Graph-based Cross Entropy Method for Solving Multi-Robot Decentralized POMDPs

Shayegan Omidshafiei[1], Ali-akbar Agha-mohammadi[2], Christopher Amato[3], Shih-Yuan Liu[1], Jonathan P. How[1], John Vian[4]

***Abstract*— This paper introduces a probabilistic algorithm for multi-robot decision-making under uncertainty, which can be posed as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP). Dec-POMDPs are inherently synchronous decision-making frameworks which require significant computational resources to be solved, making them infeasible for many real-world robotics applications. The Decentralized Partially Observable Semi-Markov Decision Process (Dec-POSMDP) was recently introduced as an extension of the Dec-POMDP that uses high-level macro-actions to allow large-scale, asynchronous decision-making. However, existing Dec-POSMDP solution methods have limited scalability or perform poorly as the problem size grows. This paper proposes a cross-entropy based Dec-POSMDP algorithm motivated by the combinatorial optimization literature. The algorithm is applied to a constrained package delivery domain, where it significantly outperforms existing Dec-POSMDP solution methods.**

I. Introduction

Consider the problem of unmanned underwater exploration using a fleet of autonomous robots. This can be posed as a complex decentralized decision-making problem, where each robot must not only decide which region of the domain to explore, but also which route to take to complete the task efficiently. Additionally, the robots operate in an uncertain environment, using noisy sensors to detect their surroundings, and have limited or no communication with each other during execution. This team of robots must cooperate to achieve their joint task, while considering the sources of uncertainty present in the domain.

This problem, in its most general form, can be posed as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [4], [18]. The Dec-POMDP is a sequential, synchronized decision-making framework which extends the single-robot POMDP to decentralized, multi-robot domains.

Due to the decentralized nature of Dec-POMDPs, POMDP algorithms cannot be easily extended to solve them; finding the optimal solution for a finite-horizon Dec-POMDP is NEXP-complete, and the infinite-horizon problem is undecidable [9]. Numerous algorithms have emerged for obtaining exact solutions for finite-horizon Dec-POMDPs and approximate solutions for the infinite-horizon case [5], [8], [10]–[12], [18], [20], [21], [25], [27], [28]. Despite this, the complexity of Dec-POMDPs and high-memory requirements for large domains limit the applicability of these solution methods to small problems [8], [9], [28].

Dec-POMDPs are computationally difficult to solve as they suffer from the curses of dimensionality and history. To improve scalability, recent efforts have cast Dec-POMDPs into higher-level frameworks which use macro-actions (MAs): temporally-extended actions that have successfully aided representation and solution in single robot MDPs and POMDPs [1], [13], [14], [26]. In [3], [6], [7], integration of MAs into Dec-POMDPs was presented, though until recently selection and design of MAs relied on a human domain expert.

Principled integration of automatically-generated MAs into Dec-POMDPs was considered in [22], which introduced the Decentralized Partially Observable Semi-Markov Decision Process (Dec-POSMDP) as well as a heuristics-based solution method (Masked Monte Carlo Search). The Dec-POSMDP is an inherently asynchronous decision-making framework that is well-suited to real-world robotics applications. In real-world settings, asynchronous decision-making is critical in scenarios where the cost of idling robots (in order to synchronize their decision-making) is high.

This paper provides an extension of the work presented in [22]. The main contribution of this paper is the Graph-based Direct Cross-Entropy method (G-DICE), a Dec-POSMDP solution algorithm with probabilistic convergence guarantees. This paper also presents more rigorous definitions of many of the Dec-POSMDP concepts introduced in the earlier work, and complexity analysis for G-DICE. Experiments conducted for a package delivery domain are presented and compared to the results from [22], showing significant improvement in solution quality provided by the proposed algorithm.

II. Problem Statement

This section introduces the decentralized decision-making under uncertainty problem as a Dec-POMDP and formally defines its transformation into a MA-based Dec-POSMDP.

A. Dec-POMDPs

The Dec-POMDP [9] is a sequential decision-making problem where multiple robots operate under uncertainty based on different streams of observations. At each step, every robot chooses an action (in parallel) based purely on its local observations, resulting in an immediate reward and an observation for each individual robot based on stochastic (Markovian) models over continuous state, action, and observation spaces.

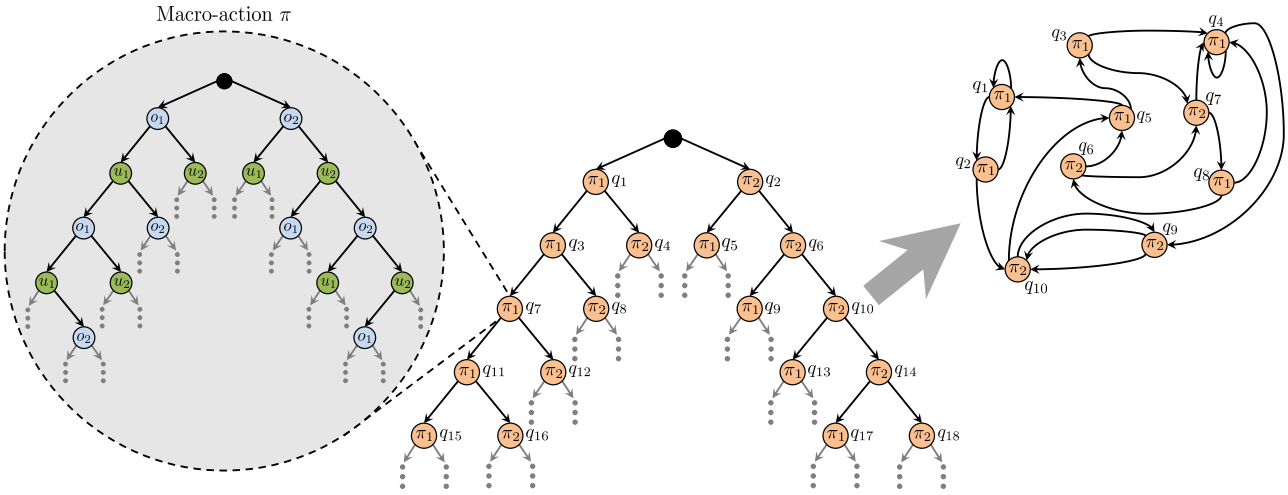
*This work was supported by The Boeing Company.

[1]Laboratory for Information and Decision Systems (LIDS), MIT, Cambridge, MA 02139, USA {shayegan,syliu,jhow}@mit.edu

[2]Qualcomm Research, San Diego, CA, 92121, USA aliagha@qualcomm.com

[3]Department of Computer Science, University of New Hampshire, Durham, NH 03824, USA camato@cs.unh.edu

[4]Boeing Research & Technology, Seattle, WA 98108, USA john.vian@boeing.com



(a) The selected MA, $\pi^{(i)}$, for each robot i is itself a low-level policy mapping the robot's history of action-observations $H^{(i)}$ to its next low-level (primitive) action, $u^{(i)} \in \mathbb{U}$.

