Multiagent Reinforcement Learning: Rollout and Policy Iteration for POMDP with Application to Multi-Robot Problems

Sushmita Bhattacharya, Siva Kailash, Sahil Badyal, Stephanie Gil, Dimitri Bertsekas

***Abstract*—In this paper, 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 off-line approximate policy iteration scheme, and we apply an additional 'on-line play' scheme enhancing off-line approximation architectures. 2) We consider the imperfect communication case and provide various extensions to our rollout methods to deal with this case. 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 [1] and MADDPG [2] and work well where other methods, including POMCP [1], DESPOT [3] fail to scale up.**

***Index Terms*—Multiagent reinforcement learning, POMDP, Multiagent rollout, Approximate policy iteration, On-line play policy, Imperfect communication.**

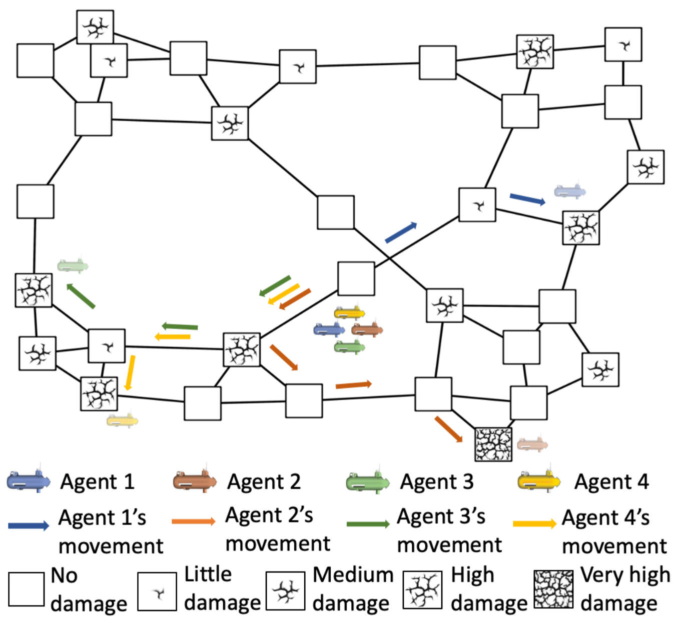


Fig. 1. We study a multiagent sequential decision-making problem where four agents coordinate in repairing a partially observable network of 32 vertices, each with a different level of damage.

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 realistic concerns, 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 multi-robot repair problems under partial state observation. Fig. 1 shows an instance of the repair problem where 4 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 (DP) is typically intractable. In this paper, 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. It is based on the multiagent rollout and policy iteration ideas first proposed in paper [4] and discussed at length in the research monograph [5]. Important distinctions of the current work include explicit treatment of the partial state observation case and the development of a multiagent decision-making framework within the partially observable context that extends to the imperfect agent communication cases.

The standard form of rollout, as described in several sources (e.g., the reinforcement learning book [6]), starts with some easily implementable policy, called the *base policy*, and produces another policy, the *rollout policy*, using one-step or multistep lookahead optimization. Its key property is *the*

S. Bhattacharya, and S. Gil are with the REACT lab in the Computer Science Department at the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02139 USA, email: sushmita_bhattacharya@g.harvard.edu, sgil@seas.harvard.edu.

S. Kailash, S. Badyal are with the REACT lab as external collaborators, email: siva.kailas10@gmail.com, sbadyal@asu.edu.

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

Manuscript received May 16, 2022;

policy improvement property: here, the rollout policy improves performance over the base policy for all initial 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 also implemented approximately using truncated rollout and a terminal cost function approximation. In this case, the policy improvement property applies in an approximate form, which is quantified by an error bound that is no worse than the one for the corresponding standard rollout.

Lastly, we employ our multiagent rollout algorithm in an approximate policy iteration framework in order to improve the policy in an iterative fashion where successive policies are approximated using neural networks. The performance of a purely off-line trained policy may degrade in a complex and dynamic environment. Thus we study an 'on-line play policy' that performs an on-line lookahead optimization with an off-line trained policy as the base policy and an optional terminal cost approximation, trained off-line. The concept of such on-line play in the context of rollout was first introduced in [7] for the perfect observation case. The difference between [7] and our work is that we consider the application of the on-line play policy in problems that are much more complex due to the partial state observation. We show in our simulation experiments that the performance of the on-line play policy is superior than both our multiagent rollout (without any off-line training) and our approximate policy iteration algorithms (with off-line approximations). This performance improvement can be attributed to the fact that the on-line play is considered a true Newton step that has the superlinear convergence property, making the on-line play policy converge to the optimal policy much faster than the off-line approximations (see [7] for detailed analysis).

We demonstrate the implementation of our multiagent rollout methods on a challenging class of multi-robot repair problems. In the repair problem, it is important to use a policy that can identify and execute critical repairs in minimum time by leveraging coordination among the agents. We demonstrate the power of our method by applying it to a complex repair problem involving a network of 32 partially observable, potentially damaged locations, and as many as 10 repair robots/agents (see Fig. 1, where 4 agents are employed). This is a number of agents (10 agents) that is well beyond the capabilities of standard POMDP methods. In particular, we present favorable comparisons (with 4 agents) of our proposed method with the state-of-art software Partially Observable Monte-Carlo Planning (POMCP [1]) and Multi-Agent Deep Deterministic Policy Gradient (MADDPG [2]) that also provide an approximate solution for POMDP problems.

Up until this point, our treatment of multiagent rollout, approximate policy iteration, and on-line play policy assume perfect communication. 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. Additionally, we recover the policy improvement property for our multiagent rollout with an intermittent communication architecture. In this intermittent communication architecture, we assume the presence of an intermittently available cloud server, which, when available, 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 present extensive numerical results comparing the performance of various imperfect communication architectures for multiagent rollout in the large and complex POMDP setup for the multi-robot repair problem where agents may not always share belief states and computed controls.

This paper is an evolved paper from our earlier conference publication [8], and the new contributions in this paper which weren't included in [8] are the following. 1) We discuss the on-line 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. In this context, we show that if the computed controls are not shared between agents, finite termination can be hindered, and subsequently, the policy improvement property will not hold. We propose an approach of using randomization in case of no control communication and analytically show that finite termination is guaranteed in this case. Finally, we analytically show that the policy improvement property is recoverable using an intermittent control communication architecture. The analytical results for imperfect communication cases bolster and rationalize our numerical results. 3) We present extensive numerical simulations on a multi-robot repair problem using the on-line play policy and show that the on-line play significantly improves policy and cost approximations trained using an offline approximate policy iteration. We experimentally validate the robustness and adaptability of the on-line play policy with dynamically changing system parameters, which signifies the role of the online replanning.

II. Related work

Several reinforcement learning algorithms for POMDP problems have been proposed in the literature. In particular, [9] describes a general solution method for POMDP, [10] discusses a policy search method using finite state controllers, [6], [11], [12] discuss aggregation-based methods, and [13]–[15] 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 trade off 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 [3]) 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 [16]–[18] have been proposed. Among them, [19] 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 papers [2], [20] 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 whenever available. In Section VII, we compare the performance of our methods to several of these state-of-art POMDP methods.

The paper [21] proposes rollout and policy iteration methods that can address POMDP, but does not deal with multiagent problems and has difficulty dealing with a large control space. The rollout policy improvement property, given in [21], also holds for the multiagent version of this paper. The proof was given in [4] and an associated performance bound was given in the research monograph [5]. The paper [7] talks about the on-line play policy in the context of rollout as a Newton step to improve off-line trained policies with an on-line optimization and their superlinear convergence to the optimal policy but does not consider partial state observation and its associated challenges. AlphaZero [22] talks about a setup where MCTS-based on-line lookahead is used to improve off-line approximation architectures given by deep neural networks in both policy and value spaces for perfect-state observation cases. In contrast, this paper considers on-line play for partially observable multiagent decision-making problems with a smaller but denser lookahead optimization that improves over off-line trained policy and value approximations with shallow networks.

According to [23], robotic exploration is one of the biggest challenges in the field. The methods discussed in this paper are well suited for related multiagent contexts such as search and rescue applications [24]–[29]. Researchers in robotics have proposed various methods for addressing robotic exploration and large state spaces [30]–[33], also in a POMDP setting [34], [35]. The algorithms developed here are compatible with various coordination issues [27], [34], [36]–[38].

This paper is most closely related to the multiagent rollout methods developed in [4], on-line play policy [7], and autonomous repair problems [21]. Our work differs in various important ways from this prior work: 1) we treat the case of the partially observable state, leading to an explosion in the size of the state space, not addressed in [4], 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 [21], 3) we employ the on-line play policy to a partially observable multi-robot repair problem (not discussed in [7]), 4) we consider more general repair problems described over arbitrary graph topologies (in contrast with the linear or strict grid topology in [21]) and treat the significantly more challenging and realistic non-terminating case where previously repaired locations can fall into disrepair (not treated in [21]), 5) we consider practical issues of imperfect communication in the multiagent rollout that are not discussed in [4].

