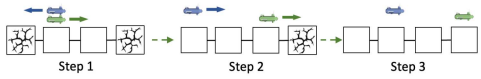


Fig. 14. Trajectory of the one-at-a-time rollout, where the second agent makes a control decision with the knowledge of the first agent's computed control (takes two steps to visit all damaged locations).

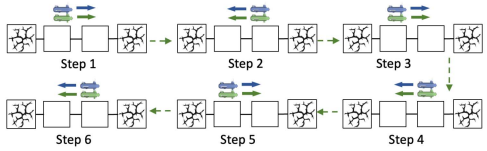


Fig. 15. Partial view of a nonterminating trajectory with the AMR using base policy signaling (no communication of controls). At each step, an agent thinks that the other agent will visit the nearest partially observable damaged location, and moves away from it and the two agents oscillate infinitely.

them will visit the nearest partially observable damaged location. This behavior can continue infinitely, thereby not terminating the problem in a finite time.

Figs. 13 and 14 show the trajectories followed by the base policy, and the one-agent-at-a-time rollout using two agents that want to visit two partially observable damaged locations at the two ends of a linear graph with four vertices. Fig. 15 shows a partial view of an infinitely long trajectory given by the AMR that does not share computed control components among agents; instead, each agent estimates other agents' control using the base policy. In the next section, we show that termination is possible with a simple modification to the approximate rollout with base policy signaling in the case of no control communication.

E. Finite Termination Without Communication of Controls by Using Randomization of the Policy

In this section, we address the problem of nontermination with no communication of controls by using a randomized control policy. We consider a randomized multiagent rollout policy that selects a control $\bar{u}^r$ as follows:

$$\bar{u}^r = \begin{cases} u^r & \text{with probability } \epsilon \\ \bar{u} & \text{with probability } 1-\epsilon \end{cases} \tag{5}$$

where $0 < \epsilon < 1$, u^r is a random control chosen from U, and $\bar{u}$ is the control given by the AMR with base policy signaling using (3) by putting $u' = (\pi_1(b), \ldots, \pi_{\ell-1}(b), u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b))$. Here, $\pi(b) = (\pi_1(b), \ldots, \pi_m(b))$ is the base policy's control at belief state

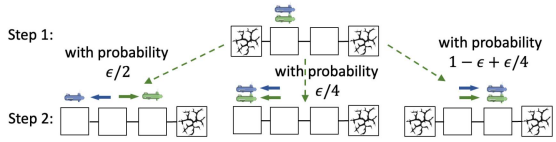


Fig. 16. Possible outcomes of the one-time application of the randomized policy on a problem with two agents and two partially observable damaged locations. The randomization ensures the agents do not get stuck into the infinite limit cycle of nontermination since every single application of the randomized policy has a positive probability of making progress toward termination in this problem.

b, and the future cost $\tilde{J}(F(b,u',z))$ is given by the cost of applying the base policy π until termination, starting at belief state $F(b,u',z)$. The intuition behind doing this is that most of the time, the multiagent rollout with base policy signaling makes good decisions to direct agents toward strategic locations, except for rare cases where agents enter into a limit cycle with no termination, as described in Section VIII-D. For these cases, randomization helps break the oscillation and essentially reinitializes, in a random fashion, the starting points of the agents for the next decision. Fig. 16 shows possible outcomes of a single application of this randomized policy that has a positive probability of making progress toward termination with two agents to visit two partially observable damaged locations in a linear graph, where a repaired location does not fall into disrepair.

Now, we discuss how randomization resolves the issue of nontermination in our AMR scheme that does not use any control communication for a problem where the agents need to visit a set of damaged vertices in a finite graph. In particular, we consider m agents that aim to visit η vertices on a given finite connected graph $G = (V, E)$, where $\eta \leq |V|$. The vertex set V denotes the physical locations, out of which η vertices are believed to be damaged, and E is the edge set defined over the vertices. Once visited, a damaged vertex is automatically repaired and does not fall into disrepair. The problem terminates when all η damaged vertices are visited. We will argue that an agent executing the randomized multiagent rollout with base policy signaling (5), and without communication of controls leads to termination in a finite time in this problem.

To prove finite termination with randomization, we first reformulate the original problem that uses the randomized policy as a modified Markov chain. We state the conditions where the Markov chain for the reformulated problem terminates in finite time. We conclude the proof by showing that these conditions are achievable in a finite connected graph.

Each state in the reformulated Markov chain corresponds to a possible number of vertices (believed to be damaged) yet to be visited. We denote the vertices with nonzero damage belief as the target vertices in the graph. The states are denoted by $X_0, X_1, \ldots, X_\eta$, where X_χ denotes χ target vertices that remain to be visited, $\chi = \{0, 1, \ldots, \eta\}$. X_0 is the terminal state where no more target vertices remain to be visited. We consider that each state transition in the modified Markov chain is equivalent to D applications of the randomized policy in the original

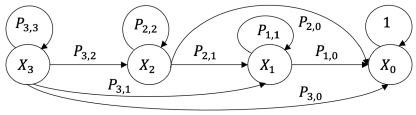


Fig. 17. Reformulated Markov chain for a problem where agents need to visit three target vertices. States X_3, X_2, X_1, X_0 denote 3, 2, 1, and 0 target vertex(s) remain to be visited, respectively. Each state transition in this Markov chain is equivalent to D applications of the randomized policy in the original problem.

problem. We denote this D as the step-size. The state transition probability of the modified problem is given by

$$\begin{bmatrix} P_{\eta,\eta} & P_{\eta,\eta-1} & \dots & P_{\eta,0} \\ P_{\eta-1,\eta} & P_{\eta-1,\eta-1} & \dots & P_{\eta-1,0} \\ \vdots & \vdots & \dots & \vdots \\ P_{0,\eta} & P_{0,\eta-1} & \dots & P_{0,0} \end{bmatrix}$$

where $P_{\chi,\chi'}$ is the probability that after starting with χ unvisited target vertices, χ' target vertices remain to be visited by applying the randomized policy (5) D times in the original problem, $\chi, \chi' = \{0, 1, \dots, \eta\}$. We assume that a visited (repaired) vertex cannot fall into disrepair, or equivalently $P_{\chi,\chi'} = 0, 0 \leq \chi < \chi' \leq \eta$. Thus, the state transition matrix becomes an upper triangular matrix. Fig. 17 shows the reformulated Markov chain with $\eta = 3$ target vertices.

Finite termination in the modified Markov chain is possible if the steady-state probability of all the nonterminal states is 0. We apply the steady-state convergence theorem [35] to show that if all nonterminal states are transient, then a unique steady-state distribution exists where the steady-state probability of all the nonterminal states is 0, and the steady-state probability of the terminal state is 1. All nonterminal states $(X_1, \dots, X_\eta)$ are transient only if each diagonal entry corresponding to the nonterminal states in the state transition matrix is less than 1. In particular, the finite termination in this Markov chain is possible when $P_{\chi,\chi} < 1$, where $0 < \chi \leq \eta$ for the choice of the step-size D, since we already established that the state transition matrix is upper triangular ($P_{\chi,\chi'} = 0$, where $0 \leq \chi < \chi' \leq \eta$).

