Scalable Accelerated Decentralized Multi-Robot Policy Search in Continuous Observation Spaces

Shayegan Omidshafiei[1], Christopher Amato[2], Miao Liu[3], Michael Everett[1], Jonathan P. How[1], John Vian[4]

Abstract— **This paper presents the first ever approach for solving *continuous-observation* Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) and their semi-Markovian counterparts, Dec-POSMDPs. This contribution is especially important in robotics, where a vast number of sensors provide continuous observation data. A continuous-observation policy representation is introduced using Stochastic Kernel-based Finite State Automata (SK-FSAs). An SK-FSA search algorithm titled Entropy-based Policy Search using Continuous Kernel Observations (EPSCKO) is introduced and applied to the first ever continuous-observation Dec-POMDP/Dec-POSMDP domain, where it significantly outperforms state-of-the-art discrete approaches. This methodology is equally applicable to Dec-POMDPs and Dec-POSMDPs, though the empirical analysis presented focuses on Dec-POSMDPs due to their higher scalability. To improve convergence, an entropy injection policy search acceleration approach for both continuous and discrete observation cases is also developed and shown to improve convergence rates without degrading policy quality.**

I. Introduction

Decision-making under uncertainty is a ubiquitous robotics problem wherein a robot collects data from its environment and decides subsequent tasks to execute. While low-cost robotics platforms and sensors have increased the affordability of multi-robot systems, derivation of policies dictating robot decisions remains a challenge. This decision-making problem is even more complex in noisy settings with imperfect communication, requiring a formal framework for its treatment.

A general representation of the multi-agent planning under uncertainty problem is the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [1], which extends single-agent POMDPs to decentralized domains. Due to Dec-POMDPs' usage of primitive actions (atomic actions assumed to each take a single time unit to execute) they have exceedingly large policy spaces which severely limits planning scalability. Recent efforts have extended Dec-POMDPs to use macro-actions (temporally extended actions), resulting in the Decentralized Partially Observable *Semi*-Markov Decision Process (Dec-POSMDP) [2], [3]. The result is a scalable asynchronous multi-robot decision-making framework which plans over the space of high-level robot tasks (e.g., *Open-the-valve* or *Find-the-key*) with non-deterministic durations.

*This work was supported by The Boeing Company.

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

[2]College of Computer and Information Science (CCIS), Northeastern University, Boston, MA 02115, USA camato@ccs.neu.edu

[3]IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA miao.liu1@ibm.com

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

Despite the increased action-space scalability offered by Dec-POSMDPs, they have so far been limited to planning over the space of discrete observations. To date, no algorithms exist for continuous-observation Dec-POSMDPs (or Dec-POMDPs [4]). This is a major research gap, especially important in the context of robotics where a vast number of real-world sensors provide continuous observation data. Application of Dec-POSMDPs to continuous problems such as robot navigation currently mandates observation space discretization, resulting in loss of valuable sensor information which could otherwise be used to better inform the decision-making policy. Several approaches have targeted single-agent continuous-observation POMDPs. These include partitioning of continuous spaces into lossless discrete spaces [5], Gaussian mixtures for belief representation [6], use of continuous-observation classifiers [7], and learned discrete representations for continuous state spaces [8]. This paper expands this body of work beyond the single-agent case, targeting scalable treatment of continuous-observation Dec-POSMDPs. The methods presented are applicable to domains with continuous underlying state spaces, as shown in some of the experiments used for evaluation.

In order to develop solvers for continuous-observation Dec-POSMDPs, we build on current state-of-the-art discrete policy search methods [2], [3], [9]. Unfortunately, these algorithms suffer from convergence speed limitations—an issue which was identified in prior work but remains untreated [9]. A major gap exists in addressing these issues before extending the foundations of these discrete algorithms to the continuous case, where such convergence issues are exacerbated. To resolve this, we first introduce a maximal entropy injection approach targeting convergence acceleration for both discrete and continuous algorithms, without degrading overall policy quality. The approach is shown to significantly outperform existing search acceleration methods.

The paper's key contribution is a stochastic kernel-based policy representation and search algorithm, allowing direct mapping of continuous observations to robot decisions (with no discretization necessary). This algorithm leverages the proposed entropy injection acceleration method and is evaluated on a multi-robot nuclear contamination domain—the first ever continuous-observation Dec-POMDP/Dec-POSMDP domain—in which discrete policy search algorithms perform extremely poorly. Failure modes of discrete methods are analyzed and compared to the superior continuous policy behavior. The contributions introduced in this paper can be readily applied to Dec-POMDPs *and* Dec-POSMDPs. However, as we are motivated by applications to extremely large action-observation spaces, the notation used and experiments conducted focus on the more scalable Dec-POSMDP framework.

II. Background

A. Decentralized Planning using Macro-Actions

This section summarizes the Dec-POSMDP, a multi-robot decentralized decision-making under uncertainty framework targeting action-space scalability. For a more detailed introduction to Dec-POSMDPs, we refer readers to [2], [3], [9].

The Dec-POSMDP is a belief-space framework in which agents execute macro-actions (temporally-extended actions) with non-deterministic completion times, and receive noisy high-level observations of their post-MA state. Macro-actions (MAs) are abstractions of low-level POMDPs involving primitive actions $u_t^{(i)}$ and observations $o_t^{(i)}$, allowing execution of high-level tasks (e.g., *Park-the-car*)[1]. Each MA executes until an ϵ-neighborhood of its belief milestone $\check{b}^{goal}$ is reached. This neighborhood defines the MA termination condition or *goal belief node*, denoted $B^{goal}=\{b : \|b-\check{b}^{goal}\| \leq \epsilon\}$ [3].

Upon completion of an MA, each robot makes a macro (or high-level) observation $o^{e(i)}$ of the underlying high-level system state $x^e \in \mathbb{X}^e$. It also calculates its own final belief state, $b^{f(i)}$. Thus far, both Dec-POMDPs and Dec-POSMDPs have only seen limited applications to *finite discrete* observation spaces. Due to its action-space scalability, let us focus on the Dec-POSMDP, defined as follows:

- $\mathbb{I}=\{1,2,\ldots,n\}$ is the set of heterogeneous robots.
- $\mathbb{B}^{(1)}\times\mathbb{B}^{(2)}\times\ldots\times\mathbb{B}^{(n)}\times\mathbb{X}^e$ is the belief space, with local belief milestones $\mathbb{B}^{(i)}$ and joint environment (or high-level) space $\mathbb{X}^e$.
- $\bar{\mathbb{T}}=\mathbb{T}^{(1)}\times\mathbb{T}^{(2)}\ldots\times\mathbb{T}^{(n)}$ is the joint MA space, where $\mathbb{T}^{(i)}$ is the finite set of MAs for the i-th robot.
- $\bar{\breve{\mathbb{O}}}^e=\{\bar{\breve{o}}^e\}$ is the space of all joint MA-observations.
- $P(\bar{b}',x^{e'},k|\bar{b},x^e;\bar{\pi})$ is the high-level transition probability model under MAs $\bar{\pi}$ from $(\bar{b},x^e)$ to $(\bar{b}',x^{e'})$.
- $\bar{R}^\tau(\bar{b},x^e;\bar{\pi})$ is the high-level reward of taking a joint MA $\bar{\pi}$ at $(\bar{b},x^e)$.
- $P(\bar{\breve{o}}^e|\bar{b},x^e)$ is the joint observation likelihood model, with joint observation $\bar{\breve{o}}^e=\{\breve{o}^{e(1)},\breve{o}^{e(2)},\ldots,\breve{o}^{e(n)}\}$.
- $\gamma\in[0,1)$ is the reward discount factor.

