To appear in the Proceedings of the 2014 IEEE Intl. Conf. on Robotics and Automation (ICRA2014)

Fast Stochastic Motion Planning with Optimality Guarantees using Local Policy Reconfiguration

Ryan Luna, Morteza Lahijanian, Mark Moll, and Lydia E. Kavraki

***Abstract*— This work presents a framework for fast reconfiguration of local control policies for a stochastic system to satisfy a high-level task specification. The motion of the system is abstracted to a class of uncertain Markov models known as bounded-parameter Markov decision processes (BMDPs). During the abstraction, an efficient sampling-based method for stochastic optimal control is used to construct several policies within a discrete region of the state space in order for the system to transit between neighboring regions. A BMDP is then used to find an optimal strategy over the local policies by maximizing a continuous reward function; a new policy can be computed quickly if the reward function changes. The efficacy of the framework is demonstrated using a sequence of online tasks, showing that highly desirable policies can be obtained by reconfiguring existing local policies in just a few seconds.**

I. Introduction

The objective for traditional robotic motion planning is to compute a trajectory that will move the robot between two valid poses while respecting all physical constraints [1]–[3]. In practice, however, robots suffer from unexpected events like noisy actuation of a wheeled base when navigating uneven terrain, imperfect observations due to unfavorable sensing conditions, or an incorrect map if objects have moved around a room. These kinds of disturbances force the robot to deviate from its current course into a state from which there may be no clear path to the goal. Replanning is one option to address these detours, but this can be computationally prohibitive if the deviations are frequent or constant. A more robust strategy to combat motion planning under uncertainty is to model the problem as a stochastic decision process where the solution is not a single trajectory, but rather a control policy over all possible states of the system to maximize an objective function [3], [4].

Ideally, one would compute a stochastic control policy over the continuum of states and actions at every point in time. However, finding an optimal policy requires solving the Hamilton-Jacobi-Bellman PDE, which does not have a general closed-form solution [5]. Markov decision processes (MDPs) which discretize the state and action spaces are common in the literature for computing a policy which approximates the optimal solution [6]–[10]. This work goes one step further by quickly computing optimal policies to arbitrary regions of the state space by reusing many local policies obtained with a sampling-based method to robustly achieve a task.

Work by Ryan Luna has been supported in part by NASA NNX13AD09G and a NASA Space Technology Research Fellowship. Work by Morteza Lahijanian, Mark Moll, and Lydia Kavraki has been supported in part by NSF 1317849, 1139011, 1018798, and ARL/ARO W911NF-09-1-0383.

The authors are with the Department of Computer Science at Rice University, Houston, TX, USA. {rluna, morteza, mmoll, kavraki}@rice.edu

Sampling-based motion planners have proven effective in computing motion plans for dynamical systems. Specifically, probabilistic sampling of a continuous space mitigates the *curse of dimensionality* that classical planning methods suffer from. Sampling-based tree planners [11]–[13] are particularly useful for computing kinodynamic motion plans which return a single trajectory. Sampling-based algorithms which compute control *policies*, however, are a much more recent endeavor.

For systems with linear dynamics, methods exist that synthesize closed-loop feedback control policies around randomly sampled states. Typical approaches construct a tree [14] or a graph [15] of stabilizing controllers for single or multiple queries, respectively. These methods are highly robust to disturbances and provide strong guarantees. However, such approaches are only locally optimal and do not generalize well to systems with non-linearizable dynamics.

In the *stochastic motion roadmap* (SMR) [8], a set of discrete actions is employed along with sampling of the continuous state space to construct a directed roadmap. An MDP is formed over the roadmap, where the transition probabilities are empirically computed by sampling a number of possible destinations for each state-action pair and finding the closest existing representative. SMR computes a policy which optimizes a reward function defined over the discrete state and action spaces. However, no guarantees are made regarding optimization of a continuous reward function.

A recent work, the incremental Markov decision process (iMDP) [10], approximates the optimal policy for a continuous-time continuous-space stochastic system through sampling of both the state and control spaces. A discrete model of the underlying problem is continually refined by adding new states and controls obtained through sampling. The model is locally optimized using asynchronous value iterations and asymptotically converges to the optimal control policy.

A top-down approach for satisfying a complex goal specification for a system with uncertain dynamics, however, requires computation of multiple policies, one for each *point-to-point* action that must be performed. Existing works in the literature compute either a very large MDP which must be solved for each policy (SMR), or a single Markov chain approximation which must be recomputed for each task (iMDP). This work proposes a method where local, near-optimal policies can be optimally reconfigured online to satisfy each step of a goal specification represented by a sequence of tasks. The proposed technique operates in two phases: an offline step in which policies are computed within discrete regions of the state space to reach local goals, and an online step where a policy is selected for each region to

maximize a reward function specific to each task. Selection of local policies is modeled as a bounded-parameter Markov decision process (BMDP) [16], a stochastic decision algorithm where transition probabilities and rewards are a range of real values. This work builds upon existing techniques that utilize a BMDP to compute policies for uncertain systems [17] by maximizing a continuous reward function representing the task to be executed.