So far, we prove that the modified Markov chain terminates in finite time if $P_{\chi,\chi} < 1$, where $0 < \chi \leq \eta$. In other words, the original problem terminates with the randomized policy if $\exists D$, such that there is a positive probability that at least one target vertex will be visited after D applications of the randomized policy in the original problem. Now it is only left to prove the existence of D for our problem that involves a finite graph. We prove such D exists by using the properties of a finite graph (with a finite diameter) and the nature of the randomized policy that applies a randomized control with a probability of ϵ. The finite, connected graph $G = (V, E)$ in the original problem guarantees at least one path between two given vertices $v_1, v_2 \in V$ exists with length at most the diameter of the graph $\mathcal{D} = \max_{v_1,v_2 \in V} \text{shortest\_path\_length}(v_1, v_2)$. We denote such a path by ρ_{v_1,v_2}, and the set of all such paths by $\mathcal{P}_{v_1,v_2}$. The probability that an agent at v_1 visits v_2 in $\mathcal{D}$ applications of the randomized policy in the original problem

$$\begin{aligned} &> \sum_{\rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}} \prod_{v \in \rho_{v_1,v_2}} \frac{\epsilon}{\delta_v} \\ &\geq \prod_{v \in \rho_{v_1,v_2}} \frac{\epsilon}{\delta_v} \quad [\text{where } \rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}] \\ &\geq \prod_{v \in \rho_{v_1,v_2}} \frac{\epsilon}{\max_v \delta_v} \geq \left(\frac{\epsilon}{\max_v \delta_v}\right)^{\mathcal{D}} > 0. \end{aligned} \tag{6}$$

The first step comes from the fact that an agent located at vertex $v \in \rho_{v_1,v_2}$ that applies the randomized policy (5) moves to the next vertex in the path ρ_{v_1,v_2}, adjacent to v, with a probability of at least $\frac{\epsilon}{\delta_v}$ (where δ_v is the degree of vertex v). The second inequality in the second line holds since there is at least one path in $\mathcal{P}_{v_1,v_2}$ due to the graph being connected. Hence, the sum of probabilities of reaching v_2 from v_1 using all paths in $\mathcal{P}_{v_1,v_2}$ is at least as big as the probability of reaching v_2 from v_1 using one path $\rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}$. The fourth inequality holds since the path length of ρ_{v_1,v_2} is $\leq \mathcal{D}$. The fifth inequality holds since $\epsilon > 0$, and both $\mathcal{D}$ and $\max_v \delta_v$ are finite.

Expression (6) ensures that there is a positive probability that an agent moves from a given vertex v_1 to visit a target vertex v_2 within $\mathcal{D}$ applications of the randomized policy in the original problem. In other words, there exists a finite step-size D equal to the diameter of the graph ($\mathcal{D}$), which ensures a probability strictly less than one that no target vertex is visited in one state transition of the modified problem, i.e., $P_{\chi,\chi} < 1, \forall \chi = \{1, \dots, \eta\}$. The fact $P_{\chi,\chi} < 1, \forall \chi = \{1, \dots, \eta\}$ that implies all nonterminating states are transient, in conjunction with the steady-state convergence theorem [35], proves the finite termination with the randomized policy in a finite graph.

Here, we prove that a randomized policy can recover finite termination without any communication of controls. However, we may not claim the policy improvement property for the randomized policy in the case of no control communication. The following section discusses an imperfect communication architecture where controls can be shared intermittently, that recovers the policy improvement property.

F. Policy Improvement Property With Intermittent Communication of Controls

So far in the case of imperfect communication, we consider that controls are not shared with other agents. In this section, we discuss an AMR scheme, where controls are intermittently shared. The resulting policy recovers the policy improvement property with intermittent communication of controls. We further consider that the base policy (π) is used when the controls are not communicated. Particularly, the hybrid policy (4) takes the form

$$\pi_{\text{hybrid}}(b) = \begin{cases} \tilde{\pi}(b) & \text{with probability } \rho \\ \pi(b) & \text{with probability } 1 - \rho. \end{cases} \tag{7}$$

The control $\tilde{\pi}(b) = (\tilde{\pi}_1(b), \dots, \tilde{\pi}_m(b))$ is given by the one-agent-at-a-time rollout policy using (2) at belief state b. In the pure form, we replace the cost function $\tilde{J}(F(b, u', z))$ in the

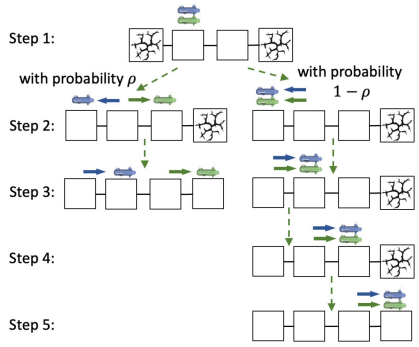


Fig. 18. Possible outcomes of π_{hybrid} policy with intermittent control communication with two agents and two partially observable locations. All damaged locations are visited faster than the base policy.

right-hand side of (2) by $J_\pi(F(b,u',z))$. $J_\pi(F(b,u',z))$ is the cost of applying the base policy π starting at the belief state $F(b,u',z)$ till termination. We will use the one-step lookahead one-at-a-time rollout in the pure form in this section, that is

$$\tilde{\pi}_\ell(b) \in \arg\min_{u_\ell \in U_\ell} \left[\hat{g}(b,u') + \alpha \sum_{z\in Z} \hat{p}(z|b,u') J_\pi(F(b,u',z)) \right] \tag{8}$$

where $u' = (\tilde{\pi}_1(b), \ldots, \tilde{\pi}_{\ell-1}(b), u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b))$. Fig. 18 shows the hybrid policy π_{hybrid} applied to a problem where two agents want to visit two partially observable damaged locations at the two ends of a linear graph of four vertices. We note that agents visit all locations that are believed to be damaged in 2 time steps with a probability of ρ, and 4 time steps with a probability of $1-\rho$. In expectation, the hybrid policy π_{hybrid} needs $2\rho + 4(1-\rho)$ steps to visit both locations; faster than the base policy, that takes four steps if $\rho > 0$.

Now, we show that the AMR with intermittent communication of controls, where the base policy is used without communication of controls, improves cost over the base policy. For this hybrid policy, denoted by π_{hybrid} [defined in (7)], we have $J_{\pi_{\text{hybrid}}}(b) \le J_\pi(b), \quad \forall b$.

Here, $J_{\pi_{\text{hybrid}}}(b)$ and $J_\pi(b)$ denote the costs of applying the hybrid policy π_{hybrid} and the base policy π, starting at belief state b until termination, respectively. The policy improves when the probability of connecting to the cloud server $\rho > 0$. With $\rho = 0$, the policy π_{hybrid} is given the base policy π.

We first define the Bellman operator T for a cost function J, policy π, and belief state b, as , $(T_\pi J)(b) = \hat{g}(b,\pi(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\pi(b)) J(F(b,\pi(b),z))$.

The cost of policy π, at state b, can be expressed recursively, $J_\pi(b) = \hat{g}(b,\pi(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\pi(b)) J_\pi(F(b,\pi(b),z))$.