Macro-observations and final beliefs are jointly denoted as *MA-observation* $\breve{o}^{e(i)}=(o^{e(i)},b^{f(i)})$. Trajectories of MAs and received MA-observations are denoted as the *MA-history*,

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

Transition probability $P(\bar{b}',x^{e'},k|\bar{b},x^e;\bar{\pi})$ from $(\bar{b},x^e)$ to $(\bar{b}',x^{e'})$ under joint MA $\bar{\pi}=\{\pi^{(1)},\ldots,\pi^{(n)}\}$ in k timesteps is,

$$\begin{aligned}P(\bar{b}',x^{e'},k|\bar{b}_0,x_0^e,o_k^e;\bar{\pi})&=P(x_k^e,\bar{b}_k|\bar{b}_0,x_0^e,o_k^e;\bar{\pi})\\&=\sum_{x_{k-1}^e,\bar{b}_{k-1}}\Big[P(x_k^e|x_{k-1}^e,o_k^e;\bar{\pi}(\bar{b}_{k-1}))\times \quad (2)\\&P(\bar{b}_k|x_{k-1}^e,\bar{b}_{k-1};\bar{\pi}(\bar{b}_{k-1}))P(x_{k-1}^e,\bar{b}_{k-1}|x_0^e,\bar{b}_0;\bar{\pi}(\bar{b}_0))\Big].\end{aligned}$$

The generalized high-level team reward for a discrete-time Dec-POSMDP during execution of joint MA $\bar{\pi}$ is defined [9],

$$\bar{R}^\tau(\bar{b},x^e;\bar{\pi})=\mathbb{E}\left[\sum_{t=0}^{\tau-1}\gamma^t\bar{R}(\bar{x}_t,x_t^e,\bar{u}_t)|P(\bar{x}_0)=\bar{b},x_0^e=x^e;\bar{\pi}\right] \quad (3)$$

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

The *joint high-level policy*, $\bar{\phi}=\{\phi^{(1)},\ldots,\phi^{(n)}\}$, dictates MA selection. High-level policy $\phi^{(i)}$ maps the i-th robot's MA-history $\xi_k^{(i)}$ to the next MA $\pi^{(i)}$ to be executed. Joint Dec-POSMDP value under policy $\bar{\phi}$ is then [9],

$$\begin{aligned}\bar{V}^{\bar{\phi}}(\bar{b},x^e)&=\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{t_k}\bar{R}^\tau(\bar{b}_{t_k},x_{t_k}^e;\bar{\pi}_{t_k})|\bar{b}_0,x_0^e;\bar{\phi}\right] \quad (4)\\&=\bar{R}^\tau(\bar{b},x^e;\bar{\pi})+\\&\sum_{k=1}^{\infty}\gamma^{t_k}\sum_{\bar{b}',x^{e'},o^{e'}}P(\bar{b}',x^{e'},o^{e'},k|\bar{b},x^e;\bar{\pi})\bar{V}^{\bar{\phi}}(\bar{b}',x^{e'}). \quad (5)\end{aligned}$$

The optimal joint high-level policy is,

$$\bar{\phi}^*=\operatorname*{argmax}_{\bar{\phi}}\bar{V}^{\bar{\phi}}(\bar{b},x^e). \quad (6)$$

Solving the Dec-POSMDP results in joint high-level decision-making policy $\bar{\phi}$ dictating the MA $\pi^{(i)}$ executed by each robot based on its MA-history. Each MA is, itself, a policy over low-level actions $u_t^{(i)}$ and observations $o_t^{(i)}$. Thus, decision-making using the Dec-POSMDP allows abstraction of task-level actions from low-level actions, leading to significantly improved planning scalability over Dec-POMDPs.

B. Dec-POSMDP Policy Search Algorithms

So far, research efforts have focused on Dec-POSMDP policy search for discrete observation spaces, resulting in several algorithms: Masked Monte Carlo Search (MMCS) [3], MacDec-POMDP Heuristic Search (MDHS) [2], and Graph-based Direct Cross Entropy method (G-DICE) [9]. These algorithms use Finite State Automata (FSAs) for policy representation. FSA-based policy $\phi^{(i)}$ for robot i consists of N_n FSA nodes, $\{q_1^{(i)},\ldots,q_{N_n}^{(i)}\}$. FSA-based decision-making is two-fold: each robot begins execution in FSA node $q^{(i)}$, where *MA output function* $\pi^{(i)}=\lambda^{(i)}(q^{(i)})$ assigns it an MA, $\pi^{(i)}$. Following MA execution, the robot receives a high-level observation and selects its next FSA node using *transition function* $q'^{(i)}=\delta^{(i)}(q^{(i)},\breve{o}^{e(i)})$. The graph-based nature of FSAs allows their application to infinite-horizon domains.

Though Dec-POSMDPs have increased the size of solvable planning domains beyond Dec-POMDP counterparts, major algorithm limitations still exist. MMCS is a greedy algorithm which succumbs to local optimality issues [3]. MDHS uses lower and upper bound value heuristics to bias search towards promising policy regions, by initiating an empty (partial) FSA and incrementally assigning nodes actions $\lambda^{(i)}$ and transitions $\delta^{(i)}$. Partial policies with high upper bounds are expanded incrementally. Yet, each expansion involves $|\mathbb{T}|N_n^{|\breve{\mathbb{O}}|}$ child policies, severely limiting usage for large observation spaces.

