Decentralized Control of Partially Observable Markov Decision Processes using Belief Space Macro-actions

Shayegan Omidshafiei, Ali-akbar Agha-mohammadi, Christopher Amato, Jonathan P. How

***Abstract*— The focus of this paper is on solving multi-robot planning problems in continuous spaces with partial observability. Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) are general models for multi-robot coordination problems, but representing and solving Dec-POMDPs is often intractable for large problems. To allow for a high-level representation that is natural for multi-robot problems and scalable to large discrete and continuous problems, this paper extends the Dec-POMDP model to the Decentralized Partially Observable Semi-Markov Decision Process (Dec-POSMDP). The Dec-POSMDP formulation allows asynchronous decision-making by the robots, which is crucial in multi-robot domains. We also present an algorithm for solving this Dec-POSMDP which is much more scalable than previous methods since it can incorporate closed-loop belief space macro-actions in planning. These macro-actions are automatically constructed to produce robust solutions. The proposed method's performance is evaluated on a complex multi-robot package delivery problem under uncertainty, showing that our approach can naturally represent multi-robot problems and provide high-quality solutions for large-scale problems.**

I. Introduction

Many real-world multi-robot coordination problems operate in continuous spaces where robots possess partial and noisy sensors. In addition, asynchronous decision-making is often needed due to stochastic action effects and the lack of perfect communication. The combination of these factors makes control very difficult. Ideally, high-quality controllers for each robot would be automatically generated based on a high-level domain specification. In this paper, we present such a method for both formally representing multi-robot coordination problems and automatically generating local planners based on the specification. While these local planners can be a set of hand-coded controllers, we also present an algorithm for automatically generating controllers that can then be sequenced to solve the problem. The result is a principled method for coordination in probabilistic multi-robot domains.

The most general representation of the multi-robot coordination problem is the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [8]. Dec-POMDPs have a broad set of applications including networking problems, multi-robot exploration, and surveillance [9], [10], [20], [21]. Unfortunately, current Dec-POMDP solution methods are limited to small discrete domains and require synchronized decision-making. This paper extends promising recent work on incorporating macro-actions, temporally

S. Omidshafiei, A. Agha, and J.P. How are with the Laboratory for Information and Decision Systems (LIDS), MIT, Cambridge, MA. C. Amato is with the Dept. of Computer Science at the University of New Hampshire, Durham, NH. Work for this paper was completed while all authors were at MIT. {shayegan,aliagha,camato,jhow}@mit.edu

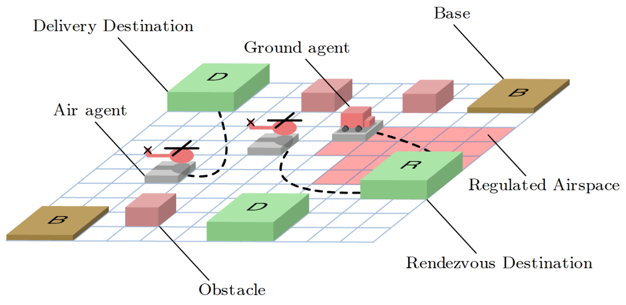

Fig. 1. Package delivery domain with key elements labeled.

extended actions, [4], [5] to solve continuous and large-scale problems which were infeasible for previous methods.

Macro-actions (MAs) have provided increased scalability in single-agent MDPs [18] and POMDPs [1], [11], [14], but are nontrivial to extend to multi-agent settings. Some of the challenges in extending MAs to decentralized settings are:

- In the decentralized setting, synchronized decision-making is problematic (or even impossible) as some robots must remain idle while others finish their actions. The resulting solution quality would be poor (or not implementable), resulting in the need for MAs that can be chosen asynchronously by the robots (an issue that has not been considered in the single-agent literature).
- Incorporating principled asynchronous MA selection is a challenge, because it is not clear how to choose optimal MAs for one robot while other robots are still executing. Hence, a novel formal framework is needed to represent Dec-POMDPs with asynchronous decision-making and MAs that may last varying amounts of time.
- Designing these variable-time MAs also requires characterizing the stopping time and probability of terminating at every goal state of the MAs. Novel methods are needed that can provide this characterization.

MA-based Dec-POMDPs alleviate the above problems by no longer attempting to solve for a policy at the primitive action level, but instead considering temporally-extended actions, or MAs. This also addresses scalability issues, as the size of the action space is considerably reduced.

In this paper, we extend the Dec-POMDP to the Decentralized Partially Observable *Semi*-Markov Decision Process (Dec-POSMDP) model, which formalizes the use of closed-loop MAs. The Dec-POSMDP represents the theoretical basis for asynchronous decision-making in Dec-POMDPs. We also automatically design MAs using graph-based planning techniques. The resulting MAs are closed-loop and the completion time and success probability can be characterized analytically, allowing them to be directly integrated into the Dec-POSMDP framework. As a result, our framework can generate efficient decentralized plans which take advantage of estimated completion times to permit asynchronous decision-making. The proposed Dec-POSMDP framework enables solutions for large domains (in terms of state/action/observation space) with long horizons, which are otherwise computationally intractable to solve. We leverage the Dec-POSMDP framework and design an efficient discrete search algorithm for solving it, and demonstrate the performance of the method for the complex problem of multi-robot package delivery under uncertainty (Fig. 1).

II. Problem Statement

A Dec-POMDP [8] is a sequential decision-making problem where multiple agents (e.g., robots) operate under uncertainty based on different streams of observations. At each step, every agent chooses an action (in parallel) based purely on its local observations, resulting in an immediate reward and an observation for each individual agent based on stochastic (Markovian) models over continuous states, actions, and observation spaces.

We define a notation aimed at reducing ambiguities when discussing single agents and multi-agent teams. A generic parameter p related to the i-th agent is noted as $p^{(i)}$, whereas a joint parameter for a team of n agents is noted as $\bar{p} = (p^{(1)}, p^{(2)}, \cdots, p^{(n)})$. Environment parameters or those referring to graphs are indicated without parentheses, for instance p^i refers to a parameter of a graph node, and p^{ij} to a parameter of a graph edge.

Formally, the Dec-POMDP problem we consider in this paper is described by the following elements:

- $\mathbb{I} = \{1, 2, \cdots, n\}$ is a finite set of agents' indices.
- $\bar{\mathbb{S}}$ is a continuous set of joint states. Joint state space can be factored as $\bar{\mathbb{S}} = \bar{\mathbb{X}} \times \mathbb{X}^e$ where $\mathbb{X}^e$ denotes the environmental state and $\bar{\mathbb{X}} = \times_i \mathbb{X}^{(i)}$ is the joint state space of robots, with $\mathbb{X}^{(i)}$ being the state space of the i-th agent. $\mathbb{X}^{(i)}$ is a continuous space. We assume $\mathbb{X}^e$ is a finite set.
- $\bar{\mathbb{U}}$ is a continuous set of joint actions, which can be factored as $\mathbb{U} = \times_i \mathbb{U}^{(i)}$, where $\mathbb{U}^{(i)}$ is the set of actions for the i-th agent.
- State transition probability density function is denoted as $p(\bar{s}'|\bar{s}, \bar{u})$, that specifies the probability density of transitioning from state $\bar{s} \in \bar{\mathbb{S}}$ to $\bar{s}' \in \bar{\mathbb{S}}$ when the actions $\bar{u} \in \bar{\mathbb{U}}$ are taken by the agents.
- $\bar{R}$ is a reward function: $\bar{R} : \bar{\mathbb{S}} \times \bar{\mathbb{U}} \to \mathbb{R}$, the immediate reward for being in joint state $\bar{x} \in \bar{\mathbb{X}}$ and taking the joint action $\bar{u} \in \bar{\mathbb{U}}$.
- $\bar{\Omega}$ is a continuous set of observations obtained by all agents. It is factored as $\bar{\Omega} = \bar{\mathbb{Z}} \times \bar{\mathbb{Z}}^e$, where $\bar{\mathbb{Z}} = \times_i \mathbb{Z}^{(i)}$ and $\bar{\mathbb{Z}}^e = \times_i \mathbb{Z}^{e(i)}$. The set $\mathbb{Z}^{(i)} \times \mathbb{Z}^{e(i)}$ is the set of observations obtained by the i-th agent. $\mathbb{Z}^{e(i)}$ is the observation that is a function of the environmental state $x^e \in \mathbb{X}^e$. We assume the set of environmental observations $\mathbb{Z}^{e(i)}$ is a finite set for any agent i.
- Observation probability density function $h(\bar{o}|\bar{s}, \bar{u})$ encodes the probability density of seeing observations $\bar{o} \in \bar{\Omega}$ given joint action $\bar{u} \in \bar{\mathbb{U}}$ is taken which resulted in joint state $\bar{s} \in \bar{\mathbb{S}}$.

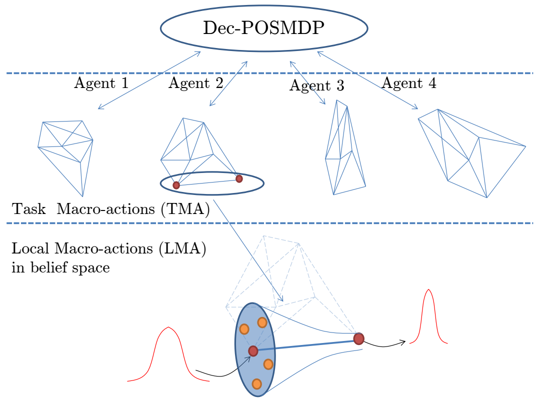

Fig. 2. Hierarchy of the proposed planner. In the highest level, a decentralized planner assigns a TMA to each robot. Each TMA encompasses a specific task (e.g. picking up a package). Each TMA in turn is constructed as a set of local macro-actions (LMAs). Each LMA (the lower layer) is a feedback controller that acts as a funnel. LMAs funnel a large set of beliefs to a small set of beliefs (termination belief of the LMA).

Note that a general Dec-POMDP need not have a factored state space such as the one given here.

The solution of a Dec-POMDP is a collection of decentralized policies $\bar{\eta} = (\eta^{(1)}, \eta^{(2)}, \cdots, \eta^{(n)})$. Because (in general) each agent does not have access to the observations of other agents, each policy $\eta^{(i)}$ maps the individual data history (obtained observations and taken actions) of the i-th agent into its next action: $u_t^i = \eta^{(i)}(H_t^{(i)})$, where $H_t^{(i)} = \{o_1^{(i)}, u_1^{(i)}, o_2^{(i)}, u_2^{(i)}, \cdots, o_{t-1}^{(i)}, u_{t-1}^{(i)}, o_t^{(i)}\}$, where $o^{(i)} \in \mathbb{Z}^{(i)}$. Note that the final observation $o_t^{(i)}$ after action $u_{t-1}^{(i)}$ is included in the history.

According to above definition, we can define the value associated with a given policy $\bar{\eta}$ starting from an initial joint state distribution $\bar{b}$:

$$V^{\bar{\eta}}(\bar{b}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \bar{R}(\bar{s}_t, \bar{u}_t) | \bar{\eta}, p(\bar{s}_0) = \bar{b}\right] \quad (1)$$

Then, a solution to a Dec-POMDP formally can be defined as the optimal policy:

$$\bar{\eta}^* = \arg\max_{\bar{\eta}} V^{\bar{\eta}} \quad (2)$$

The Dec-POMDP problem stated in (2) is undecidable over continuous spaces without additional assumptions. In discrete settings, recent work has extended the Dec-POMDP model to incorporate macro-actions which can be executed in an asynchronous manner [5]. In planning with MAs, decision making occurs in a two layer manner (see Fig. 2). A higher-level policy will return a MA for each agent and the selected MA will return a primitive action to be executed. This approach is an extension of the *options framework* [18] to multi-agent domains while dealing with the lack of synchronization between agents. The options framework is a formal model of MAs [18] that has been very successful in aiding representation and solutions in single robot domains [12], [14]. Unfortunately, this method requires a full, discrete model of the system (including macro-action policies of all agents, sensors and dynamics).

As an alternative, we propose a Dec-POSMDP model which only requires a high-level model of the problem. The Dec-POSMDP provides a high-level discrete planning formalism which can be defined on top of continuous spaces. As such, we can approximate the continuous multi-robot coordination problems with a tractable Dec-POSMDP formulation. Before defining our more general Dec-POSMDP model, we first discuss the form of MAs that allow for efficient planning within our framework.

III. Hierarchical Graph-based Macro-actions

This section introduces a mechanism to generate complex MAs based on a graph of lower-level simpler MAs, which is a key point in solving Dec-POSMDPs without explicitly computing success probabilities, times, and rewards of MAs in the decentralized planning level. We refer to the generated complex MAs as Task MAs (TMAs).

To clarify the concepts, before describing task macro-actions, we distinguish between open-loop and closed-loop MAs. In general, MAs refer to temporally-extended actions [19]. An l-step long open-loop MA is a sequence of predefined actions such as by $u_{0:l} = \{u_0, u_1, \cdots, u_l\}$. However, a closed-loop MA is policy $\pi(\cdot)$, i.e., a mapping from histories to actions. As we discuss in the next section, for a seamless incorporation of MAs in Dec-POMDP planning, we need to design closed-loop MAs, which is a challenge in partially-observable settings.

Generating a closed-loop MA that accomplishes a single-agent task such as picking up an object, delivering a package, opening a door, and so on, itself requires solving a POMDP problem. In this paper, we utilize information roadmaps [1] as a substrate to generate such task MAs. We start by discussing the structure of feedback controllers in partially-observable domains.

A macro-action $\pi^{(i)}$ for the i-th agent maps the histories $H^{\pi^{(i)}}$ of actions and observations that have occurred to the next action the agent should take, $u = \pi^{(i)}(H^{\pi^{(i)}})$. Note that the environmental observations are only obtained when the MA terminates. We compress this history into a belief $b^{(i)} = p(x^{(i)}|H^{\pi^{(i)}})$, with joint belief for the team denoted by $\bar{b} = (b^{(1)}, b^{(2)}, \cdots, b^{(n)})$. It is well known [13] that making decisions based on belief $b^{(i)}$ is equivalent to making decisions based on the history $H^{\pi^{(i)}}$ in a POMDP.