$J_\pi(b)$ can also be given in terms of the Bellman operator T, $J_\pi(b) = \lim_{K\to\infty}(T_\pi^K J)(b), \forall b. (T_\pi^K J)(b)$ is the discounted sum of stage costs starting at belief state b, where the Bellman operator T for policy π is applied K times, followed by the cost approximation J.

We prove the policy improvement property of policy π_{hybrid} for two agents. It is straightforward to extend the argument for $m > 2$ agents. We first show that $(T_{\pi_{hybrid}} J_\pi)(b) \le J_\pi(b), \forall b$, which is essential in showing $(T^K_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b), \forall b$, for a finite K. Finally, taking the limit $K \to \infty$, we show that $J_{\pi_{\text{hybrid}}}(b) = \lim_{K\to\infty}(T^K_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b), \forall b$.

In order to show $(T_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b)$, we expand $(T_{\pi_{\text{hybrid}}} J_\pi)(b)$, the Bellman operator for the policy π_{hybrid} and the cost approximation J_π. Now for all belief states, b, we have

$$(T_{\pi_{\text{hybrid}}} J_\pi)(b) = \hat{g}(b,\pi_{\text{hybrid}}(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b, \pi_{\text{hybrid}}(b)) J_\pi(F(b,\pi_{\text{hybrid}}(b),z)) \tag{9a}$$

$$= \rho \left[\hat{g}(b,\tilde{\pi}(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\tilde{\pi}(b)) J_\pi(F(b,\tilde{\pi}(b),z)) \right] + (1-\rho)[\hat{g}(b,\pi(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\pi(b)) J_\pi(F(b,\pi(b),z))] \tag{9b}$$

$$= (1-\rho) J_\pi(b) + \rho \min_{u_2\in U_2} [\hat{g}(b,(\tilde{\pi}_1(b),u_2)) + \alpha \sum_{z\in Z} \hat{p}(z|b,(\tilde{\pi}_1(b),u_2)) J_\pi(F(b,(\tilde{\pi}_1(b),u_2),z))] \tag{9c}$$

$$\le (1-\rho) J_\pi(b) + \rho[\hat{g}(b,(\tilde{\pi}_1(b),\pi_2(b))) + \alpha \sum_{z\in Z} \hat{p}(z|b,(\tilde{\pi}_1(b),\pi_2(b))) J_\pi(F(b,(\tilde{\pi}_1(b),\pi_2(b)),z))] \tag{9d}$$

$$= (1-\rho) J_\pi(b) + \rho \min_{u_1\in U_1} [\hat{g}(b,(u_1,\pi_2(b))) + \alpha \sum_{z\in Z} \hat{p}(z|b,(u_1,\pi_2(b))) J_\pi(F(b,(u_1,\pi_2(b)),z))] \tag{9e}$$

$$\le (1-\rho) J_\pi(b) + \rho[\hat{g}(b,(\pi_1(b),\pi_2(b))) + \alpha \sum_{z\in Z} \hat{p}(z|b,(\pi_1(b),\pi_2(b))) J_\pi(F(b,(\pi_1(b),\pi_2(b)),z))] \tag{9f}$$

$$= (1-\rho) J_\pi(b) + \rho J_\pi(b) = J_\pi(b). \tag{9g}$$

Step (9a) is the definition of the Bellman operator for policy π_{hybrid}. Step (9b) expands line (9a) using the definition of the policy π_{hybrid} [see (7)]. Step (9c) (the first part of the summation) and step (9g) come from the recursive definition of the cost of the base policy π. Step (9c) (the second part of the summation) and step (9e) come from the definition of the one-agent-at-a-time rollout policy $\tilde{\pi}$ [see (8)]. Steps (9d) and (9f) come from the minimization operations.

From the aforementioned derivation, we get $(T_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b), \forall b$. This inequality gives us the following relations. $(T^K_{\pi_{\text{hybrid}}} J_\pi)(b) = (T^{K-1}_{\pi_{\text{hybrid}}})(T_{\pi_{\text{hybrid}}} J_\pi)(b) \le (T^{K-1}_{\pi_{\text{hybrid}}} J_\pi)(b) = (T^{K-2}_{\pi_{\text{hybrid}}})(T_{\pi_{\text{hybrid}}} J_\pi)(b) \le (T^{K-2}_{\pi_{\text{hybrid}}} J_\pi)(b) = (T^{K-3}_{\pi_{\text{hybrid}}})(T_{\pi_{\text{hybrid}}} J_\pi)(b) \le (T^{K-3}_{\pi_{\text{hybrid}}} J_\pi)(b) \le \ldots \le (T_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b)$. The cost of the policy π_{hybrid} is, $J_{\pi_{\text{hybrid}}}(b) = \lim_{K\to\infty}(T^K_{\pi_{\text{hybrid}}} J_\pi)(b) \le J_\pi(b), \forall b$.

TABLE III
OUR METHODS USED IN THE IMPERFECT COMMUNICATION CASES

Acronym	Name	Signaling	Comm. strategy
AMR-B	AMR with base policy signaling	Base policy	None
AMR-N	AMR with neural network signaling	Policy net. approximating rollout	None
AMR-PI	AMR with PI signaling	Policy net. of the best policy iter.	None
AMR-LC	AMR with local communication	Base policy for nonlocal agents	Local
AMR-ILC	AMR with local and intermittent communication	Base policy w/o cloud and local connectivity	Local + intermittent
AMR-IAC	AMR with intermittent asynchronous agent communication	Base policy w/o connectivity	Intermittent
AMR-IB0	AMR with intermittent communication and base policy w/o optimization	None	Intermittent
AMR-IB1	AMR with base policy signaling intermittent communication	Base policy w/o cloud connectivity	Intermittent

Next, we talk about the numerical results for our rollout methods with various imperfect communication architectures.

G. Results on the Multirobot Repair Problem With Imperfect Communication of Controls (Shared Belief States)

This section considers the case where agents cannot communicate their choice of control with others, although we assume that the belief state is still shared. This case may arise when agents have access to information that changes more slowly, such as beliefs, but cannot necessarily share information at tighter timescales, such as chosen controls. Although with the shared belief state, agents can reconstruct the controls for all other agents, it is computationally expensive for a decentralized architecture where agents need to repeat work. Instead, using a signaling policy helps with speed up by leveraging parallel control computation. Next, we discuss cases where both belief states and controls are shared imperfectly.

Simulation setup: Similar to the simulation setup in Section VII-A, we use the Markov chain shown in Fig. 8 for damage level degradation with parameters $\gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.05, \gamma_0 = 0.01$, and $\alpha = 0.95$, where a repaired location may evolve to the next damage level for eight and ten agents. We use $\gamma_1 = 0.01, \gamma_2 = 0.02, \gamma_3 = 0.03, \gamma_0 = 0$, and $\alpha = 0.99$ for four agents (except for the comparison study with A3C3 [8]) where we use different values of α). We use $t = 10$ and the steady-state terminal cost approximation for all AMR methods. Table III summarizes different architectures for imperfect communication strategies we study. Each AMR method's name contains an abbreviation of the signaling policy, which the agents use to estimate other agents' computed control components. "I" represents intermittent cloud server, and "LC" represents local communication.