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 $i = 1, \dots, n$, and that the control u consists of m components, $u = (u_1, u_2, \dots, u_m)$. 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 $u_\ell, \ell \in \{1, 2, \dots, m\}$, 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.

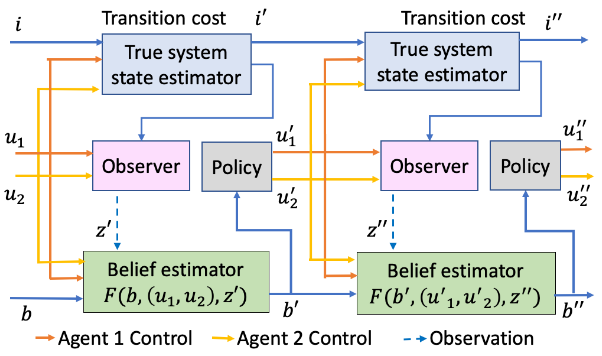


Fig. 2. Composite system simulator for POMDP for a given policy involving 2 agents (adapted from [21]). The starting state i of a trajectory is generated randomly using the belief state b.

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 [21]. In particular, the belief state is the conditional probability vector $b = \big(b(1), \dots, b(n)\big)$, 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 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). The optimal cost function $J^*(b)$ is the unique solution of the Bellman equation

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

Here 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 $\widetilde{\mu}$ is obtained by the one-step lookahead minimization

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

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 μ as the cost approximation in Eq. (1) ($\tilde{J}(b') = J_\mu(b')$, where $J_\mu(b')$ is the discounted cost of applying the policy μ starting from belief state b' until termination). In the truncated rollout version, $\tilde{J}(b')$ is the discounted cost of applying a base policy μ 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 $\bar{b}$ after t stages, 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 μ, and the terminal cost function approximation $\hat{J}$. The parameters l and t are usually chosen based on a trade-off between implementation complexity and obtained performance. The base policy can be a greedy policy, and the terminal cost

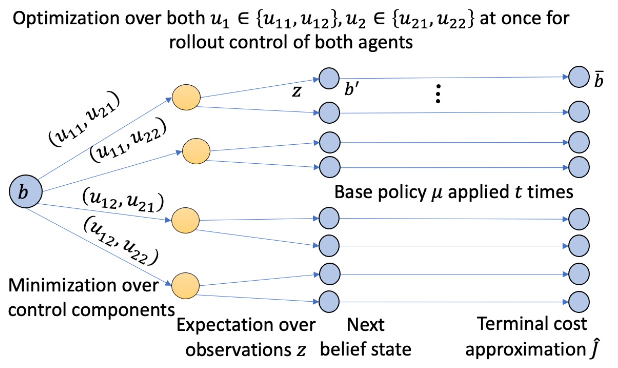

Optimization over both $u_1 \in \{u_{11}, u_{12}\}$, $u_2 \in \{u_{21}, u_{22}\}$ at once for rollout control of both agents
b
(u_{11}, u_{21})
(u_{11}, u_{22})
(u_{12}, u_{21})
(u_{12}, u_{22})
z
b'
$\bar{b}$
Base policy μ applied t times
Minimization over control components
Expectation over observations z
Next belief state
Terminal cost approximation $\hat{J}$

Fig. 3. Standard truncated rollout algorithm for 2 agents: one-step lookahead followed by t applications of the base policy μ, and cost approximation $\hat{J}$.

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 [21] (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 $\widetilde{\mu}$ improves over the base policy μ as the terminal cost function approximation $\hat{J}$ gets closer to the base policy cost J_μ.

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 [4]). The branch corresponding to control u is associated with a Q-factor corresponding to (b,u), which is $\hat{g}(b,u) + \alpha\sum_{z\in Z}\hat{p}(z|b,u)\tilde{J}\big(F(b,u,z)\big)$, the expression in brackets in Eq. (1). The standard rollout algorithm chooses the control that is associated with minimal Q-factor, cf. Eq. (1). This rollout algorithm (in the pure form, where $\tilde{J}$ is given by J_μ in equation (1)) possesses the policy improvement property in an exact form

$$J_{\widetilde{\mu}}(b) \leq J_\mu(b),$$

where μ is the base policy and $\widetilde{\mu}$ 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 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 (1-at-a-time)*

In order to reduce the algorithmic complexity of the standard rollout algorithm, the minimization over the control branches in the above formulation needs to be simplified.

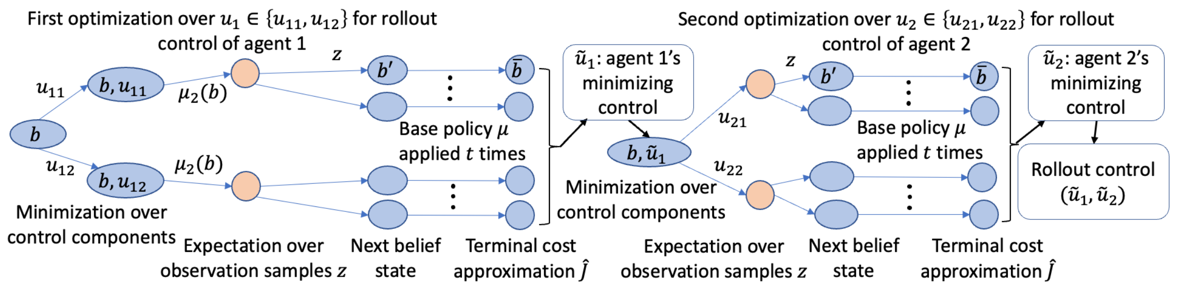

First optimization over $u_1 \in \{u_{11}, u_{12}\}$ for rollout control of agent 1
u_{11}
u_{12}
b
b, u_{11}
b, u_{12}
$\mu_2(b)$
$\mu_2(b)$
z
b'
$\bar{b}$
Base policy μ applied t times
Minimization over control components
Expectation over observation samples z
Next belief state
Terminal cost approximation $\hat{J}$
$\tilde{u}_1$: agent 1's minimizing control
$b, \tilde{u}_1$
Second optimization over $u_2 \in \{u_{21}, u_{22}\}$ for rollout control of agent 2
u_{21}
u_{22}
$\tilde{u}_2$: agent 2's minimizing control
Rollout control $(\tilde{u}_1, \tilde{u}_2)$

Fig. 4. One-agent-at-a-time truncated rollout algorithm for 2 agents using base policy μ with terminal cost approximation $\hat{J}$ on the reformulated state space $(b), (b, \tilde{u}_1)$.

To achieve improved algorithmic complexity, we introduce an equivalent problem formulation where the control $u = (u_1, u_2, \dots, u_m)$ 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 $\{b, (b, u_1), (b, u_1, u_2), \dots, (b, u_1, u_2, \dots, u_{m-1}), b'\}$ assuming the agents choose their controls sequentially in a fixed order. The last transition from $(b, u_1, u_2, \dots, u_{m-1})$ to b' involves the choice of the last component u_m and includes the cost $\hat{g}(b, u)$ of choosing control $u = (u_1, u_2, \dots, u_m)$ 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 $\tilde{\mu}(b) = (\tilde{\mu}_1(b), \dots, \tilde{\mu}_m(b))$, where $\tilde{\mu}_\ell(b)$ is the control component for agent ℓ at belief state b, $\ell = \{1, \dots, m\}$. When optimizing over the Q-factors of the component corresponding to agent ℓ , we set $u_1, \dots, u_{\ell-1}$ to $\tilde{\mu}_1(b), \dots, \tilde{\mu}_{\ell-1}(b)$, the optimized values calculated earlier by the rollout algorithm, and we set $u_{\ell+1}, \dots, u_m$ to the values dictated by the base policy at belief state b.