(b) High-level tree-based policy, $\phi^{(i)}$, with nodes $q^{(i)}$ representing the MAs the robot should execute and edges representing the MA-observation the robot receives after the MA execution.

(c) Graph-based controllers (or FSAs) are used to compress policy representation, reducing the size of the policy search space and allowing execution in infinite-horizon domains.

Fig. 1: Overview of hierarchical decision-making using the Dec-POSMDP. The above illustrates policy, $\phi^{(i)}$, for the i-th robot. Each robot's policy is represented as a high-level graph-based controller (right), which can be considered a compression of a tree-based policy over MAs (middle). Each MA is, in turn, a low-level policy (left).

The Dec-POMDP model used in this paper is defined as[1]:

- $\mathbb{I} = \{1, 2, \cdots, n\}$ is the set of robots, which may be heterogeneous.
- $\bar{\mathbb{S}} = \bar{\mathbb{X}} \times \mathbb{X}^e$ is the joint super-state space, where $\mathbb{X}^e$ is the environment state (e-state) and $\bar{\mathbb{X}} = \times_i \mathbb{X}^{(i)}$ is the joint state space of the robots. Note that $\mathbb{X}^e$ is a finite set describing the state of the environment (for instance, the location of packages in a warehouse). $\mathbb{X}^{(i)}$ denotes the continuous state space of the i-th robot.
- $\bar{\mathbb{U}} = \times_i \mathbb{U}^{(i)}$ is the set of continuous joint actions, where $\mathbb{U}^{(i)}$ is the set of actions for the i-th robot.
- $P(\bar{s}'|\bar{s}, \bar{u})$ is the joint state transition probability density function, indicating the team's probability of transitioning to state $\bar{s}' \in \bar{\mathbb{S}}$ when joint action $\bar{u} \in \bar{\mathbb{U}}$ is taken in state $\bar{s} \in \bar{\mathbb{S}}$.
- $\bar{R} : \bar{\mathbb{S}} \times \bar{\mathbb{U}} \to \mathbb{R}$, is the joint reward function, denoting the reward for being in joint state $\bar{s} \in \bar{\mathbb{S}}$ and taking the joint action $\bar{u} \in \bar{\mathbb{U}}$.
- $\bar{\Omega} = \times_i \Omega^{(i)} = \bar{\mathbb{Z}} \times \bar{\mathbb{Z}}^e$ is the set of continuous joint observations obtained by the robots, where $\bar{\mathbb{Z}} = \times_i \mathbb{Z}^{(i)}$ and $\bar{\mathbb{Z}}^e = \times_i \mathbb{Z}^{e(i)}$. Note that $\Omega^{(i)} = \mathbb{Z}^{(i)} \times \mathbb{Z}^{e(i)}$ is the set of observations obtained by the i-th robot, and $o^{e(i)} \in \mathbb{Z}^{e(i)}$ is the environmental observation (e-obs) that is a function of the e-state $x^e \in \mathbb{X}^e$.
- $P(\bar{o}|\bar{s}', \bar{u})$ is the joint observation probability density function, denoting the probability of seeing joint observation $\bar{o} \in \bar{\Omega}$ given joint action $\bar{u} \in \bar{\mathbb{U}}$ which resulted in joint state $\bar{s}' \in \bar{\mathbb{S}}$.
- $\gamma \in [0, 1]$ is a discount factor on rewards, used to prioritize actions which yield earlier rewards.

[1]The typical Dec-POMDP definition does not include factored states [18].

The full action-observation history is then defined as

$$\breve{H}_t^{(i)} = \{\breve{o}_0^{(i)}, u_0^{(i)}, \breve{o}_1^{(i)}, u_1^{(i)}, \cdots, \breve{o}_{t-1}^{(i)}, u_{t-1}^{(i)}, \breve{o}_t^{(i)}\}, \quad (1)$$

where $\breve{o}^{(i)} \in \Omega^{(i)}$. The policy, $\eta^{(i)}$, for the i-th robot is defined as a mapping from the full action-observation history to the next action the robot should take, $u_t^{(i)} = \eta^{(i)}(\breve{H}_t^{(i)})$. The collection of policies for all the robots is referred to as the joint policy, $\bar{\eta} = \{\eta^{(1)}, \eta^{(2)}, \cdots, \eta^{(n)}\}$.

The joint value of a given joint policy $\bar{\eta}$ starting from an initial joint belief (or state distribution) $\bar{b} = P(\bar{s})$ is defined as the discounted sum of joint rewards,

$$\bar{V}(\bar{b}; \bar{\eta}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \bar{R}(\bar{s}_t, \bar{u}_t) | P(\bar{s}_0) = \bar{b}; \bar{\eta}\right]. \quad (2)$$

The solution to the Dec-POMDP is the optimal joint policy

$$\bar{\eta}^* = \arg\max_{\bar{\eta}} \bar{V}(\bar{b}; \bar{\eta}). \quad (3)$$

B. Dec-POSMDPs

The Dec-POMDP is a general framework for multi-robot decision-making problems. However, due to its reliance on primitive (low-level) actions, it is typically infeasible to use for large problems [8], [9], [28]. The use of primitive actions also enforces synchronized decision-making amongst the robots, which is limiting in real-world scenarios. This motivates the need for the Dec-POSMDP [22], which is a belief-space framework allowing tractable solving of large-scale planning problems by using automatically-generated MAs.

Macro-action $\pi^{(i)}$ for the i-th robot provides a mapping from the robot's action-observation history to the next action it should take, $u_t = \pi^{(i)}(H_t^{(i)})$, similar to the policy $\eta^{(i)}$ defined in the previous section. MAs have probabilistic completion

times and success rates, meaning that the Dec-POSMDP is an inherently asynchronous, semi-Markovian decision-making framework as the robots' actions take varying amounts of time to complete.

Each MA is considered to be successfully executed when it takes a robot from an initial belief, b, to an ϵ-neighborhood of a milestone (or goal) belief, $\breve{b}^{goal}$. This ϵ-neighborhood is referred to as a *goal belief node* for the MA, and is denoted $B^{goal} = \{b \ : \ \|b - \breve{b}^{goal}\| \leq \epsilon\}$. A detailed discussion of belief nodes is presented in [22].

To allow implementation of MAs in decentralized decision-making frameworks, three properties are required for each MA, π, at any initial belief state, b:

1) The value of the MA, $V(b;\pi)$. This is necessary when the MA takes the environmental state into account and abstracts these values into a generalized one-step joint reward $\bar{R}$ for the team. In practice, $V(b;\pi)$ is also useful for efficient, high-level simulations of MAs when searching for joint policies.
2) The probabilistic completion time of the MA, $T(B^{goal}|b;\pi)$. This is necessary for appropriate discounting of each MA's reward, since the MAs take different amounts of time to complete.
3) The success probability of the MA, $P(B^{goal}|b;\pi)$.