We consider several communication architectures with different signaling policies where belief states are shared but controls are not perfectly shared.

Fig. 19. Scenario for AMR-IAC where a subset of agents communicate with each other. We apply one-agent-at-a-time rollout with the agent order $\{2, 4, 1\}$, and we apply the base policy for agent 3.

1) Approximate Multiagent Rollout With Base Policy Signaling (AMR-B): This is the method where the agents estimate other agents' controls using the base policy. This represents the extreme case of no communicated controls.

2) Approximate Multiagent Rollout With Neural Network Signaling (AMR-N): In this approach, the predecessors' control components are predicted by a neural network that approximates the one-agent-at-a-time rollout policy, and successors' control components estimated by the base policy. This also falls into the extreme case of no communicated controls.

3) Approximate Multiagent Rollout With Best PI Policy Signaling (AMR-PI): This approach is similar to AMR-N. Instead of using the neural network policy that approximates the one-agent-at-a-time rollout, the predecessors' control components are given by the neural network corresponding to one of the approximate PIs (possibly the best iteration), and the base policy is given by the previous PI.

4) Approximate Multiagent Rollout With Local Communication (AMR-LC): This approach considers a local communication scheme where the computed predecessors' control components are communicated among agents when the corresponding agents are less than r hops away on a network, and all other controls are estimated by the base policy.

5) Approximate Multiagent Rollout With Intermittent as well as Local Communication (AMR-ILC): In this approach, we assume intermittent connectivity to a centralized cloud server that provides access to computed predecessors' control components with probability $\rho > 0$. With a probability $1 - \rho$, the method assumes local communication with a radius of r. In other words, when the cloud is available, the one-agent-at-a-time rollout is performed, and otherwise, the method follows AMR-LC, as described earlier. This architecture strikes a practical tradeoff since it takes advantage of a rich centralized information source whenever possible and uses clustered local communication when the server is unreachable.

6) Approximate Multiagent Rollout With Intermittent Asynchronous Agent Communication (AMR-IAC): In AMR-ILC, we consider that controls always shared with other agent whenever the intermittent cloud is available. Now, we discuss an AMR scheme, where each agent has an independent probability p of sharing its computed control with other agents, thereby inducing an agent communication graph. We consider P to be the longest-directed communication path over all agents $\{1, \ldots, m\}$. For example, in Fig. 19, the largest communication path $P = \{2, 4, 1\}$, where the elements of P are indices of communicating agents. The one-at-a-time rollout is applied using agent order given by P since each vertex (agent) in this path can receive information from their corresponding predecessor vertices. Agents not belonging to the path apply the base policy.

TABLE IV
COST COMPARISON OF DIFFERENT MULTIAGENT ROLLOUT POLICIES WITH IMPERFECT COMMUNICATION OF CONTROLS (WITH SHARED BELIEF)

Methods	Four agents	Eight agents	Ten agents
Base policy	3297	5347	4667
One-at-a-time rollout	1906	992	799
AMR-B	2983	2513	2487
AMR-N	2255	1712	1533
AMR-PI	1785	1590	1428
AMR-LC r=2	1914	1010	813
AMR-ILC ρ=0.3, r=2	2081	1005	809
AMR-ILC ρ=0.5, r=2	2001	998	807
AMR-ILC ρ=0.8, r=2	1935	992	804
AMR-IAC p=0.5	2090	1144	1002
AMR-IAC p=0.7	2040	1074	902

The resulting policy recovers the policy improvement property (see Appendix B). Fig. 19 shows an example where only a subset out of four agents communicate. We show simulation results with various communication probabilities p in Table IV.

Table IV presents performance results of different communication architectures with imperfect control sharing (but perfect belief sharing) and compares them with our one-at-a-time rollout with perfect communication for four, eight, and ten agents. We observe that AMR-B gives worse performance than other multiagent rollout methods. This is because it performs local optimizations for each agent without any coordination, i.e., no communicated controls. This method is also susceptible to oscillations (as discussed in Section VIII-D). AMR-N performs better since it uses a neural network to approximate the one-at-a-time rollout policy, which is then used to estimate predecessor agent controls. AMR-PI performs better than AMR-N, which can be explained by its usage of better base and signaling policies. AMR-LC works very well for our problem since the spatial clustering of agents makes sense in this network repair context. However, this approach is heavily dependent on the communication radius r (we use $r = 2$) and can exhibit poor behavior if coordination is required across agent clusters (i.e., the damage is in a distant part of the environment). The assumption of a local communication radius may not be a practical assumption in other multiagent POMDP problems. AMR-ILC performs best among all other AMR approaches for a large number of agents (eight and ten agents) by utilizing all predecessors' controls whenever available by means of accessing the centralized cloud. Naturally, the cost of generated policy improves with better connection probability ρ. The results suggest that our methods produce intelligent policies in imperfect communication cases.

7) Comparison With Other Work in Partial Communication: We compare our approach with a recent approach A3C3 [8], where the agents learn to communicate using an actor–critic framework. We apply A3C3 with four agents, where each agent learns five and ten communication bits from every other agent. We use the authors' source code for A3C3 with a thorough hyperparameter search. From Table V, we see that our AMR-B, which provides the worst cost for the imperfect communication cases, outperforms A3C3. We provide results for various discount factors α since the authors use smaller α, including 0.01 for many tasks with four or more agents.

TABLE V
COMPARISON STUDY WITH A3C3 FOR FOUR AGENTS

α	Our AMR-B	A3C3 (five comm bits)	A3C3 (ten comm bits)
0.5	579.5	751.5	739.2
0.2	395.5	453.0	439.9
0.01	328.4	369.2	362.3

TABLE VI
COST COMPARISON OF MULTIAGENT ROLLOUT METHODS WITH INTERMITTENT BELIEF AND CONTROL SHARING WITH PROBABILITY ρ

Agents	Base policy	One-at-a-time rollout	AMR-IB1 $\rho = 0.4$	AMR-IB0 $\rho = 0.4$	AMR-IB1 $\rho = 0.8$	AMR-IB0 $\rho = 0.8$
4	3297	1906	2772	2752	2320	2251
8	5347	992	1513	1713	1128	1140
10	4667	799	1265	1399	960	921

H. Results on the Multirobot Repair Problem With Imperfect Communication of Belief States and Controls

Here we consider the case where the agents do not share their belief states and cannot communicate their choice of control with one another at all times. Each agent knows its location and can obtain a perfect observation of the damage at its location. However, agents may not always perfectly perceive other agents' locations and knowledge of their observations. In this way, each agent may have a local belief state that is different from the true global belief state. We consider the existence of a centralized cloud server having access to the global belief states with an intermittent connection probability of $\rho \in (0, 1)$. If the cloud is reachable, every agent synchronizes with the global belief state. During that time step, the one-agent-at-a-time rollout is performed with the computed predecessors' control components given by the cloud, and each agents' belief state is evolved forward using this information accordingly. If the cloud is not accessible, the local belief corresponding to an agent evolves by applying the locally computed control component and the base policy's control components for other agents. In this context, we consider the following communication architectures and perform an extensive performance simulation study. The simulation setup related to the Markov chain dynamics and rollout parameters used in this section is the same as that of Section VIII-G. Each AMR method's name includes the number of optimizations performed by an agent when the belief and controls are not shared.