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

where $u' = (\tilde{\mu}_1(b), \dots, \tilde{\mu}_{\ell-1}(b), u_\ell, \mu_{\ell+1}(b), \dots, \mu_m(b))$, and $\mu(b) = (\mu_1(b), \dots, \mu_m(b))$ is the base policy's control at belief state 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 the paper [4] and the research monograph [5], 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 [5]). 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 [5] 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. 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 $\ell = 1, \dots, m$ 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 $\ell_1, \dots, \ell_m$ 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 policy iteration (PI)

We will now discuss the approximate policy iteration (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 on-line fashion by l-step lookahead minimization over the simulated trajectories (using t times base policy μ application followed by the terminal cost approximation $\hat{J}$) at each stage. The subsequent iterations can be expedited by replacing the on-line evaluation of the rollout policy with an approximation architecture (namely, a neural network) as shown in Fig. 5. See [6], 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 these

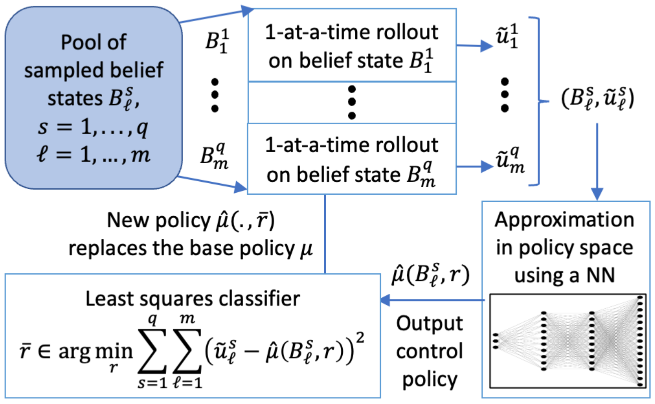

Pool of sampled belief states B^s_ℓ, $s = 1,\ldots,q$ $\ell = 1,\ldots,m$
B^1_1
B^q_m
1-at-a-time rollout on belief state B^1_1
1-at-a-time rollout on belief state B^q_m
$\tilde{u}^1_1$
$\tilde{u}^q_m$
$(B^s_\ell, \tilde{u}^s_\ell)$
Approximation in policy space using a NN
Output control policy
$\hat{\mu}(B^s_\ell, r)$
Least squares classifier
$\bar{r} \in \arg\min_r \sum_{s=1}^{q}\sum_{\ell=1}^{m}\left(\tilde{u}^s_\ell - \hat{\mu}(B^s_\ell, r)\right)^2$
New policy $\hat{\mu}(.,\bar{r})$ replaces the base policy μ

Fig. 5. Approximate policy iteration (PI) algorithm based on multiagent rollout and approximation in policy space. This figure is adapted from [21], which we extend by applying the one-agent-a-time rollout in the policy improvement phase for tackling multiple agents (not discussed in [21]).

multiagent truncated rollout schemes 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 $\hat{\mu}(b, \bar{r})$, 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 $\hat{\mu}(., \bar{r})$ 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 [39], and is also described in the book [6], 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-agent-at-a-time case discussed next.

B. Approximate PI with truncated one-agent-at-a-time rollout (1-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 $\ell = \{1, \ldots, m\}$ is given by the following equation at each stage:

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

where $u'^s = (\tilde{u}^s_1, \ldots, \tilde{u}^s_{\ell-1}, u_\ell, \mu_{\ell+1}(b^s), \ldots, u_m(b^s))$. Here, $\mu(b^s) = (\mu_1(b^s), \ldots, \mu_m(b^s))$ 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 $B^s_\ell = (b^s, \ell, \tilde{u}^s_1, \ldots, \tilde{u}^s_{\ell-1}, \mu_{\ell+1}(b^s), \ldots, \mu_m(b^s))$ are used to train the approximation architecture and obtain the policy network $\hat{\mu}(., \bar{r})$. The policy network is trained with qm 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 $B_1 = (b, 1, \mu_2(b), \ldots, \mu_m(b))$. Invoking the policy network $\hat{\mu}(B_1, \bar{r})$ 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 $B_2 = (b, 2, \tilde{u}_1, \mu_3(b), \ldots, \mu_m(b))$ and invoking the policy network $\hat{\mu}(B_2, \bar{r})$. Repeating this process will produce the entire rollout control $\tilde{u} = \{\tilde{u}_1, \ldots, \tilde{u}_m\}$ (see Fig. 5).

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. ON-LINE PLAY POLICY

Given an off-line trained policy, obtained through the use of multiple approximate policy iterations, we can improve substantially its performance with an on-line play algorithm that is based on one-step or multistep lookahead minimization. In the context of our problem, this can be done by implementing on-line a truncated rollout algorithm with the off-line trained policy used as the base policy. The recent book [7] has focused on the substantial beneficial effects of on-line play built on top of off-line training, and has interpreted the on-line 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 off-line trained policy, or more accurately, its neural network representation.

An additional benefit of policy improvement by on-line play is that it works well with changing problem parameters and on-line replanning, similar to classical methods of indirect adaptive control. When the problem parameters change, the policy obtained off-line by an approximate PI algorithm is degraded, since it was trained with the old parameters. However, in the on-line 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 on-line 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 on-line play policy where controls are computed by employing an on-line

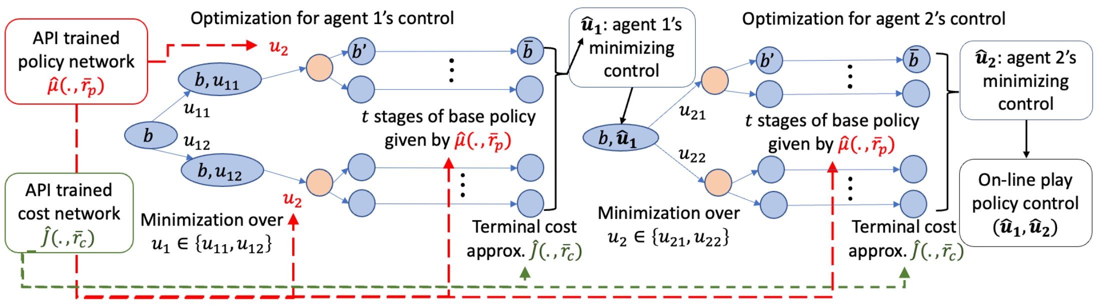

API trained policy network $\hat{\mu}(.,\bar{r}_p)$
API trained cost network $\hat{J}(.,\bar{r}_c)$
Optimization for agent 1's control
u_2
b
b, u_{11}
b, u_{12}
u_{11}
u_{12}
b'
$\bar{b}$
t stages of base policy given by $\hat{\mu}(.,\bar{r}_p)$
Minimization over $u_1 \in \{u_{11}, u_{12}\}$
Terminal cost approx. $\hat{J}(.,\bar{r}_c)$
$\hat{\mathbf{u}}_1$: agent 1's minimizing control
$b, \hat{\mathbf{u}}_1$
Optimization for agent 2's control
u_{21}
u_{22}
Minimization over $u_2 \in \{u_{21}, u_{22}\}$
$\hat{\mathbf{u}}_2$: agent 2's minimizing control
On-line play policy control $(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2)$

Fig. 6. On-line play with the one-agent-at-a-time rollout using a base policy $\hat{\mu}(,.\bar{r}_p)$ and a terminal cost approximation $\hat{J}(,.\bar{r}_c)$ for 2 agents. The control component u_2 used in the first optimization (left side) is obtained after calling the network $\hat{\mu}(B_2, \bar{r}_p)$ using feature B_2, constructed from belief state b (see Section V-B for feature construction for policy network).

one-at-a-time truncated rollout using a policy network $\hat{\mu}(.,\bar{r}_p)$ 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 off-line one-at-a-time approximate PI algorithm.

Starting from a belief state b, we perform m on-line sequential optimizations, one agent at a time, $\ell = \{1, \dots, m\}$, in that order. First, we construct the base policy's control $\mu(b) = (\mu_1(b), \dots, \mu_m(b))$ by calling the policy network $\hat{\mu}(.,\bar{r}_p)$ given by 1-at-a-time approximate PI method m times from belief state b (as described in Section V-B). The on-line 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,

$$\hat{u}_\ell \in \arg\min_{u_\ell \in U_\ell} \hat{g}(b, u') + \alpha \sum_{z \in Z} \hat{p}(z \,|\, b, u') \tilde{J}\big(F(b, u', z)\big).$$

Here, $u' = (\hat{u}_1, \dots, \hat{u}_{\ell-1}, u_\ell, \mu_{\ell+1}(b), \dots, \mu_m(b))$. The cost approximation $\tilde{J}(b')$ at belief state b' is the discounted cost of t applications of the control given by the off-line 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 $\hat{\mu}(.,\bar{r}_p)$ starting from b'. Note that the control at each of the t stages is given by calling the policy network $\hat{\mu}(.,\bar{r}_p)$, m times. Fig. 6 illustrates an on-line play algorithm with two agents. In what follows, we present an extensive simulation study of the on-line play policy on a large and complex POMDP problem. Additionally, we show that the on-line play policy largely overcomes the difficulties of off-line training in cases involving dynamically changing system parameters.

VII. Simulation Studies and Comparative Results on a Multi-Robot Repair Problem

In this section, we provide computational results and a comparative study with existing POMDP methods applied to a partially observable multi-robot 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 vs. 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 multi-robot repair problem significantly outperform existing methods, and work well where other methods fail to scale up (e.g., with 10 agents), and 4) an on-line play policy that utilizes the off-line trained policy and cost approximations during on-line 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 multi-robot repair problem.