Automatic MA-generation using a graph-based approach is presented in [22], which also details calculation of the aforementioned three characteristic properties for each MA.

The joint MA, $\bar{\pi} = \{\pi^{(1)}, \cdots, \pi^{(n)}\}$, is defined as the collection of MAs being executed by the entire team at a given time. Upon completion of its MA, each robot i receives a partial observation of the e-state, x^e, denoted as the e-obs $o^{e(i)}$. In conjunction, the robot has access to its final local belief, $b^{f(i)}$, at the end of the MA. This pair (the e-obs and the final belief) is referred to as the MA-observation, defined as $\breve{o}^{e(i)} = (o^{e(i)}, b^{f(i)})$ for the i-th robot. Assuming $o^{e(i)}$ changes its value only at the completion of a MA, then $\breve{o}^{e(i)}$ abstracts all the primitive observations $o^{(i)}$ within the MA along with the e-obs $o^{e(i)}$ at the end of the MA.

The Dec-POSMDP framework is defined as follows:

- $\mathbb{I} = \{1, 2, \cdots, n\}$ is the set of robots, which may be heterogeneous.
- $\mathbb{B}^{(1)} \times \mathbb{B}^{(2)} \times \ldots \times \mathbb{B}^{(n)} \times \mathbb{X}^e$ is the underlying belief space, where $\mathbb{B}^{(i)}$ is the set of belief milestones of the i-th robot's MAs
- $\bar{\mathbb{T}} = \mathbb{T}^{(1)} \times \mathbb{T}^{(2)} \ldots \times \mathbb{T}^{(n)}$ is MA space, where $\mathbb{T}^{(i)}$ is the set of MAs for the i-th robot. Each MA is automatically generated using the procedure outlined in [22].
- $P(\bar{b}', x^{e'}, k|\bar{b}, x^e; \bar{\pi})$ is the transition probability under MAs $\bar{\pi}$ from $(\bar{b}, x^e)$ to $(\bar{b}', x^{e'})$ as described below.
- $\bar{R}^\tau(\bar{b}, x^e; \bar{\pi})$ denotes the generalized reward of taking a joint MA $\bar{\pi}$ at $(\bar{b}, x^e)$ as described further below.
- $P(\bar{\breve{o}}^e|\bar{b}, x^e)$ denotes the joint observation likelihood model, where the joint observation vector is $\bar{\breve{o}}^e = \{\breve{o}^{e(1)}, \breve{o}^{e(2)}, \cdots, \breve{o}^{e(n)}\}$.
- $\bar{\mathbb{O}}^e = \{\bar{\breve{o}}^e\}$ is the set of all joint MA-observations.
- $\gamma \in [0, 1]$ is a discount factor on rewards.

The MA-history for the i-th robot is then defined as the history of executed MAs and received MA-observations,

$$\xi_k^{(i)} = \{\breve{o}_0^{e(i)}, \pi_0^{(i)}, \breve{o}_1^{e(i)}, \pi_1^{(i)}, \cdots, \breve{o}_{k-1}^{e(i)}, \pi_{k-1}^{(i)}, \breve{o}_k^{e(i)}\}. \quad (4)$$

$\Xi^{(i)}$ denotes the space of MA-histories for the i-th robot.

Extending the notion of state transition probabilities from the Dec-POMDP, the transition probability from $(\bar{b}, x^e)$ to $(\bar{b}', x^{e'})$ under joint MA $\bar{\pi}$ in k timesteps is derived,

$$\begin{aligned} P(\bar{b}', x^{e'}, k|\bar{b}, x^e; \bar{\pi}) &= P(x_k^{e'}, \bar{b}'_k|x_0^e, \bar{b}_0; \bar{\pi}) \quad (5) \\ &= \sum_{x_{k-1}^e, \bar{b}_{k-1}} [P(x_k^{e'}|x_{k-1}^e; \bar{\pi}(\bar{b}_{k-1})) \cdot \\ &P(\bar{b}'_k|x_{k-1}^e, \bar{b}_{k-1}; \bar{\pi}(\bar{b}_{k-1}))P(x_{k-1}^e, \bar{b}_{k-1}|x_0^e, \bar{b}_0; \bar{\pi}(\bar{b}_0))]. \end{aligned}$$

Similarly extending the one-step joint reward $\bar{R}$ of the Dec-POMDP, the generalized reward of joint MA $\bar{\pi}$ taken at initial joint belief $\bar{b}$ and e-state x^e is defined as

$$\bar{R}^\tau(\bar{b}, x^e; \bar{\pi}) = \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t \bar{R}(\bar{x}_t, x_t^e, \bar{u}_t)|P(\bar{x}_0) = \bar{b}, x_0^e = x^e; \bar{\pi}\right] \quad (6)$$

where $\tau = \min_i \min_t \{t : b_t^{(i)} \in B^{(i),goal}\}$ is the timestep at which any robot i completes its current MA, $\pi^{(i)}$.

The solution to the Dec-POSMDP is a joint high-level decentralized policy, $\bar{\phi} = \{\phi^{(1)}, \cdots, \phi^{(n)}\}$, where $\phi^{(i)} : \Xi^{(i)} \rightarrow \mathbb{T}^{(i)}$ is the high-level policy for the i-th robot, mapping its MA-history to a subsequent MA to be executed.

The joint value of the decentralized policy $\bar{\phi}$, starting from joint belief $\bar{b}$ and e-state x^e is

$$\begin{aligned} \bar{V}^{\bar{\phi}}(\bar{b}, x^e) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^{t_k} \bar{R}^\tau(\bar{b}_{t_k}, x_{t_k}^e; \bar{\pi}_{t_k})|\bar{b}_0, x_0^e; \bar{\phi}\right] \quad (7) \\ &= \bar{R}^\tau(\bar{b}, x^e; \bar{\pi}) + \sum_{k=1}^{\infty} \sum_{\bar{b}', x^{e'}} \gamma^{t_k} P(\bar{b}', x^{e'}, k|\bar{b}, x^e; \bar{\pi}) \bar{V}^{\bar{\phi}}(\bar{b}', x^{e'}). \end{aligned}$$

The optimal Dec-POSMDP policy is then,

$$\bar{\phi}^* = \underset{\bar{\phi}}{\operatorname{argmax}} \bar{V}^{\bar{\phi}}(\bar{b}, x^e). \quad (8)$$

Fig. 1 provides a visual summary of the hierarchical Dec-POSMDP decision-making scheme. Solving the Dec-POSMDP results in the high-level joint policy, $\bar{\phi}$, dictating