G-DICE is a cross entropy-based algorithm which iteratively updates policies using two *sampling distributions* at each FSA node: MA distribution $f(\pi^{(i)}|q^{(i)};\theta^{(i)(\pi|q)})$

[1]We denote a generic parameter p of the i-th robot as $p^{(i)}$, a joint team parameter as $\bar{p}$, and a joint team parameter at timestep k as $\bar{p}_k$.

and node transition distribution $f(q^{(i)'}|q^{(i)}, \breve{o}^{e(i)}; \theta^{(i)(q'|q,\breve{o}^e)})$, where $\theta^{(i)}$ are parameter vectors. Each iteration samples the distributions N_s times, resulting in N_s deterministic FSA policies. Maximum likelihood estimates (MLE) of parameters $\theta^{(i)}$ are calculated using the $N_b \leq N_s$ best policies. To prevent convergence to local optima, smooth parameter updates,

$$\theta_{k+1} \leftarrow \alpha\theta_{k+1} + (1-\alpha)\theta_k, \tag{7}$$

are used, with iteration number k and learning rate $\alpha \in (0, 1]$. For sufficiently small values of α, this process minimizes cross entropy between each sampling distribution and a unit mass centered at the optimal policy [10]. G-DICE is executed until convergence, after which the best *deterministic* policy from the history of samples is returned.

Using smooth parameter updates and sampling distributions initiating from a uniform distribution allows G-DICE to tradeoff exploration and exploitation in the policy space, outperforming other Dec-POSMDP search approaches given a fixed computational budget. Yet, G-DICE suffers from sample degeneracy and convergence issues related to the sampling distributions, and in its current form only applies to discrete observation settings. The following sections resolve these issues, resulting in a scalable, accelerated continuous-observation search algorithm.

III. Accelerated Policy Search

Prior to extending to continuous observations, this section treats the sampling distribution degeneracy issue in sampling-based Dec-POSMDP approaches. It also introduces a maximal entropy injection scheme which is then embedded in the proposed continuous-observation Dec-POSMDP algorithm.

A. Sampling Distribution Degeneracy Problem

A major issue with sampling distribution-based approaches, such as G-DICE, occurs when a low enough learning rate α is not used, causing underlying sampling distributions to rapidly converge to *degenerate distributions* far from the optimum [11]. All subsequent search iterations return identical samples of the policy space, stifling exploration altogether. Yet, one benefit of a high learning rate is fast convergence, especially useful for complex Dec-POSMDPs with large observation spaces and computationally expensive trajectory sampling and evaluation. Sampling distribution-based approaches such as G-DICE often require *hand-tuned* selection of α for good performance, even after which convergence may be excessively slow and can hinder experimentation and analysis. This trade-off was noted in [9], where it was left as future work. Recall the motivation behind the Dec-POSMDP framework is scalability to very large multi-robot planning domains. Despite the fact that policy search is conducted offline, hindrance of human-in-the-loop analysis due to slow convergence is undesirable. A naïve solution is to set α arbitrarily low, but this implies arbitrarily high convergence time (on the order of many days for complex domains). These foundational issues must first be resolved before extending these algorithms to treat the more complex continuous observation case.

Several works have targeted this degeneracy problem. One approach uses dynamic smoothing of learning rates [12],

$$\alpha_k = \alpha_0 - \alpha_0(1-k^{-1})^{\beta}, \tag{8}$$

where α_0 is the baseline rate (typically close to 1) and β is the drop-off rate (typically between 5 to 10). The result is a monotonically decreasing α_k which initially starts high.

Another approach involves the addition of a noise term ω_k to the sampling distribution at each iteration k to prevent degeneration. Linearly decreasing noise injection,

$$\omega_k = \max(\omega_{\max} - rk, 0), \tag{9}$$

was investigated in [13]. In the above, $\omega_{\max}$ is the maximum allowable noise and r is the noise drop-off rate.

These approaches are not ideal as they are agnostic to Dec-POSMDP value function convergence, meaning they do not adapt to domain-specific behaviors. Thus, sub-parameters (α_0, β, $\omega_{\max}$, r) typically need significant tuning to alleviate convergence issues for individual domains.

B. Maximal Entropy Injection

A principled approach combining policy exploration with fast convergence is desired, without reliance on sensitive dynamic smoothing or noise terms. As degenerate distributions have minimal entropy [14], an intuitive idea is to simultaneously monitor policy value convergence and underlying sampling distribution entropy to alleviate degeneracy issues.

In the proposed acceleration approach, search is conducted as usual for iterations where policy value has not converged, allowing policy space exploration. Once convergence occurs, entropies of sampling distributions $f(\pi^{(i)}|q^{(i)}; \theta^{(i)(\pi|q)})$ and $f(q^{(i)'}|q^{(i)}, \breve{o}^{e(i)}; \theta^{(i)(q'|q,\breve{o}^e)})$ are calculated. If a distribution's entropy is significantly below the max entropy for its distribution family, degeneracy has likely occurred [14]. Max entropy distributions are well-studied and closed form results for many families and constraint sets are known [15]. For Dec-POSMDPs, these entropy calculations are computationally cheap as sampling distributions are categorical, with corresponding discrete uniform maximal entropy distributions.

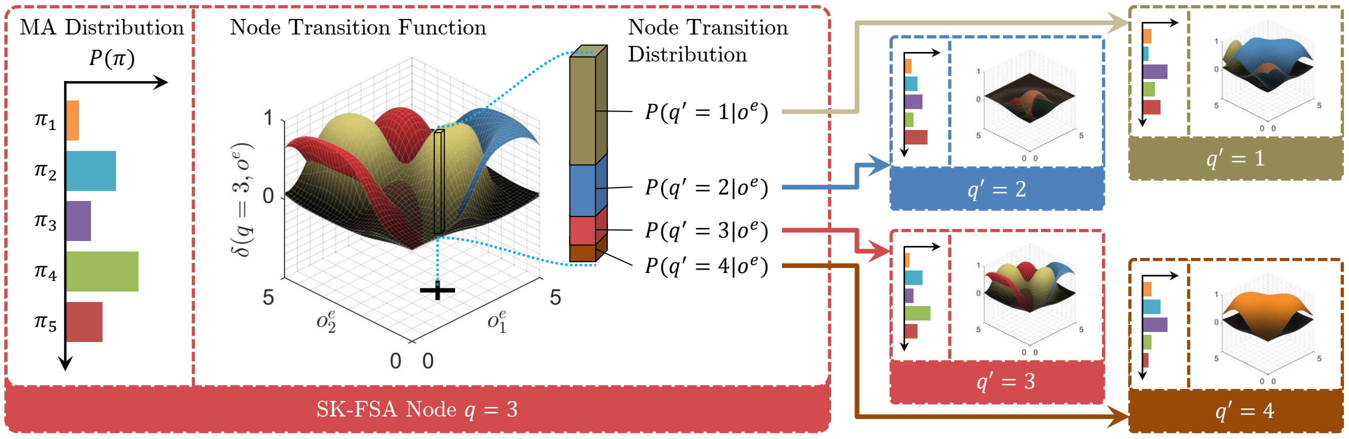
In post-degeneracy iterations, each sampling distribution's entropy is increased by incrementally combining its parameters $\theta^{(i)}$ with the max entropy distribution parameters θ_{ME},

$$\theta_{k+1} \leftarrow (1-\alpha_{EI})[\alpha\theta_{k+1} + (1-\alpha)\theta_k] + \alpha_{EI}\theta_{ME}, \tag{10}$$

where α_{EI} is the entropy injection rate. This encourages policy space exploration while still allowing usage of high learning rates (e.g., $\alpha > 0.5$) for fast convergence. In practice, entropy injection rate α_{EI} has a low value (between 1% - 3% per iteration). As this process is repeated *only* in post-convergence iterations, there is low sensitivity to α_{EI} as entropy is incrementally increased *whenever necessary*. Injection stops as soon as the policy value diverges, allowing unhindered exploration. This acceleration approach is evaluated in Section V-A and also integrated into the proposed continuous-observation search algorithm in the next section.