TABLE I
Methods Used in the Comparison Study for Solving the Partially Observable Multi-Robot Repair Problem

Acronym	Name
All-at-once	Standard rollout
1-at-a-time	One-agent-at-a-time rollout
Order optimized	Order-optimized rollout
1-at-a-time API	One-agent-at-a-time approximate policy iteration
On-line play policy	On-line play policy with off-line trained approximations
POMCP [1]	Partially Observable Monte-Carlo Planning
DESPOT [3]	Determinized Sparse Partially Observable Tree
MADDPG [2]	Multi-Agent Deep Deterministic Policy Gradient

A. Multi-robot repair problem

Here, we are interested in solving a challenging multi-robot 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 $(0, 1, \dots, \nu-1)$ 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 $\lambda+1$, $\lambda = \{0, \dots, \nu-2\}$. A non-zero 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 [21]. The generalized graph topology and possible decay of a repaired location make the problem in this paper a significantly more difficult infinite horizon problem than that described

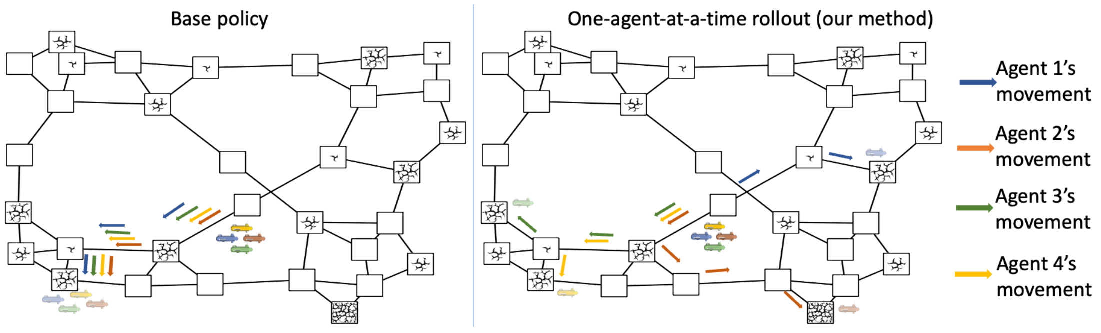


Fig. 7. Trajectories generated by a base policy (left) vs. our one-agent-at-a-time rollout policy (right) on a network with several damaged locations. Note the coordinated splitting behavior of the one-agent-at-a-time rollout policy in contrast to the base policy.

in [21]. 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 $d^v = (d_0^v, \ldots, d_{\nu-1}^v)$, 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 $d^v, \forall v \in V$.

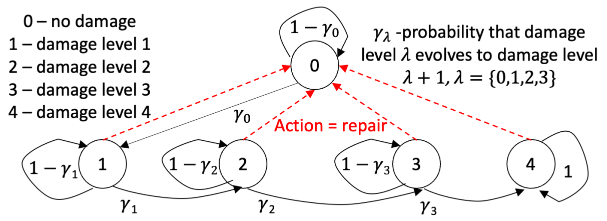


Fig. 8. Markov chain for the damage level of each network location. We use $\gamma_0 > 0$ for 8 and 10 agents, which is a generalization from [21] for a graph topology where repaired locations can fall into disrepair. We use $\gamma_0 = 0$ for all experiments with 4 agents.

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.

a) ***Simulation setup:*** We implement the multiagent rollout methods on a graph topology (shown in Fig. 1) with 32 vertices and $4, 8, 10$ 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 $\gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.05$ for all experiments in this section. We use the transition probability $\gamma_0 = 0.01$ and a discount factor $\alpha = 0.95$ for all experiments with 8 and 10 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 non-trivial 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 $\gamma_0 = 0$ and $\alpha = 0.99$ for all experiments with 4 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 towards 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 in this section are aggregated over 1000 random initial states.

b) ***Performance of multiagent rollout:*** Table II shows the cost comparison between the base policy and the one-agent-at-a-time rollout policy (with $l = 1, t = 10$ 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 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

TABLE II
COST COMPARISON OF THE BASE POLICY, AND 1-AT-A-TIME ROLLOUT POLICY ON THE MULTI-ROBOT REPAIR PROBLEM

Agent	Base policy	1-at-a-time rollout
8	5347	992
10	4667	799

(Fig. 7 left 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-agent-at-a-time rollout, the agent starting in a mildly damaged section moves to the most damaged section to assist other agents.

*c) **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 $l=1, t=10$) for policy improvement at each iteration. We use 500000 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 [39], [40]. 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 [6], Ch. 5.

Fig. 9 shows the performance of neural network policies generated by approximate PI with 8 and 10 agents. The 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.

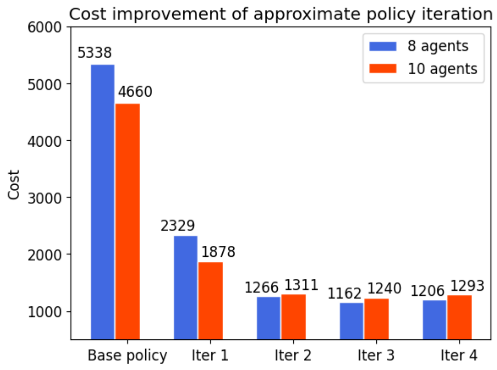


Fig. 9. Cost of base policy and several iterations of approximate PI with 1-at-a-time rollout on the multi-robot repair problem.

*d) **Performance comparison to existing methods:*** This section presents cost comparisons of the base policy, several existing methods, standard rollout, our one-agent-at-a-time rollout, and our order-optimized rollout. Note that here we cap

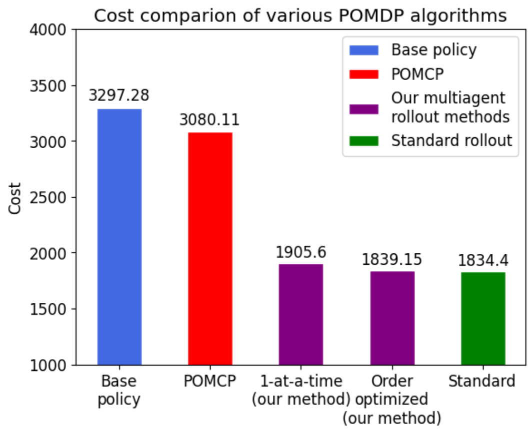


Fig. 10. Cost comparison of POMCP, base policy, rollout, and online play policy on the multi-robot repair problem with 4 agents. Our methods perform comparably well as the standard rollout with dramatically less computation (linear and polynomial computational complexity of one-agent-at-a-time rollout and order optimized rollout, respectively, in comparison with standard rollout's exponential computational complexity in the number of agents).

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 Partially Observable Monte-Carlo Planning (POMCP [1]). In fact, due to scalability issues, Determinized Sparse Partially Observable Tree (DESPOT [3]) was unsuccessful for the 4-agent problem within a reasonable time limit (1 second per stage). For the same time limit, POMCP was able to deal with the 4 agent case, but not a larger number of agents due to a combination of scalability problems involving computation time and memory requirements. Fig. 10 shows that standard rollout outperformed all other approaches, and our one-agent-at-a-time rollout methods performed comparably well as the standard rollout with dramatically less computation. The per-stage computational complexity of our one-agent-at-a-time rollout is only $O(4C)$, 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 shown in Fig. 1. This performance behavior was observed on a broad range of tests involving up to 4 agents. At the same time, the standard rollout method could not solve the problem with 8 and 10 agents due to the scalability issue. Furthermore, as expected, our order-optimized rollout (with per-stage $O(4^2C)$ computations) outperforms the one-agent-at-a-time rollout. Notably, all of our rollout algorithms outperform the base policy significantly.

We compare our methods with two existing learning methods, POMCP [1] and Multi-Agent Deep Deterministic Policy Gradient (MADDPG [2]). POMCP uses MCTS-based lookahead. For our implementation, we use default parameters for POMCP (given in [3]), 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. 10 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 used 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. 10).

e) ***On-line play policy:*** We use the policy network training using our 1-at-a-time approximate PI after iteration 4 in our on-line play policy (see the details of the policy network in Paragraph c). The cost network used in our on-line play policy 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 500000 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 on-line play, we compute the control by employing a 1-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.

The red bars of Fig. 11 show the performance of the base policy, the one-agent-at-a-time rollout (without off-line trained approximations), the off-line trained policy network using approximate PI, and the on-line play policy that uses the off-line trained policy and cost networks in the multi-robot repair problem with 4 agents. The red bars of Fig. 11 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 off-line trained policy network, by itself, is not as good as the on-line rollout with an initial base policy. This behavior can be attributed to several approximation errors in the approximate PI setting. However, using these off-line trained networks as the base policy (policy network) and the terminal cost approximation (cost network), the on-line play policy for the original Markov chain produces a cost of 1725. The on-line 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. 10) in the multi-robot repair problem with 4 agents.