1) Approximate Multiagent Rollout With Base Policy Signaling and Intermittent Communication (AMR-IB1): If the belief and control are not accessible via the cloud server, each agent performs one optimization to estimate its own control component evaluated at its local belief state, assuming other agents' control components are given by the base policy.

2) Approximate Multiagent Rollout With Intermittent Communication and Base Policy (AMR-IB0) w/o Optimization: If the belief and control are not accessible via the cloud server, each agent applies the base policy control evaluated at its local belief state independently from the team (without estimating or taking into consideration the actions of the other agents).

Table VI shows the cost comparison between the base policy, one-at-a-time rollout policy, and different architectures involving intermittent communication with imperfect belief and control sharing for four, eight, and ten agents. We observe that all the multiagent rollout methods with intermittent communication improve over the base policy, and the cost improves with a better connection probability ρ. With a high probability $\rho \to 1$, the methods perform similar to the one-at-a-time rollout. With a low probability $\rho \to 0$, the methods produce similar to the base policy. Interestingly, we see that for the same intermittent communication probability ρ, AMR-IB0 outperforms AMR-IB1 in some cases and, at the same time, reduces the computations by a factor of m when the cloud is not reachable. In AMR-IB1, each agent chooses its control component, thinking the other agents will apply the base policy. This, in effect, might miss some damaged locations before the global belief is shared. In contrast, AMR-IB0 does not try to make any smarter moves until the cloud is reachable and uses the base policy's controls until then. AMR-IB0 has less chance of missing a damage location than AMR-IB1 and has better cost.

I. Discussion

Our results suggest that our multiagent rollout algorithm and its variants are suitable for a practical class of multiagent systems involving imperfect belief and control communication and imperfect state observation. Our methods take advantage of a centralized information source whenever possible and subsequently leverage our computation efficient rollout methods. Our methods are also extended in the case when the server is unreachable, and agents select controls in a distributed fashion with imperfect knowledge of the true global belief state. Such distributed computation achieves significant speedup through parallelization. **A base policy** can be a simple greedy or a heuristic-based policy that performs reasonably well on a problem. For a discounted infinite horizon problem, discussed in this article, the rollout policy improves over the base policy, when the base policy is a stationary policy [36], that produces same action on a given state independent of the time of application. A base policy should also maintain stability [6] such that the online rollout policy maintains the performance improvement. A base policy is stable as long as the stage costs are finite for a discounted infinite horizon problem, and hence the total cost is uniformly bounded. **The comparison study signifies** our approach provides a good policy with a shorter lookahead tree with more precise Q-factor estimation and by best utilizing the information available from other agents. Other sample-based methods including POMCP, PA-POMCPOW may fail to estimate Q-factor precisely due to the use of a long and sparse lookahead tree optimization or a sparse sampling of observations. Our comparison with MADDPG, that follows a centralized training decentralized execution philosophy, reveals that communicating during policy execution is key for problems with fast dynamics like the multiagent repair problem where damage levels are degraded rapidly unless a cooperative policy is applied within a critical time frame. Other gradient-based methods, including A3C3, may fail to learn a good policy without explicit communication available from peer agents. This is because learning to communicate messages with an actor–critic framework can be very challenging in a complex sequential repair problem, with a large horizon, large belief, and control space.

J. Limitations and Future Directions

1) **Discrete Space:** In this article, we consider a discrete environment represented by a known graph and discrete control space for each agent. This article does not deal with problems with continuous action space, including but not limited to multiagent path finding problems, which we will consider in the future.
2) **Safety:** It is important to note that an online policy needs to follow safety considerations when applied to real-world environments. However, we do not consider the safety aspect in this article but we believe that it would be an interesting direction of future study.
3) **Delayed Communication:** Although we do not focus on a delayed communication setup in this article, AMR-IB0 and AMR-IB1 policies may be applied if the local beliefs and control components are shared in finite intervals. However, communication delays may involve various additional challenges we intend to focus in the future.

IX. Conclusion

In this article, we present various multiagent rollout methods, approximate PI and online play policy with offline trained approximations for handling challenging large-scale POMDP problems. We experimentally verify the policy improvement property of one-agent-at-a-time rollout, similar to standard rollout, with dramatically less computation requirements. Similarly, we show that multiagent approximate PI improves the policy at each iteration in order to find the approximately optimal policy. We show that the online play policy enhances the solution quality of offline trained architectures, especially for problems with changing system parameters, by providing adaptive controls. We present extensions of our multiagent rollout methods, analytically justify their performance guarantees, and report numerical results for the imperfect communication case. Based on our results, the methods discussed here work well for robotics problems, particularly when a large team of multiple robots needs to collaborate on a complex task over long horizons, with a large state space with partial observations and a large action space (induced by a large number of agents), under partial communication. Future extensions to our POMDP algorithms include but are not limited to asynchronous communication of the belief state, imperfect knowledge of the agents' own locations, and partitioning of the belief state space to achieve distributed learning.

Appendix A
Policy Improvement With Arbitrary Agent Order

In this section, we show that policy improvement holds for a random-order rollout in partially observable cases, by extending the argument for policy improvement in order-optimized rollout

for perfectly observable MDPs given in [4]. We show that the cost of the one-at-a-time rollout improves the cost of the base policy J_π irrespective of the order of agents at each step with two agents, but can be easily extended for more agents. We consider the policy π_{ro}, that applies the one-at-a-time rollout to optimize for agent 1's control component followed by agent 2's control component with a probability p_1, and uses the other agent order with a probability $p_2 = 1 - p_1$. Using the definitions of cost function J_π, Bellman operator T_π, we first show that $(T_{\pi_{\text{ro}}} J_\pi)(b) \leq J_\pi(b), \forall b$, which is needed to show $J_{\pi_{\text{ro}}}(b) \leq J_\pi(b)$. Now, for all belief b, $(T_{\pi_{\text{ro}}} J_\pi)(b)$

$$= \hat{g}(b, \pi_{\text{ro}}(b)) + \alpha \sum_{z \in Z} \hat{p}(z|b, \pi_{\text{ro}}(b)) J_\pi(F(b, \pi_{\text{ro}}(b), z))$$

$$= \hat{g}(b, (\pi_{\text{ro},1}(b), \pi_{\text{ro},2}(b))) + \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_{\text{ro},1}(b), \pi_{\text{ro},2}(b)))$$

$J_\pi(F(b, (\pi_{\text{ro},1}(b), \pi_{\text{ro},2}(b)), z))$

$= p_1 \text{min}_{u_2 \in U_2} [\hat{g}(b, (\pi_{\text{ro},1}(b), u_2))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_{\text{ro},1}(b), u_2)) J_\pi(F(b, (\pi_{\text{ro},1}(b), u_2), z))]$