IV. Continuous-Observation Dec-POSMDP Search

This section focuses on multi-robot policy search in continuous observation spaces. It first presents an extension of traditional discrete, deterministic FSAs to allow representation of continuous policies. A continuous-observation Dec-POSMDP search algorithm is then introduced.



(a) The robot samples an MA using its node's MA distribution.

(b) Given a high-level continuous observation made following MA execution (e.g., $o^e = (2.5, 1.5)$ above), the transition *function* outputs a categorical transition *distribution* over next-nodes q'. The robot samples this distribution to select its next node, q'.

(c) The robot repeats this decision-making process at the next SK-FSA node q' (one of the 4 nodes above), conducting the stochastic MA and transition selection process indefinitely.

Fig. 1: Overview of continuous-observation decision-making using SK-FSAs. A given robot's policy is represented by a set of stochastic FSA nodes, each containing an MA sampling distribution and node transition function. A 4 node SK-FSA ($N_n = 4$) is illustrated above, with the robot starting policy execution at SK-FSA node $q = 3$ (on the left).

A. Stochastic Kernel-Based Finite State Automata

We first extend the notion of deterministic policies used in existing Dec-POSMDP algorithms to stochastic policies. In a stochastic FSA, MA output function $\lambda^{(i)}$ and node transition function $\delta^{(i)}$ provide robots with a probability distribution over MAs and next-nodes q' during policy execution, rather than deterministic MA and transition assignments. The resulting stochastic decision-making scheme allows robots to escape cycles of incorrect decisions which may otherwise occur in deterministic FSAs [16]. While it has been shown that finite-horizon Dec-POMDPs have at least one optimal deterministic policy (i.e., guaranteed to at least equal performance of the optimal stochastic policy) [17], in approximate searches, stochastic FSAs often result in a higher joint value [16], [18]. One can readily modify cross entropy-based search to provide such a stochastic policy by simply using the underlying sampling distributions $f(\pi^{(i)}|q^{(i)};\theta^{(i)(\pi|q)})$ and $f(q^{(i)'}|q^{(i)},\breve{o}^{e(i)};\theta^{(i)(q'|q,\breve{o}^e)})$ to define the policy, rather than the *best sampled deterministic policy* (as done in G-DICE).

A second issue is extension of FSAs to support continuous observations, a formidable task as continuous observation spaces are uncountably infinite. Existing Dec-POSMDP algorithms are, thus, inapplicable. To resolve this, we assume policy smoothness over the observation space, a characteristic which occurs naturally in many robotics domains. In other words, the controller structure should induce similar decisions from similar observation chains. This typical assumption is also made by the continuous state-action MDP and POMDP literature [7], [8], [19].

We exploit this smoothness assumption and introduce Stochastic Kernel-based Finite State Automata (SK-FSAs) for policy representation (Fig. 1), which have similar structure to the controllers used in [7]. Policy execution in SK-FSAs is similar to traditional FSAs. Each robot's SK-FSA node (e.g., node $q = 3$ in Fig. 1) outputs categorical MA distribution $f(\pi^{(i)}|q^{(i)};\theta^{(i)(\pi|q)})$, which the robot samples to select its next MA (Fig. 1a). Following MA execution, the robot receives a *continuous* high-level observation, which the SK-FSA node transition *function* $\delta^{(i)}$ uses to output a corresponding node transition *distribution* $f(q^{(i)'}|q^{(i)},\breve{o}^{e(i)};\theta^{(i)(q'|q,\breve{o}^e)})$. Note the distinction between transition *function* and transition *distribution*—the transition function maps continuous observations to the N_n-dimensional simplex. Given an observation, $\delta^{(i)}$ outputs an infinitesimal 'slice' representing a categorical transition distribution over next-nodes q'. Fig. 1b illustrates such a slice, evaluated at high-level observation $o^e = (2.5, 1.5)$. The robot samples this categorical distribution, transitions to its next SK-FSA node q', and repeats this process indefinitely.

We propose use of kernel logistic regression (KLR) to represent node transition functions. KLR is a non-parametric multi-class classification model (i.e., model complexity grows with the number of kernel points). In SK-FSAs, node transition functions use KLR with high-level observation inputs, o^e, and output probabilities over next-nodes q'. KLR is a natural model for stochastic policies as it is a probabilistic classifier (i.e., SK-FSA transition distributions correspond to KLR probabilities) [20]. Our approach uses KLR with radial basis function (RBF) kernels over the observation space,

$$K(o^e, o^{e'}) = \exp(-0.5\sigma^{-2}||o^e - o^{e'}||^2), \qquad (11)$$

where σ is the kernel radius. RBF kernels are preferred as they provide smooth classification outputs while allowing non-linear decision boundaries [20], in contrast to linear kernels. The next section discusses SK-FSA policy search, including details on kernel basis selection and kernel weight training.

B. Entropy-based Policy Search over SK-FSAs

This section introduces an SK-FSA search algorithm titled Entropy-based Policy Search using Continuous Kernel Observations (EPSCKO). EPSCKO consists of 3 steps: cross entropy search for MA distributions (as done in G-DICE), memory-bounded KLR training for SK-FSA node transition functions, and entropy injection for search acceleration (as in

Section III-B). In each EPSCKO iteration, decision trajectories are sampled from the SK-FSA policy. The N_b best trajectories (evaluated using (4)) are used for policy update.

We first detail the KLR training approach and then present the overall algorithm. As transition function $\delta^{(i)}$ uses a kernel-based representation over the observation space, it requires a set of observation kernel basis points and weights. In EPSCKO, kernel weights constitute the node transition parameter vector $\theta^{(i)(q'|q,\breve{o}^e)}$. To simplify notation, references to $\theta^{(i)}_{q'}$ in this section refer to this transition parameter vector.