II. Problem Formulation

Consider a noisy robot whose dynamics are given by the following stochastic system:

$$dx = f(x(t), u(t))dt + F(x(t), u(t))dw, \qquad (1)$$
$$x \in X \subset \mathbb{R}^{n_x} \ , \ u \in U \subset \mathbb{R}^{n_u},$$

where X and U are compact, $w(\cdot)$ is an n_w-dimensional Wiener process (i.e. Brownian motion) on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, and $f : X \times U \to \mathbb{R}^{n_x}$ and $F : X \times U \to \mathbb{R}^{n_x \times n_w}$ are bounded measurable and continuous functions. It is assumed that the pair $(u(\cdot), w(\cdot))$ is admissible [18], and the matrix $F(\cdot, \cdot)$ has full rank. Furthermore, it is assumed that stochastic process $x(t)$ is fully observable for all $t \geq 0$ and stops as soon as it exits the interior of X.

The robot represented by system (1) evolves in workspace $\mathcal{W}$ that consists of regions of interest such as obstacles and goal regions. Let $\mathcal{R} \subseteq \mathcal{W}$ denote the set of all the regions of interest, i.e., $\mathcal{R} = \{r_1, \ldots, r_{n_r}\}$ where r_i is a region of interest. The object of this work is to develop a theoretical and computational framework which allows for planning for system (1) from a high-level specification $\mathcal{T}$ given over $\mathcal{R}$. It is assumed that $\mathcal{T}$ consists of an ordered sequence of tasks, i.e., $\mathcal{T} = T_1 \to T_2 \to \ldots \to T_{n_T}$, where each task T_i has an initial and a terminal condition taking place at times t_0^i and t_f^i, respectively. Furthermore, executing task T_i optimally is equivalent to maximizing a reward function of the form

$$J_i = \mathbb{E}\Big[\int_{t_0^i}^{t_f^i} \gamma^{(t-t_0^i)} g_i(x(t), u(t))dt + \gamma^{t_f^i} h_i(x(t_f^i)) \Big], \qquad (2)$$

where $g_i : X \times U \to \mathbb{R}$ and $h_i : X \to \mathbb{R}$ are bounded measurable and continuous functions called the *reward rate function* and *terminal reward function*, respectively, and $\gamma \in [0, 1)$ is the *discount rate*. The function g_i returns the reward received by taking action $u(t)$ at state $x(t)$. The function h_i returns the reward for reaching a terminal state $x(t_f^i)$.

Our ultimate goal is a planning framework for noisy robots that is capable of both online and optimal planning for complex tasks. In other words, not only can the goal specification be complex, but it can also change online by modifying an existing task or adding new tasks. Accommodating both online requests and generating optimal motion policies is challenging and ambitious. As a first step toward such a framework, the proposed method focuses on the problem of generating control policies online which are near-optimal. A formal statement of this problem follows.

Problem 1: Given stochastic system (1) representing a noisy robot in workspace $\mathcal{W}$ and specification $\mathcal{T}$, find a control strategy for the robot online that achieves $\mathcal{T}$ and approximates the reward function J_i given by (2) corresponding to each task $T_i \in \mathcal{T}$.

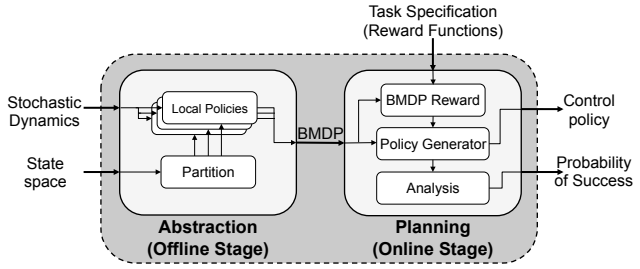


Fig. 1: A block representation of the approach to Problem 1.

For example, consider a two-task specification for a stochastic robot. The first task may be time sensitive and should be completed quickly. In this case, actions would receive rewards inversely proportional to the distance to the goal. The second task may require the system to minimize energy consumption. Therefore, actions requiring more fuel would receive less reward; the system may take a longer time to achieve the second task.

A. Approach

This work approaches the problem in two distinct stages. First, a discretization of the state space is performed. Then local control policies are computed using the asymptotically optimal iMDP algorithm; local policies are computed within a discrete region of the state space with the intent of moving the system to a neighboring region. The choice of selecting a policy to transit between regions is abstracted using a bounded-parameter Markov decision process (BMDP), which models the transition probabilities and reward of each action using a range of possible values.

Given the BMDP abstraction, a control policy for the stochastic system can be quickly computed by optimizing a reward function over the local policies within the BMDP. That is, for each task $T_i \in \mathcal{T}$, a policy which maximizes the reward function J_i (given in (2)) associated with T_i is obtained by selecting a local policy within each discrete region. A high-level diagram of the method appears in Figure 1. Experiments show that optimal policy selection can be obtained in a matter of seconds, indicating that the method can be used online. It is stressed that the final policy over the BMDP is optimal only with respect to the discretization. Further discussion of optimality is given in Section V. Details of the Markov models used in the framework are given in the next section.