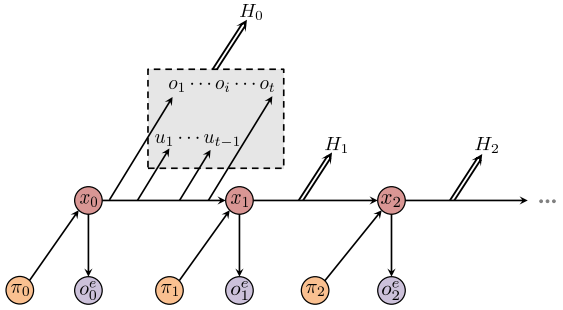


Fig. 2: Overview of the Dec-POSMDP sequential decision-making process. The i-th robot moves from state x_0 through a sequence of MAs, $\pi_0, \pi_1, \cdots, \pi_k$. Under each MA, it executes primitive actions $u_1, \cdots, u_{t-1}$ and perceives primitive observations $o_1, \cdots, o_t$. At the end of each MA, it also obtains environmental observations $o_0^e, \cdots, o_k^e$.

the MA each robot should take based on its MA-history (the history of executed MAs and received MA-observations). The selected MA, $\pi^{(i)}$, for each robot i is itself a low-level policy mapping the robot's history of action-observations $H^{(i)}$ to its next low-level (primitive) action, $u^{(i)} \in \mathbb{U}$ (Fig. 1a). Thus, the Dec-POSMDP solves the same base Dec-POMDP problem and results in a sequential decision-making process eventually leading to low-level actions and observations (Fig. 2). However, its efficacy comes from solving the problem in a hierarchical manner — first selecting the MA for each robot, and then selecting the low-level action.

Although Dec-POSMDPs significantly reduce the Dec-POMDP problem through the use of MAs and the resulting hierarchical decision-making scheme, the optimization problem in (8) is still challenging to solve. Section II-C outlines a graph-based policy representation for infinite-horizon Dec-POSMDPs. Section III presents a probabilistic algorithm to search for the optimal joint policy.

C. Policy Representation as FSAs

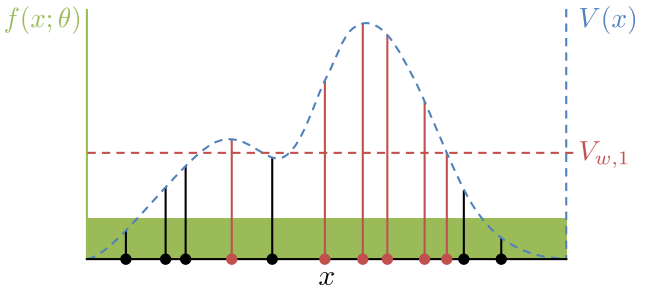
Recall that the joint policy $\bar{\phi}$ is the collection of the individual policies of the robots $\{\phi^{(1)}, \cdots, \phi^{(n)}\}$. Policy execution is done sequentially (yet asynchronously) by the robots. Specifically, the i-th robot executes MA $\pi_k^{(i)}$ at the k-th timestep. The robot will then receive its MA-observation, $\breve{o}_{k+1}^{e(i)}$, which it uses to select its next MA, $\pi_{k+1}^{(i)}$.

Each robot can represent its policy using a tree or graph-based controller (Fig. 1), where the nodes represent the MAs the robot should execute and edges represent the MA-observation the robot receives after the MA execution. Formally, given controller node $q^{(i)}$, the decision tree/graph output function $\pi^{(i)} = \lambda^{(i)}(q^{(i)})$ designates MA $\pi^{(i)}$ to the i-th robot (Fig. 1b and 1c). Following this MA, the robot transitions to the next decision node as determined by the transition function $q'^{(i)} = \delta^{(i)}(q^{(i)}, \breve{o}^{e(i)})$.

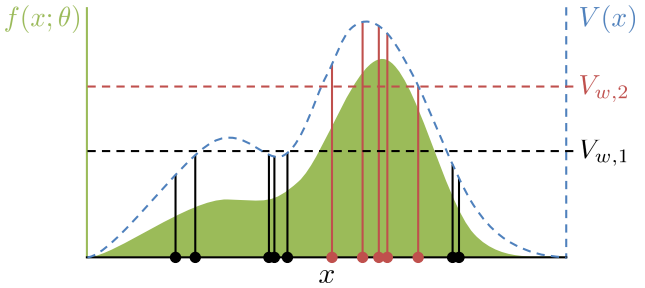
Note that a tree with infinite levels is required to store the policy for an infinite-horizon Dec-POSMDP (Fig. 1a), making trees feasible only for finite-horizon problems. On the other hand, graphs or Finite State Automata (FSAs) allow solving of infinite-horizon Dec-POSMDPs since loops in the policy graph allow it to be executed indefinitely (Fig. 1c). The number of nodes, N_n, in the FSA can be chosen a priori in order to constrain the policy size, alleviating out-of-memory issues which are prevalent in domains with long time horizons [25]. Although an optimal policy may require an infinite-sized graph for full representation [16], it has been shown that a finite-sized FSA can be used to represent a policy within ϵ of the optimal value [8].

III. Solving Dec-POSMDPs using Probabilistic Optimization

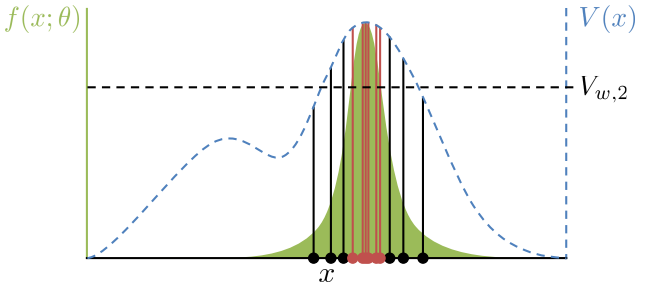
This section introduces a probabilistic algorithm for solving Dec-POSMDPs using a graph-based policy representation. Existing methods for solving Dec-POSMDPs include Masked Monte Carlo Search (MMCS), which is heuristics-based and has no probabilistic guarantees, and Monte Carlo Search (random sampling of the joint policy space), which has poor empirical performance for large domains [22].



(a) Iteration 0. Since $V_{w,0}$ is initialized to $-\infty$, all $N_b = 6$ samples (red dots) are used to estimate θ_1 for the next iteration.



(b) Iteration 1. Note that one of the current iteration's N_b samples has a value lower than $V_{w,1}$ (the minimum value from the previous iteration's N_b samples). Thus, this iteration only uses the best 5 samples (not 6) to estimate θ_2 for the next iteration.



(c) Iteration 2. $f(x;\theta_k)$ incrementally converges towards a Dirac delta distribution centered at the optimal value.

Fig. 3: Overview of the CE method, with $f(x;\theta_k)$ and $V(x)$ plotted in green and blue, respectively. In this example, $N_k = 3$ (number of iterations), $N_s = 12$ (number of samples per iteration), and $N_b = 6$ (number of best samples used to update θ_k in each iteration). The circles anchored to the x axis indicate the samples drawn from $f(x;\theta_k)$ in each iteration, with the N_b best samples highlighted in red.