For any given agent, a feedback controller in partially-observable environment comprises a Bayesian filter $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ that evolves the belief and a separated controller $u_{k+1} = \mu(b_{k+1})$ that generates control signals based on the current belief (figure). Therefore, a feedback controller $\mathcal{L}$ in belief space can be viewed as a function that maps the current belief b_k, control u_k, and observation z_{k+1} to the pair of next belief b_{k+1} and control u_{k+1}; i.e., $(b_{k+1}, u_{k+1}) = \mathcal{L}(b_k, u_k, z_{k+1}) = (\tau(b_k, u_k, z_{k+1}), \mu(\tau(b_k, u_k, z_{k+1})))$.

A local MA we consider herein is a feedback controller that is effective (has basin of attraction) locally in a region of the state/belief space. Many controllers that rely on linearization fall into this category as the linearization assumption is valid in locally around the linearization point. The goal of LMA in the partially-observable setting is to drive the system's belief to a particular belief. In [1] it has been shown that in Gaussian belief space, utilizing a combination of Kalman filter and linear controllers, the system's belief can be steered toward certain probability distributions. In other words, LMAs act like a funnel in belief space. Starting from a belief in the mouth of funnel (see Fig. 2), the LMA drives the belief toward a target belief that is referred to as a *milestone* herein.

Since LMAs act locally on belief space, we can locally linearize the system and design corresponding simple LMAs. In this paper, we assume the belief space is Gaussian and thus belief can be represented with a mean vector $\hat{x}^+$ and covariance matrix P^+ denoted as $b \equiv (\hat{x}^+, P^+)$. For a given state point $\mathbf{v}$, we linearize the nonlinear process and measurement equations to get a stationary linear system with Gaussian noises. Associated with this linear system, we design a stationary Kalman filter and a linear separated controller, $\mu(b) = -L(\hat{x}^+ - \mathbf{v})$. Thus, the utilized LMA is parametrized by feedback gain matrix L and point $\mathbf{v}$; i.e., $\mu(b; \theta)$, where $\theta = (L, \mathbf{v})$. It can be shown that under appropriate choice of L and mild observability conditions, this Linear LMA acts as a funnel in belief space that drives the belief toward the milestone $\check{b} \equiv (\check{x}, \check{P})$, where $\check{x} = \mathbf{v}$ and $\check{P}$ is the solution of the Riccati equation corresponding to the Kalman filter [1].

A chain of funnels is a sequence of funnels where the target belief of each funnel falls into the mouth (or pre-image) of the next funnel in the chain. A richer way of combining funnels is via graphication (to form a graph of funnels). An information roadmap is defined as a graph of funnels, where each node of this graph is a milestone and each edge is an LMA funnel.

To construct a graph of LMAs, we sample a set of parameters $\{\theta^j\}$ and generate corresponding LMAs $\{\mathcal{L}^j\}$. Associated with the j-th LMA, we compute the j-th milestone $\check{b}^j$. We define the j-th node of our LMA graph as an ϵ-neighborhood around the milestone; i.e., $B^j = \{b : \|b - \check{b}^j\| \leq \epsilon\}$ and the set of all nodes as $\mathbb{V} = \{B^j\}$. We connect B^j to its k-nearest neighbors via their corresponding LMAs. If neighboring nodes i and j are so far from each other that the j-th LMA $\mathcal{L}^j$ cannot take the belief from i to j (since the linearization used to construct $\mathcal{L}^j$ is not valid around B^i), we utilize an edge controller (as detailed in [1]). An edge controller is a finite-time controller whose role is to take the mean of distribution close enough to the node B^j via tracking a trajectory that connects $\mathbf{v}^i$ to $\mathbf{v}^j$ in the state space. Once the distribution mean gets close enough to the target node, the system's control is handed over to the funnel associated with the target node. We denote the concatenation of the edge controller and the funnel utilized to take the belief from B^i to B^j by $\mathcal{L}^{ij}$, which is defined as the (i,j)-th graph edge. The set of all edges are denoted by $\mathbb{L} = \{\mathcal{L}^{ij}\}$. We denote the set of available LMAs at B^i by $\mathbb{L}(i)$. To incorporate the lower-level state constraints (e.g., obstacles) and control constraints, we consider B^0 as a hypothetical node, hitting which represents violation of constraints. We add B^0 to the set of nodes $\mathbb{V}$. Therefore, taking any $\mathcal{L}^{ij}$ there is a chance that system ends up in B^0.

We can simulate the behavior of LMA $\mathcal{L}^{ij}$ at B^i offline and compute the probability of landing in any given node B^r, which is denoted by $P(B^r|B^i,\mathcal{L}^{ij})$. Similarly, we can compute the reward of taking LMA $\mathcal{L}^{ij}$ at B^i offline, which is denoted by $R(B^i,\mathcal{L}^{ij})$ and defined as the sum of one-step rewards under this LMA. Finally, by $\mathcal{T}^{ij}=\mathcal{T}(B^i,\mathcal{L}^{ij})$ we denote the time it takes for LMA $\mathcal{L}^{ij}$ to complete its execution starting from B^i.

A. Utilizing TMAs in the Decentralized Setting

In a decentralized setting, the following properties of the macro-action need to be available to the high-level decentralized planner: *(i)* TMA value from any given initial belief, *(ii)* TMA completion time from any given belief, and *(iii)* TMA success probability from any given belief. What makes computing these properties challenging is the requirement that they need to be calculated for *every* possible initial belief. Every belief is needed because when one agent's TMA terminates, the other agents might be in any belief while still continuing to execute their own TMA. This information about the progress of agents' TMAs is needed for nontrivial asynchronous TMA selection.

In the following, we discuss how the graph-based structure of our proposed TMAs allows us to compute a closed-form equation for the success probability, value, and time. As a result, when evaluating the high-level decentralized policy, these values can be efficiently retrieved for any given start and goal states. This is particularly important in decentralized multi-agent planning since the state/belief of the j-th agent is not known a priori when the MA of i-th agent terminates.

A TMA policy is defined as a policy that is found by performing dynamic programming on the graph of LMAs. Consider a graph of LMAs that is constructed to perform a simple task such as open-the-door, pick-up-a-package, move-a-package, etc. An important feature of this graph is that it is multi-query, meaning that it is valid for any starting and goal belief. Depending on the goal belief of the task, we can solve the dynamic programming problem on the LMA graph that leads to a policy which achieves the goal while trying to maximize the accumulated reward and taking into account the probability of hitting failure set B^0. Formally, we need to solve the following DP:

$$V^*(B^i,\pi^*)=\max_{\mathcal{L}\in\mathbb{L}(i)}\left\{R(B^i,\mathcal{L})+\sum_j P(B^j|B^i,\mathcal{L})V^*(B^j)\right\},\forall i \tag{3}$$

$$\pi^*(B^i)=\arg\max_{\mathcal{L}\in\mathbb{L}(i)}\left\{R(B^i,\mathcal{L})+\sum_j P(B^j|B^i,\mathcal{L})V^*(B^j)\right\},\forall i$$

where $V^*(\cdot)$ is the optimal value defined over the graph nodes with $V(B^{goal})$ set to zero and $V(B^0)$ set to a suitable negative reward for violating constraints. $\pi^*(\cdot)$ is the resulting TMA. The primitive actions can be retrieved from TMA via a two-stage computation: TMA picks the best LMA at each milestone and LMA generates the next action based on the perceived observations until belief reaches the next milestone; i.e., $u_{k+1}=\mathcal{L}(b_k,u_k,z_{k+1})=\pi^*(B)(b_k,u_k,z_{k+1})$ where B is the last visited milestone and $\mathcal{L}=\pi^*(B)$ is the best LMA chosen by TMA at milestone B. The space of TMAs is denoted as $\mathbb{T}=\{\pi\}$.