III. Stochastic Modeling Techniques

A. Markov Decision Process

Markov decision processes (MDPs) provide a modeling framework for decision making in the presence of randomness where a choice of action must be taken at each state of a discrete system. A formal definition of an MDP follows.

Definition 1 (MDP): An MDP is a tuple $\mathcal{M} = (Q, A, P, R)$, where:

- Q is a finite set of states;

- A is a set of actions, and $A(q)$ denotes the set of actions available at state $q \in Q$;
- $P : Q \times A \times Q \to [0, 1]$ is a transition probability function, where $P(q, a, q')$ gives the transition probability from state $q \in Q$ to state $q' \in Q$ under action $a \in A(q)$ and $\sum_{q' \in Q} P(q, a, q') = 1$;
- $R : Q \times A \to \mathbb{R}$ is a reward function that maps each state-action pair (q, a), where $q \in Q$ and $a \in A(q)$, to a real value $R(q, a)$.

A policy defines a choice of action at each state of an MDP and is formally defined as follows.

Definition 2 (Policy): A *policy* maps states to actions, $\pi : Q \to A$. The set of all policies is denoted by Π.

Under a fixed policy $\pi \in \Pi$, MDP $\mathcal{M}$ becomes a Markov chain (MC) denoted by $\mathcal{M}_\pi$. Moreover, the probability of making a transition from q to q' is given by $P_\pi(q, q') = P(q, \pi(q), q')$.

B. Bounded-Parameter Markov Decision Process

A bounded-parameter Markov decision process (BMDP) [16] is an MDP where the exact probability of a state transition and reward is unknown. Instead, these values lie within a range of real numbers. A formal definition of a BMDP follows.

*Definition 3 (*BMDP*):* A BMDP is a tuple $\mathcal{B} = (Q, A, \check{P}, \hat{P}, \check{R}, \hat{R})$ where:

- Q is a finite set of states;
- A is a set of actions, and $A(q)$ denotes the set of actions available at state $q \in Q$;
- $\check{P} : Q \times A \times Q \to [0, 1]$ is a pseudo transition probability function, where $\check{P}(q, a, q')$ gives the lower bound of the transition probability from state q to the state q' under action $a \in A(q)$;
- $\hat{P} : Q \times A \times Q \to [0, 1]$ is a pseudo transition probability function, where $\hat{P}(q, a, q')$ gives the upper bound of the transition probability from state q to the state q' under action $a \in A(q)$;
- $\check{R} : Q \times A \to \mathbb{R}$ is a reward function, where $\check{R}(q, a)$ gives the minimum reward of choosing action $a \in A(q)$ at state $q \in Q$;
- $\hat{R} : Q \times A \to \mathbb{R}$ is a reward function, where $\hat{R}(q, a)$ gives the maximum reward of choosing action $a \in A(q)$ at state $q \in Q$.

For all $q, q' \in Q$ and any $a \in A(q)$, $\check{P}(q, a, \cdot)$ and $\hat{P}(q, a, \cdot)$ are pseudo distribution functions such that $0 \leq \check{P}(q, a, q') \leq \hat{P}(q, a, q') \leq 1$ and

$$0 \leq \sum_{q' \in Q} \check{P}(q, a, q') \leq 1 \leq \sum_{q' \in Q} \hat{P}(q, a, q').$$

Furthermore, a BMDP $\mathcal{B}$ defines a set of uncountably many MDPs $\mathcal{M}_i = (Q, A, P_i, R_i)$ where

$$\check{P}(q, a, q') \leq P_i(q, a, q') \leq \hat{P}(q, a, q'),$$
$$\check{R}(q, a, q') \leq R_i(q, a, q') \leq \hat{R}(q, a, q'),$$

for all $q, q' \in Q$ and $a \in A(q)$.

C. Incremental Markov Decision Process

The incremental Markov decision process (iMDP) is an algorithm that approximates the optimal policy of stochastic system (1) through probabilistic sampling of the continuous state and control spaces [10]. iMDP incrementally builds a sequence of discrete MDPs with probability transitions and reward functions that consistently approximate the continuous counterparts. The algorithm refines the discrete MDPs by adding new states into the current approximate model. At every iteration, the optimal policy of the approximating MDP is computed using value iteration. As the number of sampled states and controls approaches infinity, this policy converges to the optimal policy of the continuous system (1).

IV. Planning Framework

A policy reconfiguration framework, as described in Section II-A, is presented in this section. Local policies within discrete regions of the state space are computed offline, and selection of a local policy for each region online is modeled using a BMDP. Since much of the computation is performed offline, reconfiguration of local policies relies only on finding a strategy over the BMDP, a polynomial-time operation. Details of the framework are given in the following sections.