A. Cross-Entropy Method

The Cross-Entropy (CE) method is a probabilistic approach to stochastic optimization, targeted towards combinatorial optimization problems [23]. It performs well in large search spaces with many local optima, and has previously been used for policy search in MDPs [17] and Dec-POMDPs [19], where it was named the Direct CE Method (DICE) due to its direct search of the un-pruned joint policy space.

For a given optimization problem,

$$x^* = \underset{x}{\operatorname{argmax}}\, V(x), \tag{9}$$

the CE method maintains a sampling distribution $f(x;\theta)$, parameterized by θ, which it uses to sample solutions x in the

search space. To solve (9), the CE method iteratively updates $f(x;\theta)$ such that samples drawn from it get progressively closer to the global optimum. This process is summarized as follows (see Fig. 3):

1) Generate a set of samples $\mathbf{X}$ from $f(x;\theta_k)$, where k is the iteration number and θ_0 is the initial parameter vector. Fig. 3 shows samples as circles on the x axis.
2) Use the best N_b samples $\mathbf{X_b}$ to calculate the Maximum Likelihood Estimate (MLE) of the parameter vector, $\theta_{k+1} = \text{argmax}_\theta f(\mathbf{X_b};\theta)$. Note that samples with value $V(x)$ less than $V_{w,k}$ (the worst-performing sample from the previous iteration's best N_b samples) are rejected to counter convergence towards local optima. Fig. 3b illustrates an example of sample rejection.
3) Apply smoothed update to θ_{k+1} (to further counter convergence towards local optima)

$$\theta_{k+1} \leftarrow \alpha\theta_{k+1} + (1-\alpha)\theta_k, \tag{10}$$

where $\alpha \in (0,1]$ is the learning rate.
4) Repeat until convergence, return best sample x_b.

This process minimizes the KL-divergence between indicator function $I_{(V(x)\geq V_{w,k})}$ and $f(x;\theta_k)$,

$$D_{KL}(I_{(V(x)\geq V_{w,k})}||f(x;\theta_k)), \tag{11}$$

and empirically causes $f(x;\theta_k)$ to converge towards a Dirac delta distribution centered at the optimal value [23] (Fig. 3c).

B. G-DICE

The CE method is extended to solve Dec-POSMDPs while using FSAs for policy representation. The resulting algorithm is called Graph-based Direct Cross Entropy (G-DICE).

Recall that the FSA for each robot, i, is characterized by two functions, the output function $\lambda^{(i)}(q^{(i)})$ and the transition function $\delta^{(i)}(q^{(i)}, \breve{o}^{e(i)})$. Thus, G-DICE maintains two probability distributions at each FSA node $q^{(i)}$:

1) MA output function $f(\pi^{(i)}|q^{(i)};\theta_k^{(i)(\pi|q)})$, parameterized by $\theta_k^{(i)(\pi|q)}$.
2) FSA transition function $f(q^{(i)'}|q^{(i)}, \breve{o}^{e(i)};\theta_k^{(i)(q'|q,\breve{o}^e)})$, parameterized by $\theta_k^{(i)(q'|q,\breve{o}^e)}$.

For a simple implementation, a categorical underlying sampling distribution can be used (although more complex distributions may benefit certain domains). For each robot, both parameters are updated using the CE method. The result is a policy dictating deterministic selection of MAs $\pi^{(i)}$ and node transitions based on each robot's MA-observations.

G-DICE is outlined Alg. 1. To search for the joint policy using G-DICE, an initial FSA with N_n nodes and $|\bar{\breve{\mathbb{O}}}^e|$ outgoing edges from each node is created (Alg. 1, Line 4), where N_n is chosen empirically based on system memory limitations and desired accuracy of the joint policy. The best-joint-value-so-far, $\bar{V}_b$, and worst-joint-value-so-far, $\bar{V}_w$, are initialized to $-\infty$ (Alg. 1, Lines 5-6). The parameter vectors $\theta_k^{(i)(\pi|q)}$ and $\theta_k^{(i)(q'|q,\breve{o}^e)}$ are initialized either using a priori knowledge, or more typically such that their associated distributions are uniform (Alg. 1, Lines 8-9).

In each iteration, batches of N_s policies are sampled using parameter vectors $\theta_k^{(i)(\pi|q)}$ and $\theta_k^{(i)(q'|q,\breve{o}^e)}$ (Alg. 1, Line 14).

Algorithm 1: G-DICE

1 **Procedure** : G-DICE($\bar{\mathbb{T}}, \bar{\breve{\mathbb{O}}}^e, \mathbb{I}, N_n, N_k, N_s, N_b, \alpha$)
2 **Input** : MA space $\bar{\mathbb{T}}$, MA-observation space $\bar{\breve{\mathbb{O}}}^e$, robots $\mathbb{I}$, number of nodes in graph N_n, number of iterations N_k, number of samples per iteration N_s, number of best samples retained N_b, learning rate α
3 **Output** : best joint policy $\bar{\phi}_b$
4 For each robot, initialize policy graph with N_n nodes and $|\bar{\breve{\mathbb{O}}}^e|$ edges per node;
5 $\bar{V}_b \leftarrow -\infty$;
6 $\bar{V}_{w,0} \leftarrow -\infty$;
7 **for** $i = 1$ *to* n **do**
8 Initialize $\theta_0^{(i)(\pi|q)} \ \ \forall q$;
9 Initialize $\theta_0^{(i)(q'|q,\breve{o})} \ \ \forall q, \breve{o}$;
10 **for** $k = 0$ *to* $N_k - 1$ **do**
11 $\bar{\phi}_{list} \leftarrow \varnothing$;
12 **for** $s = 1$ *to* N_s **do**
13 **for** $i = 1$ *to* n **do**
14 $\phi^{(i)} \leftarrow$ sample π from $f(\pi|q;\theta_k^{(i)(\pi|q)}) \ \forall q$, sample q' from $f(q'|q,\breve{o}^e;\theta_k^{(i)(q'|q,\breve{o}^e)}) \ \forall q, \breve{o}$;
15 $\bar{V}^{\bar{\phi}} \leftarrow$ Evaluate $\bar{\phi}$=$\{\phi^{(1)}, \cdots, \phi^{(n)}\}$;
16 **if** $\bar{V}^{\bar{\phi}} \geq \bar{V}_{w,k}$ **then**
17 $\bar{\phi}_{list} \leftarrow \bar{\phi}_{list} \cup \bar{\phi}$;
18 $\bar{V}_{list} \leftarrow \bar{V}_{list} \cup \bar{V}^{\bar{\phi}}$;
19 **if** $\bar{V}^{\bar{\phi}} > \bar{V}_b$ **then**
20 $\bar{V}_b \leftarrow \bar{V}^{\bar{\phi}}$;
21 $\bar{\phi}_b \leftarrow \bar{\phi}$;
22 $\bar{\phi}_{b,list}, \bar{V}_{b,list} \leftarrow$ best N_b policies in $\bar{\phi}_{list}$;
23 $\bar{V}_{w,k+1} \leftarrow \min(\bar{V}_{b,list})$;
24 **for** $i = 1$ *to* n **do**
25 $\theta_{k+1}^{(i)(\pi|q)} \leftarrow$ MLE of $\theta^{(i)(\pi|q)}$ using $\bar{\phi}_{b,list} \ \ \forall q$;
26 $\theta_{k+1}^{(i)(\pi|q)} \leftarrow \alpha\theta_{k+1}^{(i)(\pi|q)} + (1-\alpha)\theta_k^{(i)(\pi|q)}$;
27 $\theta_{k+1}^{(i)(q'|q,\breve{o})} \leftarrow$ MLE of $\theta^{(i)(q'|q,\breve{o})}$ using $\bar{\phi}_{b,list} \ \ \forall q, \breve{o}$;
28 $\theta_{k+1}^{(i)(q'|q,\breve{o})} \leftarrow \alpha\theta_{k+1}^{(i)(q'|q,\breve{o})} + (1-\alpha)\theta_k^{(i)(q'|q,\breve{o}^e)}$;
29 **return** $\bar{\phi}_b$;