For a given optimal TMA π^*, the associated optimal value $V^*(B^i,\pi^*)$ from any node B^i is computed via solving (3). Also, using Markov chain theory we can analytically compute the probability $P(B^{goal}|B^i,\pi^*)$ of reaching the goal node B^{goal} under the optimal TMA π^* starting from any node B^i in the offline phase [1].

Similarly, we can compute the time it takes for the TMA to go from B^i to B^{goal} under any TMA π as follows:

$$\begin{aligned}T^g(B^i;\pi)&=\mathcal{T}(B^i,\pi(B^i))\\&+\sum_j P(B^j|B^i,\pi(B^i))T^g(B^j;\pi),\ \forall i\end{aligned} \tag{4}$$

where $T^g(B;\pi)$ denotes the time it takes for TMA π to take the system from B to TMA's goal. Defining $\mathcal{T}^i=\mathcal{T}(B^i,\pi(B^i))$ and $\bar{\mathcal{T}}=(\mathcal{T}^1,\mathcal{T}^2,\cdots,\mathcal{T}^n)^T$ we can write (4) in its matrix form as:

$$\bar{T}^g=\bar{\mathcal{T}}+\bar{P}\bar{T}^g\Rightarrow\bar{T}^g=(I-\bar{P})^{-1}\bar{\mathcal{T}} \tag{5}$$

where $\bar{T}^g$ is a column vector with i-th element equal to $T^g(B^i;\pi)$ and $\bar{P}$ is a matrix with (i,j)-th entry equal to $P(B^j|B^i,\pi(B^i))$.

Therefore, a TMA can be used in a higher-level planning algorithm as a MA whose success probability, execution time, and reward can be computed offline.

B. Environmental State and Updated Model

We also extend TMAs to the multi-agent setting where there is an environmental state that is locally observable by agents and can be affected by other agents.

We denote the environment state (e-state) at the k-th time step as $x^e_k\in\mathbb{X}^e$. It encodes the information in the environment that can be manipulated and observed by different agents. We assume x^e_k is only locally (partially) observable. An example for x^e_k in the package delivery application (presented in Section V) is "there is a package in the base". An agent can only get this measurement if the agent is in the base (hence it is partial).

Any given TMA π is only available at a subset of e-states, denoted by $\mathbb{X}^e(\pi)$. In many applications $\mathbb{X}^e(\pi)$ is a small finite set. Thus, we can extend the cost and transition probabilities of TMA π for all $x^e\in\mathbb{X}^e$ by performing the TMA evaluation described in Section III-A for all $x^e\in\mathbb{X}^e$.

We extend transition probabilities $P(B^{goal}|b,\pi)$ to take the e-state into account, i.e., $P(B^{goal},x^{e'}|b,x^e,\pi)$, which denotes the probability of getting to the goal region B^{goal} and e-state $x^{e'}$ starting from belief b and e-state x^e under the TMA policy π. Similarly, the TMA's value function $V(b,\pi)$ is extended to $V(b,x^e,\pi)$, for all $x^e\in\mathbb{X}^e(\pi)$.

The joint reward $\bar{R}(\bar{x},x^e,\bar{u})$ encodes the reward obtained by the entire team, where $\bar{x}=(x^{(1)},\cdots,x^{(n)})$ is the set of states for different agents and $\bar{u}=(u^{(1)},\cdots,u^{(n)})$ is the set of actions taken by all agents.

We assume the joint reward is a multi-linear function of a set of reward functions $\{R^{(1)},\cdots,R^{(n)}\}$ and R^E, where $R^{(i)}$ only depends on the i-th agent's state and R^E depends on all the agents. In other words, we have:

$$\begin{aligned}\bar{R}(\bar{x},x^e,\bar{u})=g\Big(&R^{(1)}(x^{(1)},x^e,u^{(1)}),R^{(2)}(x^{(2)},x^e,u^{(2)}),\\&\cdots,R^{(n)}(x^{(n)},x^e,u^{(n)}),R^E(\bar{x},x^e,\bar{u})\Big)\end{aligned} \tag{6}$$

In multi-agent planning domains, often computing R^E is computationally less expensive than computing $\bar{R}$, which is the property we exploit in designing the higher-level decentralized algorithm.

The joint reward $\bar{R}(\bar{b}, x^e, \bar{u})$ encodes the reward obtained by the entire team, where $\bar{b} = (b^{(1)}, \cdots, b^{(n)})$ is the joint belief and $\bar{u}$ is the joint action defined previously.

Similarly, the joint policy $\bar{\phi} = \{\phi^{(1)}, \cdots, \phi^{(n)}\}$ is the set of all decentralized policies, where $\phi^{(i)}$ is the decentralized policy associated with the i-th agent. In the next section, we discuss how these decentralized policies can be computed based on the Dec-POSMDP formulation.

Joint value $\bar{V}(\bar{b}, x^e_0, \bar{\phi})$ encodes the value of executing the collection $\bar{\phi}$ of decentralized policies starting from e-state x^e_0 and initial joint belief $\bar{b}$.

IV. The Dec-POSMDP Framework

In this section, we formally introduce the Dec-POSMDP framework. We discuss how to transform a continuous Dec-POMDP to a Dec-POSMDP using a finite number of TMAs, allowing discrete domain algorithms to generate a decentralized solution for general continuous problems.

We denote the high-level decentralized policy for the i-th agent by $\phi^{(i)} : \Xi^{(i)} \to \mathbb{T}^{(i)}$, where $\Xi^{(i)}$ is the macro-action history for the i-th agent (as opposed to the action-observation history), which is formally defined as:

$$\Xi^{(i)}_k = (o^{e(i)}_1, \pi^{(i)}_1, H^{(i)}_1, o^{e(i)}_2, \pi^{(i)}_2, H^{(i)}_2, \ldots, \pi^{(i)}_{k-1}, H^{(i)}_{k-1}, o^{e(i)}_k)$$

which includes the chosen macro-actions $\pi^{(i)}_{1:k-1}$, the action-observation histories under chosen macro-actions $H^{(i)}_{1:k-1}$, and the environmental observations $o^{e(i)}_{1:k}$ received at the termination of macro-actions.

Accordingly, we can define a joint policy $\bar{\phi} = (\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(n)})$ for all agents and a joint macro-action policy as $\bar{\pi} = (\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(n)})$.

