To appear in the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014), Quebec City, Canada

Optimal and Efficient Stochastic Motion Planning in Partially-Known Environments

Ryan Luna and **Morteza Lahijanian** and **Mark Moll** and **Lydia E. Kavraki**
Department of Computer Science, Rice University
Houston, Texas 77005, USA
{*rluna, morteza, mmoll, kavraki*}*@rice.edu*

Abstract

A framework capable of computing optimal control policies for a continuous system in the presence of both action and environment uncertainty is presented in this work. The framework decomposes the planning problem into two stages: an offline phase that reasons only over action uncertainty and an online phase that quickly reacts to the uncertain environment. Offline, a bounded-parameter Markov decision process (BMDP) is employed to model the evolution of the stochastic system over a discretization of the environment. Online, an optimal control policy over the BMDP is computed. Upon the discovery of an unknown environment feature during policy execution, the BMDP is updated and the optimal control policy is efficiently recomputed. Depending on the desired quality of the control policy, a suite of methods is presented to incorporate new information into the BMDP with varying degrees of detail online. Experiments confirm that the framework recomputes high-quality policies in seconds and is orders of magnitude faster than existing methods.

1 Introduction

Uncertainties arise in decision making when the consequence of taking an action is not known exactly or when the confidence in an observation is questionable. For example, consider a search-and-rescue scenario where a robot is tasked with finding survivors after a disaster. Motion plans must be computed to combat (a) uncertainty in the robot's actions when moving over loose or uneven terrain, (b) uncertainty in the environment if a building's structure changes after a catastrophe, and (c) uncertainty in the robot's observations due to unfavorable sensing conditions. However, finding motion plans that consider all of these uncertainties is a difficult computational challenge because the sources of the uncertainty derive from both internal and external processes.

Robust motion planning under uncertainty can be cast as a problem where the goal is to map every possible state and observation of the robot to a single action. This mapping is referred to as a *policy*. Construction of an *optimal policy* with respect to a reward function representing the task requires reasoning over all possible situations that the system may encounter. In real-time problems, like the search-and-rescue example, constructing an optimal policy that considers all sources of uncertainty is computationally prohibitive. Consequently, current state-of-the-art methods for real-time applications reason over a single kind of uncertainty. In contrast, this work considers motion planning scenarios where the robot must make quick decisions under both action and environmental uncertainties.

Related Work

Motion planning under action uncertainty is generally accomplished by computing a control policy *a priori* using discrete Markov modeling techniques alongside a model of the evolution of the robot (Dean et al. 1995; Burlet, Aycard, and Fraichard 2004; Ong et al. 2010). Sampling-based algorithms have proven particularly effective for computing stochastic control policies for continuous systems with complex dynamics. The *stochastic motion roadmap* (SMR) constructs a Markov decision process (MDP) by sampling a set of states and simulating a discrete set of actions at each state to obtain the transition probabilities (Alterovitz, Simeon, and Goldberg 2007). The result is a multi-query MDP that is able to compute a policy to reach any state in the approximating MDP. In the iMDP algorithm (Huynh, Karaman, and Frazzoli 2012), an asymptotically optimal control policy for a continuous stochastic system is obtained by iteratively constructing an approximating MDP through sampling of the state and control spaces. In contrast to SMR, iMDP is a single-query structure that must be recomputed to find a policy to a different state. More recent work (Luna et al. 2014) bridges the gap between SMR and iMDP by discretizing the environment into regions, computing a set of local policies within each region, and selecting one local policy from each region using a *bounded-parameter Markov decision process* (BMDP) (Givan, Leach, and Dean 2000). The resulting control policy is optimal with respect to the discretization, and the approximating data structure allows for fast policy computation by reasoning only over regions at runtime.