$+ p_2 \text{min}_{u_1 \in U_1} [\hat{g}(b, (u_1, \pi_{\text{ro},2}(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (u_1, \pi_{\text{ro},2}(b))) J_\pi(F(b, (u_1, \pi_{\text{ro},2}(b)), z))]$

(by definition)

$\leq p_1 [\hat{g}(b, (\pi_{\text{ro},1}(b), \pi_2(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_{\text{ro},1}(b), \pi_2(b))) J_\pi(F(b, (\pi_{\text{ro},1}(b), \pi_2(b)), z))]$

$+ p_2 [\hat{g}(b, (\pi_1(b), \pi_{\text{ro},2}(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_1(b), \pi_{\text{ro},2}(b))) J_\pi(F(b, (\pi_1(b), \pi_{\text{ro},2}(b)), z))]$

$= p_1 \text{min}_{u_1 \in U_1} [\hat{g}(b, (u_1, \pi_2(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (u_1, \pi_2(b))) J_\pi(F(b, (u_1, \pi_2(b)), z))]$

$+ p_2 \text{min}_{u_2 \in U_2} [\hat{g}(b, (\pi_1(b), u_2))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_1(b), u_2)) J_\pi(F(b, (\pi_1(b), u_2), z))]$

(by definition)

$\leq p_1 [\hat{g}(b, (\pi_1(b), \pi_2(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_1(b), \pi_2(b))) J_\pi(F(b, (\pi_1(b), \pi_2(b)), z))]$

$+ p_2 [\hat{g}(b, (\pi_1(b), \pi_2(b)))$

$+ \alpha \sum_{z \in Z} \hat{p}(z|b, (\pi_1(b), \pi_2(b))) J_\pi(F(b, (\pi_1(b), \pi_2(b)), z))]$

$= p_1 J_\pi(b) + p_2 J_\pi(b) (\text{by definition of } J_\pi)$

$= J_\pi(b) (\text{since } p_2 = 1 - p_1).$

Now that we show $(T_{\pi_{\text{ro}}} J_\pi)(b) \leq J_\pi(b), \forall b$, we have for a $K \in \mathbb{N}$, the following, $(T^K_{\pi_{\text{ro}}} J_\pi)(b) = (T^{K-1}_{\pi_{\text{ro}}})(T_{\pi_{\text{ro}}} J_\pi)(b) \leq (T^{K-1}_{\pi_{\text{ro}}} J_\pi)(b) = (T^{K-2}_{\pi_{\text{ro}}})(T_{\pi_{\text{ro}}} J_\pi)(b) \leq (T^{K-2}_{\pi_{\text{ro}}} J_\pi)(b) = (T^{K-3}_{\pi_{\text{ro}}})(T_{\pi_{\text{ro}}} J_\pi)(b) \leq (T^{K-3}_{\pi_{\text{ro}}} J_\pi)(b) \leq \ldots \leq (T_{\pi_{\text{ro}}} J_\pi)(b) \leq J_\pi(b)$. The cost of the policy π_{ro} is, $J_{\pi_{\text{ro}}}(b) = \lim_{K \to \infty} (T^K_{\pi_{\text{ro}}} J_\pi)(b) \leq J_\pi(b), \forall b$.

APPENDIX B
POLICY IMPROVEMENT WITH PARTIAL COMMUNICATION

Here, we discuss that policy improvement property holds for the AMR-IAC (described in Section VIII-G6). We consider the agent communication graph $G = (E, N)$, where is N represents agents $N = \{1, \ldots, m\}$ and a directional edge $e_{i,j} \in E$ is present between agent $i \in N$ to agent $j \in N$, if agent i can communicate with agent j. We denote an ordering P of agents given by the largest path (with length $\bar{m}$) in the graph G. We reindex all m agents so that $\bar{m}$ agents that belong to path P (the longest-directed path in the agent communication graph) come before agents that do not belong to P. We apply the one-agent-at-a-time rollout to the first $\bar{m}$ agents where the rollout order is given by the agent order in path P, and all other $m - \bar{m}$ agents apply the base policy π. The resulting policy is $\pi_{\text{partial}}(b) = (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}}(b), \pi_{\bar{m}+1}(b), \ldots, \pi_m(b)), \forall b$, where $\bar{\pi}_\ell = \text{argmin}_{u_\ell \in U_\ell(b)} \{\hat{g}(b, u') + \alpha \sum_{z \in Z} \hat{p}(z|b, u') J_\pi(F(b, u', z))\}$, with $(\ell \leq \bar{m})$, and

$$u' = (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\ell-1}(b), u_\ell, \pi_{\ell+1}(b), \ldots, \pi_m(b)).$$

Using the definitions of cost function J_π, Bellman operator T_π, we first show that $(T_{\pi_{\text{partial}}} J_\pi)(b) \leq J_\pi(b), \forall b$, which is essential in showing $J_{\pi_{\text{partial}}}(b) \leq J_\pi(b), \forall b$. Now, we define an operator $H(b, \pi(b), J) = (T_\pi J)(b)$. Now, for all belief b , we have

$T_{\pi_{\text{partial}}} J_\pi(b) = H(b, \pi_{\text{partial}}(b), J_\pi)$

$= H(b, (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}}(b), \pi_{\bar{m}+1}(b), \ldots, \pi_m(b)), J_\pi)$

$= \text{min}_{u_{\bar{m}} \in U_{\bar{m}}} H(b, (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}-1}(b), u_{\bar{m}},$

$\pi_{\bar{m}+1}(b), \ldots, \pi_m(b)), J_\pi)$

$\leq H(b, (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}-1}(b), \pi_{\bar{m}}(b), \ldots, \pi_m(b)), J_\pi)$

$= \text{min}_{u_{\bar{m}-1} \in U_{\bar{m}-1}} H(b, (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}-2}(b), u_{\bar{m}-1},$

$\pi_{\bar{m}}(b), \ldots, \pi_m(b)), J_\pi)$

$\leq H(b, (\bar{\pi}_1(b), \ldots, \bar{\pi}_{\bar{m}-2}(b), \pi_{\bar{m}-1}(b), \ldots, \pi_m(b)), J_\pi)$

$\leq \ldots \leq H(b, (\bar{\pi}_1(b), \pi_2(b), \ldots, \pi_m(b)), J_\pi)$

$= \text{min}_{u_1 \in U_1} H(b, (u_1, \pi_2(b), \ldots, \pi_m(b)), J_\pi)$

$\leq H(b, (\pi_1(b), \ldots, \pi_m(b)), J_\pi) = H(b, \pi(b), J_\pi)$

$= T_\pi J_\pi(b) = J_\pi(b).$

The aforementioned derivation gives $(T_{\pi_{\text{partial}}} J_\pi)(b) \leq J_\pi(b), \forall b$. For a finite $K \in \mathbb{N}$, this inequality gives us the following. $(T^K_{\pi_{\text{partial}}} J_\pi)(b) = (T^{K-1}_{\pi_{\text{partial}}})(T_{\pi_{\text{partial}}} J_\pi)(b) \leq (T^{K-1}_{\pi_{\text{partial}}} J_\pi)(b) = (T^{K-2}_{\pi_{\text{partial}}})(T_{\pi_{\text{partial}}} J_\pi)(b) \leq (T^{K-2}_{\pi_{\text{partial}}} J_\pi)(b) = (T^{K-3}_{\pi_{\text{partial}}})(T_{\pi_{\text{partial}}} J_\pi)(b) \leq (T^{K-3}_{\pi_{\text{partial}}} J_\pi)(b) \leq \ldots \leq (T_{\pi_{\text{partial}}} J_\pi)(b) \leq J_\pi(b)$. The cost

of the policy π_{partial} is, $J_{\pi_{\text{partial}}}(b) = \lim_{K\to\infty}(T^K_{\pi_{\text{partial}}}J_\pi)(b) \leq J_\pi(b), \forall b$.