We next demonstrate the robustness of the on-line play policy that provides adaptive controls when system parameters differ during the policy evaluation time from the parameters used during the off-line 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 multi-robot repair problems (4 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. 11 showcases the robustness of the on-line play policy, which is evaluated using new parameters and compares the relative performance with the original Markov chain. The policy approximation performs poorly during on-line evaluation using the new parameters since the controls come from the policy trained with the original parameters. The on-line play policy outperforms the base policy and our one-at-a-time rollout in the multi-robot repair problem (both of which do not use off-line trained architectures and are evaluated on-line with new parameters). The results show that the on-line play policy adapts to the new parameter changes with the on-line replanning while utilizing the off-line trained architectures as much as possible.

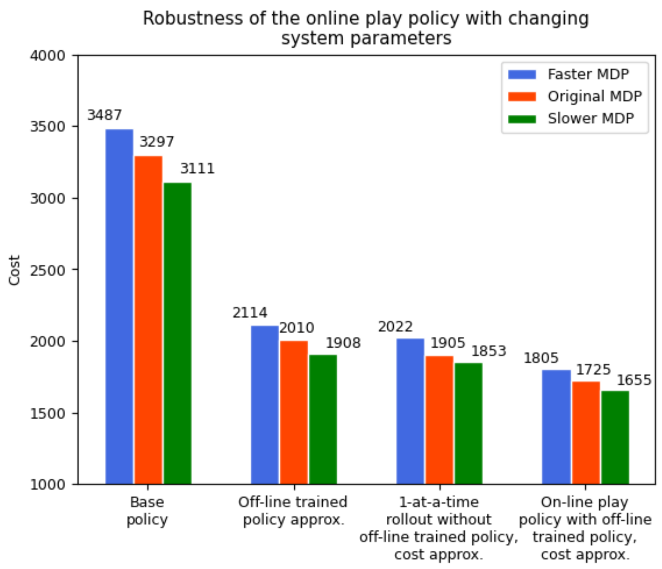


Fig. 11. Performance and robustness study of the on-line play policy on the multi-robot repair problem with 4 agents, where the damage levels evolve differently with different parameters of the Markov chain (shown in Fig. 8). The original MDP uses $\gamma_0 = 0, \gamma_1 = 0.01, \gamma_2 = 0.02, \gamma_3 = 0.03$. 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 $\gamma_0 = 0, \gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.04$. Off-line policy and cost approximations used in the on-line play policy are trained using approximate PI with the original MDP parameters.

VIII. Practical 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 state and intermittently share the computed controls.

*a) **Approximate multiagent rollout 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 approximate multiagent rollout with a signaling policy gives a control $\bar{u} = (\bar{u}_1, \dots, \bar{u}_m)$ at belief state b. Agent ℓ's control component is determined as follows,

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

where $u' = (\bar{\bar{u}}_1, \dots, \bar{\bar{u}}_{\ell-1}, u_\ell, \mu_{\ell+1}(b), \dots, \mu_m(b))$. Here, control components $\bar{\bar{u}}_1, \dots, \bar{\bar{u}}_{\ell-1}$ are given by the signaling policy, which agent ℓ uses as a proxy to the computed control components of agents $\{1, \dots, \ell-1\}$. The control $\mu(b) = (\mu_1(b), \dots, \mu_m(b))$ denotes the base policy's control at belief state b.

*b) **Approximate multiagent rollout 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 $\breve{\mu}$, 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 $\tilde{\mu}$ (using Eq. (2)) with a probability ρ when the cloud is accessible. The hybrid policy uses another policy $\underline{\mu}$ with a probability $1 - \rho$ when the computed control components are not accessible via the cloud server.

$$\breve{\mu}(b) = \begin{cases} \tilde{\mu}(b) & \text{with probability } \rho \\ \underline{\mu}(b) & \text{with probability } 1 - \rho \end{cases} \quad (4)$$

Control $\underline{\mu}(b)$ can be given by a policy that does not require the exact computed control components of the other agents. Examples of such a control $\underline{\mu}(b)$ include the base policy's control $\mu(b)$, and the approximate multiagent rollout with a signaling policy (See Eq (3)). This hybrid communication infrastructure is similar to the one analyzed in [41].

A. 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 multi-robot repair problem, we consider the state of termination when there are no more locations left to repair.

A non-termination case is more perturbing than the loss of policy improvement property since the cost of a non-terminating 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-B, 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-C, we show that termination is possible for the multi-robot 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-D, 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-agent-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-E, where belief states are shared, but the computed controls are not perfectly shared, and in Section VIII-F, where both belief states and the computed controls are imperfectly shared.

B. Failure to terminate with no communication of controls

The approximate multiagent rollout 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 approximate multiagent rollout where agents estimate other agents' controls using the base policy. The control $\bar{u} = (\bar{u}_1, \dots, \bar{u}_m)$ given by this policy at belief state b, is obtained using Eq. 3, where $u' = (\mu_1(b), \dots, \mu_{\ell-1}(b), u_\ell, \mu_{\ell+1}(b), \dots, u_m(b))$. Here $\mu(b) = (\mu_1(b), \dots, \mu_m(b))$ is the base policy's control at belief state b. The future cost is given by $\tilde{J}(F(b, u', z))$, which is the cost of applying the base policy μ until termination, starting at belief state $F(b, u', z)$.

We will use a counterexample to show that an agent executing the approximate multiagent rollout 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 counterexample, we consider a group of agents wanting to visit several partially damaged locations in a discrete space. Additionally, 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

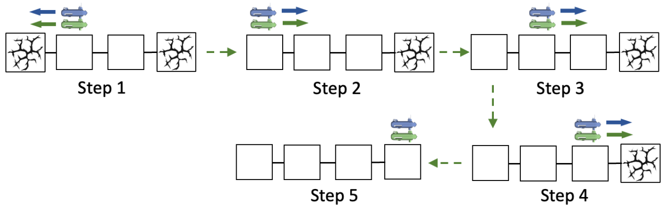


Fig. 12. Trajectory given by the base policy that moves an agent towards its nearest partially visible damaged location without communicating with the other agent. This policy takes 4 steps to visit all damaged locations.

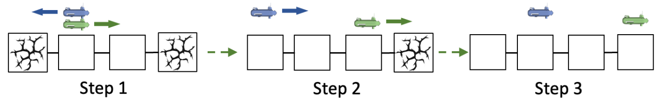


Fig. 13. Trajectory given by the one-agent-at-a-time rollout, where the second agent makes a control decision with the knowledge of the first agent's computed control component. This rollout policy takes 2 steps to visit all damaged locations.

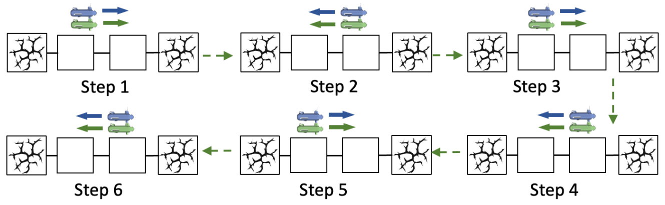


Fig. 14. Partial view of a non-terminating trajectory given by the approximate multiagent rollout using base policy signaling with no communication of controls. At each step, an agent thinks that the other agent will visit the nearest partially observable damaged location, and moves away from it. Two agents oscillate infinitely many times between the second and the third (from left) locations.

simplified instance of several real-world multiagent sequential decision problems with spatial controls, including multi-robot repair, search and rescue, and taxicab pickup problems. In the approximate multiagent rollout, 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 them will visit the nearest partially observable damaged location. This behavior can continue infinitely, thereby not terminating the problem in a finite time.

Fig. 12 and Fig. 13 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 4 vertices. Fig. 14 shows a partial view of an infinitely long trajectory given by the approximate multiagent rollout 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.

C. Finite termination without communication of controls by using randomization of the policy

In this section, we address the problem of non-termination 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 approximate multiagent rollout with base policy signaling using Eq. 3 by putting $u' = (\mu_1(b), \dots, \mu_{\ell-1}(b), u_\ell, \mu_{\ell+1}(b), \dots, \mu_m(b))$. Here $\mu(b) = (\mu_1(b), \dots, \mu_m(b))$ is the base policy's control at belief state b, and the future cost $\tilde{J}(F(b, u', z))$ is given by the cost of applying the base policy μ 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 towards strategic locations, except for rare cases where agents enter into a limit cycle with no termination, as described in Section VIII-B. For these cases, randomization helps break the oscillation and essentially re-initializes, in a random fashion, the starting points of the agents for the next decision. Fig. 15 shows possible outcomes of a single application of this randomized policy that has a positive probability of making progress towards termination with 2 agents to visit 2 partially observable damaged locations in a linear graph, where a repaired location does not fall into disrepair.

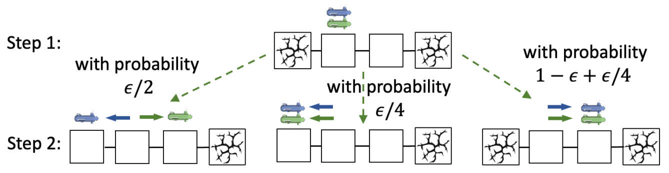


Fig. 15. Possible outcomes of the one-time application of the randomized policy on a problem with 2 agents and 2 partially observable damaged locations. The randomization ensures the agents do not get stuck into the infinite limit cycle of non-termination since every single application of the randomized policy has a positive probability of making progress towards termination in this problem.