A. Offline Stage

The offline stage begins by discretizing the state space into distinct regions, and then computing control policies for states within each region to transit to all neighboring regions. Once all control policies have been computed, the transition probability for each non-terminal state in each policy to reach all neighboring regions is computed.

1) Discretization: A discretization of the state space is used to denote the boundaries of the local policies. The choice of discretization is dependent on the problem to be solved, the dynamics of the system, and the optimization objective. A coarse discretization requires computation of relatively few control policies, but the range of transition probabilities within a large region is likely to be large as well. Conversely, a fine discretization is likely to have a small range of transition probabilities at the expense of computing many local control policies. Moreover, the number of discrete regions affects the runtime of the online stage.

The evaluation of this work uses a Delaunay triangulation [19] of workspace $\mathcal{W}$ that respects obstacle boundaries. Note that $\mathcal{W}$ is the projection of X onto $\mathbb{R}^{n_\mathcal{W}}$ where $n_\mathcal{W}$ is the dimension of $\mathcal{W}$. Hence, partitioning $\mathcal{W}$ induces a discretization in X. A property of Delaunay triangulations is that the circumcircle for each triangle does not enclose any other vertex. In other words, this triangulation maximizes the minimum angle of all triangles, avoiding *skinny* triangles. Using triangles also reduces the number of policies to three per region, one to transition to each neighboring triangle.

Let $D = \{d_1, \ldots, d_{n_D}\}$ denote the set of discrete regions (triangles), where n_D is the total number of regions. By definition, $d_i \cap d_j = \emptyset$ for all $d_i, d_j \in D$. Furthermore, the triangulation is performed with respect to the regions of interest in $\mathcal{R} \subseteq \mathcal{W}$. That is, each region of interest $r \in \mathcal{R}$

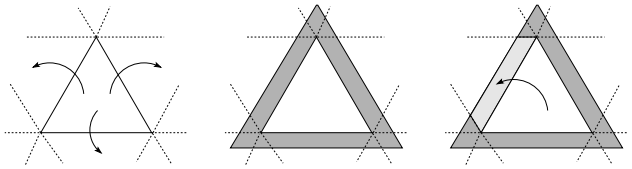


Fig. 2: Concepts of offline (local) policy generation.
(a) A policy is computed to each neighboring region.
(b) Terminal states bleed into the surrounding area.
(c) Terminal states within the desired destination have high reward; all others have negative reward.

is decomposed into a set of triangles $D_r \subseteq D$ such that for $r, r' \in \mathcal{R}$ where $r \neq r'$, $D_r \cap D_{r'} = \emptyset$. A demonstration of this triangulation is shown in Figures 3a and 4a.

2) Local Policy Generation: Control policies within each discrete region of the state space are computed which take the system to each of the neighboring discrete regions; the number of local policies in each region is equal to the number of its neighbors (Figure 2a). Generation of each local policy uses the iMDP algorithm [10].

Formally, a local policy π_d^i is a Markov chain for transitioning between discrete region d and d's neighbor along face i. Let X_d denote the set of sampled states in region d and z_k denote the discrete MDP process that approximates continuous system (1) in region d. Policy π_d^i is calculated by maximizing the discrete reward function corresponding to the (discounted) probability of success:

$$J^i_{\text{iMDP}}(x_d, \pi_d^i(x_d)) = \mathbb{E}[\gamma^{t_{k_N}} h^i_{\text{iMDP}}(z_{k_N}) | z_0 = x_d], \qquad (3)$$

where $h^i_{\text{iMDP}}(\cdot)$ is the terminal reward function, γ is the discount rate, t_k is the total time up to step k, k_N is the expected first exit time of z_k from the discrete region, and z_{k_N} is a terminal state (i.e., a sampled state in one of the neighboring regions) in the approximating MDP. The method of sampling the terminal states and the assignment of their reward values are explained below.

Each region considers a fixed area around the discrete geometry as terminal, shown in Figure 2b. States from this fixed area can then be sampled uniformly as terminal states in the MDP. Terminal states that lie in the desired destination region of the policy (i.e., neighbor that shares face i) yield a high terminal reward. Terminal states that are not in the desired destination region are treated as obstacle states (zero reward) to encourage a policy which avoids all regions other than the destination (Figure 2c).

The use of the terminal region around the discrete geometry has the added benefit of ensuring a well-connected final policy. If, instead, the terminal states for the policy were those that lie exactly on the border of two regions, the control policy would exhibit very short control durations for states near the border since the boundary condition must be met exactly; states outside of the terminal region are considered obstacles. These small durations can lead to oscillations or deadlocks at the boundary. By allowing incursions into the next region, continuity between disjoint policies is ensured since relatively long control durations are valid for all states in the region.

3) Markov Chain Evaluation: Evaluating the probability for each state in a region to end up at each terminal state can be computed using an absorbing Markov chain analysis [20]. These probabilities are used later in the BMDP abstraction. Let L be the one-step transition probability matrix between transient states. Then the probability of reaching a transient state from any other transient state in any number of discrete steps is given by the matrix $B = (I - L)^{-1}$, where I is the identity matrix. Let C be the matrix of one-step probabilities between transient and terminal states. Then the absorbing probabilities are given by $E = BC$, where the probability of transient state i being absorbed by terminal state j is the entry (i, j) in the matrix E.