The computational cost of training KLR models is $O(N_d^3)$ [20], where N_d is the training input size. For a sustainable training time, EPSCKO uses a memory-bounded kernel basis consisting of continuous observations received during evaluation of the N_b best policies in each of the latest N_{KLR} iterations. In each iteration, the *bundle* of observations in the N_b best decision trajectories is pushed to a first-in, first-out (FIFO) circular queue of length N_{KLR}. KLR training outputs are the corresponding sampled node transitions taken along these same trajectories. The non-parametric nature of KLR ensures that node transition function complexity increases in regions with high observation density, so the policy naturally focuses on prominent observation space regions. The result is a compact yet informative policy representation.

To counter convergence to locally optimal SK-FSAs, EPSKCO uses a weighted log-likelihood function to train the KLR model. Weights are discounted such that observations sampled in earlier algorithm iterations are given higher value. Given learning rate α, the following weight set is used,

$$w_b = \begin{cases} (1-\alpha)^{N_{KLR}-1} & b = 1 \\ \alpha(1-\alpha)^{N_{KLR}-b} & b \in \{2, \cdots, N_{KLR}\} \end{cases} \tag{12}$$

where w_b is the training weight for the b-th observation bundle in the FIFO kernel queue. This weighting is derived from recursive application of (7), and is analogous to the smoothing step used in G-DICE. For each robot i, the weighted log-likelihood function is maximized over $\theta^{(i)}_{q'}$ for KLR training,

$$l^{(i)}(\theta^{(i)}) = \sum_{b=1}^{N_{KLR}} w_b \log \frac{\exp(\theta^{(i)T}_{q'_b} o^{e(i)}_b)}{\sum_{k=1}^{N_n} \exp(\theta^{(i)T}_{q_k} o^{e(i)}_b)}, \tag{13}$$

where $o^{e(i)}_b$, ${q'_b}^{(i)}$, and $\theta^{(i)}_{q'_b}$ are transition function training inputs, outputs, and kernel weights for the b-th observation bundle. The partial derivative with respect to the j-th component of each parameter is,

$$\frac{\partial}{\partial\theta^{(i)}_{q',j}} l(\theta^{(i)}) = \sum_{b=1}^{N_{KLR}} w_b o^{e(i)}_{b,j} \left[\mathbb{I}(q'_b = q') - \frac{\exp(\theta^{(i)T}_{q'} o^{e(i)}_b)}{\sum_{k=1}^{N_n} \exp(\theta^{(i)T}_{q_k} o^{e(i)}_b)} \right], \tag{14}$$

where $\mathbb{I}(\cdot)$ is the indicator function. The log-likelihood can be maximized using a quasi-Newton method (our implementation uses the Broyden-Fletcher-Goldfarb-Shanno algorithm). To improve the generalization of the learned model, L_2 regularization is used during weight training.

EPSCKO is outlined in Alg. 1. It begins by specifying an empty SK-FSA policy and N_{KLR}-length FIFO circular kernel basis queue for each robot (Alg. 1, Lines 2-3). The

Algorithm 1: EPSCKO

```
1  Procedure: $\bar{\phi}_b =$
       EPSCKO$(\bar{\mathbb{T}}, \bar{\breve{\mathbb{O}}}^e, \mathbb{I}, N_n, N_k, N_s, N_b, N_{KLR}, \alpha, \alpha_{EI})$
2  For each robot, initialize SK-FSA policy with $N_n$ nodes;
3  $\bar{Q}_{KLR} \leftarrow$ initFIFOQueue$(N_{KLR})$;
4  $\bar{V}_b, \bar{V}_{w,0} \leftarrow -\infty$;
5  for $i = 1$ to $n$ do
6      Initialize $\theta_0^{(i)(\pi|q)}$ $\forall q$ and $\theta_0^{(i)(q'|q,\breve{o})}$ $\forall q, \breve{o}$;
7  for $k = 0$ to $N_k - 1$ do
8      allowEntropyInject, entropyInjected $\leftarrow$ False;
9      $\bar{\pi}_{list}, \overline{KLR}_{list}, \bar{V}_{list} \leftarrow \varnothing$;
10     for $s = 1$ to $N_s$ do
11         $\bar{V}^{\bar{\phi}}, \{\bar{\pi}\}_s, \{\bar{\breve{o}}^e, \bar{q}'\}_s \leftarrow$ Evaluate$(\bar{\phi})$;
12         if $\bar{V}^{\bar{\phi}} \geq \bar{V}_{w,k}$ then
13             $\bar{\pi}_{list} \leftarrow \bar{\pi}_{list} \cup \{\bar{\pi}\}_s$;
14             $\overline{KLR}_{list} \leftarrow \overline{KLR}_{list} \cup \{\bar{\breve{o}}^e, \bar{q}'\}_s$;
15             $\bar{V}_{list} \leftarrow \bar{V}_{list} \cup \bar{V}^{\bar{\phi}}$;
16         if $\bar{V}^{\bar{\phi}} > \bar{V}_b$ then
17             $\bar{V}_b, \bar{\phi}_b \leftarrow \bar{V}^{\bar{\phi}}, \bar{\phi}$;
18     $\bar{\pi}_{b,list}, \overline{KLR}_{b,list}, \bar{V}_{b,list} \leftarrow$ best $N_b$ policies in $\bar{V}_{list}$;
19     $\bar{Q}_{KLR}$.push$(\overline{KLR}_{b,list})$;
20     $\bar{V}_{w,k+1} \leftarrow$ min$(\bar{V}_{b,list})$;
21     if ValueConverged() then
22         allowEntropyInject $\leftarrow$ True;
23     for $i = 1$ to $n$ do
24         $\theta_{k+1}^{(i)(\pi|q)} \leftarrow$ MLE of $\theta^{(i)(\pi|q)}$ using $\bar{\pi}_{b,list}$ $\forall q$;
25         $\theta_{k+1}^{(i)(\pi|q)} \leftarrow \alpha\theta_{k+1}^{(i)(\pi|q)} + (1-\alpha)\theta_k^{(i)(\pi|q)}$;
26         $\theta_{k+1}^{(i)(q'|q,\breve{o})} \leftarrow$ trainWeightedKLR$(Q_{KLR}^{(i)}, \alpha)$;
27         if allowEntropyInject then
28             entropyInjected$\leftarrow$tryInject$(\theta_{k+1}^{(i)(\pi|q)}, \theta_{k+1}^{(i)(q'|q,\breve{o})})$;
29     if entropyInjected then
30         $\bar{V}_{w,k+1} \leftarrow -\infty$;;
31 return $\bar{\phi}_b$;
```

best-value-so-far, $\bar{V}_b$, and worst-joint-value, $\bar{V}_{w,0}$, are set to $-\infty$ (Alg. 1, Line 4). To encourage policy space exploration, SK-FSA parameter vectors are initialized such that associated distributions are uniform (Alg. 1, Line 6).