Many planning algorithms exist for handling uncertainty in the environment. When the environment changes over time, a Markov model can be used to compute a policy that provides motion commands to the robot depending on the current state of the environment (LaValle and Sharma 1997). In dynamic environments when the trajectories of moving

obstacles are unknown, previous methods are typically reactive rather than deliberative. For example, a partially closed-loop receding horizon algorithm can be employed to approximate future configurations of obstacles (Toit and Burdick 2012). In static environments where there is no knowledge of the environment uncertainty, fast replanning approaches are employed to compensate for the unknown environment (Stentz 1994; Fox, Burgard, and Thrun 1997; Kewlani, Ishigami, and Iagnemma 2009). A popular class of algorithms for this problem derive from discrete search, including D* (Stentz 1994; 1995; Koenig and Likhachev 2005), ARA* (Likhachev et al. 2008), and LPA* (Likhachev and Ferguson 2009). These algorithms admit fast, optimal replanning for a deterministic system in a discretized environment. Their success derives from the ability to quickly and repeatedly compute an optimal path, rather than a policy, often by reusing information from previous searches.

Planning under sensing uncertainty introduces a computational challenge whose complexity dominates both action and environment uncertainty (Papadimitriou and Tsitsiklis 1987). When the system has noise in observation, planning is typically performed in the space of *beliefs* (Thrun et al. 2005). Reasoning over *belief states* incurs significant overhead since taking an action or making an observation induces a convolution of probability distributions. For real-time planning problems, however, this overhead is computationally prohibitive. For linear systems with Gaussian noise, methods exist to construct a multi-query roadmap in belief space from which an optimal policy to any state can be computed (Prentice and Roy 2009; Agha-mohammadi, Chakravorty, and Amato 2014).

Although many previous works consider just a single kind of uncertainty, the computational complexity has not deterred methods that attempt to consider multiple kinds of uncertainty. The partially-observable Markov decision process (POMDP) provides a modeling framework that is capable of finding an optimal policy when faced with all three kinds of uncertainty. However, computation of an optimal POMDP policy is notoriously intractable (Papadimitriou and Tsitsiklis 1987), limiting POMDPs to a restricted class of real-time problems (Simmons and Koenig 1995; He, Brunskill, and Roy 2011; Marthi 2012).

Contribution

The contribution of this work is a top-down framework for efficient computation of an optimal control policy for a continuous system with complex dynamics in the presence of both action and environment uncertainty. There is no knowledge of the environment uncertainty; only partial information is known *a priori*. Sensing is presumed to be perfect. Although previous works like D* for navigation in an unknown environment are both fast and optimal, these algorithms assume a deterministic system operating in a discrete world with discrete actions. A continuous system with complex and stochastic dynamics does not conform well to these assumptions and is likely to repeatedly deviate from a trajectory; a policy is preferable in this case. The literature on sampling-based motion planning for action uncertainty does not emphasize computation time, and existing methods are unsuitable for the fast policy recomputation required when the environment is partially known. SMR can be used directly in a recomputation framework, but finding the policy requires performing *value iteration* over an enormous MDP. iMDP is attractive for its optimality properties, but the approximating data structure relies on a specific reward function and must be completely reconstructed to obtain a policy when the reward changes (e.g., an obstacle is observed).

This paper builds upon the BMDP framework in (Luna et al. 2014) which is able to quickly and effectively compute a control policy for a continuous stochastic system at runtime. However, the framework is unable to account for environment uncertainty because the construction of the BMDP is performed offline and with respect to the environment. In a partially-known environment where features exist that are not known *a priori*, the BMDP abstraction is invalidated once an additional feature is discovered online. Moreover, the resulting control policy may no longer be optimal with respect to the reward function representing the task.

The aforementioned BMDP framework is extended in this paper to handle both action uncertainty and environment uncertainty in the form of unknown features. The BMDP is updated during execution when new environment information arrives, and an optimal control policy is quickly recomputed. The efficiency of the recomputation derives from reasoning over regions rather than individual states at runtime. Three different methods are presented to update the BMDP when new environment information is obtained: (1) a fast, conservative approach that assumes minimum reward across a whole region, (2) recomputation of the expected reward for existing local policies, and (3) recomputation of the local policies given the new environment information and reward structure. The choice of the discretization and update method determine the quality and speed of policy recomputation, and simulated results evaluating the update methods confirm this trade-off. Nevertheless, a comparison with SMR shows that the BMDP abstraction is able to compute a high quality control policy in significantly shorter time.