B. Online Stage

The online stage utilizes all of the data in the offline stage to construct a BMDP which is used to find an optimal selection of local policies for a particular task.

1) BMDP Construction: In the BMDP model, each discrete region $d \in D$ is associated with one state of the BMDP $q \in Q$. The actions available at each state of the BMDP $A(q)$ correspond to the policies computed for the associated region d. For instance, in the case where d is a triangle, $A(q) = \{a_q^1, a_q^2, a_q^3\}$, where a_q^i is policy π_q^i which takes system (1) out of region d from facet i.

Recall that each state-action pair (q, a_q^i) of the BMDP is associated with a range of transition probabilities $[\check{P}(q, a_q^i, q'), \hat{P}(q, a_q^i, q')]$. These probabilities are the range of transition probabilities from d to d' under policy π_d^i. Let $pr(x_d, \pi_d^i, d')$ denote the transition probability of the iMDP sampled state x_d in region d to the neighboring region d' under policy π_d^i. These probability values can be calculated using the Markov chain evaluation method described in section IV-A.3. Then, the BMDP upper- and lower-bound transition probabilities are given as:

$$\check{P}(q, a_q^i, q') = \min_{x_d \in X_d} pr(x_d, \pi_d^i, d'),$$
$$\hat{P}(q, a_q^i, q') = \max_{x_d \in X_d} pr(x_d, \pi_d^i, d'),$$

where X_d is the set of states sampled by iMDP within region d to find local policy π_d^i, and d and d' are the associated discrete regions to the BMDP states q and q', respectively. Note that these probabilities do not change depending on the task and can be computed during the offline phase.

Note that for $\check{P}$ and $\hat{P}$ to be correct in the context of a BMDP, for each action their sum must be ≤ 1 and ≥ 1 respectively. The following lemma proves this statement.

Lemma 1: For policy π_d^i, the following properties hold.

$$\sum_{d' \in D} \min_{x_d \in X_d} pr(x_d, \pi_d^i, d') \leq 1,$$
$$\sum_{d' \in D} \max_{x_d \in X_d} pr(x_d, \pi_d^i, d') \geq 1.$$

A formal proof is omitted for space considerations, but the idea begins by presuming one state in the policy has the minimum probability to reach all other neighboring regions. These probabilities sum to one. As each additional state is

checked, the minimum probability to reach a particular region can only decrease, ensuring that the sum over the minimum probabilities is at most one. A similar argument shows that the sum over maximum probabilities is at least one.

To simplify notations and improve clarity, with an abuse of the notation q is used to refer to both the BMDP state and its associated discrete region, and π is used to point to both a local policy and the corresponding BMDP action.

2) BMDP *policy generation:* Once the BMDP abstraction is created, a policy can be computed for each task $T_i \in \mathcal{T}$. Recall that T_i has an associated reward function J_i as shown in (2). The discrete reward interval for each BMDP state-action pair can be computed from the continuous-time function J_i. Theorem 1 shows the method to calculate these reward values.

Theorem 1: Given stochastic system (1), continuous reward rate function $g(x,u)$, approximating MDP process z_k in region q and local policy π, the BMDP reward functions are:

$$\check{R}(q,\pi) = \min_{x_q \in X_q} \mathbb{E}\Big[\sum_{i=0}^{\Delta k_q - 1} \gamma^{t_i} g(z_i, \pi_{z_i}) \Delta t(z_i, \pi_{z_i}) \Big| z_0 = x_q \Big],$$

$$\hat{R}(q,\pi) = \max_{x_q \in X_q} \mathbb{E}\Big[\sum_{i=0}^{\Delta k_q - 1} \gamma^{t_i} g(z_i, \pi_{z_i}) \Delta t(z_i, \pi_{z_i}) \Big| z_0 = x_q \Big],$$

where Δk_q is the expected first exit time step of z_k from X_q and t_k is the total time up to step k.

A formal proof is omitted due to space limitations. The main idea is to decompose the continuous reward function (2) using the analysis from [18] to derive the discrete approximation of (2) over the BMDP. Given the approximation, the range of rewards for each state $q \in Q$ is computed by taking the min and max over the expected reward for each state in the underlying policy, resulting in $\check{R}$ and $\hat{R}$.

Note that the reward function $g(x,u)$ refers to the notion of the reward for a single action. In many applications, g corresponds to notions like energy usage or fuel consumption and is invariant to the task. In such a case, $\check{R}$ and $\hat{R}$ can be computed offline. For general applications, however, $\check{R}$ and $\hat{R}$ must be computed online to optimize for each task. It is argued, however, that computation of these values can be performed fast, within a few seconds.