The main algorithm loop updates the SK-FSA policy over N_k iterations, using the maximal entropy injection scheme detailed in Section III-B to accelerate search. Entropy injection is initially disabled and a flag indicating successful entropy injection in the current iteration is set to False (Alg. 1, Line 8). The team's SK-FSA policies are evaluated N_s times, with perceived continuous observation and node transition trajectories saved for KLR training (Alg. 1, Line 11). MA selections and node transitions from policies exceeding the previous iteration's worst joint value are tracked in $\overline{KLR}_{list}$ (Alg. 1, Lines 13-15). The best-value-so-far, $\bar{V}_b$, is saved (Alg. 1, Line 17). Trajectory lists are pruned to retain only the best N_b trajectories (Alg. 1, Line 18). Continuous observations and node transitions from this list are pushed to the FIFO queue, causing old trajectories to be popped (Alg. 1, Line 19).

The iteration's worst joint value, $\bar{V}_{w,k+1}$, is then updated.

At this point, the algorithm checks if the Dec-POSMDP joint value has converged. If so, entropy injection is enabled to counter convergence to a local optima (Alg. 1, Line 22). This does not imply entropy injection *will* occur, only that it is *allowed to* occur. Each robot subsequently updates its MA distribution parameter vector, $\theta^{(i)(\pi|q)}$, using a smoothed MLE approach (Alg. 1, Lines 24-25). As discussed earlier, weighted log-likelihood maximization is used to train the KLR model for each node transition function (Alg. 1, Line 26).

Next, if maximal entropy injection is allowed, entropies of sampling distributions are calculated and (if necessary) injection occurs (Alg. 1, Line 28). As transition function $\delta^{(i)}$ is continuous and non-linear, an approximate measure of its entropy is calculated using transition distributions sampled at its underlying set of observation kernels. This approximation was found to work well in practice (Section V-B) and is computationally efficient as it avoids domain re-sampling. To increase entropy of the node transition function, a continuous uniform distribution injection is done using update rule (10). If entropy injection is conducted for any robot, the current iteration's worst joint value, $\bar{V}_{w,k+1}$, is set to $-\infty$ (Alg. 1, Line. 30). This critical step ensures trajectories sampled in the next iteration can actually be used for policy exploration.

EPSKCO is an anytime algorithm applicable to continuous-observation Dec-POMDPs and Dec-POSMDPs. This approach also offers memory advantages to discretization as SK-FSA memory usage is $O(N_{KLR}N_bN_n|\breve{\bar{\mathbb{O}}}|)$, in contrast to $O(d^{|\breve{\bar{\mathbb{O}}}|}N_n^2)$ for FSAs with discretization resolution d.

V. Experiments

This section first validates maximal entropy search acceleration, which resolves a long-standing convergence issue for sampling-based Dec-POSMDP algorithms. Then, EPSCKO is evaluated against discrete approaches in the first ever continuous-observation Dec-POMDP/Dec-POSMDP domain.

A. Accelerated Policy Search

We evaluate policy search acceleration approaches discussed in Section III on the benchmark Navigation Among Movable Obstacles (NAMO) domain [21] with horizon $h = 25$ and a 6×6 grid. Fig. 2 shows convergence trends for all approaches. A low learning rate of 0.15 is needed in G-DICE [9] to find the optimal policy (taking $k = 200$ iterations). 50 policies are sampled per iteration, with 1000 trajectories used to approximate policy value in each iteration, so $1e7$ total policy evaluations are conducted. This computationally expensive evaluation becomes prohibitively large as domain complexity grows. Increasing learning rate to $\alpha = 0.5$ causes fast convergence to a sub-optimal solution, after which exploration stops due to sampling distribution degeneration.

Existing search acceleration approaches are also evaluated. Dynamic smoothing with a moderate baseline rate ($\alpha_0 = 0.5$, $\beta = 15$) slightly improves value. However, decay rate β is static with no closed-loop feedback from underlying sampling distributions. The result is a sub-optimal policy (found around iteration $k = 35$) which then quickly converges to the same value as the baseline approach with $\alpha = 0.5$.

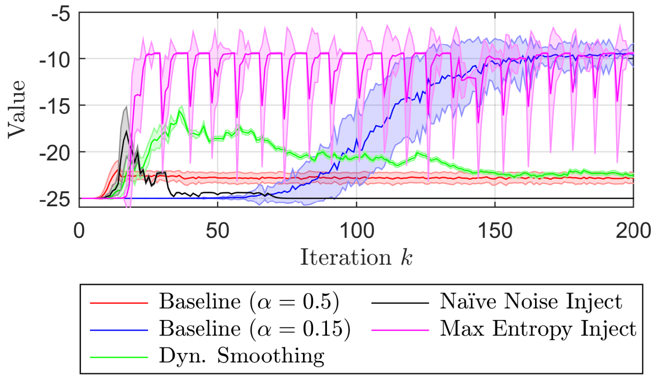


Fig. 2: Comparison of search acceleration approaches for NAMO domain, using a $N_n = 5$ node policy.

Linearly decreasing noise injection with $\omega_{\max} = 0.02$ and $r = 2000^{-1}$ performs similarly, with fast initial increase in value and subsequent degeneration to a sub-optimal policy.

The proposed entropy injection method significantly outperforms the above approaches. The same baseline learning rate as previous methods ($\alpha = 0.5$) is used with a 3% entropy injection rate, resulting in much faster convergence (around $k = 20$). Sensitivity to α and injection rate is low as value convergence monitoring is conducted in all iterations. While some initial tuning of entropy injection rate is necessary, the key insight is that post-tuning results converge much faster and are more conducive to additional experimentation and analysis (e.g., with domain/policy structure). Oscillations in plots are due to post-convergence injections, which reset underlying sampling distributions and forces further policy space exploration. In practice, the best policy found in a fixed number of iterations would be returned by the algorithm.

B. Continuous Observation Domain

To evaluate EPSCKO, a multi-robot continuous-observation nuclear contamination domain is considered (Fig. 4a). This first-ever continuous-observation Dec-POMDP/POSMDP domain involves 3 robots cleaning up nuclear waste. MAs are *Navigate to base*, *Navigate to waste zone*, *Correct position*, and *Collect nuclear contaminant*. Following MA execution, each robot receives a noisy high-level observation o^e of its 2D (x, y) state. The above MAs have non-deterministic durations and a 30% failure probability (due to nuclear contaminant degrading the robots). This causes poor performance of observation-agnostic policies which memorize chains of MAs, rather than make informed decisions using the observations.