2 Problem Formulation

Consider a robot with noisy actuation moving about a static but partially-known environment. The stochastic dynamics of the robot are presumed to be given in the following form:

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

where X and U are compact sets representing the state and control spaces, $w(\cdot)$ is an n_w-dimensional Wiener process (i.e., Brownian motion) on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$. Functions $f : X \times U \to \mathbb{R}^{n_x}$ and $F : X \times U \to \mathbb{R}^{n_x \times n_w}$ are bounded measurable and continuous. It is assumed that $F(\cdot, \cdot)$ in its matrix form has full rank, and the pair $(u(\cdot), w(\cdot))$ is admissible (Kushner and Dupuis 2001). The stochastic process $x(t)$ is fully observable for all $t \geq 0$ and stops as soon as the interior of X is left.

The robot moves in a workspace $\mathcal{W}$ that contains a set of disjoint features $\mathcal{W}_f$ and a set of disjoint goal regions $\mathcal{W}_g$. $\mathcal{W}_f$ may consist of both obstacles and navigable regions of

interest. The robot has only partial, *a priori* knowledge of $\mathcal{W}_f$, but is equipped with a sensor that can perfectly detect all features within a radius $r_{\text{DETECT}} < \infty$. For simplicity, it is assumed that if any portion of the feature is within range, the entire feature is observed. This assumption can easily be relaxed with an incremental representation of the feature.

The mission of the robot in $\mathcal{W}$ is specified by a continuous reward function of the form

$$J = \mathbb{E}\left[\int_0^T \gamma^t g(x(t), u(t))dt + \gamma^T h(x(T))\right], \qquad (2)$$

where $g : X \times U \to \mathbb{R}$ and $h : X \to \mathbb{R}$ are bounded measurable and continuous functions called the *reward rate function* and *terminal reward function*, respectively, and $0 \leq \gamma < 1$ is the *discount rate*. Conceptually, $g(x(t), u(t))$ is the reward at state x for taking action u at time t, $h(x(T))$ is the reward for reaching a terminal state (goal or obstacle) x at time T, and the goal is to maximize expectation over the sum of action and terminal rewards. In a search-and-rescue example, $g(\cdot, \cdot)$ could correspond to the energy cost of a specific action, and $h(\cdot)$ is the reward for finding a survivor. T denotes the first time that the system (1) reaches a terminal state. From the stochastic dynamics, T is random and depends on the control $u(t)$ at state $x(t)$ for all $0 \leq t < T$.

The objective is to quickly compute a control policy $\pi : X \to U$ that maximizes J (2) for a robot with stochastic dynamics (1) moving in a partially-known workspace $\mathcal{W}$. Since the workspace is not completely known *a priori*, discovery of new features necessarily changes $g(\cdot, \cdot)$ and $h(\cdot)$ and forces recomputation of the optimal policy at runtime.

3 Methodology

A fast policy computation framework is presented that is able to handle both action and environmental uncertainty when planning for a continuous stochastic system. The complexity of considering two kinds of uncertainty is managed by reasoning over just one kind of uncertainty at a time. Action uncertainty is captured with a control policy, and the system reacts to environmental changes during execution.

The efficiency of the approach derives from the aggregation of states into regions and reasoning only over regions at runtime. The framework consists of two stages. First, an offline stage discretizes the space into regions and computes several local policies within each region, one to reach every neighboring region. Abstracting the system's evolution into regions induces a second decision process during the online stage that selects a local policy for each region. A Markov decision process is not capable of fully representing the transition probabilities and rewards of this abstraction because each local policy has a distribution of transition probabilities and rewards that depend on the initial state. A bounded-parameter Markov decision process (BMDP), however, provides a vehicle for state aggregation and tractable optimal policy computation without loss of information. This framework employs a BMDP to capture the probabilities and rewards when a stochastic system moves between discrete regions. Given a BMDP abstraction computed offline, the proposed framework incorporates new environment features into the BMDP and recomputes a global policy online.

