Planning for Decentralized Control of Multiple Robots Under Uncertainty

Christopher Amato, George Konidaris, Gabriel Cruz, Christopher A. Maynor,
Jonathan P. How and Leslie P. Kaelbling

***Abstract*— This paper presents a probabilistic framework for synthesizing control policies for general multi-robot systems that is based on decentralized partially observable Markov decision processes (Dec-POMDPs). Dec-POMDPs are a general model of decision-making where a team of agents must cooperate to optimize a shared objective in the presence of uncertainty. Dec-POMDPs also consider communication limitations, so execution is decentralized. While Dec-POMDPs are typically intractable to solve for real-world problems, recent research on the use of macro-actions in Dec-POMDPs has significantly increased the size of problem that can be practically solved. We show that, in contrast to most existing methods that are specialized to a particular problem class, our approach can synthesize control policies that exploit any opportunities for coordination that are present in the problem, while balancing uncertainty, sensor information, and information about other agents. We use three variants of a warehouse task to show that a single planner of this type can generate cooperative behavior using task allocation, direct communication, and signaling, as appropriate. This demonstrates that our algorithmic framework can automatically optimize control and communication policies for complex multi-robot systems.**

I. Introduction

Most multi-robot systems are controlled by hand-built special-purpose algorithms that are difficult to design, implement and verify. For single robots, automatic planning systems provide a flexible general-purpose strategy for constructing plans given high-level declarative domain specifications, even in the presence of substantial stochasticity and partial observability [1]. We show that this strategy can be effectively extended to multi-robot systems. Our methods allow automatic off-line construction of robust multi-robot policies that support coordinated action. As a natural consequence of the approach, they can even generate communication strategies that exploit the domain dynamics to share critical information in service of achieving the group's overall objective.

Specifically, we are interested in problems where the robots share the same objective function and each individual robot (and, in fact, the collection of robots) can only make noisy, partial observations of the environment. The decentralized partially observable Markov decision process (Dec-POMDP) is a general framework for representing these problems and by solving them, we can automatically derive near-optimal control strategies. Dec-POMDPs have been studied in fields such as control [2], [3], operations research [4] and artificial intelligence [5]. Like the POMDP [6] models that they extend, Dec-POMDPs consider general probabilistic dynamics, sensor and cost models. To make better use of cooperation and computational resources, we consider the case where we produce a cooperative solution of the problem off-line which results in separate policies for individual robots that are executed online in a decentralized manner. These decentralized policies optimize movement, sensing and communication actions while considering uncertainty in outcomes, sensors and information about the other agents.

Fig. 1. The warehouse domain with three robots.

For example, consider the multi-robot warehousing problem (shown in Figure 1) that we present in the experiments. A team of robots is tasked with finding a set of large and small boxes in the environment and returning them to a shipping location. Large boxes require multiple robots to push. As a result, coordination is needed not just for assigning robots to push specific boxes, but also requires that two robots push the larger box at the same time. There is stochasticity in the movements of robots and partial observability with respect to the location of the boxes and other robots (both can be only be detected when they are within range). We also consider cases where the robots can send communication signals to each other, but we do not define the meaning of the messages. Therefore, our planner must determine where the robots should navigate, what boxes they should push and what communication messages should be sent (if at all) at each step of the problem to optimize the solution for the team. The robots must make these decisions based solely on the information they individually receive during execution (e.g., each robot's estimate of its own location as well as where and when boxes and other robots have been seen).

C. Amato is with the Department of Computer Science at the University of New Hampshire, Durham, NH. G. Konidaris is with the Department of Computer Science and the Department of Electrical and Computer Engineering at Duke University, Durham, NC. G. Cruz and L. P. Kaelbling are with CSAIL at MIT, Cambridge, MA. C. A. Maynor and J. P. How are with LIDS at MIT, Cambridge, MA. Email: camato@cs.unh.edu, gdk@cs.duke.edu, gcruz@csail.mit.edu, maynorc@mit.edu, jhow@mit.edu, lpk@csail.mit.edu.
Work for this paper was completed while all authors were at MIT and the research was supported in part by AFOSR MURI project #FA9550-091-0538 and and ONR MURI project #N000141110688.

This multi-robot warehousing problem — as well as any other problem where multiple robots share a single overall reward or cost function — can be formalized as a Dec-POMDP. Therefore, a Dec-POMDP solver could potentially automatically generate control policies (including policies over when and what to communicate) for very rich decentralized control problems, in the presence of uncertainty. Unfortunately, this generality comes at a cost: Dec-POMDPs are typically infeasible to solve except for very small problems [4], [7].

One reason for the intractability of solving large Dec-POMDPs is that current methods model problems at a low level of granularity, where each robot's actions are primitive operations lasting exactly one time step. Recent research has addressed the more realistic *MacDec-POMDP* case where each robot has *macro-actions*: temporally extended actions which may require different amounts of time to execute [7]. Macro-actions enable systems to be modeled at a higher level of abstraction so that coordination decisions only occur at the level of deciding which macro-actions to execute. MacDec-POMDPs retain the ability to coordinate robots while allowing near-optimal solutions to be generated for significantly larger problems than would have been possible using other Dec-POMDP-based methods.