Robots are initially at the base and must first navigate to the waste zone prior to collection attempt. Robots which execute the *Navigate to base* MA terminate with a random continuous state in a region centered on the base (brown region marked 'B' in Fig. 4c). The *Navigate to waste zone* MA results in a random terminal state within two large regions surrounding the nuclear zone (everything *interior* of gray regions marked 'L' in Fig. 4c, *including* the green regions marked 'S'). Collection attempts are only possible if the robot is *within* the waste zone (green regions marked 'S' in Fig. 4c). Collections attempted outside these small contamination regions result in wasted time, which further discounts the team's future joint rewards. Robot can attempt a *Correct position* MA, which re-samples their state to be within these smaller regions. However, repeated attempts may be necessary due to the 30% MA failure probabilities.

After successful collection, each robot must return to the base to deposit the waste before attempting another collection. Each collection results in $+1$ joint team reward (with discount factor $\gamma = 0.9$). This domain is particularly challenging due to the high failure rate of MAs, and the presence of a continuous, non-linear decision boundary in the nuclear zone center, where the trade-off between the correction and collection MAs must be considered by robots given their noisy observations.

Fig. 3 compares best values obtained using continuous-observation and discrete-observation policy search (EPSCKO, G-DICE with maximal entropy injection, and MDHS). Time horizon $h = 40$ was used for evaluation, with each MA taking an average of 1-4 time units to complete. $N_n = 6$ nodes were used for both discrete and continuous policies. G-DICE and MDHS results are shown for observation discretization factors $d \in \{2, \ldots, 10\}$, with uniform discretization in each observation dimension. EPSCKO significantly outperforms the discrete approaches, more than doubling the mean policy value of the best discrete-observation case ($d = 4$). MDHS faces the policy expansion issues discussed in Section II-B.

G-DICE policy values initially increase with higher discretization resolutions ($d = 2$ to $d = 4$), yet a drop-off occurs beyond $d = 5$. While initially counterintuitive, as higher discretization factors imply increased precision regarding important decision boundaries in the continuous domain, Figs. 4b and 4d reveal the underlying problem. These plots show the normalized count of observation samples used to compute $N_n = 5$ node discrete policies for the $d = 10$ and $d = 5$ cases, with discounting of old observation samples using (7). In other words, they provide a measure of discrete observation bins which have informed each G-DICE policy throughout its iterations. The core issue for discrete policies is that no correlation exists between decisions at nearby observation bins. Fine discretization meshes, as in Fig. 4b, result in cyclic processes where observation bins with no previous samples are encountered, therefore causing the robot to make a poor MA selection. Nearby observation bins do not inform the robot during this process, leading it to repeatedly make incorrect decisions. This issue is especially compounded in this domain due to delays caused by high MA failure probabilities, which reduce the overall number of observations received by robots. The result is a highly uninformative policy with no observations made in many bins, in contrast to policies with lower discretization factor (Fig. 4d).

To build intuition on continuous-policy decision-making, Fig. 5 plots transition functions for a 6-node EPSCKO policy. For each node q, colored 3D manifolds represent probabilities of transitioning to next-nodes, q', given a continuous observation. Circles plotted beneath transition functions indicate base and nuclear zone locations. Colorbars indicate the transition manifold color associated with each node and the *highest-probability* MA, $\pi_{\max}$, executed in it.

Consider a robot policy starting at node $q = 1$ (far left in Fig. 5) which has two major manifolds (beige and green). Observations under a prominent green manifold region indicate high probability of transitioning to node $q' = 3$ (as

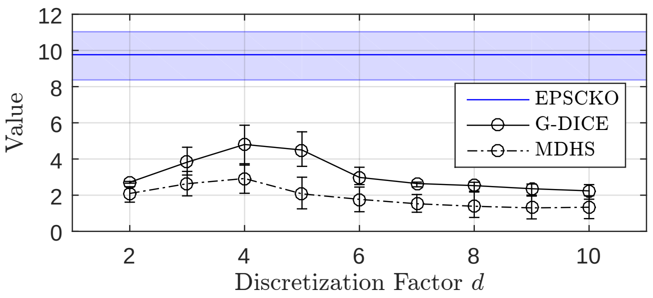


Fig. 3: Comparison of discrete and continuous policy search approaches for nuclear contamination domain.

its colorbar is green), which has $\pi_{\max} =$ *Navigate to waste zone*. For $q = 1$, this green manifold is centered on the base, which makes intuitive sense as the *Navigate to waste zone* MA should only be executed if the robot is confident it is at the base. Thus, the robot most likely transitions to node $q = 3$, and a complex transition function manifold is encountered. Two beige peaks are centered on the small inner regions of the nuclear zone, indicating transition to node $q' = 5$, which has $\pi_{\max} =$ *Collect nuclear contaminant*. Thus, when the robot is in $q = 3$ and confident that it is in the center of the nuclear zone, it attempts a collection MA. Yet, for observations outside the inner nuclear zone, the red and blue manifolds are most prominent. These indicate high probabilities of transitioning to $q' = 1$ and $q' = 2$, which have $\pi_{\max} =$ *Correct position*. Thus, the robot most likely performs a heading correction before continuing policy execution and attempting waste collection. This process continues indefinitely or until the time horizon is reached. Recall that SK-FSA policies are stochastic, so these discussions provide an intuition of the 'most likely' continuous-policy behaviors.

VI. Conclusion

This paper presented an approach for solving continuous-observation multi-robot planning under uncertainty problems. Entropy injection for policy search acceleration was presented, targeting convergence issues of existing algorithms, which are exacerbated in the continuous case. Stochastic Kernel-based Finite State Automata (SK-FSAs) were introduced for policy representation in continuous domains, with the Entropy-based Policy Search using Continuous Kernel Observations (EPSCKO) algorithm for continuous policy search. EPSCKO was shown to significantly outperform discrete search approaches for a complex multi-robot continuous-observation nuclear contamination mission—the first ever Dec-POMDP/Dec-POSMDP domain. Future work includes extending the framework to continuous-time planning.

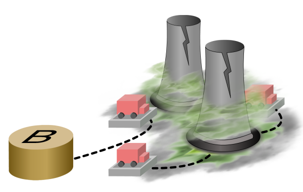
References

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

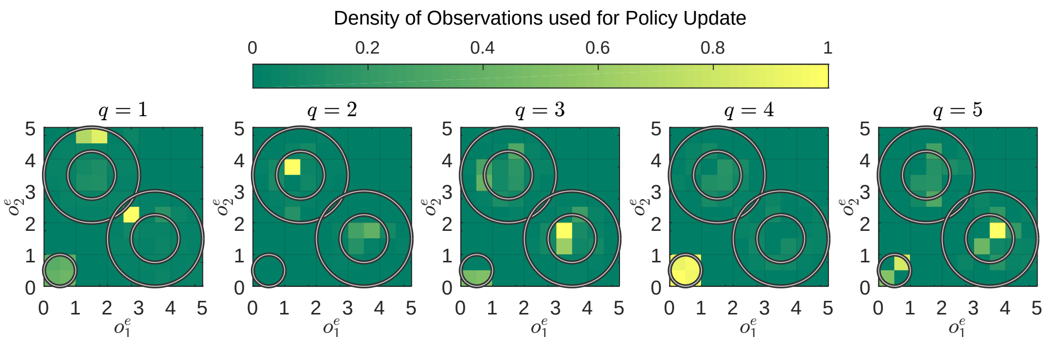
[2] C. Amato, G. Konidaris, A. Anders, G. Cruz, J. How, and L. Kaelbling, "Policy search for multi-robot coordination under uncertainty," in *Robotics: Science and Systems XI (RSS)*, 2015.