The policies are then evaluated using (7). Policies with value lower than the previous iteration's worst value, $\bar{V}_{w,k}$, are rejected (Alg. 1, Lines 17-18) and the best-policy-so-far, $\bar{\phi}_b$, is noted (Alg. 1, Line 21). This process can be performed very efficiently through parallelization, since each sampled joint policy $\bar{\phi}$ is evaluated independently.

Following this, the best N_b policies from the list of sampled policies, $\bar{\phi}_{list}$, are used to calculate MLEs of parameter vectors (Alg. 1, Line 25 and Line 27). Parameter vectors are then updated in a smooth manner (Alg. 1, Line 26 and Line 28) and the process is repeated until N_k iterations are completed. Since the parameters and the best-policy-so-far, $\bar{\phi}_b$, are updated in each iteration, G-DICE can be stopped at any point to produce an approximation of the optimal joint policy as $\bar{\phi}_b \approx \bar{\phi}^*$. G-DICE is an anytime algorithm offering several

advantages to the tree-based approaches utilized previously for Dec-POMDPs [19]. Usage of graphs allows solving infinite-horizon Dec-POSMDPs, since loops in the policy graph allow it to be executed indefinitely. Additionally, the number of nodes in the graph, N_n, can be fixed a priori to impose memory bounds on the policy, alleviating the memory issues that are prevalent in problems with long time horizons [25].

C. Complexity Analysis

For a Dec-POMDP with n robots and finite horizon h, the total number of possible joint policies is

$$O\Big[(|\mathbb{U}^*|^{\frac{|\Omega^*|^h-1}{|\Omega^*|-1}})^n\Big], \tag{12}$$

where $|\mathbb{U}^*| = \max\{|\mathbb{U}|^{(1)}, \cdots, |\mathbb{U}^{(n)}|\}$ and $|\Omega^*| = \max\{|\Omega^{(1)}|, \cdots, |\Omega^{(n)}|\}$, the largest action-set and observation-set for any robot [19]. In [19], the cost of evaluating each policy using (2) is shown to be

$$O\Big[|\bar{\mathbb{S}}| \cdot \frac{|\Omega^*|^{nh}-1}{|\Omega^*|^n-1}\Big]. \tag{13}$$

For comparison, the complexity of solving Dec-POSMDPs using G-DICE can be considered by analyzing the joint policy sampling and evaluation steps, which are the most computationally-intensive portions of Alg. 1.

In Alg. 1, Line 14, a joint policy (FSA) is constructed by sampling a MA from $f(\pi^{(i)}|q^{(i)};\theta_k^{(i)(\pi|q)})$, which has complexity $O[N_n \cdot |\mathbb{T}^*|]$, where $|\mathbb{T}^*| = \max\{|\mathbb{T}|^{(1)}, \cdots, |\mathbb{T}^{(n)}|\}$. Additionally, in each FSA node, the transition to the next node is sampled from $f(q^{(i)'}|q^{(i)}, \breve{o}^{e(i)};\theta_k^{(i)(q'|q,\breve{o}^e)})$ based on each robot's MA-observation history, which has complexity

$$O\Big[(N_n)^2 \cdot \frac{|\breve{\mathbb{O}}^{e*}|^h-1}{|\breve{\mathbb{O}}^{e*}|-1}\Big]. \tag{14}$$

For most domains, this term dominates FSA construction. In each of the N_k iterations in G-DICE, N_s FSAs are constructed for each of the n robots. Therefore, the computational complexity of FSA construction in G-DICE is

$$O\Big[n \cdot N_k \cdot N_s \cdot (N_n)^2 \cdot \frac{|\breve{\mathbb{O}}^{e*}|^h-1}{|\breve{\mathbb{O}}^{e*}|-1}\Big]. \tag{15}$$

In Alg. 1, Line 15, the joint policy is evaluated, which involves considering all possible state-observation histories of n robots. This evaluation occurs in each of the N_s samples for each of the N_k iterations, resulting in complexity

$$O\Big[N_k \cdot N_s \cdot |\bar{\mathbb{S}}| \cdot \frac{|\breve{\mathbb{O}}^{e*}|^{nh}-1}{|\breve{\mathbb{O}}^{e*}|^n-1}\Big]. \tag{16}$$

IV. Experiments

This section presents a constrained multi-robot package delivery domain, as well as experimental results comparing G-DICE and MMCS [22] joint policy quality. The considered domain poses significant challenges for decentralized decision-making, due to its use of joint MAs requiring cooperation between multiple robots. The significant size of the policy search space ($5.622e+17$ cardinality when using a FSA with $N_n = 13$ nodes) makes this problem extremely difficult for existing Dec-POMDP methods, thus making it a useful benchmark domain for Dec-POSMDPs.

A. Domain Overview

The constrained package delivery domain considered was introduced in [22] and a single-robot POMDP variant was considered in [2]. This domain involves a team of robots retrieving packages from 2 base locations and transporting them to 3 delivery destinations. The team consists of two aerial robots (e.g., quadrotors) and one ground robot. The aerial robots can handle pickup and deliveries outside a regulated airspace, whereas the ground robot is restricted to deliveries within the regulated airspace. The aerial robots can transfer packages to the ground robot at a rendezvous destination. Each package has a size and delivery destination descriptor, which is observed by the robots as an environment observation o^e. The team is rewarded only when a package is delivered to the correct delivery destination.

Single and multi-robot MAs are used in this domain, and can be generated automatically using the procedure outlined in [22]. Due to the underlying sources of uncertainty in the domain, the MAs have probabilistic success rate and completion times. The MAs are summarized as follows:

- Go to base $Base_j$ for $j \in \{1, 2\}$
- Go to delivery destination $Dest_j$ for $j \in \{1, 2, r\}$, where $Dest_r$ is the destination in the regulated airspace.
- Jointly go to delivery destination $Dest_j$ for $j \in \{1, 2\}$
- Individually/jointly pick up package
- Individually/jointly put down package
- Go to rendezvous location
- Place package on ground robot
- Wait at current location for 1 timestep

Refer to [22] for detailed domain and MA descriptions.