APPENDIX C
CONVERGENCE WITH API WITH ONE-AT-A-TIME ROLLOUT

In this section, we discuss convergence of approximate PI with multiagent one-at-a-time rollout as the policy improvement step by extending the convergence proof for approximate PI with standard rollout given in the book [36], Proposition 5.1.4. The book [36] gives the convergence proof for single agent approximate PI with standard rollout, we extend that proof for multiagent approximation PI with one-agent-at-a-time rollout. Let $\tilde{\pi}^k$ be the approximate PI policy at iteration k where one-at-a-time rollout is used in the policy improvement step, and $J_{\tilde{\pi}^k}$ be the cost of policy $\tilde{\pi}^k$. In addition, we denote $\tilde{J}_k$ to approximate the cost $J_{\tilde{\pi}^k}$ at iteration k in the policy evaluation step. Policy $\bar{\pi}^k$ is the Approximate PI policy at iteration k where standard (all-at-once) rollout is used in the policy improvement step, and $J_{\bar{\pi}^k}$ is the cost of policy $\bar{\pi}^k$. For any iteration k, any cost function J, the discount factor α, with $0 < \alpha < 1$ and with $||.||$ to denote max norm, $\exists \delta, \epsilon, \epsilon' > 0$, if the following assumptions hold:

$$||\tilde{J}_k - J_{\tilde{\pi}^k}|| \leq \delta \tag{10a}$$

$$||T_{\bar{\pi}^k}J - TJ|| \leq \epsilon \tag{10b}$$

$$||T_{\tilde{\pi}^k}J - T_{\bar{\pi}^k}J|| \leq \epsilon' \tag{10c}$$

then cost $J_{\tilde{\pi}^k}$ has the bound with the optimal cost J^*, $\limsup_{k\to\infty}||J_{\tilde{\pi}^k} - J^*|| \leq \frac{\epsilon'+\epsilon+2\alpha\delta}{(1-\alpha)^2}$.

Proof sketch: Rewriting (10a) with max norm definition

$$J_{\tilde{\pi}^k} \leq \tilde{J}_k + \delta v, \text{ and } \tilde{J}_k \leq J_{\tilde{\pi}^k} + \delta v \tag{11}$$

where v is a vector of all 1's. Applying the bellman operator T for policy $\tilde{\pi}^{k+1}$ on both sides of the left expression of (11)

$$\begin{aligned}
&T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} \leq T_{\tilde{\pi}^{k+1}}\tilde{J}_k + \alpha\delta v \quad (\text{the contraction property of } T_\pi)\\
&= (T_{\tilde{\pi}^{k+1}}\tilde{J}_k - T_{\bar{\pi}^{k+1}}\tilde{J}_k) + (T_{\bar{\pi}^{k+1}}\tilde{J}_k - T\tilde{J}_k) + T\tilde{J}_k + \alpha\delta v\\
&\leq T\tilde{J}_k + (\epsilon' + \epsilon + \alpha\delta)v \quad [\text{from (10c)and (10b)}]\\
&\leq TJ_{\tilde{\pi}^k} + \alpha\delta v + (\epsilon' + \epsilon + \alpha\delta)v \quad [\text{using right expression of}\\
&(11)\text{and the contraction property of } T]\\
&\leq T_{\tilde{\pi}^k}J_{\tilde{\pi}^k} + (\epsilon' + \epsilon + 2\alpha\delta)v \quad (\text{since } TJ \leq T_\pi J)\\
&= J_{\tilde{\pi}^k} + (\epsilon' + \epsilon + 2\alpha\delta)v \quad (\text{since } T_\pi J_\pi = J_\pi).
\end{aligned}$$

From the aforementioned derivation, we get

$$T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} \leq TJ_{\tilde{\pi}^k} + (\epsilon' + \epsilon + 2\alpha\delta)v \tag{12a}$$

$$T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} \leq J_{\tilde{\pi}^k} + (\epsilon' + \epsilon + 2\alpha\delta)v. \tag{12b}$$

The cost of policy $\tilde{\pi}^{k+1}$ is

$$\begin{aligned}
&J_{\tilde{\pi}^{k+1}} = \lim_{K\to\infty} T^K_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} \qquad (\text{from definition of } J_\pi)\\
&= \lim_{K\to\infty} T^{K-1}_{\tilde{\pi}^{k+1}}(T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k})\\
&\leq \lim_{K\to\infty} T^{K-1}_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} + \alpha^{K-1}(\epsilon' + \epsilon + 2\alpha\delta)v \ [\text{from (12b)}]\\
&\ldots\\
&\leq T^2_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} + \lim_{K\to\infty}[(\alpha^2 + \ldots + \alpha^{K-1})(\epsilon' + \epsilon + 2\alpha\delta)v]\\
&= T_{\tilde{\pi}^{k+1}}(T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k}) + \frac{\alpha^2}{1-\alpha}(\epsilon' + \epsilon + 2\alpha\delta)v\\
&\leq T_{\tilde{\pi}^{k+1}}J_{\tilde{\pi}^k} + \left(\alpha + \frac{\alpha^2}{1-\alpha}\right)(\epsilon' + \epsilon + 2\alpha\delta)v \ [\text{from (12b)}]\\
&\leq TJ_{\tilde{\pi}^k} + \left(1 + \frac{\alpha}{1-\alpha}\right)(\epsilon' + \epsilon + 2\alpha\delta)v \ [\text{from (12a)}].
\end{aligned}$$

From the aforementioned derivation, we get $J_{\tilde{\pi}^{k+1}} \leq TJ_{\tilde{\pi}^k} + \frac{\epsilon+\epsilon'+2\alpha\delta}{1-\alpha}v$. Subtracting both sides with J^* and taking max norm, $||J_{\tilde{\pi}^{k+1}} - J^*|| \leq ||TJ_{\tilde{\pi}^k} - J^*|| + \frac{\epsilon+\epsilon'+2\alpha\delta}{1-\alpha} = ||TJ_{\tilde{\pi}^k} - TJ^*|| + \frac{\epsilon+\epsilon'+2\alpha\delta}{1-\alpha} \leq \alpha||J_{\tilde{\pi}^k} - J^*|| + \frac{\epsilon+\epsilon'+2\alpha\delta}{1-\alpha}$. The last two inequalities come from the fact $TJ^* = J^*$ and the contraction of T, respectively. Taking $\limsup$ to both sides as $k \to \infty$, we get the desired result, $\limsup_{k\to\infty}||J_{\tilde{\pi}^k} - J^*|| \leq \frac{\epsilon+\epsilon'+2\alpha\delta}{(1-\alpha)^2}$.