Computing $\check{R}$ and $\hat{R}$ for each local policy requires finding the expected discounted reward, where the reward at transient states is $g(x_q, \pi(x_q))$, and terminal reward is zero. These values can be computed using value iteration where the discount factor is $\gamma^{\Delta t(x_q, \pi(x_q))}$. Alternatively, a system of linear equations can be solved for more consistent computation times. For instances where there are a large number of discrete regions, the size of the policies within each region is expected to be small (i.e., hundreds of states) and can be solved in milliseconds.

Since the BMDP is defined over a range of transition probabilities and reward values, an optimal policy over the BMDP results in a range of expected values $[\check{V}(q), \hat{V}(q)]$ for each state $q \in Q$. Computing these values is performed using *interval value iteration* (IVI) [16], analogous to *value iteration*. The only difference is that representative MDPs are selected at each iteration to compute $\check{V}(q)$ and $\hat{V}(q)$. MDP selection is problem dependent. For the evaluation of this work (section VI), a pessimistic policy is computed which maximizes the lower bound $\check{V}(q)$. To ensure that IVI maximizes the reward function properly, the discount factor must be chosen carefully to reflect the time taken to transition between regions in the discretization. The following lemma gives this value and proves its correctness.

Lemma 2: The dynamic programming formulas to maximize the discrete approximation of (1) are:

$$\check{V}(q) = \max_{\pi \in A} \big[\check{R}(q,\pi) + \gamma^{\Delta T(x_q,\pi)} \sum_{q' \in Q} \underline{P}_{\text{IVI}}(q,\pi,q') \check{V}(q') \big],$$

$$\hat{V}(q) = \max_{\pi \in A} \big[\hat{R}(q,\pi) + \gamma^{\Delta T(x_q,\pi)} \sum_{q' \in Q} \bar{P}_{\text{IVI}}(q,\pi,q') \hat{V}(q') \big],$$

where Q is the set of states in the BMDP, $\underline{P}_{\text{IVI}}$ and $\bar{P}_{\text{IVI}}$ are the transition probabilities of the MDP representatives selected by IVI for transitioning between states q and q', and $\Delta T(x_q, \pi)$ is the expected first exit time of policy π from q with the initial condition of $z_0 = x_q$.

Proof is omitted due to space, but these equations are a direct result of the discrete reward function derived in Theorem 1.

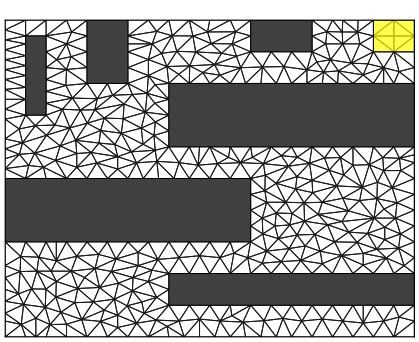
Computation of $\Delta T(x_q, \pi)$ can be performed offline for each x_d in every local policy using value iteration, replacing rewards with holding time for transient states. Online, the value of $\gamma^{\Delta T(x_q,\pi)}$ can be retrieved using the same x_q which minimizes $\check{R}$ (maximizes $\hat{R}$).

Given a policy over the BMDP computed using IVI, a complete control policy over the entire space can be constructed by concatenating the local policies that correspond to the actions selected by IVI. States that lie outside of the discrete region for each local policy are discarded during this process since they are treated as terminal during the offline computation and do not have controls associated with them. Since IVI runs in polynomial time, and the number of states in the BMDP is radically smaller than the total number of states in the local policies, it is expected that the runtime of IVI will be very short. This implies that a complete control policy to different goal locations can be computed online.

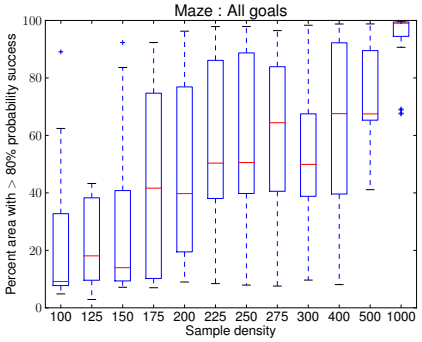
V. Analysis

A brief discussion is given in this section regarding the quality of the resulting BMDP policy. It is possible to quickly compute the probability of success for the resulting policy. By collapsing the terminal states into *goal* or *obstacle*, the computation reduces to maximum reachability probability of the *goal* which can be solved using a system of linear equations. Since the policy will have relatively few non-zero probabilities, the probability of success for policies with hundreds of thousands of states can be computed in seconds.

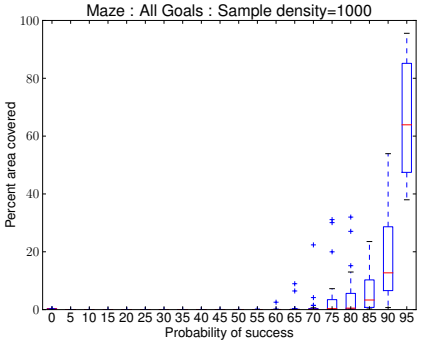
It is important to note that the policy over the BMDP optimizes the continuous reward function (2), and is proven in Theorem 1 and Lemma 2. Optimality does not extend to the local policies since these maximize the probability of transiting between neighboring regions (3). For the local policies to be optimal, expected reward at the local terminal states and the action reward function g must be known.