Now, we discuss how randomization resolves the issue of non-termination in our approximate multiagent rollout 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 nodes 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 (see Eq. (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 non-zero 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 problem. We denote this D as the step-size. The state transition probability of the modified problem is given as follows,

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

where $P_{\chi,\chi'}$ denotes the probability that after starting with χ unvisited target vertices, χ' target vertices remain to be visited by applying the randomized policy (given by Eq. (5)) D times in the original problem, $\chi, \chi' = \{0, 1, \ldots, \eta\}$. We assume that a visited (and repaired) vertex cannot fall into disrepair, or equivalently $P_{\chi,\chi'} = 0, 0 \leq \chi < \chi' \leq \eta$. Hence the state transition matrix becomes an upper triangular matrix. Fig. 16 shows the reformulated Markov chain for the problem with $\eta = 3$ target vertices.

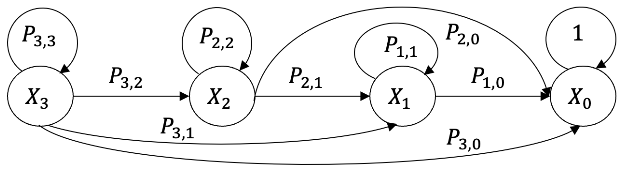


Fig. 16. The reformulated Markov chain for a problem where agents need to visit 3 target vertices. States X_3, X_2 and X_1 denote $3, 2$ and 1 target vertex(s) remain to be visited, respectively. State X_0 is the terminal state denoting that no target vertex remains to be visited. Each state transition in this Markov chain is equivalent to D applications of the randomized policy in the original problem.

Finite termination in the modified Markov chain is possible if the steady-state probability of all the non-terminal states is 0. We apply the steady-state convergence theorem [42] to show that if all non-terminal states are transient, then a unique steady-state distribution exists where the steady-state probability of all the non-terminal states is 0, and the steady-state probability of the terminal state is 1. All non-terminal states $(X_1, \ldots, X_\eta)$ are transient only if each diagonal entry corresponding to the non-terminal 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 $\mathcal{D}$, where $\mathcal{D} = \max_{v_1,v_2 \in V}$ 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 located at v_1 visits a target vertex v_2 within $\mathcal{D}$ applications of the randomized policy in the original problem is at least

$$\begin{aligned} & \sum_{\rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}} \prod_{v \in \rho_{v_1,v_2}} \frac{\epsilon}{\delta_v} \qquad \text{[where } \delta_v \text{ is the degree of vertex } v\text{]} \\ & \geq \prod_{\substack{v \in \rho_{v_1,v_2} \\ \text{where } \rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}}} \frac{\epsilon}{\delta_v} \\ & \geq \prod_{\substack{v \in \rho_{v_1,v_2} \\ \text{where } \rho_{v_1,v_2} \in \mathcal{P}_{v_1,v_2}}} \frac{\epsilon}{\max_v \delta_v} \\ & \geq \Big(\frac{\epsilon}{\max_v \delta_v}\Big)^{\mathcal{D}} \\ & > 0, \end{aligned} \tag{6}$$

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 (Eq. (5)) moves to the next vertex in the path ρ_{v_1,v_2}, adjacent to v, with a probability of at least $\frac{\epsilon}{\delta_v}$. The first 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 third inequality holds since the path length of ρ_{v_1,v_2} is $\leq \mathcal{D}$. The fourth 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 non-terminating states are transient, in conjunction with the steady-state convergence theorem [42], 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.

D. *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 approximate multiagent rollout 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 (μ) is used when the controls are not communicated. In particular, the hybrid policy (given in Eq. (4)) takes the following form,

$$\breve{\mu}(b) = \begin{cases} \tilde{\mu}(b) & \text{with probability } \rho \\ \mu(b) & \text{with probability } 1-\rho \end{cases} \tag{7}$$

The control $\tilde{\mu}(b) = (\tilde{\mu}_1(b), \dots, \tilde{\mu}_m(b))$ is given by the one-agent-at-a-time rollout policy using Eq. (2) at belief state b. In the pure form, we replace the cost function $\tilde{J}(F(b,u',z)$ in the right side of Eq. (2) by $J_\mu(F(b,u',z)$. $J_\mu(F(b,u',z)$ is the cost of applying the base policy μ 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, which takes the following form.

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

where $u' = (\tilde{\mu}_1(b), \dots, \tilde{\mu}_{\ell-1}(b), u_\ell, \mu_{\ell+1}(b), \dots, \mu_m(b))$.

Fig. 17 shows the hybrid policy $\breve{\mu}$ applied to a problem where 2 agents want to visit 2 partially observable damaged locations at the two ends of a linear graph of 4 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 $\breve{\mu}$ needs $2\rho + 4(1-\rho)$ time steps to visit both locations; faster than the base policy, which takes 4 time steps since $\rho > 0$.

Now, we show that the approximate multiagent rollout 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 $\breve{\mu}$ (defined in Eq. (7)), we have

$$J_{\breve{\mu}}(b) \le J_\mu(b) \quad \forall b.$$

Here $J_{\breve{\mu}}(b)$ and $J_\mu(b)$ denote the costs of applying the hybrid policy $\breve{\mu}$ and the base policy μ, starting at belief state b until termination, respectively.

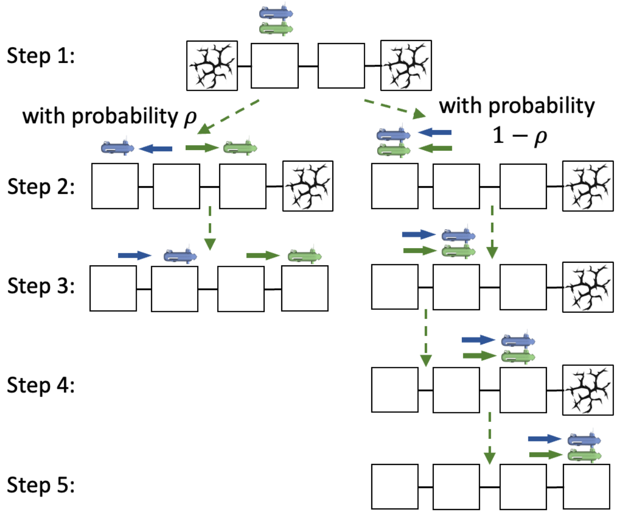


Fig. 17. Possible outcomes of the hybrid policy ($\breve{\mu}$) with intermittent communication of controls with 2 agents and 2 partially observable damaged locations. All damaged locations are visited in $2\rho+4(1-\rho)$ time steps which is sooner than the base policy that takes 4 time steps to visit all the damaged locations (See Figure 12).

We first define the Bellman operator T. For a cost function J, policy μ.

$$(T_\mu J)(b) = \hat{g}(b,\mu(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\mu(b)) J(F(b,\mu(b),z))$$

$\forall b$. The cost of policy μ, at belief state b, can be expressed recursively,

$$J_\mu(b) = \hat{g}(b,\mu(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\mu(b)) J_\mu(F(b,\mu(b),z)), \forall b.$$

Alternatively, $J_\mu(b)$ can be given in terms of the Bellman operator T,

$$J_\mu(b) = \lim_{K\to\infty} (T_\mu^K J)(b), \forall b.$$

$(T_\mu^K J)(b)$ is the discounted sum of stage costs starting at belief state b, where the Bellman operator T for policy μ is applied K times, followed by the cost approximation J.