B. Results

Existing Dec-POSMDP solution methods such as MMCS [22] are heuristics-based and generally have no probabilistic guarantees. In this section, MMCS is used as a baseline for comparisons with the proposed algorithm in this paper, G-DICE, applied to the constrained package delivery domain.

Table I presents an overview of joint policy value for G-DICE (14.44), MMCS (4.53), and Monte Carlo Search (2.07). Parameter values used for G-DICE results in this table were $N_k = 100$, $N_s = 100$, $N_b = 10$, $N_n = 13$, and $\alpha = 0.1$. G-DICE significantly outperforms MMCS in policy quality. Due to the large cardinality of the policy space, an exhaustive search for the optimal policy is infeasible.

G-DICE policy value as a function of learning rate, α, is shown in Fig. 4a. As the learning rate increases, the policy tends to converge to a local optima and therefore suffers in quality. This is illustrated in Fig. 4b, which shows convergence characteristics for G-DICE with varying learning rates. Lower learning rates increase convergence time, although with near-optimal policy value. The trade-off between policy quality and convergence time is prevalent in stochastic search algorithms, and recent research has focused on developing adaptive learning rates or removing them altogether for algorithms such as stochastic gradient descent [24]. In the package delivery domain, a learning rate of $\alpha = 0.2$ was empirically found to provide a suitable balance between solution quality and convergence time. This analysis, however, motivates future

Algorithm	Max Policy Value
Monte Carlo Search	2.07
MMCS	4.53
G-DICE	14.44

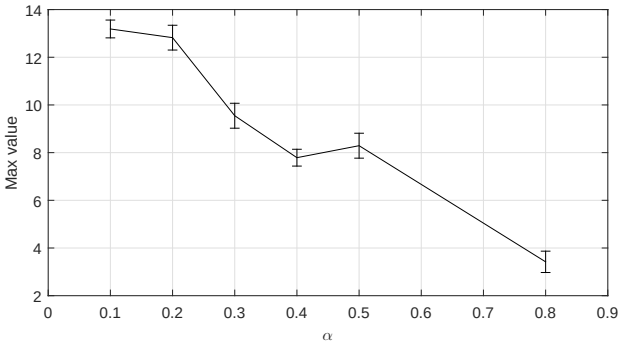
TABLE I: Comparison of the maximum policy values.

work in the area of developing adaptive learning rates for G-DICE, which would be highly beneficial for online planning with limited computational resources.

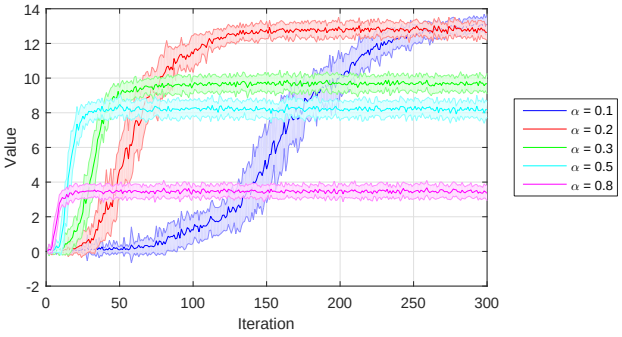
Fig. 5a shows G-DICE policy values as a function of the number of FSA nodes, N_n. Increasing the number of FSA nodes improves solution quality, primarily since the policy fidelity is improved. Larger policy graphs are able to store more accurate representation of action-observation histories for the robots, improving the decision-making. However, as Fig. 5b illustrates, graphs with more nodes require longer time for convergence, since the cardinality of the policy search space increases. Once a large enough N_n is chosen, increasing the number of nodes does not lead to significant improvement in the policy value. In the package delivery example, $N_n \geq 10$ results in policies with similar values. These plots suggest a trade-off related to the graph size, similar to the one presented for learning rate, where increasing policy size improves solution quality at the expense of convergence time. Investigation of automatic determination of graph size using a Bayesian nonparametric approach was done in [15]. A key advantage of the FSA representation is that for a given problem, the number of nodes can be automatically selected in order to adhere to memory limitations.

Case studies of varying policies as a function of controller size can be done using Fig. 5b. For instance, the $N_n = 2$ policy has zero joint value, since there are not enough nodes to perform the sequence of MAs required for even a single package delivery (which needs at least 3 MAs: pickup, transport, and dropoff). The $N_n = 3$ graph, as expected, yields a policy with positive value. This policy involves each robot picking up a package, delivering it to one of the destinations, dropping it off, and remaining at the delivery destination. Once the delivery is completed using the $N_n = 3$ policy, the combination of MAs possible in a 3-node graph have been exhausted, and the robot does not have the ability to travel to a base location for an additional pickup. This policy is a direct result of the constraints imposed on graph size, and serves as a check to ensure that the joint policies produced by G-DICE are intuitively valid.

To further quantify the performance advantage of G-DICE over MMCS and Monte Carlo Search, Fig. 6 compares success probability of delivering a given minimum number of packages for all 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 Search policy, probability of successfully delivering 3 or more packages given a fixed time horizon is near-zero. The MMCS policy successfully delivers a larger number of packages, although struggling to deliver more than 4 or 5. The G-DICE policy (with $N_n = 13$ and $\alpha = 0.2$) clearly outperforms the other methods, delivering up to 3 packages with probability 1.0, and up to 6 packages with probability of approximately 0.8. The probability of delivering



(a) Max joint value obtained for the package delivery domain using G-DICE, with varying learning rate α.



(b) Effect of learning rate, α, on convergence.

Fig. 4: G-DICE policy results for constrained package delivery domain, with varying learning rate α.

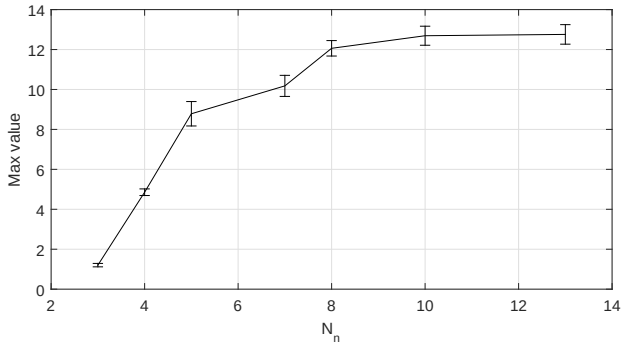
9 or more packages becomes near-zero, due to the imposed time cutoff constraining the number of deliveries possible. These results indicate that G-DICE significantly outperforms MMCS in terms of policy quality. This is primarily due to the greedy policy selection process used in MMCS, which often results in the algorithm getting stuck in local optima. In contrast, G-DICE takes specific steps to avoid this, by applying a smoothed update to its parameter vectors (Alg. 1, Line 26 and Line 28) while tracking performance of the worst policies using $\bar{V}_{w,k}$. While the approach used in [3] can be used for this problem domain, we believe that G-DICE will be more scalable (but without the same convergence guarantees). A direct comparison to [3] will be considered in future work.