(a) Maze environment (684 triangles)

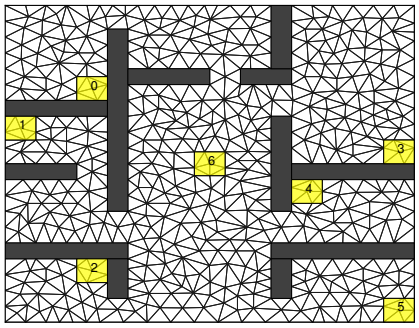


(b) Success as sampling density increases

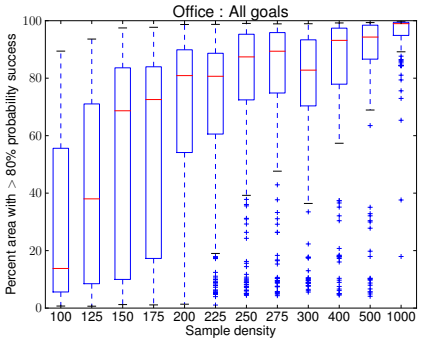


(c) Prob. success for sampling density 1000

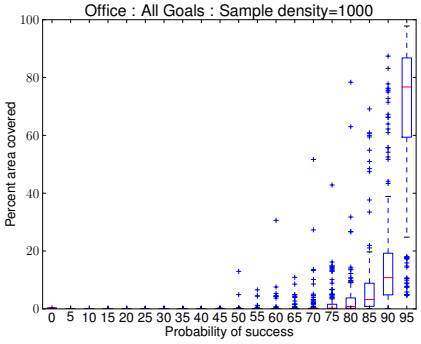
Fig. 3: The maze environment, probability of success for increasing sampling densities, and a breakdown of probability of success for the largest sampling density. Values are taken over 25 policies computed to reach the goal region.



(a) Office environment (938 triangles)



(b) Success as sampling density increases



(c) Prob. success for sampling density 1000

Fig. 4: The office environment, probability of success for increasing sampling densities, and a breakdown of probability of success for the largest sampling density. Values taken over 25 offline runs and seven distinct goal regions, 175 policies total.

VI. Evaluation

To evaluate the quality of the policies generated by the framework, experiments are performed in two different environments. There are 684 triangles in the maze environment (Figure 3a) with one goal region (marked yellow). The office environment (Figure 4a) has 938 triangles and seven sequential goals. No triangle occupies more than 0.1% of the workspace area. Both environments are 20x20. All computations took place on a 2.4 GHz Intel Xeon CPU. The implementation uses the *Open Motion Planning Library* [21].

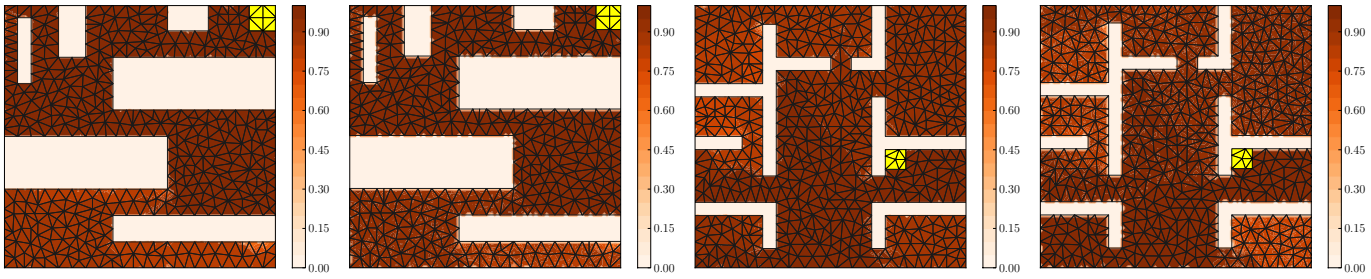
The system evaluated has stochastic single integrator dynamics $f(x,u) = u$ and $F(x,u) = 0.1I$, where I is the identity matrix (as in [10]). The system receives a terminal reward of 1 for reaching the goal region and a terminal reward of 0 for hitting an obstacle. All action rewards are zero. The local control policies use a discount factor γ of 0.95.

The performance of the method as the sampling density of the local policies changes is examined in the first experiment. In other words, this experiment evaluates how the amount of time devoted offline affects the quality of the policy computed in the online step. An analogous experiment would be to fix a sampling density and allow the triangles to encompass a larger area. For a fixed triangulation, the number of sampled states per unit area is increased from 100 to 1000. For each resulting Markov chain, the probability of success for each state is computed using the analysis technique in section V. The Voronoi regions for each state are computed, and the probability of success for each state is associated with the area defined by the corresponding Voronoi region. Figures 3b and 4b plot the distributions of workspace area with an expected probability of success of at least 80% in the maze and office environments. As the density of the samples increases, the quality of the policy improves and the overall probability of success increases dramatically. This result is inline with the asymptotic optimality property of an iMDP policy.