ACKNOWLEDGMENT

The authors would like to thank the reviewers who provided useful and insightful comments for this work and the colleagues who contributed to the ideas.

REFERENCES

[1] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. 23rd Int. Conf. NeurIPS*, 2010, pp. 2164–2172.

[2] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[3] D. P. Bertsekas, "Multiagent rollout algorithms and reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 2, pp. 249–272, Feb. 2021.

[4] D. P. Bertsekas, *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Belmont, MA, USA: Athena Sci., 2020.

[5] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Belmont, MA, USA: Athena Sci., 2019.

[6] D. P. Bertsekas, *Lessons From Alphazero for Optimal, Model Predictive, and Adaptive Control*. Nashua, NH, USA: Athena Sci., 2022.

[7] J. Mern, A. Yildiz, L. Bush, T. Mukerji, and M. J. Kochenderfer, "Improved POMDP tree search planning with prioritized action branching," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11888–11894.

[8] D. Simões, N. Lau, and L. P. Reis, "Multi-agent actor centralized-critic with communication," *Neurocomputing*, vol. 390, pp. 40–56, 2020.

[9] S. Bhattacharya, S. Badyal, T. Wheeler, S. Gil, and D. Bertsekas, "Reinforcement learning for POMDP: Partitioned rollout and policy iteration with application to autonomous sequential repair problems," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 3967–3974, Jul. 2020.

[10] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent rollout and policy iteration for POMDP with application to multi-robot repair problems," in *Proc. Conf. Robot Learn.*, 2021, pp. 1814–1828. [Online]. Available: https://proceedings.mlr.press/v155/bhattacharya21a.html

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, 1998.

[12] N. Meuleau, K. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving POMDPs by searching the space of finite policies," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999.

[13] R. Zhou and E. A. Hansen, "An improved grid-based approximation algorithm for POMDPs," in *Proc. 17th Int. Joint Conf. Artif. Intell.*, 2001, pp. 707–714.

[14] H. Yu and D. P. Bertsekas, "Discretized approximations for POMDP with average cost," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 619–627.

[15] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. AI Res.*, vol. 15, pp. 319–350, 2001.

[16] H. Yu, "A function approximation approach to estimation of policy gradient for POMDP with structured policies," in *Proc. 21st Conf. Uncertainty Artif. Intell.*, 2005, pp. 642–649.
[17] R. M. Estanjini, K. Li, and I. C. Paschalidis, "A least squares temporal difference actor-critic algorithm with applications to warehouse management," *Nav. Res. Logistics*, vol. 59, pp. 197–211, 2012.
[18] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Proc. Adv. NeurIPS*, 2013, pp. 1772–1780.
[19] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017, *arXiv:1707.06347*.
[21] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned POMDPs," *Int. J. Robot. Res.*, vol. 32, no. 6, pp. 650–671, 2013.
[22] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.
[23] J. Paulos, S. W. Chen, D. Shishika, and V. Kumar, "Decentralization of multiagent policies by learning what to communicate," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 7990–7996.
[24] J. P. Inala et al., "Neurosymbolic transformers for multi-agent communication," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 13597–13608.
[25] D. Silveret al., "A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
[26] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *Proc. Int. Conf. Emerg. Secur. Technol.*, 2010, pp. 142–147.
[27] A. Cassandra, "A survey of POMDP applications," in *Proc. Work. Notes AAAI Fall Symp. Plan. Partially Observable Markov Decis. Processes*, 1998, pp. 17–24.
[28] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus, "Experiments with cooperative control of underwater robots," *Int. J. Robot. Res.*, vol. 28, pp. 815–833, 2009.
[29] M. Lauri, J. Pajarinen, and J. Peters, "Multi-agent active information gathering in discrete and continuous-state decentralized POMDPs by policy graph improvement," *Auton. Agents Multi-Agent Syst.*, vol. 34, pp. 1–44, 2020.
[30] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.
[31] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 4797–4803.
[32] M. G. Lagoudakis and R. Parr, "Reinforcement learning as classification: Leveraging modern classifiers," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 424–431.
[33] C. Dimitrakakis and M. G. Lagoudakis, "Rollout sampling approximate policy iteration," *Mach. Learn.*, vol. 72, pp. 157–171, 2008.
[34] M. Yemini, S. Gil, and A. Goldsmith, "Exploiting local and cloud sensor fusion in intermittently connected sensor networks," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–7.
[35] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to probability*. Belmont, MA, USA: Athena Sci., 2000.
[36] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Belmont, MA, USA: Athena Sci., 2019.

Sushmita Bhattacharya is currently working toward the Ph.D. degree in multiagent sequential decision-making with realistic considerations with REACT Lab, Computer Science Department, Harvard University, Cambridge, MA, USA, advised by Dr. Stephanie Gil.

Her research interests include multirobot sequential decision-making and distributed learning with uncertainty.

Siva Kailas received the B.S. degree in computer science from Arizona State University, , Tempe, AZ, USA, in 2020. He is currently working toward the M.S. degree in computer science with Carnegie Mellon University.

He was with REACT Lab, Arizona State University. His research interests include multiagent systems, information gathering, and path planning.

Mr. Kailas is a Member of the AART Lab, Carnegie Mellon University, Pittsburgh, PA, USA.

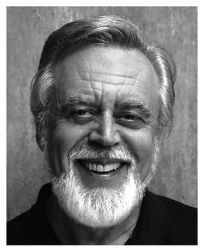
Sahil Badyal received the M.S. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2021.

He is currently a Data Scientist with Capital One, McLean, VA, USA, and was with REACT Lab with Arizona State University.

Stephanie Gil (Member, IEEE) received the Ph.D. degree in communication and control in multirobot systems from Aero/Astro Department, Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014.

She is currently an Assistant Professor with Computer Science Department, School of Engineering and Applied Sciences, Harvard University, Cambridge, where she directs the Robotics, Embedded Autonomy, and Communication Theory Lab. Her research interests include multirobot systems, where she studies the impact of information exchange and communication on resilience and trusted coordination.

Dr. Gil was the recipient of the 2019 Faculty Early Career Development Program Award from the National Science Foundation (NSF CAREER), and the Alfred P. Sloan Fellowship, in 2020. She was selected as an Office of Naval Research Young Investigator, in 2021.



Dimitri Bertsekas received the Ph.D. degree in system science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1971.

He has held Faculty positions with Stanford University, Stanford, CA, USA; University of Illinois Urbana-Champaign, Champaign, IL, USA, MIT, and Arizona State University, Tempe, AZ, USA, respectively. He has authored or coauthored numerous research papers and books. His research interests include optimization, control, large-scale computation, and data communication networks.

Dr. Bertsekas was the recipient of the INFORMS 1997 Prize, the 2000 Greek National Award, the 2014 AACC Bellman Control Heritage Award, the 2022 IEEE Control Systems Award, and the INFORMS John von Neumann Theory Prize (2018). In 2001 he was elected the US National Academy of Engineering.