Each time an agent completes a TMA, it receives an observation of the e-state x^e, denoted by o^e. Also, due to the special structure of the proposed TMAs, we can record the agent's final belief b^f at TMA's termination (which compresses the entire history H under that TMA). We denote this pair as $\breve{o} = (b^f, o^e)$. As a result, we can compress the macro-action history as $\Xi^{(i)}_k = \{\breve{o}^{(i)}_1, \pi^{(i)}_1, \breve{o}^{(i)}_2, \pi^{(i)}_2, \cdots, \breve{o}^{(i)}_{k-1}, \pi^{(i)}_{k-1}, \breve{o}^{(i)}_k\}$.

The value of joint policy $\bar{\phi}$ is

$$\bar{V}^{\bar{\phi}}(\bar{b}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \bar{R}(\bar{s}_t, \bar{u}_t) | \bar{\phi}, \{\bar{\pi}\}, p(\bar{s}_0) = \bar{b}\right], \quad (7)$$

but it is unclear how to evaluate this equation without a full (discrete) low-level model of the domain. Even in that case, it would often be intractable to calculate the value directly. Therefore, we will formally define the Dec-POSMDP problem, which has the same goal as the Dec-POMDP problem (finding the optimal policy), but in this case we seek the optimal policy for choosing macro-actions in our semi-Markov setting:

$$\bar{\phi}^* = \arg\max_{\bar{\phi}} \bar{V}^{\bar{\phi}}(\bar{b}) \quad (8)$$

Definition 1: **(Dec-POSMDP)** The Dec-POSMDP framework is described by the following elements

- $\mathbb{I} = \{1, 2, \cdots, n\}$ is a finite set of agents' indices.
- $\mathbb{B}^{(1)} \times \mathbb{B}^{(2)} \times \ldots \times \mathbb{B}^{(n)} \times \mathbb{X}^e$ is the underlying state space for the proposed Dec-POSMDP, where $\mathbb{B}^{(i)}$ is the set of belief milestones of i-th agent's TMAs (i.e., $\mathbb{T}^{(i)}$).
- $\mathbb{T} = \mathbb{T}^{(1)} \times \mathbb{T}^{(2)} \ldots \times \mathbb{T}^{(n)}$ is the space of high-level actions in Dec-POSMDP, where $\mathbb{T}^{(i)}$ is the set of TMAs for the i-th agent.
- $P(\bar{b}', x^{e'}, k | \bar{b}, x^e, \bar{\pi})$ denotes the transition probability under joint TMA policy $\bar{\pi}$ from a given $\bar{b}, x^e$ to $\bar{b}', x^{e'}$ as described below.
- $\bar{R}^\tau(\bar{b}, x^e, \bar{\pi})$ denotes the reward/value of taking TMA $\bar{\pi}$ at $\bar{b}, x^e$ as described below.
- $\breve{\mathbb{O}}$ is the set of environmental observations.
- $P(\breve{o} | \bar{b}, x^e)$ denotes the observation likelihood model.

Again, a general Dec-POSDMP need not have a factored state space. Also, while the state space is very large, macro-actions allow much of it to be ignored once these high-level transition, reward and observation functions are calculated.

Below, we describe the above elements in more detail. For further explanation and derivations, please see [17]. The planner $\bar{\phi} = \{\phi^{(1)}, \phi^{(2)}, \cdots, \phi^{(n)}\}$ that we construct in this section is fully decentralized in the sense that each agent i has its own policy $\phi^{(i)} : \Xi^{(i)} \to \mathbb{T}^{(i)}$ that generates the next TMA based on the history of TMAs taken and the observation perceived solely by the i-th agent.

Each policy $\phi^{(i)}$ is a discrete controller, represented by a (policy) graph [3], [15]. Each node of the discrete controller corresponds to a TMA. Note that different nodes in the controller could use the same TMA. Each edge in this graph is an $\breve{o}$. An example discrete controller for a package delivery domain is illustrated in Fig. 3.

Consider a joint TMA $\bar{\pi} = (\pi^{(1)}, \pi^{(2)}, \cdots, \pi^{(n)})$ for the entire team. Incorporating terminal conditions for different agents, we can evaluate the set of decentralized TMAs until at least one of them stops as:

$$\bar{R}^\tau(\bar{b}, x^e, \bar{\pi}) = \mathbb{E}\left[\sum_{t=0}^{\bar{\tau}_{min}} \gamma^t \bar{R}(\bar{x}_k, x^e_k, \bar{u}_k) | \bar{\pi}, p(\bar{x}_0) = \bar{b}, x^e_0 = x^e\right]$$

$$\text{where} \qquad \bar{\tau}_{min} = \min_i \min_t \{t : b^{(i)}_t \in B^{(i),goal}\} \quad (9)$$

It can be shown that the probability of transitioning between two configurations (from $\bar{b}, x^e$ to $\bar{b}', x^{e'}$) after k steps under the joint TMA $\bar{\pi}$ is given by:

$$\begin{aligned} &P(\bar{b}', x^{e'}, k | \bar{b}, x^e, \bar{\pi}) = P(x^{e'}_k, \bar{b}'_k | x^e_0, \bar{b}_0, \bar{\pi}) \\ &= \sum_{x^e_{k-1}, \bar{b}_{k-1}} \Big[P(x^{e'}_k | x^e_{k-1}, \bar{\pi}(\bar{b}_{k-1})) \times \\ &P(\bar{b}'_k | x^e_{k-1}, \bar{b}_{k-1}, \bar{\pi}(\bar{b}_{k-1})) P(x^e_{k-1}, \bar{b}_{k-1} | x^e_0, \bar{b}_0, \bar{\pi}) \Big] \quad (10) \end{aligned}$$

Joint value $\bar{V}^{\bar{\phi}}(\bar{b}, x^e)$ then encodes the value of executing the collection $\bar{\phi}$ of decentralized policies starting from e-state x^e_0 and initial joint belief $\bar{b}_0$. The below equation describes the value transformation from the primitive actions to TMAs, which is vital for allowing us to efficiently perform evaluation. Details of this derivation can be found in [17].

$$\begin{aligned}\bar{V}^{\bar{\phi}}(\bar{b}_0, x^e_0) &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \bar{R}(\bar{x}_t, x^e_t, \bar{u}_t)|\bar{\phi}, \bar{b}_0, x^e_0\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{t_k} \bar{R}^{\tau}(\bar{b}_{t_k}, x^e_{t_k}, \bar{\pi}_{t_k})|\bar{\phi}, \bar{b}_0, x^e_0\right] \quad (11)\end{aligned}$$

where $t_k = \min_i \min_t\{t > t_{k-1} : b^{(i)}_t \in B^{(i),goal}\}$ and $\bar{\pi} = \bar{\phi}(\bar{b}, x^e)$.

The dynamic programming formulation corresponding to the defined joint value function over MAs is:

$$\bar{V}^{\bar{\phi}}(\bar{b}, x^e) = \bar{R}^{\tau}(\bar{b}, x^e, \bar{\pi}) + \sum_{k=0}^{\infty}\gamma^{t_k}\sum_{\bar{b}', x^{e'}} P(\bar{b}', x^{e'}, k|\bar{b}, x^e, \bar{\pi})\bar{V}^{\bar{\phi}}(\bar{b}', x^{e'}) \quad (12)$$

The critical reduction from the continuous Dec-POMDP to the Dec-POSMDP over a finite number of macro-actions is a key factor in solving large Dec-POMDP problems. In the following section we discuss how we compute a decentralized policy based on the Dec-POSMDP formulation.