Figures 3c and 4c plot the distribution of the probability of success for density 1000. Note that the distributions are heavily concentrated in the 95-100% range, with a median value near 65% of the free workspace area in the maze and 75% of the free workspace area in the office environment, indicating that the resulting policies have very high probability of success over much of the space.

The next evaluation compares the expected probability of success obtained through analysis of the final policy with the observed probability of success obtained by simulating the policy. Figure 5 shows such a comparison in the maze and office worlds. A visual comparison of the contours shows that the simulated results follow very well from the theoretical results, indicating a highly accurate theoretical model.

The final evaluation compares the proposed BMDP abstraction with iMDP directly. The iMDP algorithm is run until the final policy has the same number of states per unit area as the BMDP (1000). Table I compares the runtimes and policy quality of the two approaches. From the table, computing



(a) Maze: Theoretical prob. success (b) Maze: Simulated prob. success (c) Office: Theoretical prob. success (d) Office: Simulated prob. success

Fig. 5: An example comparison of theoretical and simulated probability of success in the Maze and Office environments. Goal regions are shown in yellow. Simulations are taken over 40000 points at 150 simulations each; 6,000,000 total simulations.

	Method	Offline (s)	Online (s)	% prob > 0.8
Maze	Proposed	1743.01	0.81	92.18%
	iMDP	n/a	5587.39	$> 99\%$
Office	Proposed	2238.35	1.03	92.84%
	iMDP	n/a	7470.88	$> 99\%$

TABLE I: Comparison of policy computation times and probability of success. Values averaged over 50 runs.

a BMDP policy takes an average of 1 second or less in the maze and office environments. iMDP, on the other hand, must recompute each control policy from scratch. Obtaining policies of comparable size to the BMDP method takes several orders of magnitude more time, about 1.5 hours in the maze and about 2 hours in the office world. The final column of Table 1 gives the average percentage of the state space covered with at least 80% probability of success. Spending hours on the iMDP control policy yields very good results, but the BMDP abstraction cedes just a few percentage points of the space covered with at least 80% probability of success and gains several orders of magnitude faster computation.

VII. Discussion

This work introduces a novel framework for fast reconfiguration of local control policies for a stochastic system using a bounded-parameter Markov decision process. The framework can quickly return a control policy over the entire space, allowing the system to achieve a sequence of goals online with minimal computation that optimizes a continuous reward function. Experiments show that reconfiguration is able to achieve a high probability of success over a large portion of the state space. Moreover, the offline stage is highly parallel; each local policy is independent from all others.

The ability to quickly compute a control policy to a set of arbitrary regions has some very exciting avenues for future work. A direct extension of this work could incorporate environmental uncertainty as well as action uncertainty. Since the policy can be reconfigured online, an observation that a particular region is no longer passable is as simple as labeling the region as an obstacle and recomputing the BMDP policy.

Acknowledgments

Special thanks to Moshe Y. Vardi for his helpful discussions and insights, as well as Ryan Christiansen and the other Kavraki Lab members for valuable input on this work.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://msl.cs.uiuc.edu/planning/

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT press, 2005.

[5] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.

[6] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning under time constraints in stochastic domains," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 35–74, 1995.

[7] J. Burlet, O. Aycard, and T. Fraichard, "Robust motion planning using Markov decision processes and quadtree decomposition," in *IEEE Int'l. Conference on Robotics and Automation*, vol. 3, 2004, pp. 2820–2825.

[8] R. Alterovitz, T. Siméon, and K. Y. Goldberg, "The stochastic motion roadmap: a sampling framework for planning with markov motion uncertainty," in *Robotics: Science and Systems*, 2007.

[9] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *Int'l Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, July 2010.

[10] V. A. Huynh, S. Karaman, and E. Frazzoli, "An incremental sampling-based algorithm for stochastic optimal control," in *IEEE Int'l. Conference on Robotics and Automation*, May 2012, pp. 2865–2872.

[11] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int'l Journal of Computational Geometry and Applications*, vol. 9, no. 4-5, pp. 495–512, 1999.

[12] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Intl. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[13] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[14] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *Int'l Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, July 2010.

[15] S. Chakravorty and S. Kumar, "Generalized sampling-based motion planners," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 3, pp. 855–866, June 2011.

[16] R. Givan, S. Leach, and T. Dean, "Bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 122, no. 1-2, pp. 71–109, 2000.

[17] D. Wu and X. Koutsoukos, "Reachability analysis of uncertain systems using bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 172, no. 8-9, pp. 945–954, 2008.

[18] H. J. Kushner and P. Dupuis, *Numerical methods for stochastic control problems in continuous time*. Springer, 2001, vol. 24.

[19] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Comp. Geometry*, vol. 22, no. 1-3, pp. 21–74, 2002.

[20] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. Springer-Verlag, 1976.

[21] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012, http://ompl.kavrakilab.org.