V. Conclusion

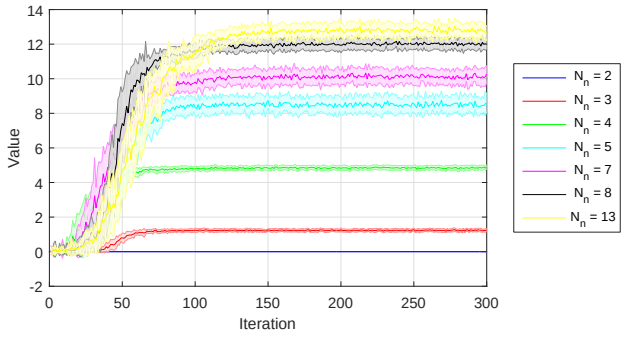
This paper proposed a probabilistic method for solving multi-robot planning under uncertainty problems posed as Dec-POSMDPs. It motivated the solving of Dec-POSMDPs from a combinatorial optimization perspective, presenting a cross-entropy based algorithm, G-DICE, and associated complexity analysis. For the constrained package delivery under uncertainty problem, G-DICE was shown to significantly outperform existing Dec-POSMDP solution methods, resulting in a high-quality joint policy despite the very large cardinality of the policy search space.

References

[1] A.-a. Agha-mohammadi, S. Chakravorty, and N. Amato, "FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 2, pp. 268–304, 2014.



(a) Max joint value obtained for the package delivery domain using G-DICE, with varying policy controller size N_n.



(b) Effect of policy controller size, N_n, on convergence.

Fig. 5: G-DICE policy results for constrained package delivery domain, with varying policy controller sizes N_n.

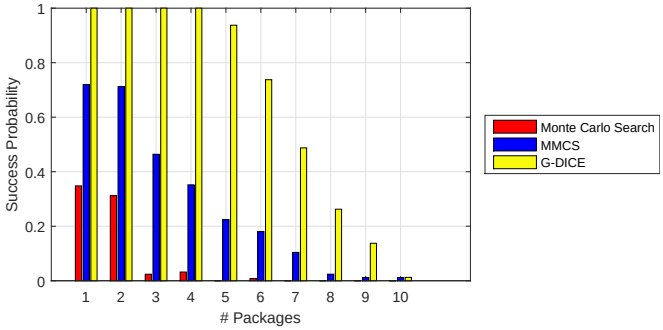


Fig. 6: Success probability of delivering a specified number of packages *or more* within a fixed time horizon, for G-DICE, MMCS [22], and Monte Carlo Search.

[2] A.-a. Agha-mohammadi, N. K. Ure, J. P. How, and J. 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] C. Amato, G. Konidaris, A. Anders, G. Cruz, J. How, and L. Kaelbling, "Policy search for multi-robot coordination under uncertainty," in *Robotics: Science and Systems XI (RSS)*, 2015. [Online]. Available: http://lis.csail.mit.edu/pubs/amato-konidaris-rss15.pdf

[4] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer, "Decentralized control of partially observable Markov decision processes," in *Proceedings of the Fifty-Second IEEE Conference on Decision and Control*, 2013, pp. 2398–2405.

[5] C. Amato, J. S. Dibangoye, and S. Zilberstein, "Incremental policy generation for finite-horizon Dec-POMDPs." in *ICAPS*, A. Gerevini, A. E. Howe, A. Cesta, and I. Refanidis, Eds. AAAI, 2009.

[6] C. Amato, G. D. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling, "Planning for decentralized control of multiple robots under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[7] C. Amato, G. D. Konidaris, and L. P. Kaelbling., "Planning with macro-actions in decentralized POMDPs," in *International Conference on Autonomous Agents and Multiagent Systems*, 2014.

[8] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein, "Policy iteration for decentralized control of Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 34, pp. 89–132, 2009.

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

[10] D. S. Bernstein, E. A. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005, pp. 1287–1292.

[11] A. Boularias and B. Chaib-draa, "Exact dynamic programming for decentralized POMDPs with lossless policy compression." in *ICAPS*, J. Rintanen, B. Nebel, J. C. Beck, and E. A. Hansen, Eds. AAAI, 2008, pp. 20–27.

[12] A. Carlin and S. Zilberstein, "Value-based observation compression for Dec-POMDPs." in *AAMAS (1)*, L. Padgham, D. C. Parkes, J. P. Mller, and S. Parsons, Eds. IFAAMAS, 2008, pp. 501–508.

[13] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238 – 1274, September 2013.

[14] Z. Lim, L. Sun, and D. J. Hsu, "Monte carlo value iteration with macro-actions," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 1287–1295.

[15] M. Liu, C. Amato, X. Liao, J. P. How, and L. Carin, "Stick-Breaking Policy Learning in DEC-POMDPs," in *Proceedings of the 24rd International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos aires, Argentina, 2015.

[16] O. Madani, S. Hanks, and A. 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)*, 1999, pp. 541–548.

[17] S. Mannor, R. Y. Rubinstein, and Y. Gat, "The cross entropy method for fast policy search." in *ICML*, T. Fawcett and N. Mishra, Eds. AAAI Press, 2003, pp. 512–519. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2003.html#MannorRG03

[18] 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, pp. 471–503.

[19] F. A. Oliehoek, J. F. Kooi, and N. Vlassis, "The cross-entropy method for policy search in decentralized POMDPs," *Informatica*, vol. 32, pp. 341–357, 2008.

[20] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 289–353, 2008.

[21] F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan, "Lossless clustering of histories in decentralized POMDPs." in *AAMAS (1)*, C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, Eds. IFAAMAS, 2009, pp. 577–584.

[22] S. Omidshafiei, A. akbar Agha-mohammadi, C. Amato, and J. P. How, "Decentralized Control of Partially Observable Markov Decision Processes using Belief Space Macro-actions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[23] Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning*. Springer-Verlag, 2004.

[24] T. Schaul, S. Zhang, and Y. LeCun, "No More Pesky Learning Rates," in *International Conference on Machine Learning (ICML)*, 2013.

[25] S. Seuken, "Memory-bounded dynamic programming for Dec-POMDPs," in *In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI*, 2007, pp. 2009–2015.

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

[27] D. Szer, F. Charpillet, and S. Zilberstein, "Maa*: A heuristic search algorithm for solving decentralized POMDPs," in *In Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 576–583.

[28] D. Szer and F. Charpillet, "An optimal best-first search algorithm for solving infinite horizon Dec-POMDPs." in *ECML*, ser. Lecture Notes in Computer Science, J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, Eds., vol. 3720. Springer, 2005, pp. 389–399.