Macro-actions are a natural model for the modular controllers often sequenced to obtain robot behavior. The macro-action approach leverages expert-designed or learned controllers for solving subproblems (e.g., navigating to a waypoint or grasping an object), bridging the gap between traditional robotics research and work on Dec-POMDPs. This approach has the potential to produce high-quality general solutions for real-world heterogeneous multi-robot coordination problems by automatically generating control and communication policies.

This paper presents a general framework for solving decentralized cooperative partially observable robotics problems and provides the first demonstration of such method running on real robots. We extend previously developed approaches [7] and apply them to a real robotics problem. We begin by formally describing the Dec-POMDP model, its solution and relevant properties as well as the MacDec-POMDP extension. We then describe a process for converting a robot domain into a MacDec-POMDP model, solving it, and using the solution to produce a set of SMACH [8] finite-state machine task controllers that can be executed on the robots. Finally, we use three variants of the warehouse problem to show that a MacDec-POMDP planner generates appropriate emergent behaviors by optimizing the available macro-actions (i.e., allocating tasks, using direct communication, and employing signaling, as appropriate). The MacDec-POMDP represents a foundational algorithmic framework for generating solutions for a wide range of probabilistic multi-robot systems.

II. Dec-POMDPs

Dec-POMDPs [4] generalize POMDPs to the multiagent, decentralized setting. As depicted in Fig. 2, multiple agents

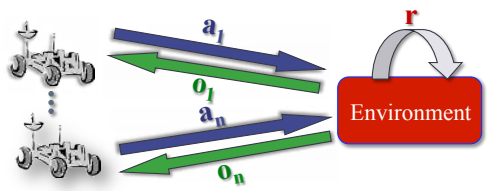


Fig. 2. Representation of an n-agent Dec-POMDP with actions a_i and observations o_i for each agent i along with a single reward r.

(e.g., robots) operate under uncertainty based on partial and local views of the world, with execution unfolding over a sequence of steps. At each step, every agent chooses an action (in parallel) based purely on locally observable information (i.e., one agent does not necessarily observe what the others are seeing or doing), resulting in an immediate reward and an observation being obtained by each individual agent. The agents share a single reward or cost function, making the problem cooperative, but their local views mean that operation is decentralized during execution.

We focus on solving sequential decision-making problems with discrete time steps and stochastic models with finite states, actions, and observations, but the model can be extended to continuous problems. A key assumption is that state transitions are *Markovian*, meaning that the state at time t depends only on the state and events at time $t-1$. Note that the robots do not perceive the state itself (only their streams of observations). The reward is used as a way to specify the objective of the problem and is not observed during execution. A Dec-POMDP is described by a tuple $\langle I, S, \{A_i\}, T, R, \{\Omega_i\}, O, h\rangle$, where

- I is a finite set of agents.
- S is a finite set of states with designated initial state distribution b_0.
- A_i is a finite set of actions for each agent i with $A = \times_i A_i$ the set of joint actions.
- T is a state transition probability function, $T : S \times A \times S \to [0, 1]$, that specifies the probability of transitioning from state $s \in S$ to $s' \in S$ when actions $\vec{a} \in A$ are taken by the agents. Hence, $T(s, \vec{a}, s') = \Pr(s'|\vec{a}, s)$.
- R is a reward function: $R : S \times A \to \mathbb{R}$, the immediate reward for being in state $s \in S$ and taking actions $\vec{a} \in A$.
- Ω_i is a finite set of observations for each agent, i, with $\Omega = \times_i \Omega_i$ the set of joint observations.
- O is an observation probability function: $O : \Omega \times A \times S \to [0, 1]$, the probability of seeing observations $\vec{o} \in \Omega$ given actions $\vec{a} \in A$ were taken which results in state $s' \in S$. Hence $O(\vec{o}, \vec{a}, s') = \Pr(\vec{o}|\vec{a}, s')$.
- h is the number of steps until the problem terminates, called the horizon.

Note that while the actions and observations are factored with one factor per agent, the state need not be. Because the full state is not directly observed, generating optimal or approximately optimal behavior generally requires each agent to remember a history of its observations. We can con-

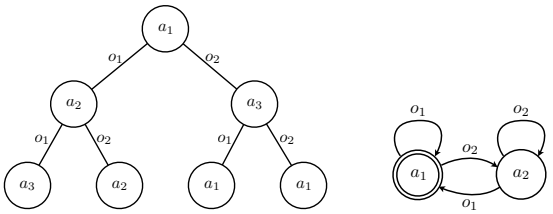


Fig. 3. A single agent's policy represented as (a) a policy tree and (b) a finite-state controller with initial state shown with a double circle.

sider an action-observation history for agent i representing the actions taken and observations seen at each step (up to step t) as $h_i^A = (a_i^0, o_i^0, \ldots, a_i^t, o_i^t)$. Unlike in POMDPs, it is not typically possible to calculate an estimate of the system state from the observation history of a single agent, because the system state depends on the behavior of all of the agents.