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

In order to show $(T_{\breve{\mu}} J_\mu)(b) \le J_\mu(b)$, we expand $(T_{\breve{\mu}} J_\mu)(b)$, the Bellman operator for the policy $\breve{\mu}$ and the cost approximation J_μ. For all belief states b, we have the following.

$$\begin{aligned}
&(T_{\breve{\mu}} J_\mu)(b) && \text{(9a)}\\
&= \hat{g}(b,\breve{\mu}(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\breve{\mu}(b)) J_\mu(F(b,\breve{\mu}(b),z)) && \text{(9b)}\\
&= \rho\Big[\hat{g}(b,\tilde{\mu}(b)) + \alpha \sum_{z\in Z} \hat{p}(z|b,\tilde{\mu}(b)) J_\mu(F(b,\tilde{\mu}(b),z))\Big] \\
&\quad + (1-\rho)\Big[\hat{g}(b,\mu(b)) && \text{(9c)}\\
&\quad + \alpha \sum_{z\in Z} \hat{p}(z|b,\mu(b)) J_\mu(F(b,\mu(b),z))\Big]
\end{aligned}$$

$$
\begin{aligned}
&= \rho \min_{u_2 \in U_2} \Big[\hat{g}(b, (\tilde{\mu}_1(b), u_2)) \\
&\quad + \alpha \sum_{z \in Z} \hat{p}(z|b, (\tilde{\mu}_1(b), u_2)) J_\mu(F(b, (\tilde{\mu}_1(b), u_2), z))\Big] \quad (9d)\\
&\quad + (1-\rho) J_\mu(b) \\
&\leq \rho \Big[\hat{g}(b, (\tilde{\mu}_1(b), \mu_2(b))) \\
&\quad + \alpha \sum_{z \in Z} \hat{p}(z|b, (\tilde{\mu}_1(b), \mu_2(b))) J_\mu(F(b, (\tilde{\mu}_1(b), \mu_2(b)), z))\Big] \\
&\quad + (1-\rho) J_\mu(b) \quad (9e)\\
&= \rho \min_{u_1 \in U_1} \Big[\hat{g}(b, (u_1, \mu_2(b))) \\
&\quad + \alpha \sum_{z \in Z} \hat{p}(z|b, (u_1, \mu_2(b))) J_\mu(F(b, (u_1, \mu_2(b)), z))\Big] \quad (9f)\\
&\quad + (1-\rho) J_\mu(b) \\
&\leq \rho \Big[\hat{g}(b, (\mu_1(b), \mu_2(b))) \\
&\quad + \alpha \sum_{z \in Z} \hat{p}(z|b, (\mu_1(b), \mu_2(b))) J_\mu(F(b, (\mu_1(b), \mu_2(b)), z))\Big] \\
&\quad + (1-\rho) J_\mu(b) \quad (9g)\\
&= \rho J_\mu(b) + (1-\rho) J_\mu(b) \quad (9h)\\
&= J_\mu(b) \quad (9i)
\end{aligned}
$$

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

From the above derivation, we get $(T_{\breve{\mu}} J_\mu)(b) \leq J_\mu(b), \forall b$. This inequality gives us the following relations.

$$
\begin{aligned}
(T_{\breve{\mu}}^K J_\mu)(b) =&(T_{\breve{\mu}}^{K-1})(T_{\breve{\mu}} J_\mu)(b) \\
\leq&(T_{\breve{\mu}}^{K-1} J_\mu)(b) = (T_{\breve{\mu}}^{K-2})(T_{\breve{\mu}} J_\mu)(b) \\
\leq&(T_{\breve{\mu}}^{K-2} J_\mu)(b) = (T_{\breve{\mu}}^{K-3})(T_{\breve{\mu}} J_\mu)(b) \\
\leq&(T_{\breve{\mu}}^{K-3} J_\mu)(b) \\
\leq&\ldots \leq (T_{\breve{\mu}} J_\mu)(b) \\
\leq&J_\mu(b).
\end{aligned}
$$

The cost of the policy $\breve{\mu}$ is,

$$
J_{\breve{\mu}}(b) = \lim_{K \to \infty} (T_{\breve{\mu}}^K J_\mu)(b) \leq J_\mu(b), \forall b.
$$

Next, we talk about the numerical results for our rollout methods with various imperfect communication architectures on the multi-robot repair problem.

E. Results on the multi-robot repair problem with imperfect communication of controls (shared belief states)

This section considers the case where agents cannot communicate their choice of control with one another, 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 without the need of a signaling policy, it is computationally expensive for a decentralized architecture where agents need to repeat work. Instead, using a signaling policy helps achieve speed up by leveraging parallel control computation. In the next section, we discuss cases where both belief states and controls are not shared perfectly.

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 $\gamma_1 = 0.02, \gamma_2 = 0.03, \gamma_3 = 0.05$. We consider a repaired location may evolve to the next damage level for 8 and 10 agents. We use $\gamma_0 = 0.01$ and $\alpha = 0.95$ for $8, 10$ agents and $\gamma_0 = 0$ and $\alpha = 0.99$ for 4 agents. We use $l = 1, t = 10$ and the steady-state terminal cost approximation for all approximate multiagent rollout methods. Table III summarizes different architectures involving imperfect communication strategies used in our simulation study. Each approximate multiagent rollout (AMR) method's name contains an abbreviation of the signaling policy, which the agents use to estimate other agents' computed control components. "I" represents the intermittent cloud communication case, and "LC" represents the local communication case.

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 policy iteration signaling	Policy net. of the best policy iter.	None
AMR-LC	AMR with local communication	Base policy for non-local agents	Local
AMR-ILC	AMR with local and intermittent communication	Base policy w/o cloud and local connectivity	Local + 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

We consider several communication architectures with different signaling policies and study their effect on the resulting performance of our multiagent rollout methods where belief states are shared but controls are not perfectly shared.

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 policy iterations (possibly the best iteration), and the base policy is given by the previous policy iteration.

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 trade-off since it takes advantage of a rich centralized information source whenever possible and uses clustered local communication when the server is unreachable.

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

Methods	4 agents	8 agents	10 agents
Base policy	3297	5347	4667
1-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

Table IV presents performance results of different communication architectures with imperfect control sharing (but perfect belief sharing) and compares them with our one-agent-at-a-time rollout with perfect communication for $4, 8$ and 10 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-B). AMR-N performs better since it uses a neural network to approximate the one-agent-at-a-time rollout policy, which is then used to estimate predecessor agent controls. The performance of this approach improves with more accurate policy networks. 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 $r = 2$ in our experiments) and can exhibit poor behavior if coordination is required across agent clusters (i.e., the damage is in a distant part of the environment). This variant is problem dependent and assumes agents can always communicate controls perfectly with other agents within r hops. This may not be a practical assumption in other multiagent POMDP problems. AMR-ILC performs best among all other approximate multiagent rollout approaches for a large number of agents (8 and 10 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. This suggests that our multiagent rollout methods can produce intelligent policies starting from a simple base policy even with an imperfect communication setting.

F. Results on the multi-robot 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-E. Each approximate multiagent rollout (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 w/o optimization (AMR-IB0):*** 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 (i.e., without estimating or taking into consideration the actions of the other agents).

Table V shows the cost comparison between the base policy, one-agent-at-a-time rollout policy, and different architectures

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

Agent	Base policy	1-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

involving intermittent communication with imperfect belief and control sharing for $4, 8$, and 10 agents. We observe that all the multiagent rollout methods under the intermittent communication assumption improve over the base policy, and the cost improves with a better connection probability ρ. With a high probability of $\rho \to 1$, the methods perform similar to the one-agent-at-a-time rollout, attaining the same behavior as the one-agent-at-a-time rollout when $\rho = 1$. With a low probability of $\rho \to 0$, we observe that the methods produce similar costs 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. This method has less chance of missing some damage location than AMR-IB1 and gives better cost. AMR-IB0 is similar to the hybrid policy discussed in Section VIII-D (Section VIII-D discusses AMR-IB0 with perfect belief sharing and intermittent control sharing). The favorable result of AMR-IB0 can be partially attributed to the policy improvement property that exists in the perfect belief sharing case.

Discussion: The simulation 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.

IX. CONCLUSION

In this paper, we present various multiagent rollout methods, approximate PI and on-line play policy with off-line 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 on-line play policy enhances the solution quality of off-line trained architectures, especially for problems with changing system parameters, by providing adaptive controls. We also present extensions of our multiagent rollout methods, analytically justify their performance guarantees, and report numerical performance results for the imperfect communication case. Based on our experimentation, 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.

ACKNOWLEDGMENTS

We thank the reviewers who provided useful comments, to colleagues who contributed to the ideas, and we gratefully acknowledge partial funding through the ONR Young Investigators Award grant number 325 28741 124363 323199 0001 45514, Sloan Research Fellowship grant number FG-2020-13998 and Amazon Research Award.

REFERENCES

[1] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Proc. 23rd International Conf. on NeurIPS*, Red Hook, NY, USA, 2010, pp. 2164–2172.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *Neural Information Processing Systems (NIPS)*, 2017.

[3] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP Planning with Regularization," in *Advances in NeurIPS 26*, 2013, pp. 1772–1780.

[4] D. P. Bertsekas, "Multiagent Rollout Algorithms and Reinforcement Learning," *arXiv preprint arXiv:1910.00120, for an extended version, see IEEE/CAA Journal of Automatica Sinica, Vol. 8, 2021, pp. 249-271*, April 2020.

[5] ——, *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Belmont, MA: Athena Scientific, 2020.

[6] ——, *Reinforcement Learning and Optimal Control*. Belmont, MA: Athena Scientific, 2019.

[7] ——, "Lessons from alphazero for optimal, model predictive, and adaptive control," *arXiv:2108.10315*, 2022.