Bounded-parameter Markov Decision Process

A bounded-parameter Markov decision process (BMDP) (Givan, Leach, and Dean 2000) is an MDP where the exact transition probabilities and rewards are unknown. Instead, these values lie within a range of real numbers. Formally, a BMDP is a tuple $(Q, A, \check{P}, \hat{P}, \check{R}, \hat{R})$ where Q is a finite set of states, A is a finite set of actions, $\check{P}$ and $\hat{P}$ are pseudo-transition probability functions that return the min and and max probabilities for transitioning between two states under a specific action, and $\check{R}$ and $\hat{R}$ are functions that return the min and max reward for choosing an action at a particular state. Conceptually, a BMDP represents a set of uncountably many MDPs whose exact transition probabilities and rewards lie within the range of values defined in the BMDP.

The following property must also hold: for all $q, q' \in Q$ and $a \in A(q)$, $\check{P}$ and $\hat{P}$ are pseudo-distribution functions where $0 \leq \check{P}(q, a, q') \leq \hat{P}(q, a, q') \leq 1$ and $\sum_{q' \in Q} \check{P}(q, a, q') \leq 1 \leq \sum_{q' \in Q} \hat{P}(q, a, q')$.

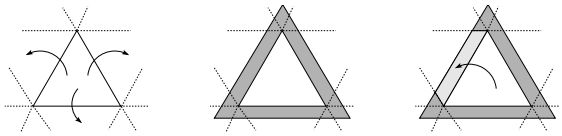
Offline BMDP Abstraction

A BMDP abstraction forms the basis for fast recomputation of an optimal policy obtained by maximizing an approximation of the reward function (2). BMDP construction is only highlighted here. For further details, see (Luna et al. 2014).

Discretization Discrete regions of the state space compose the basic building block of the BMDP abstraction. A desirable discretization depends on a number of factors including the geometry and dynamics of the system as well as the kind of task that is performed. A coarse discretization allows for fast policy computation, but the approximation of the reward function for this discretization can induce a large error. Conversely, a fine discretization represents the reward function more accurately, but the complexity of computing a policy depends on the number of discrete regions. This work utilizes a Delaunay triangulation of the workspace that respects known obstacles and goal regions (Shewchuk 2002). Empirically, this discretization also avoids *skinny* triangles that may be difficult for a dynamical system to stay within.

Local Policy Computation Once the space has been discretized, local control policies are computed within each discrete region of the space, one to transit to each neighbor of the discrete region. Computation of each local policy is performed using the iMDP algorithm (Huynh, Karaman, and Frazzoli 2012), a method that samples the state and control spaces to iteratively (and asymptotically) approximate the optimal control policy for the continuous-time, continuous-space stochastic system (1) to reach a particular goal.

For the case of a triangulated workspace, a local policy is computed to move the system through each edge of the triangles. To compute these policies, the area outside of the discrete region is considered as terminal. Terminal states within the desired destination region for the local policy receive a high reward and all other terminal states are treated as obstacles with zero or negative terminal reward. This reward structure encourages computation of local control policies that avoid all neighboring regions except the desired destination. An illustration of these concepts is shown in Figure 1.



(a) (b) (c)

Figure 1: Concepts of local policy computation. (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 other terminal states are treated as obstacles with low reward.

BMDP Construction As noted earlier, a BMDP is analogous to a Markov decision process, except that a range of real-values are specified for all transition probabilities and one-step rewards. The ranges emerge naturally as part of the state aggregation that occurs when the space is discretized. To construct the BMDP, each discrete region is added as a state, and the actions at each of these states correspond to the local policies computed within the respective region.

To complete construction of the BMDP abstraction, the probability of successfully taking each action in the BMDP and the reward associated with each action must be computed. Since the actions in the BMDP correspond to stochastic control policies within a discrete region, the probability of successfully transitioning between two neighboring regions and the associated reward depends on the initial state within the region. Since the initial state is not known *a priori*, the probabilities and rewards for the BMDP action are mapped to the range of all possible values that can emerge as a result of executing a particular local policy. In the case of the transition probabilities, the range consists of the min and max probabilities of successfully transitioning to the desired discrete region. These probabilities can be computed with an absorbing Markov chain analysis (Kemeny and Snell 1976). The reward range associated with each local policy requires computing the min and max *expected reward* over all states in the region. Formally, for BMDP state q and action (local policy) a, the lower bound expected reward is defined as:

$$\check{R}(q,a) = \min_{z \in q} \mathbb{E}\left[\int_0^{T^a_{x_0}} \gamma^t g(x, a(x)) dt \middle| x_0 = z \right],$$

where z is a configuration of the system in region q, and $T^a_{x_0}$ is the first *expected* exit time from q with initial state x_0 using local policy a. Terminal regions in which no actions are defined (obstacles or goal regions) receive a terminal reward $h(\cdot)$. When the local policies are obtained using a sampling-based method, as in iMDP, the expected reward can be computed using the following discrete approximation:

$$\check{R}(q,a) = \min_{z \in q} \mathbb{E}\left[\sum_{i=0}^{K^a_{x_0}-1} \gamma^{t_i} g(x, a(x)) \Delta t(x) \middle| x_0 = z \right],$$

where $K^a_{x_0}$ is the first expected exit time step from region q, z is a sampled state in q, $\Delta t(x)$ is the holding time of the control at state x, and t_i is the total time up to step i. This expectation can be computed using *value iteration*. The upper bound, $\hat{R}$, is analogous to $\check{R}$, except that the value is the maximum expected reward rather than the minimum.

Online Execution and Policy Recomputation

Given a BMDP abstraction that describes the evolution of the stochastic system over discrete regions of the state space, a policy over the BMDP can be quickly computed to maximize an approximation of (2). A policy over the BMDP abstraction is a selection of one local control policy for each discrete region. Computing the optimal policy in a BMDP is similar to computing a policy in an MDP. The key difference, however, is that the expected reward for each state in the BMDP is a range of possible values since the single step rewards are also a range of values. The dynamic programming analog for BMDPs is known as *interval value iteration* (IVI) (Givan, Leach, and Dean 2000). IVI computes the Bellman backup of each state during each iteration for a representative MDP selected based on the current estimation of the expected reward. Selection of the MDP representative corresponds to a pessimistic or optimistic policy that optimizes the lower bound or upper bound transition probability ranges, respectively. This framework advocates a conservative, pessimistic policy. IVI converges in polynomial-time with respect to the number of BMDP regions (Givan, Leach, and Dean 2000).

The kind of BMDP policy computed yields a set of transition probabilities for the MDP representative, denoted $\underline{P}_{\text{IVI}}$ for a pessimistic and $\bar{P}_{\text{IVI}}$ for an optimistic policy. The dynamic programming equation (IVI) to compute an optimal pessimistic policy over the BMDP is:

$$\check{V}(q) = \max_{a \in A(q)} \left[\check{R}(q,a) + \gamma^{T^a_{x_0}} \sum_{q' \in Q} \underline{P}_{\text{IVI}}(q,a,q') \check{V}(q') \right],$$

where $T^a_{x_0}$ is the *expected* exit time (x_0 minimizes $\check{R}(q,a)$). Computing an optimistic policy ($\hat{V}$) is done analogously, substituting $\hat{R}$ for $\check{R}$ and using $\bar{P}_{\text{IVI}}$ instead of $\underline{P}_{\text{IVI}}$.

Feature Discovery and BMDP Update Departing from the work in (Luna et al. 2014), new environment features that are discovered during execution are incorporated into the BMDP abstraction, and a policy over the BMDP is recomputed at runtime. As shown in the previous work, the BMDP policy is optimal with respect to the discretization of the space. However, if environmental features are discovered during execution, the BMDP policy is rendered suboptimal or even invalid because the reward that the system expected to obtain in the affected regions is incorrect. Detection of a new environmental feature can be cast as a change in the action reward function $g(\cdot,\cdot)$ and/or the terminal reward function $h(\cdot)$. Three options are presented here to update the BMDP with the new feature information so that the BMDP policy can be recomputed. These update strategies trade computation time for accuracy. The fastest update method reasons only at the region level, and the most expensive update method incorporates all new information in the local policies before recomputing the global policy.