A solution to a Dec-POMDP is a *joint policy*—a set of policies, one for each agent. Each agent's policy maps its own history of actions and observations to its next action. It is typically represented as either a policy tree, where the vertices indicate actions to execute and the edges indicate transitions conditioned on an observation, or as a finite state controller which executes in a similar manner. An example of each is given in Figure 3.

As in the POMDP case, the goal is to maximize the total cumulative reward, beginning at some initial distribution over states b_0. We assume that the model is known.

The value of a joint policy, π, from state s is $V^\pi(s) = \mathrm{E}\left[\sum_{t=0}^{h-1} \gamma^t R(\vec{a}^t, s^t) | s, \pi\right]$, which represents the expected value of the immediate reward for the set of agents summed for each step of the problem given the action prescribed by the policy until the horizon is reached. In the finite-horizon case (which we consider in this paper), the discount factor, γ, is typically set to 1. An *optimal policy* beginning at state s is $\pi^*(s) = \mathrm{argmax}_\pi V^\pi(s)$.

Unfortunately, large problem instances remain intractable. Some advances have been made in optimal algorithms [9], [10], [11], but optimally solving a Dec-POMDP is NEXP-complete [4]. Most approaches that scale well make very strong assumptions about the domain (e.g., a large amount of independence between agents) [12], [13], [14], [15].

III. Macro-Actions for Dec-POMDPs

Dec-POMDPs require synchronous decision-making: every agent determines which action to execute, and then executes it within a single time step. This restriction is problematic in robot domains for two reasons. First, robot systems typically possess a set of controllers, and planning consists of sequencing the execution of those controllers. However, due to both environmental and controller complexity, the controllers will almost always execute for an extended period, taking differing amounts of time to run. Synchronous decision-making would require waiting until all robots have completed their controller execution (and achieve common knowledge of this fact) before performing the next action selection, which is suboptimal and generally infeasible. Second, the planning complexity of a Dec-POMDP is doubly exponential in the horizon. A planner that reasons about all of the robots' possible policies at every time step will only ever be able to make very short plans.

The MacDec-POMDP formulation models a group of robots that must plan by sequencing an existing set of controllers, enabling planning at the appropriate level to compute near-optimal solutions for problems with significantly longer horizons and larger state-spaces [7]. We can gain additional benefits by exploiting known structure in the multi-robot problem. For instance, most controllers only depend on locally observable information and do not require coordination. For example, consider a controller that navigates to a waypoint. Only local information is required for navigation—the robot may detect other robots but their presence does not change its objective, and it simply moves around them—but choosing the target waypoint likely requires the planner to consider the locations and actions of all robots. Macro-actions with independent execution allow coordination decisions to be made only when necessary (i.e., when choosing macro-actions) rather than at every time step. Because MacDec-POMDPs are built on top of Dec-POMDPs, macro-action choice may depend on history, but during execution macro-actions may depend only on a single observation or on any number of steps of history, or even represent the actions of a set of robots. That is, macro-actions are very general and can be defined in such a way to take advantage of the knowledge available to the robots during execution.

It is also worth noting that our approach can incorporate state-of-the-art solution methods for solving more restricted scenarios. The widespread use of techniques for solving much more restricted scenarios has led to a plethora of usable algorithms for specific problems, but no way to combine these in more complex scenarios. Our approach can build on the large amount of research in single and multi-robot systems that has gone into solving difficult problems such as navigation in a formation [16], cooperative transport of an object [17], coordination with signaling [18] or communication under various limitations [19]. The solutions to these problems could be represented as macro-actions in our framework, building on existing research to solve even more complex multi-robot problems.

A. Model

While more complex macro-actions are possible, the MacDec-POMDP model used here considers macro-actions that only depend on a single robot's information [7]. This is an extension of the *options framework* [20] to multi-agent domains while dealing with the lack of synchronization between agents. The options framework is a formal model of macro-actions [20] that has been very successful in aiding representation and solutions in single robot domains [21].

A MacDec-POMDP with local options is defined as a Dec-POMDP where we also assume M_i represents a finite set of

options for each agent, i, with $M = \times_i M_i$ the set of joint options [7]. A *local option* is defined by:

$$M_i = (\beta_{m_i}, \mathcal{I}_{m_i}, \pi_{m_i}),$$

with stochastic termination condition $\beta_{m_i} : H_i^A \rightarrow [0,1]$, initiation set $\mathcal{I}_{m_i} \subset H_i^A$ and option policy $\pi_{m_i} : H_i^A \times A_i \rightarrow [0,1]$. Note that this representation uses action-observation histories in the termination and initiation conditions as well as the option policy. Initiation and terminal conditions can also depend on states (e.g., ending execution based on unobserved events) or single observations.

MacDec-POMDP policies consider option histories (as opposed to action-observation histories) because it may be beneficial for agents to remember their histories when choosing options. An *option history*, which includes both the action-observation histories where an option was chosen and the selected options themselves, is defined as $h_i^M = (h_i^0, m_i^1, \ldots, h_i^{t-1}, m_i^t)$. Here, h_i^0 may be a null observation or an initial observation produced from the initial belief state b_0. The option history also provides a nice representation for using histories within options by allowing initiation conditions to depend on the histories of options already taken and their results. Alternatively, it is more natural for option policies and termination conditions to depend on histories that begin when the option is first executed (action-observation histories). While histories over primitive actions provide the number of steps that have been executed (because they include actions and observations at each step), an option history may require many more steps to execute than the number of options listed. A (stochastic) local policy, $\mu_i : H_i^M \times M_i \rightarrow [0,1]$ then depends on these option histories and a joint policy for all agents is given as μ.