[8] 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 *Proceedings of the 2020 Conference on Robot Learning*, vol. 155. PMLR, 2021, pp. 1814–1828. [Online]. Available: https://proceedings.mlr.press/v155/bhattacharya21a.html

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

[10] N. Meuleau, K. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving POMDPs by Searching the Space of Finite Policies," *CoRR*, vol. abs/1301.6720, 2013. [Online]. Available: http://arxiv.org/abs/1301.6720

[11] R. Zhou and E. A. Hansen, "An Improved Grid-Based Approximation Algorithm for POMDPs," in *Proc. of the 17th International Joint Conf. on Artificial Intelligence*, San Francisco, CA, USA, 2001, p. 707–714.

[12] H. Yu and D. Bertsekas, "Discretized Approximations for POMDP with Average Cost," *arXiv preprint arXiv:1207.4154*, 2012.

[13] J. Baxter and P. L. Bartlett, "Infinite-Horizon Policy-Gradient Estimation," *J. of AI Res.*, vol. 15, pp. 319–350, 2001.

[14] H. Yu, "A Function Approximation Approach to Estimation of Policy Gradient for POMDP with Structured Policies," *arXiv preprint arXiv:1207.1421*, 2012.

[15] R. M. Estanjini, K. Li, and I. C. Paschalidis, "A least squares temporal difference actor-critic algorithm with applications to warehouse management," *Naval Research Logistics*, vol. 59, pp. 197–211, 2012.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *International conference on machine learning*, 2016, pp. 1928–1937.
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.
[19] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned POMDPs," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.
[20] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.
[21] 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 Robotics and Automation Letters*, vol. 5, no. 3, pp. 3967–3974, 2020.
[22] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.
[23] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of Science Robotics," *Science Robotics*, vol. 3, 2018.
[24] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," in *2010 International Conf. on Emerging Security Technologies*, 2010, pp. 142–147.
[25] A. Cassandra, "A Survey of POMDP Applications," *Working Notes of AAAI 1998 Fall Symposium on Planning with POMDP*, 1998.
[26] S. M. Veres, L. Molnar, N. K. Lincoln, and C. P. Morice, "Autonomous vehicle control systems - a review of decision making," *Proc. of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, pp. 155–195, 2011.
[27] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus, "Experiments with Cooperative Control of Underwater Robots," *IJRR*, vol. 28, pp. 815–833, 2009.
[28] J. Das, F. Py, J. B. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *IJRR*, vol. 34, pp. 1435–1452, 2015.
[29] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.
[30] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," in *2018 IEEE/RSJ International Conf. IROS*, 2018, pp. 3052–3059.
[31] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning," *IJRR*, vol. 35, pp. 1289–1307, 2016.
[32] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *IJRR*, vol. 37, pp. 405–420, 2018.
[33] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.
[34] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, and J. P. How, "Decentralized control of Partially Observable Markov Decision Processes using belief space macro-actions," in *IEEE ICRA*, Seattle, WA, 2015, pp. 5962–5969.
[35] S. Yi, C. Nam, and K. Sycara, "Indoor Pursuit-Evasion with Hybrid Hierarchical Partially Observable Markov Decision Processes for Multi-robot Systems," in *Distributed Autonomous Robotic Systems*, vol. 9, 2019, pp. 251–264.
[36] S. Gil, S. Kumar, D. Katabi, and D. Rus, "Adaptive Communication in Multi-Robot Systems Using Directionality of Signal Strength," *IJRR*, vol. 34, pp. 946–968, 2015.
[37] W. Wang, N. Jadhav, P. Vohs, N. Hughes, M. Mazumder, and S. Gil, "Active Rendezvous for Multi-Robot Pose Graph Optimization using Sensing over Wi-Fi," *arXiv preprint arXiv:1907.05538*, 2019.
[38] S. Gil, D. Feldman, and D. Rus, "Communication Coverage for Independently Moving Robots," in *2012 IEEE/RSJ International Conf. on IROS*, Vilamoura, 2012, pp. 4865–4872.
[39] M. G. Lagoudakis and R. Parr, "Reinforcement Learning as Classification: Leveraging Modern Classifiers," in *Proc. 20th ICML*, 2003, pp. 424–431.
[40] C. Dimitrakakis and M. G. Lagoudakis, "Rollout Sampling Approximate Policy Iteration," *Mach. Learn.*, vol. 72, pp. 157–171, 2008.
[41] M. Yemini, S. Gil, and A. Goldsmith, "Exploiting local and cloud sensor fusion in intermittently connected sensor networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.
[42] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to probability*. Athena Scientinis, 2000.

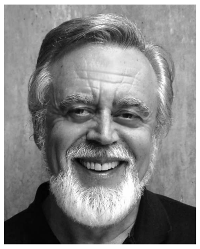
X. Biography Section

Sushmita Bhattacharya is a Computer Science Ph.D. student in the School of Engineering and Applied Sciences at Harvard University, advised by Prof. Stephanie Gil. She is a part of the Robotics, Embedded Autonomy and Communication Theory (REACT) Lab at Harvard University. Her research focuses on multi-robot sequential decision-making and distributed learning under uncertainty.

Siva Kailas is an M.S. student in the Robotics Institute at Carnegie Mellon University and is a member of the Advanced Agent-Robotics Technology Laboratory. His current research focuses on multi-agent systems, including information gathering, task allocation, and path planning. He received a B.S. in Computer Science at Arizona State University, where he was a part of the Robotics, Embedded Autonomy and Communication Theory (REACT) Laboratory.

Sahil Badyal is currently working as a Data Scientist at Capital One with about four years of experience in machine learning and data science. He also works as a research aide in the REACT Lab, where the focus of his work is Multiagent Reinforcement Learning. He is a computer science enthusiast, interested in RL, robot-learning, multi-modal machine learning, and natural language processing (NLP).

Stephanie Gil is an Assistant Professor in the Computer Science Department at the School of Engineering and Applied Sciences at Harvard University where she directs the Robotics, Embedded Autonomy and Communication Theory (REACT) Lab. Prior she was an Assistant Professor at Arizona State University. Her research focuses on multi-robot systems where she studies the impact of information exchange and communication on resilience and trusted coordination. She is the recipient of the 2019 Faculty Early Career Development Program Award from the National Science Foundation (NSF CAREER), and has been selected as a 2020 Alfred P. Sloan Fellow. She obtained her PhD from the Massachusetts Institute of Technology in 2014.



Dimtri Bertsekas undergraduate studies were in engineering at the National Technical University of Athens, Greece. He obtained his MS in electrical engineering at the George Washington University, Wash. DC in 1969, and his Ph.D. in system science in 1971 at the Massachusetts Institute of Technology.

Dr. Bertsekas has held faculty positions with the Engineering-Economic Systems Dept., Stanford University (1971-1974) and the Electrical Engineering Dept. of the University of Illinois, Urbana (1974-1979). From 1979 to 2019 he was with the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (M.I.T.), where he served as McAfee Professor of Engineering. In 2019, he was appointed Fulton Professor of Computational Decision Making, and a full time faculty member at the School of Computing and Augmented Intelligence at Arizona State University, Tempe, while maintaining a research position at MIT. His research spans several fields, including optimization, control, large-scale computation, and data communication networks, and is closely tied to his teaching and book authoring activities. He has written numerous research papers, and eighteen books and research monographs, several of which are used as textbooks in MIT classes. Most recently Dr Bertsekas has been focusing on reinforcement learning, and authored a textbook in 2019, and a research monograph on its distributed and multiagent implementation aspects in 2020.

Professor Bertsekas was awarded the INFORMS 1997 Prize for Research Excellence in the Interface Between Operations Research and Computer Science for his book “Neuro-Dynamic Programming” (co-authored with John Tsitsiklis), the 2000 Greek National Award for Operations Research, the 2001 AACC John R. Ragazzini Education Award, the 2009 INFORMS Expository Writing Award, the 2014 AACC Richard E. Bellman Control Heritage Award for “contributions to the foundations of deterministic and stochastic optimization-based methods in systems and control,” the 2014 INFORMS Khachiyan Prize for Life-Time Accomplishments in Optimization, the SIAM/MOS 2015 George B. Dantzig Prize, and the 2022 IEEE Control Systems Award. In 2018, he was awarded, jointly with his coauthor John Tsitsiklis, the INFORMS John von Neumann Theory Prize, for the contributions of the research monographs “Parallel and Distributed Computation” and “Neuro-Dynamic Programming”. In 2001, he was elected to the United States National Academy of Engineering for “pioneering contributions to fundamental research, practice and education of optimization/control theory, and especially its application to data communication networks.”

Dr. Bertsekas’ recent books are “Introduction to Probability: 2nd Edition” (2008), “Convex Optimization Theory” (2009), “Dynamic Programming and Optimal Control,” Vol. I, (2017), and Vol. II: (2012), “Abstract Dynamic Programming” (2018), “Convex Optimization Algorithms” (2015), “Reinforcement Learning and Optimal Control” (2019), and “Rollout, Policy Iteration, and Distributed Reinforcement Learning”(2020), all published by Athena Scientific.