[3] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, and J. P. How, "Decentralized control of partially observable markov decision processes using belief space macro-actions," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5962–5969.

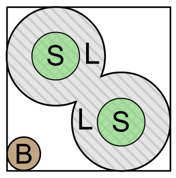
[4] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.



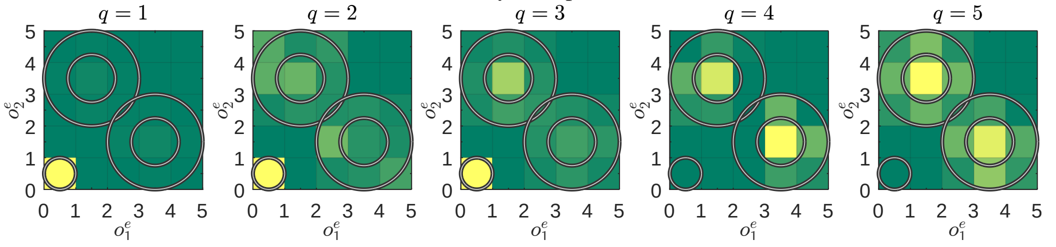
(a) Domain overview (artist's conception).



(b) Density of observations used to update discrete Dec-POSMDP policy. $N_n = 5$ node policy case with discretization factor $d = 10$. Numerous low density bins present due to fine discretization.



(c) Domain overview (key continuous regions).



(d) Density of observations used to update a discrete Dec-POSMDP policy. $N_n = 5$ node policy case with discretization factor $d = 5$. Observation density increases in the low discretization resolution case.

Fig. 4: Continuous-observation nuclear contamination domain overview and corresponding discrete policy results.

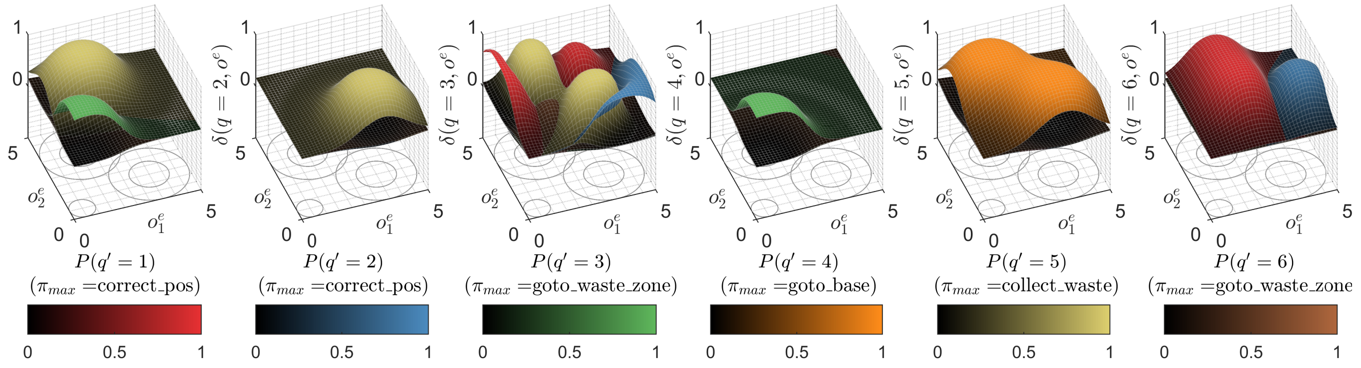


Fig. 5: Visualization of an $N_n = 6$ node SK-FSA policy transition functions for nuclear contamination domain. For each node q and observation o^e, colored 3D manifolds represent probabilities of transitioning to next-nodes q'. Colorbars indicate the color associated with each node, as well *highest-probability* MA, $\pi_{\max}$, executed in it.

[5] J. Hoey and P. Poupart, "Solving POMDPs with continuous or large discrete observation spaces," in *IJCAI*, 2005, pp. 1332–1338.

[6] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *Journal of Machine Learning Research*, vol. 7, no. Nov, pp. 2329–2367, 2006.

[7] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[8] S. Brechtel, T. Gindele, and R. Dillmann, "Solving continuous POMDPs: Value iteration with incremental learning of an efficient space representation." in *ICML (3)*, 2013, pp. 370–378.

[9] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Graph-based cross entropy method for solving multi-robot decentralized pomdps," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5395–5402.

[10] A. Costa, O. D. Jones, and D. P. Kroese, "Convergence properties of the cross-entropy method for discrete optimization." *Oper. Res. Lett.*, vol. 35, no. 5, pp. 573–580, 2007.

[11] Z. I. Botev and D. P. Kroese, "Global likelihood optimization via the cross-entropy method, with an application to mixture models." in *Winter Simulation Conference*. WSC, 2004, pp. 529–535.

[12] D. P. Kroese, S. Porotsky, and R. Y. Rubinstein, "The cross-entropy method for continuous multi-extremal optimization," *Methodology and Computing in Applied Probability*, vol. 8, no. 3, pp. 383–407, 2006.

[13] C. Thiery and B. Scherrer, "Improvements on learning tetris with cross entropy." *ICGA Journal*, vol. 32, no. 1, pp. 23–33, 2009.

[14] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*. Springer Science & Business Media, 2013, vol. 31.

[15] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mob. Comp. and Comm. Rev.*, vol. 5, no. 1, 2001.

[16] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs," *Auton. Agents and Multi-Agent Sys.*, vol. 21, no. 3, pp. 293–320, 2010.

[17] F. Oliehoek, *Value-based planning for teams of agents in stochastic partially observable environments*. Amsterdam University Press, 2010.

[18] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein, "Policy iteration for decentralized control of markov decision processes," *J. of Artif. Intell. Res.*, vol. 34, no. 1, p. 89, 2009.

[19] S. W. Carden, "Convergence of a Q-learning variant for continuous states and actions," *J. of Artif. Intell. Res.*, vol. 49, pp. 705–731, 2014.

[20] J. Zhu and T. Hastie, "Kernel logistic regression and the import vector machine," *Journal of Computational and Graphical Statistics*, 2012.

[21] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.