Because option policies are built from primitive actions, policies can be evaluated in a way that is similar to other Dec-POMDP-based approaches. Given a joint policy, the primitive action at each step is determined by the (high-level) policy, which chooses the option, and the option policy, which chooses the action. The joint policy and option policies can then be evaluated as:

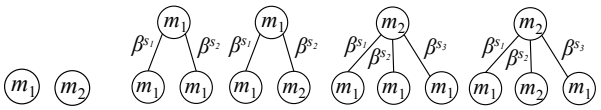
$$V^{\mu}(s) = \mathrm{E}\left[\sum_{t=0}^{h-1} \gamma^t R(\vec{a}^t, s^t) | s, \pi, \mu\right]$$

Additional details about a slightly simpler case are given in previous work [7].

The goal of MacDec-POMDP planning is to obtain a *hierarchically optimal policy*: $\mu^*(s) = \mathrm{argmax}_{\mu} V^{\mu}(s)$. This policy produces the highest expected value that can be obtained by sequencing the agent's given options. This policy may have a lower value than the optimal policy for the Dec-POMDP, because it does not include all possible history-dependent low-level policies—the policies are restricted to be sequences of macro-actions.

B. Algorithms

We use the macro-action-based memory bounded dynamic programming (MBDP) approach [7] to solve the warehouse problems. This approach is the extension of a standard Dec-POMDP algorithm [22] to consider options instead of primitive actions. MBDP searches through the space of possible policies from the last problem step back to the first. The extension is non-trivial since macro-actions may use different amounts of time, requiring the consideration of the horizon being reached during execution of a macro-action.



(a) Step 1 (b) Step 2 of DP

Fig. 4. Policies for a single agent after (a) one step and (b) two steps of dynamic programming using options m_1 and m_2 and (deterministic) terminal states as β^s.

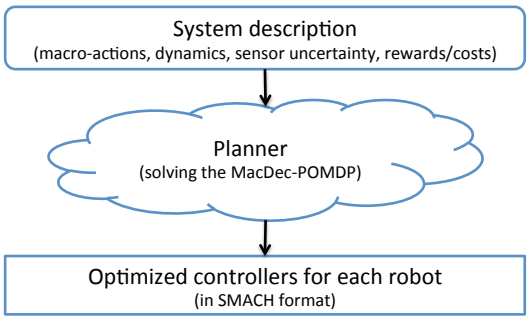


Fig. 5. A high-level system diagram.

The key modification (as shown in Figure 4) is that nodes in a policy tree now select macro-actions (rather than primitive actions) and edges correspond to terminal conditions (or, more generally, high-level observations). That is, policy trees are generated for each agent that can be executed based on that agent's possible option histories. The root node defines the option to choose in the known initial state, and options are assigned to each of the legal terminal states of that option; this continues for the depth of the tree. This tree can be evaluated up to any (primitive) horizon using the policy evaluation given above.

Our MBDP-based algorithm bounds the number of possible policies that are considered at each step. Without this bound, dynamic programming methods can guarantee that a hierarchically optimal policy is found, but when a bound is imposed the result may be suboptimal [7]. Policies are chosen that have the highest value at states that are sampled from the initial state. Because the number of policies at each step is bounded, MBDP has time and space complexity linear in the horizon. As a result, this approach works well in some relatively large MacDec-POMDPs [7].

IV. Solving Multi-robot problems with MacDec-POMDPs

The MacDec-POMDP framework is a natural way to represent and generate behavior for general multi-robot systems. A high-level description of this process is given in Figure 5. To use the MacDec-POMDP model as described above, we would assume an abstract model of the system is given

in the form of macro-action representations, which include the associated policies as well as initiation and terminal conditions. These macro-actions are controllers operating in (possibly) continuous time with continuous actions and feedback, but their operation is discretized for use with the planner. This discretization represents an underlying discrete Dec-POMDP which consists of the primitive actions, states of the system and the associated rewards. While the complexity of our method primarily depends on the size of the MacDec-POMDP model, and not the size of the underlying Dec-POMDP, it is often difficult to generate and represent a full Dec-POMDP model for real-world systems.

We extend this model to use a simulator rather than a full model of the problem. In many cases, a simulator already exists or is easier to construct than the full model. Our planner still assumes a model of the macro-actions, but while the initiation and terminal sets are known, the policies of the macro-actions as well as the underlying Dec-POMDP are not explicitly known. Instead, we make the more realistic assumption that we can simulate the macro-actions in an environment similar to the real-world domain. As such, the algorithm for generating a policy over macro-actions remains the same (since constructing policies of macro-actions only requires knowledge of the set of macro-actions and their initiation and terminal conditions), but all evaluation is conducted in the simulator (through sampling) rather than through use of the Bellman equations (which requires enumeration over all reachable states at each step).