A. Masked Monte Carlo Search (MMCS)

In this section, we propose an efficient and easy-to-implement method for solving the Dec-POSMDP, referred to as Masked Monte Carlo Search (MMCS). As demonstrated in Section V, MMCS allows extremely large multi-agent problems to be solved. It uses an informed Monte Carlo policy sampling scheme to achieve this, exploiting results from previous policy evaluations to narrow the search space.

Because the infinite-horizon problem is undecidable [16], infinite-horizon methods typically focus on producing approximate solutions given a computational budget [15], [3]. A Monte Carlo approach can be implemented by repeatedly randomly sampling from the policy space and retaining the policy with the highest expected value as an approximate solution. The search can be stopped at any point and the latest iteration of the approximate solution can be utilized.

The MMCS algorithm is detailed in Alg. 1. MMCS uses a discrete controller to represent the policy, $\phi^{(i)}$, of each agent. The nodes of the discrete controller are TMAs, $\pi \in \mathbb{T}$, and edges are observations, $\breve{o} \in \breve{\mathbb{O}}$. Each agent i transitions in the discrete controller by completing a TMA π_k, seeing an observation $\breve{o}_k$, and following the appropriate edge to the next TMA, π_{k+1}. Fig. 3 shows part of a single agent's policy.

MMCS initially generates a set of valid policies by randomly sampling the policy space (Line 13), while adhering to the constraint that the termination milestone of a TMA node in the discrete controller must intersect the set of initiation milestones of its child TMA node. The joint value, $\bar{V}^{\bar{\phi}}(\bar{b}, x^e)$, of each policy given an initial joint belief $\bar{b}$ and e-state x^e is calculated using Monte Carlo simulations (Line 15).

MMCS identifies the TMA transitions that occur most often in the set of K-best policies, in a process called 'masking' (Line 17). It includes these transitions in future iterations of the policy search by explicitly checking for the existence of a mask for that transition, preventing the transition from being re-sampled (Line 12). Note that the mask is not permanent, as re-evaluation of the mask using the K-best policies occurs in each iteration (Line 17).

Algorithm 1: MMCS

```
1  Procedure : MMCS($\mathbb{T}, \breve{\mathbb{O}}, \mathbb{I}, K$)
2  input : TMA space $\mathbb{T}$, environmental observation space
     $\breve{\mathbb{O}}$, agents $\mathbb{I}$, number of best policies to check $K$
3  output : joint policy $\bar{\phi}_{MMCS}$
4  foreach agent $i \in \mathbb{I}$ do
5      $masked^{(i)} \leftarrow$ setToFalse();
6      $\phi^{(i)}_{MMCS} \leftarrow null$;
7  for $iter_{MMCS} = 1$ to $iter_{max,MMCS}$ do
8      for $iter_{MC} = 1$ to $iter_{max,MC}$ do
9          $\bar{\phi}_{new} \leftarrow \bar{\phi}_{MMCS}$;
10         foreach agent $i \in \mathbb{I}$ do
11             foreach $(\pi^{(i)}, \breve{o}^{(i)}) \in \mathbb{T} \times \breve{\mathbb{O}}$ do
12                 if not $masked^{(i)}(\pi^{(i)}, \breve{o}^{(i)})$ then
13                     $\phi^{(i)}_{new}(\pi^{(i)}, \breve{o}^{(i)}) \leftarrow$ sample($\mathbb{T}(\pi^{(i)}, \breve{o}^{(i)})$);
14         $\bar{\phi}_{list}$.append($\bar{\phi}_{new}$);
15         $\bar{V}^{\bar{\phi}}_{list}(\bar{b}, x^e)$.append(evalPolicy($\bar{\phi}_{new}$));
16     $\bar{\phi}_{list} \leftarrow$ getBestKPolicies($\bar{\phi}_{list}, \bar{V}^{\bar{\phi}}_{list}(\bar{b}, x^e), K$);
17     $(masked, \bar{\phi}_{MMCS}) \leftarrow$ createMask($\bar{\phi}_{list}, \bar{V}^{\bar{\phi}}_{list}(\bar{b}, x^e)$);
18 return $\bar{\phi}_{MMCS}$;
```

The above process is repeated until a computational budget is reached, after which the best policy, $\bar{\phi}_{MMCS}$, is selected (Line 17). MMCS offers the advantage of balancing exploration and exploitation of the policy search space. Though 'masking' places focus on promising policies, it is done in conjunction with random sampling, allowing previously unexplored regions of the policy space to be sampled.

MMCS is designed for efficient search of the policy space for a high-quality policy. The algorithm can be extended to allow probabilistic guarantees on policy quality by allowing probabilistic resampling (in Line 13) based on the history of policy values, rather than using a binary mask. Error bounds on joint policy $\bar{\phi}$ could then be adopted from [6] as follows:

$$P[\bar{V}^{\bar{\phi}^*}(\bar{b}) - \bar{V}^{\bar{\phi}}(\bar{b}) \geq \epsilon] \leq \delta \quad (13)$$

That is, MMCS can be extended straightforwardly to ensure that with probability of at least $1-\delta$ we can construct controllers with value within ϵ of optimal (for a fixed size).

V. Experiments

In this section, we consider a package delivery under uncertainty scenario involving a team of heterogeneous robots, an application which has recently received particular attention [2], [7]. The overall objective in this problem is to retrieve and deliver packages from base locations to delivery locations using a group of robots.

Though our method is general and can be used for many decentralized planning problems, this domain was chosen due to its extreme complexity. For instance, the experiments conducted use a policy controller with 13 nodes, resulting in a policy space with cardinality $5.622e{+}17$ in the package delivery domain, making it computationally intractable to solve. Additional challenges stem from the presence of different sources of uncertainty (wind, actuator, sensor), obstacles and constraints in the environment, and different types of tasks (pick-up, drop-off, etc.). Also, the presence of joint tasks such as joint pickup of large packages introduces a significant multi-robot coordination component. This problem is formulated as a Dec-POMDP in continuous space, and hence current Dec-POMDP methods are not applicable. Even if we could modify the domain to allow their use, discretizing the state/action/observation space in this problem would lead to poor solution quality or computational intractability.

Fig. 1 illustrates the package delivery domain. Robots are classified into two categories: air vehicles (quadcopters) and ground vehicles (trucks). Air vehicles handle pickup of packages from bases, and can also deliver packages to two delivery locations, $Dest_1$ and $Dest_2$. An additional delivery location $Dest_r$ exists in a regulated airspace, where air vehicles cannot fly. Packages destined for $Dest_r$ must be handed off to a ground vehicle at a rendezvous location. The ground vehicle is solely responsible for deliveries in this regulated region. Rewards are given to the team only when a package is dropped off at its correct delivery destination.

Packages are available for pickup at two bases in the domain. Each base contains a maximum of one package, and a stochastic generative model is used for allocating packages to bases. Each package has a designated delivery location, $\delta \in \Delta = \{d_1, d_2, d_r\}$. Additionally, each base has a size descriptor for its package, $\psi \in \Psi = \{\varnothing, 1, 2\}$, where $\psi = \varnothing$ indicates no package at the base, $\psi = 1$ indicates a small package, and $\psi = 2$ indicates a large package. Small packages can be picked up by a single air vehicle, whereas large packages require cooperative pickup by two air vehicles. The descriptors of package destinations and sizes will implicitly impact the policy of the decentralized planner.