Conservative approximation The fastest BMDP update method presumes the lowest expected reward throughout all regions affected by the previously unknown feature for every local policy. Formally, for every discrete region q and local policy a in q affected by a previously unknown feature:

$$\check{R}(q,a) = \hat{R}(q,a) = \min_{z \in q} \left[g(z, a(z)) T^a_z \right].$$

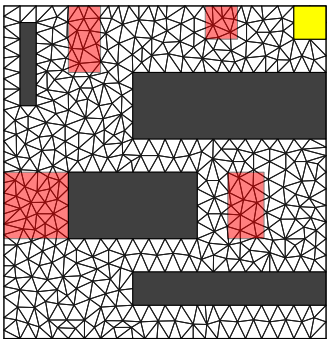


Figure 2: The *maze* environment. The goal region is shaded yellow. Obstacles not known during the offline phase are shaded in red. There are 759 triangles in the discretization.

Although the conservative approximation is very fast, this method is the least optimal of the three update options presented here. For instance, if an obstacle is observed, this method marks the entire discrete region as the obstacle even if only a tiny portion of the region is touched by the obstacle.

Recomputation of Reward A more attractive update method involves recomputing the true expected reward ranges within all affected regions and local policies. Formally, this requires recomputing the reward bounds $\check{R}$ and $\hat{R}$ for all $q \in Q$ directly impacted by the new information. By recomputing the reward, all new information obtained regarding the workspace feature is utilized, and a better approximation of the expected reward for executing each local policy in a discrete region is obtained. Note that the underlying local policies are no longer optimal with respect to the single step action reward in the discrete regions, but the reward values for executing those policies are correct.

Recomputation of Local Policies The third update method recomputes all affected local policies, then patches the BMDP with the new probability and reward ranges. Recomputing the local policies is ideal because the BMDP is patched from the bottom up to incorporate the new information. This method, however, comes at great expense since many regions could be affected, depending on the size of the observed feature, implying that many local policies must be recomputed. This method is attractive for systems with a high action uncertainty or in cluttered spaces where the previous two methods may fail to yield a robust policy.

4 Experiments

Experiments are conducted in a continuous 20x20 *maze* environment (Figure 2) to evaluate the computation time and quality of the resulting policy (probability of success) for the three BMDP update methods when new obstacles are observed. The *maze* has four unknown obstacles that the system may observe before reaching the goal. Careful consideration is given to what the proposed methods can be compared with. SMR (Alterovitz, Simeon, and Goldberg 2007) allows for direct comparison since an MDP can be constructed offline to yield a policy of comparable size to the resulting BMDP abstraction. The policy from SMR can be recomputed online after removing states that intersect with obstacles. Experiments consider an SMR with approximately 300,000 states, eight unit controls spanning the cardinal and ordinal directions, and a convergence threshold of 10^{-4}. iMDP (Huynh, Karaman, and Frazzoli 2012) is a one-shot method and the time needed to recompute a similarly sized policy is comparable to the BMDP and SMR abstraction times (on the order of hours). This comparison can safely be omitted. This problem can also be modeled with a POMDP, but a distribution of the environment uncertainty is not assumed. Furthermore, the complexity of optimally solving a POMDP prohibits online computation once the number of states exceeds a few dozen (Papadimitriou and Tsitsiklis 1987).

	BMDP Update Strategy			
	Conservative	Reward	Policies	SMR
Time (s)	1.30	9.34	62.97	184.90

Table 1: Policy recomputation times when an unknown obstacle is discovered. All values averaged over 50 runs.

For BMDP construction, the *maze* is discretized using a Delaunay triangulation that respects known obstacles, where no triangle exceeds more than 0.1% of the free space, resulting in 759 discrete regions. Three local policies are computed within each triangle until there are 1000 states per unit area in each policy. All computations are performed using a 2.4 GHz quad-core Intel Xeon (Nahalem) CPU.

The system evaluated has 2D single integrator dynamics with Gaussian noise. The dynamics are in the form of (1), where $f(x, u) = u$ and $F(x, u) = 0.1I$, and I is the identity matrix. The system receives a terminal reward of 1 when it reaches the goal and a terminal reward of 0 for colliding with an obstacle. The reward rate for all non-terminal states is zero ($g = 0$), and a discount rate of 0.95 is employed for the local policies. The system can perfectly sense all unknown features within a radius r_{DETECT} of 2 unit length.