Specifically, a fixed policy can be evaluated by sampling starting at an initial state (or belief state), choosing an action for each agent according to the policy, sampling an observation from the system, updating the current position in the policy (i.e., the current node in each agent's policy tree) and then continuing this process until some maximum time step has been reached. The value of the k-th sample-based trajectory starting at s_0 and using policy π is given by $V^{\pi,k}(s_0) = r_0^k + \ldots + \gamma^T r_T^k$, where r_t^k is the reward given to the team on the t-th step. After K trajectories, $\hat{V}^{\pi}(s_0) = \sum_{k=1}^{K} \frac{V^{\pi,k}(s_0)}{K}$. As the number of samples increases, the estimate of the policy's value will approach the true value. This sample-based evaluation is necessary in large or continuous state spaces.

Given the macro-actions and simulator, our off-line planner can automatically generate a solution which optimizes the value function with respect to the uncertainty over outcomes, sensor information and other robots. The planner generates the solution in the form of a set of policy trees (as in Figure 4) which are parsed into a corresponding set of SMACH controllers [8], one for each robot. SMACH controllers are hierarchical state machines for use in a ROS [23] environment. Just like the policy trees they represent, each node in the SMACH controller represents a macro-action which is executed on the robot and each edge corresponds to a terminal condition. Our system is thus able to automatically generate SMACH controllers, which are typically designed by hand, for complex, general multi-robot systems.

V. MacDec-POMDPs in the Warehouse Domain

We test our methods in a warehousing scenario using a set of iRobot Creates (Figure 1) where we will vary communication capabilities. This is the first time that Dec-POMDP-based methods have been used to solve large multi-robot domains, and we do not compare with other Dec-POMDP-based methods because they cannot solve problems of this size. The results demonstrate that our methods can automatically generate the appropriate motion and communication behavior while considering uncertainty over outcomes, sensor information and other robots.

A. The Warehouse Domain

We consider three robots in a warehouse that are tasked with finding and retrieving boxes of two different sizes: large and small. Robots can navigate to known depot locations (rooms) to retrieve boxes and bring them back to a designated drop-off area. The larger boxes can only be moved effectively by two robots (if a robot tries to pick up the large box by itself, it will move to the box, but fail to pick it up). While the locations of the depots are known, the contents (the number and type of boxes) are unknown. Our planner generates a SMACH controller for each of the robots off-line which are then executed online in a decentralized manner.

In each scenario, we assumed that each robot could observe its own location, see other robots if they were within (approximately) one meter, observe the nearest box when in a depot and observe the size of the box if it is holding one. In the simulator used by the planner to evaluate solutions, the resulting state space includes the location of each robot (discretized into nine possible locations) and the location of each of the boxes (in a particular depot, with a particular robot or at the goal). The primitive actions are to move in four different directions as well as pickup, drop and communication actions. Note that this primitive state and action representation is used for evaluation purposes and not actually implemented on the robots (which just utilize the SMACH controllers). Higher fidelity simulators could also be used, but running time may increase if the simulations are computationally intensive (average solution times for the policies presented below were approximately one hour). The three-robot version of this scenario has 1,259,712,000 states, which is several orders of magnitude larger than problems typically solvable by Dec-POMDP approaches. These problems are solved using the option-based MBDP algorithm initialized with a hand coded heuristic policy.

Navigation has a small amount of noise in the amount of time required to move to locations (reflecting the real-world dynamics): this noise increases when the robots are pushing the large box (reflecting the need for slower movements and turns in this case). We defined macro-actions that depend only on the observations above, but option selection depends on the history of options executed and observations seen as a result (the option history). Note that the MacDec-POMDP framework is very general so other types of macro-actions and observations could also be used (including observation of other failures).

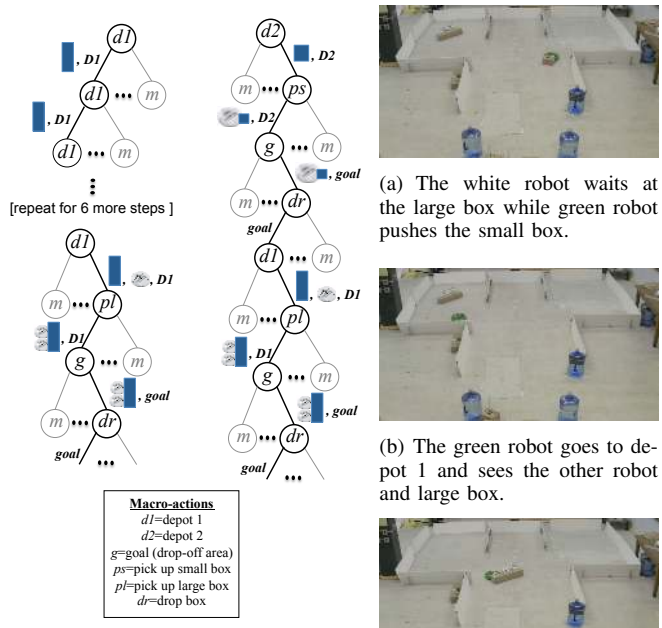


Fig. 6. Path executed in policy trees by the white robot (left) and the green robot (right). Only macro-actions executed (nodes) and observations seen (edges, with box sizes and robots given pictorially) are shown.