To allow coordination of cooperative TMAs, an e-state stating the availability of nearby vehicles, $\phi \in \Phi = \{0, 1\}$, is observable at any base or at the rendezvous location, where $\phi = 1$ signifies that another robot is at the same milestone (or location) as the current robot and $\phi = 0$ signifies that all other robots are outside the current robot's milestone.

A robot at any base location can, therefore, observe the e-state $x^e = (\psi, \delta, \phi) \in \Psi \times \Delta \times \Phi$, which contains details about the availability and the size of the package at the base (if it exists), the delivery destination of the package, and availability of nearby robots (for performing cooperative tasks). A robot at the rendezvous location can observe $x^e = \phi \in \Phi$ for guidance of rendezvous TMAs.

We assume one ground robot and two air robots are used, with two base locations $Base_1$ and $Base_2$. Air robots are initially located at $Base_1$ and the ground robot is at $Dest_r$. If $b^{(i)} \in Base_j$, we say the i-th robot is at the j-th base.

The available TMAs in this domain are:

- Go to base $Base_j$ for $j \in \{1, 2\}$
  - *robots involved:* 1 air robot.
  - *Initiation set:* Air robot i is available, where $i \in \{i_{a,1}, i_{a,2}\}$.
  - *Termination set:* robot i available and its belief is $b^i \in B^{h_j}$.
- Go to delivery destination $Dest_j$ for $j \in \{1, 2, r\}$
  - *robots involved:* 1 (any type)
  - *Initiation set:* robot i available, can be at any location.
  - *Termination set:* robot i available and its belief is $b^i \in B^{d_j}$.
- Joint go to delivery destination $Dest_j$ for $j \in \{1, 2\}$
  - *robots involved:* 2 air robots.
  - *Initiation set:* Air robots $i_{a,1}$ and $i_{a,2}$ are available.
  - *Termination set:* robots $i_{a,1}$ and $i_{a,2}$ are available and their beliefs are $b^{i_{a,1}} \in B^{d_j}$, $b^{i_{a,2}} \in B^{d_j}$, where $j \in \{1, 2\}$.
- Pick up package
  - *robots involved:* 1 air robot
  - *Initiation set:* Air robot i is available, where $i \in \{I_{a,1}, I_{a,2}\}$. $x^e = \{\psi = 1, \delta \in \Delta, \phi \in \Phi\}$
  - *Termination set:* Ground robot i is carrying package and is unavailable.
- Joint pick up package
  - *robots involved:* 2 air robots.
  - *Initiation set:* Air robots $i_{a,1}$ and $i_{a,2}$ are available. $x^e = \{\psi = 2, \delta \in \Delta, \phi = exact\}$
  - *Termination set:* robots $i_{a,1}$ and $i_{a,2}$ are carrying package and are unavailable.
- Put down package
  - *robots involved:* 1 ground robot or 1 air robot.
  - *Initiation set:* robot $i \in I$ is carrying package, $b^i \in B^{d_j}$, where $j \in \{1, 2\}$
  - *Termination set:* robot i is available.
- Joint put down package
  - *robots involved:* 2 air robots.
  - *Initiation set:* Air robots $i_{a,1}$ and $i_{a,2}$ are available and jointly carrying a package. $b^{i_1} \in B^{d_j}$ and $b^{i_2} \in B^{d_j}$, where $j \in \{1, 2\}$.
  - *Termination set:* robots $i_{a,1}$ and $i_{a,2}$ are available and their beliefs are $b^{i_{a,1}} \in B^{d_j}$, $b^{i_{a,2}} \in B^{d_j}$, where $j \in \{1, 2\}$.
- Go to rendezvous location
  - *robots involved:* 1 ground robot or 1 air robot
  - *Initiation set:* robot $i \in I$ is available.

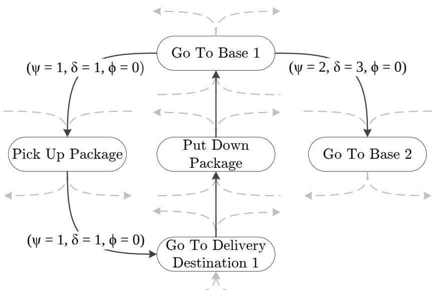

Fig. 3. Partial view of a single robot's policy obtained using MMCS for the package delivery domain. In this discrete policy controller, nodes represent TMAs and edges represent e-states. Greyed out edges represent connections to additional nodes which have been excluded to simplify the figure.

  - *Termination set:* robot i is available and its belief is $b^i \in B^{d_r}$.
- Place package on truck
  - *robots involved:* 1 air robot, 1 ground robot.
  - *Initiation set:* Air robot $i_a \in \{I_{a,1}, I_{a,2}\}$ and ground robot i_g are available and located at rendezvous location, where $b^{i_a} \in B^{d_r}$ and $b^{i_a} \in B^{d_r}$.
  - *Termination set:* robots i_a and i_g are available and their beliefs are $b^{i_a} \in B^{d_j}$, $b^{i_g} \in B^{d_j}$, where $j \in \{1, 2\}$.
- Wait at current location
  - *robots involved:* 1 ground robot or 1 air robot.
  - *Initiation set:* robot $i \in I$ is available.
  - *Termination set:* robot i is available.

Each TMA is defined with an associated initiation and termination set. For instance, "Go to $Base_j$" TMA is defined:

- *Robots involved:* 1 air robot.
- *Initiation set:* An air robot i is available, as indicated by e-state ϕ at the current belief milestone.
- *Termination set:* robot i available and $b^{(i)} \in Base_j$.

The robot's belief and the e-state affect the TMA's behavior. For instance, the behavior of the "Go to $Base_j$" TMA will be affected by the presence of obstacles for different robots. For details on the definition of other TMAs, see [17].

To generate a closed-loop policy corresponding to each TMA, we follow the procedure explained in Section III. For example, for the "Go to $Base_j$" TMA, the state of system consists of the air robot's pose (position and orientation). Thus, the belief space for this TMA is the space of all possible probability distributions over the system's pose. To follow the procedure in Section III, we incrementally sample beliefs in the belief space, design corresponding LMAs, generate a graph of LMAs, and use dynamic programming on this graph to generate the TMA policy. Fig. 4 shows performance of an example TMA ("Go to $Dest_1$"). The results show that the TMA is optimized to achieve the goal in the manner that produces the highest value, but its performance is robust to noise and tends to minimize the constraint violation probability. A key observation is that the value function is available over the entire space. The same is true for the success probability and completion time of the TMA. This information can then be directly used by MMCS to perform evaluation of policies that use this macro-action.

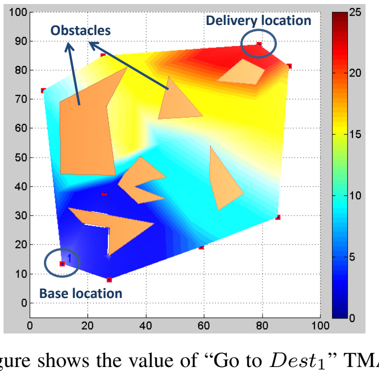

Fig. 4. This figure shows the value of "Go to $Dest_1$" TMA over its belief space (only the 2D mean part is shown).