A comparison of the computation times between SMR and the proposed BMDP abstraction update methods is given in Table 1. As expected, the conservative BMDP update performs the fastest; computation time is dominated by IVI, taking just over 1 second on average. Recomputing the BMDP rewards takes about 9 seconds on average, and recomputing all affected local policies affected by an unknown obstacle takes an average of about one minute. Contrast these times to a comparably sized SMR abstraction, where *value iteration* takes an average of about three minutes, two orders-of-magnitude slower than the conservative BMDP update and when the BMDP rewards are recomputed, and one order of magnitude slower than recomputing all affected local policies in the BMDP when an unknown obstacle is observed.

Comparing the control policies emitted from the different methods is more difficult since the quality of a policy is not easily quantified with a single value. Instead, the probability of each state in the final control policy reaching the goal is computed; each probability is weighted with the area of the Voronoi cell induced by the state. For a given probability of success, the percentage of the environment area that does not meet this probability is computed and shown in Figure 3. Results are averaged over 50 runs and the standard errors for each bar are indicated with error bars. Although SMR is com-

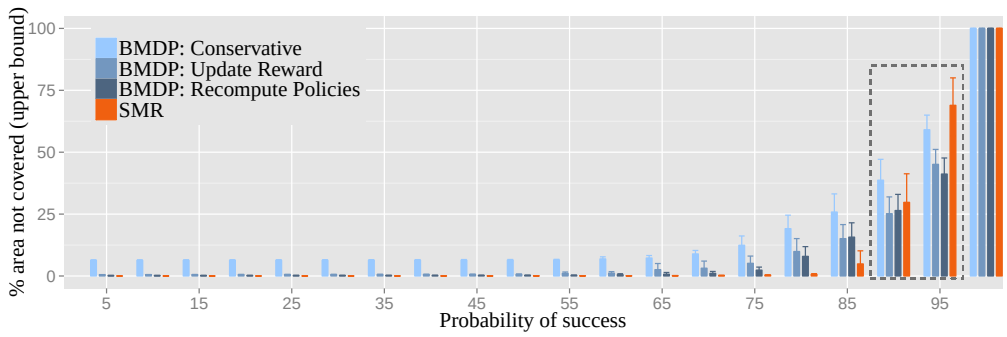
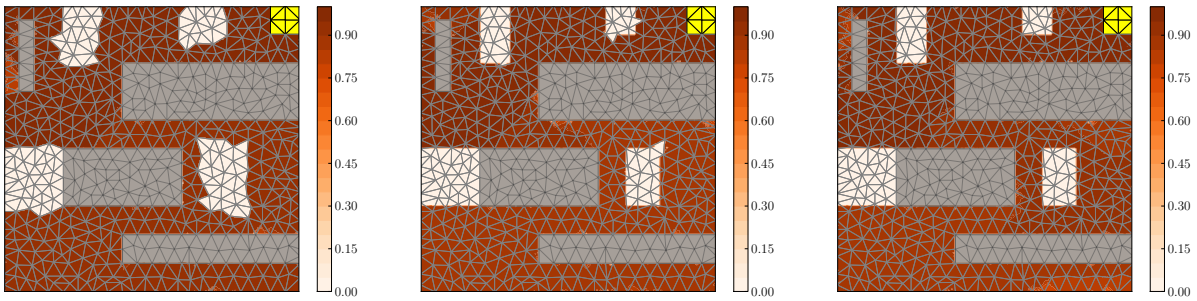


Figure 3: The percent area of the environment not covered with a given probability of success over three BMDP update methods and the SMR method (lower is better). Results are averaged over 50 runs. Mean standard error is indicated with error bars.



(a) Conservative approximation (b) Recompute reward (c) Recompute local policies

Figure 4: An example of the expected probability of success for each BMDP update method. Note the differences near the unknown obstacles, particularly between (a) and (c). Recomputation of the reward for each affected local policy (b) tends to negatively affect only those regions where each edge is affected by the unknown obstacle.

petitive with low success rate requirements, for high success rates (highlighted with a rectangle in Figure 3), SMR performance quickly starts to degrade. The BMDP reward update and policy recomputation methods significantly outperform SMR at the 95% probability of success rate. The BMDP conservative method has a consistent percentage of the space covered with a small probability of success; this corresponds to the area incorrectly labeled as an obstacle.