(a) The white robot waits at the large box while green robot pushes the small box.

(b) The green robot goes to depot 1 and sees the other robot and large box.

(c) The green robot moves to the box and the two robots push it back to the goal.

Fig. 7. Scenario 1 video captures (no communication).

B. Scenario 1: No Communication

In the first scenario, the robots cannot communicate with each other. Therefore, all cooperation is based on the controllers that are generated by the planner (which generates controllers for all robots when planning off-line) and observations of the other robots (when executing online). The macro-actions were: `Go to depot 1`, `Go to depot 2`, `Go to the drop-off area`, `Pick up the small box`, `Pick up the large box`, and `Drop off a box`.

The depot macro-actions are applicable anywhere and terminate when the robot is within the walls of the appropriate depot. The drop-off and drop macro-actions are only applicable if the robot is holding a box, and the pickup macro-actions are only applicable when the robot observes a box. Picking up the small box was assumed to succeed deterministically, but the model could easily be adjusted if the pickup mechanism is less robust. The macro-actions correspond to natural choices for robot controllers.

This case[1] (seen in Figure 7 along with a depiction of the executed policy in Figure 6) uses only two robots to more clearly show the optimized behavior in the absence of communication. The policy generated by the planner begins by assigning one robot to go to each of the depots. The robots then observe the contents of the depots they are in. If there is only one robot in a depot and there is a small box to push, the robot will push the small box (Figure 7(a)). If the robot is in a depot with a large box and no other robots, it will stay in the depot, waiting for another robot to come and help push the box (also Figure 7(a)). In this case, once the other robot is finished pushing the small box, it goes back to the depots to check for other boxes or robots that need help (Figure 7(b)). When it sees another robot and the large box in the depot on the left (depot 1), it attempts to help push the large box and the two robots are successful pushing the large box to the goal (Figure 7(c)). The planner has *automatically derived a strategy for dynamic task allocation*—two robots go to each room, and then search for help needed after pushing any available boxes. This behavior was generated by an optimization process that considered the different costs of actions and the uncertainty involved (in the current step and into the future) and used those values to tailor the behavior to the particular problem instance.

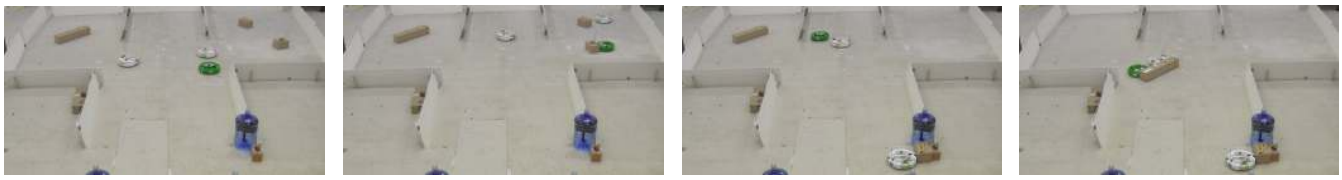
[1]All videos can be seen at `http://youtu.be/fGUHTHH-JNA`

C. Scenario 2: Local Communication

In scenario 2, robots can communicate when they are within one meter of each other. The macro-actions are the same as above, but we added ones to communicate and wait for communication. The resulting macro-action set is: `Go to depot 1`, `Go to depot 2`, `Go to the drop-off area`, `Pick up the small box`, `Pick up the large box`, `Drop off a box`, `Go to an area between the depots (the "waiting room")`, `Send signal #1`, `Send signal #2`, and `Wait in the waiting room for another robot`.

Here, we allow the robots to choose to go to a "waiting room" which is between the two depots. This permits the robots to possibly communicate or receive communications before committing to one of the depots. The waiting-room macro-action is applicable in any situation and terminates when the robot is between the waiting room walls. The depot macro-actions are now only applicable in the waiting room, while the drop-off, pick up and drop macro-actions remain the same. The wait macro-action is applicable in the waiting room and terminates when the robot observes another robot in the waiting room. The signaling macro-actions are applicable in the waiting room and are observable by other robots that are within approximately a meter of the signaling robot. Note that *we do not specify what sending each communication signal means*.

The results for this three-robot domain are shown in Figure 8. The robots go to the waiting room and then two of the robots go to depot 2 (the one on the right) and one robot goes to depot 1 (the one on the left) (Figure 8(a)). Because there are three robots, the choice for the third robot is random while one robot will always be assigned to each of the depots. Because there is only a large box to push in depot 1, the robot in this depot goes back to the waiting room to try to find another robot to help it push the box (Figure 8(b)). The robots in depot 2 see two small boxes and they choose to push these back to the goal (also Figure 8(b)). Once the small boxes are dropped off, one of the robots returns to the waiting room and then is recruited by the other robot to



(a) One robot goes to depot 1 and two robots go to depot 2. The depot 1 robot sees a large box.

(b) The robot saw a large box, so it moved to the waiting room while the other robots pushed the small boxes.

(c) The green robot goes to the waiting room to check for signals and the white robot sends signal #1.