A portion of a policy for a single air robot in the package delivery domain is illustrated in Fig. 3. The policy involves going to $Base_1$ and observing x^e. Subsequently, the robot chooses to either pick up the package (alone or with another robot) or go to $Base_2$. The full policy controller includes more nodes than shown in Fig. 3 (for all possible TMAs) as well as edges (for all possible environment observations).

Fig. 5 compares a uniform random Monte Carlo search to MMCS in the package delivery domain. Results for the Monte Carlo search were generated by repeatedly sampling random, valid policies and retaining the policy with the highest expected value. Both approaches used a policy controller with a fixed number of nodes ($n_{nodes} = 13$). Results indicate that given a fixed computational budget (quantified by the number of search iterations), MMCS outperforms standard Monte Carlo search in terms of expected value. Specifically, as seen in Table I, after 1000 search iterations in the package delivery problem, the expected value of the policy from MMCS is 118% higher than the one obtained by Monte Carlo search. Additionally, due to this problem's extremely large policy space, determination of an optimal joint value through exhaustive search is not possible. Fig. 5 illustrates MMCS's ability to exploit previous policy samples to bias towards well-performing regions of the policy space, while using random sampling for further exploration of the space.

To more intuitively quantify performance of MMCS and Monte Carlo policies in the package delivery domain, Fig. 6 compares success probability of delivering a given minimum number of packages for both methods, within a fixed mission time horizon. Results were generated over 250 simulated runs with randomly-generated packages and initial conditions. Using the Monte Carlo policy, probability of successfully delivering more than 4 packages given a fixed time horizon is near zero, whereas the MMCS policy successfully delivers a larger number of packages (in some cases up to 9).

The experiments indicate the Dec-POSMDP framework and MMCS algorithm allow solutions for complex problems, such as multi-robot package delivery, that would not be possible using traditional Dec-POMDP approaches.

VI. Conclusion

This paper proposed a layered framework to exploit the macro-action abstraction in decentralized planning for a group of robots acting under uncertainty. It formally reduces the Dec-POMDP problem to a Dec-POSMDP that can be solved using discrete space search techniques. We also formulated an algorithm, MMCS, for solving Dec-POSMDPs and showed that it performs well on a multi-robot package delivery problem under uncertainty. The Dec-POSMDP represents a general formalism for probabilistic multi-robot coordination problems and our results show that high-quality solutions can be automatically generated from this high-level domain specification.

References

[1] Ali-akbar Agha-mohammadi, Suman Chakravorty, and Nancy Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *International Journal of Robotics Research (IJRR)*, 33(2):268–304, 2014.

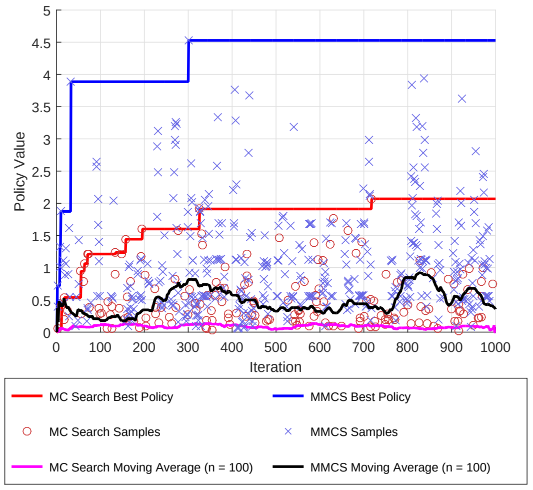

Fig. 5. Comparison of policy differences between MC and MMCS. Moving average of policy values (over 100 samples) are also indicated.

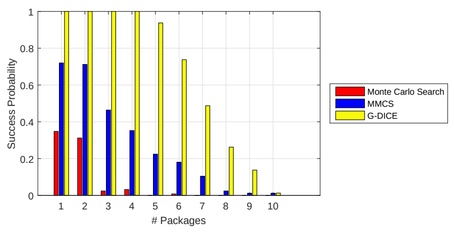

Fig. 6. Success probability (of delivering the specified number of packages *or more*) within a fixed time horizon.

[2] Ali-akbar Agha-mohammadi, N. Kemal Ure, Jonathan P. How, and John Vian. Health aware stochastic planning for persistent package delivery missions using quadrotors. In *International Conference on Intelligent Robots and Systems (IROS)*, Chicago, September 2014.

[3] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Journal of Autonomous Agents and Multi-Agent Systems*, 21(3):293–320, 2010.

[4] Christopher Amato, George D. Konidaris, Gabriel Cruz, Christopher A. Maynor, Jonathan P. How, and Leslie P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[5] Christopher Amato, George D. Konidaris, and Leslie P. Kaelbling. Planning with macro-actions in decentralized POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, 2014.

[6] Christopher Amato and Shlomo Zilberstein. Achieving goals in decentralized POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 593–600, 2009.

[7] Steve Banker. Amazon and drones – here is why it will work, December 2013.

[8] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[9] Daniel S. Bernstein, Shlomo Zilberstein, Richard Washington, and John L. Bresina. Planetary rover control as a Markov decision process. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.

[10] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Game theoretic control for robot teams. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1163–1169, 2005.

[11] Ruijie He, Emma Brunskill, and Nicholas Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40:523–570, February 2011.

[12] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238 – 1274, September 2013.

[13] P. R. Kumar and P. P. Varaiya. *Stochastic Systems: Estimation,

TABLE I

COMPARISON OF SEARCH ALGORITHMS.

Algorithm	Policy Value	Policy Iterations
Monte Carlo Search	2.068	1000
MMCS	4.528	1000
Exhaustive Search	—	5.622e+17

Identification, and Adaptive Control. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[14] Zhan Lim, Lee Sun, and Daniel J. Hsu. Monte carlo value iteration with macro-actions. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1287–1295. Curran Associates, Inc., 2011.

[15] Liam C MacDermed and Charles Isbell. Point based value iteration with optimal belief compression for Dec-POMDPs. In *Advances in Neural Information Processing Systems*, pages 100–108, 2013.

[16] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the Sixteen Conference on Artificial Intelligence (AAAI)*, pages 541–548, 1999.

[17] Shayegan Omidshafiei, Ali akbar Agha-mohammadi, Christopher Amato, and Jonathan P. How. Technical report: Decentralized control of partially observable markov decision processes using belief space macro-actions. Technical report, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, September 2014.

[18] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[19] Georgios Theocharous and Leslie Pack Kaelbling. Approximate planning in POMDPs with macro-actions. In *Advances in Neural Information Processing Systems 16 (NIPS03)*, 2004.

[20] Nazim Kemal Ure, Girish Chowdhary, Jonathan P How, and John Vian. *Planning Under Uncertainty*, chapter Multi-Agent Planning for Persistent Surveillance. MIT Press, 2013.

[21] Keith Winstein and Hari Balakrishnan. TCP ex Machina: Computer-Generated Congestion Control. In *SIGCOMM*, 2013.