(d) Signal #1 is interpreted as a need for help in depot 1, so they move to depot 1 and push the large box.

Fig. 8. Scenario 2 video captures (limited communication).

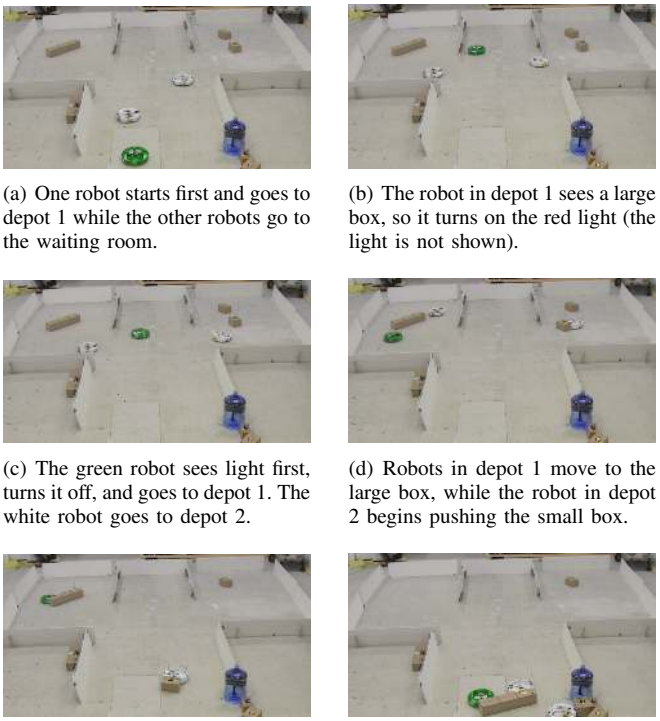
push the large box back to the goal (Figure 8(c)). The robots then successfully push the large box back to the goal (Figure 8(d)). In this case, the planning process *determines how the signals should be used to perform communication*.

D. Scenario 3: Global Communication

In the last scenario, the robots can use signaling (rather than direct communication). In this case, there is a switch in each of the depots that can turn on a blue or red light. This light can be seen in the waiting room and there is another light switch in the waiting room that can turn off the light. (The light and switch were simulated in software and not incorporated in the physical domain.) The macro-actions were: `Go to depot 1`, `Go to depot 2`, `Go to the drop-off area`, `Pick up the small box`, `Pick up the large box`, `Drop off a box`, `Go to the "waiting room")`, `Turn on a blue light`, `Turn on a red light`, and `Turn off the light`.

The first seven macro-actions are the same as for the communication case except we relaxed the assumption that the robots had to go to the waiting room before going to the depots (making both the depot and waiting room macro-actions applicable anywhere). The macro-actions for turning the lights on are applicable in the depots and the macro-actions for turning the lights off are applicable in the waiting room. While the lights were intended to signal requests for help in each of the depots, we did not assign a particular color to a particular depot. In fact, we did not assign them any meaning at all, allowing the planner to set them in any way that improves performance.

The results are shown in Figure 9. Because one robot started ahead of the others, it was able to go to depot 1 to sense the size of the boxes while the other robots go to the waiting room (Figure 9(a)). The robot in depot 1 turned on the light (red in this case, but not shown in the images) to signify that there is a large box and assistance is needed (Figure 9(b)). The green robot (the first other robot to the waiting room) sees this light, interprets it as a need for help in depot 1, and turns off the light (Figure 9(c)). The other robot arrives in the waiting room, does not observe a light on and moves to depot 2 (also Figure 9(c)). The robot in depot 2 chooses to push a small box back to the goal and the green robot moves to depot 1 to help the other robot (Figure 9(d)). One robot then pushes the small box back to the goal while the two robots in depot 1 begin pushing



(a) One robot starts first and goes to depot 1 while the other robots go to the waiting room.

(b) The robot in depot 1 sees a large box, so it turns on the red light (the light is not shown).

(c) The green robot sees light first, turns it off, and goes to depot 1. The white robot goes to depot 2.

(d) Robots in depot 1 move to the large box, while the robot in depot 2 begins pushing the small box.

(e) Robots in depot 1 begin pushing the large box and the robot in depot 2 pushes a small box to the goal.

(f) The robots from depot 1 successfully push the large box to the goal.

Fig. 9. Scenario 3 video captures (signaling).

the large box (Figure 9(e)). Finally, the two robots in depot 1 push the large box back to the goal (Figure 9(f)). This behavior is optimized based on the information given to the planner. *The semantics of all these signals as well as the movement and signaling decisions were decided on by the planning algorithm to maximize value.*

VI. Related Work

There are several frameworks for multi-robot decision making in complex domains. For instance, behavioral methods have been studied for performing task allocation over time with loosely-coupled [24] or tightly-coupled [25] tasks. These are heuristic in nature and make strong assumptions about the type of tasks that will be completed.

Linear temporal logic (LTL) has also been used to specify robot behavior [26], [27]; from this specification, reactive controllers that are guaranteed to satisfy the specification can be derived. These methods are appropriate when the world

dynamics can be effectively described non-probabilistically and when there is a useful characterization of the robot's desired behavior in terms of a set of discrete constraints. When applied to multiple robots, it is necessary to give each robot its own behavior specification. In contrast, our approach (probabilistically) models the *domain* and allows the planner to automatically optimize the robots' behavior.

Market-based approaches use traded value to establish an optimization framework for task allocation [28], [29]. These approaches have been used to solve real multi-robot problems [30], but are largely aimed to tasks where the robots can communicate through a bidding mechanism.

Emery-Montemerlo et al. [31] introduced a (cooperative) game-theoretic formalization of multi-robot systems which resulted in solving a Dec-POMDP. An approximate forward search algorithm was used to generate solutions, but because a (relatively) low-level Dec-POMDP was used scalability was limited. Their system also required synchronized execution by the robots.

VII. Conclusion

We have demonstrated—for the first time—that complex multi-robot domains can be solved with Dec-POMDP-based methods. The MacDec-POMDP model is expressive enough to capture multi-robot systems of interest, but also simple enough to be feasible to solve in practice. Our results show that a general purpose MacDec-POMDP planner can generate cooperative behavior for complex multi-robot domains with task allocation, direct communication, and signaling behavior emerging automatically as properties of the solution for the given problem model. Because all cooperative multi-robot problems can be modeled as Dec-POMDPs, MacDec-POMDPs represent a powerful tool for automatically trading-off various costs, such as time, resource usage and communication while considering uncertainty in the dynamics, sensors and other robot information. These approaches have great potential to lead to automated solution methods for general probabilistic multi-robot coordination problems with heterogeneous robots in complex, uncertain domains.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[2] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer, "Decentralized control of partially observable Markov decision processes," in *Proc. of the 52nd IEEE Conf. on Decision and Control*, 2013.

[3] A. Mahajan, "Optimal decentralized control of coupled subsystems with control sharing," *IEEE Transactions on Automatic Control*, vol. 58, 2013.

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

[5] F. A. Oliehoek, "Decentralized POMDPs," in *Reinforcement Learning: State of the Art*, ser. Adaptation, Learning, and Optimization, M. Wiering and M. van Otterlo, Eds. Springer, 2012, vol. 12.

[6] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, 1998.

[7] C. Amato, G. Konidaris, and L. P. Kaelbling., "Planning with macro-actions in decentralized POMDPs," in *Proc. 13th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2014.

[8] J. Bohren, "SMACH," http://wiki.ros.org/smach/, 2010.

[9] C. Amato, J. S. Dibangoye, and S. Zilberstein, "Incremental policy generation for finite-horizon DEC-POMDPs," in *Proc. 19th Int. Conf. on Automated Planning and Scheduling*, 2009.

[10] J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet, "Optimally solving Dec-POMDPs as continuous-state MDPs," in *Proc. 24th Int. Joint Conf. on Artificial Intelligence*, 2013.

[11] F. A. Oliehoek, M. T. J. Spaan, C. Amato, and S. Whiteson, "Incremental clustering and expansion for faster optimal planning in Dec-POMDPs," *Journal of Artificial Intelligence Research*, vol. 46, 2013.

[12] J. S. Dibangoye, C. Amato, A. Doniec, and F. Charpillet, "Producing efficient error-bounded solutions for transition independent decentralized MDPs," in *Proc. 12th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2013.

[13] F. S. Melo and M. Veloso, "Decentralized MDPs with sparse interactions," *Artificial Intelligence*, 2011.

[14] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed POMDPs: a synthesis of distributed constraint optimization and POMDPs," in *Proc. 20th National Conf. on Artificial Intelligence*, 2005.

[15] P. Velagapudi, P. R. Varakantham, K. Sycara, and P. Scerri, "Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents," in *Proc. 10th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2011.

[16] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, 1998.

[17] C. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, 2000.

[18] R. Beckers, O. Holland, and J.-L. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," in *Artificial life IV*, vol. 181, 1994.

[19] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 4. IEEE, 2004.

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

[21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The Int. Journal of Robotics Research*, vol. 32, no. 11, 2013.

[22] S. Seuken and S. Zilberstein, "Memory-bounded dynamic programming for DEC-POMDPs," in *Proc. 20th Int. Joint Conf. on Artificial Intelligence*, 2007.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[24] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, 1998.

[25] A. W. Stroupe, R. Ravichandran, and T. Balch, "Value-based action selection for exploration and dynamic target observation with robot teams," in *Proc. Int. Conf. on Robotics and Automation*, vol. 4. IEEE, 2004.

[26] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, 2007.

[27] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on LTL specifications," in *Proc. 43rd IEEE Conf. on Decision and Control*, 2004.

[28] M. B. Dias and A. T. Stentz, "A comparative study between centralized, market-based, and behavioral multirobot coordination approaches," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 3, 2003.

[29] B. Gerkey and M. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. Journal of Robotics Research*, vol. 23, no. 9, 2004.

[30] N. Kalra, D. Ferguson, and A. T. Stentz, "Hoplites: A market-based framework for planned tight coordination in multirobot teams," in *Proc. Int. Conf. on Robotics and Automation*, 2005.

[31] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Game theoretic control for robot teams," in *Proc. Int. Conf. on Robotics and Automation*, 2005.