The differences between the three BMDP update methods are further highlighted in Figure 4, which plots the probability of success for a representative control policy using each of the update methods in the *maze* environment. Note that the unknown obstacle approximations become progressively finer from Figure 4a to Figure 4c. It is difficult to characterize exactly how the recomputation of the reward (Figure 4b) reflects in the final control policy; the quality of the local policies plays an important role. Empirically, an optimal BMDP policy avoids local policies that take the system through an edge in the discretization obscured by an unknown obstacle. The reward range for those policies can be large, potentially from $[0, 1]$, and a pessimistic BMDP policy optimizes over the lower bound to avoid such a risk.

5 Discussion

A two-stage framework for efficient computation of an optimal control policy in the presence of both action and environment uncertainty is presented in this paper. The framework abstracts the evolution of the system offline using a *bounded-parameter Markov decision process* (BMDP) over a discretization of the environment, and quickly selects a local policy within each region to optimize a continuously valued reward function online. As new environment features are discovered at runtime, the BMDP is updated and experiments show a global control policy can be recomputed quickly.

The quality of the control policy obtained online depends on the amount of computation time allowed, but a comparison with the *stochastic motion roadmap* shows that even the highest fidelity BMDP update method that recomputes every local policy affected by an unknown feature yields a comparable control policy in an order of magnitude less time. For fast recomputation, a conservative update method is shown to recompute a global control policy in just over one second.

The results of the proposed framework rely on the ability to reason quickly over groups of similar states, represented in this case by discrete regions. An interesting prospect for future research is to investigate how the decomposition of the motion planning problem into a series of smaller problems and then optimally concatenating the solutions of the small problems together can potentially be extended to other difficult problems outside of robotics.

6 Acknowledgments

RL is supported by a NASA Space Technology Research Fellowship. ML, MM, and LK are supported in part by NSF NRI 1317849, NSF 113901, and NSF CCF 1018798. Computing resources supported in part by NSF CNS 0821727.

References

Agha-mohammadi, A.; Chakravorty, S.; and Amato, N. M. 2014. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *Int'l Journal of Robotics Research* 33(2):268–304.

Alterovitz, R.; Simeon, T.; and Goldberg, K. 2007. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Robotics: Science and Systems*.

Burlet, J.; Aycard, O.; and Fraichard, T. 2004. Robust motion planning using Markov decision processes and quadtree decomposition. In *IEEE Int'l. Conference on Robotics and Automation*, volume 3, 2820–2825.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76(1-2):35–74.

Fox, D.; Burgard, W.; and Thrun, S. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1):23–33.

Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *Artificial Intelligence* 122(1-2):71–109.

He, R.; Brunskill, E.; and Roy, N. 2011. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research* 40(1):523–570.

Huynh, V. A.; Karaman, S.; and Frazzoli, E. 2012. An incremental sampling-based algorithm for stochastic optimal control. In *IEEE Int'l. Conference on Robotics and Automation*, 2865–2872.

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

Kewlani, G.; Ishigami, G.; and Iagnemma, K. 2009. Stochastic mobility-based path planning in uncertain environments. In *IEEE/RSJ Int'l. Conf. on Intelligent Robotics and Systems*, 1183–1189.

Koenig, S., and Likhachev, M. 2005. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics* 21(3):354–363.

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

LaValle, S. M., and Sharma, R. 1997. On motion planning in changing, partially predictable environments. *Int'l Journal of Robotics Research* 16(6):775–805.

Likhachev, M., and Ferguson, D. 2009. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int'l Journal of Robotics Research* 28(8):933–945.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2008. Anytime search in dynamic graphs. *Artificial Intelligence* 172(14):1613–1643.

Luna, R.; Lahijanian, M.; Moll, M.; and Kavraki, L. E. 2014. Fast stochastic motion planning with optimality guarantees using local policy reconfiguration. In *IEEE Int'l. Conference on Robotics and Automation*.

Marthi, B. 2012. Robust navigation execution by planning in belief space. In *Robotics: Science and Systems*.

Ong, S. C. W.; Png, S. W.; Hsu, D.; and Lee, W. S. 2010. Planning under uncertainty for robotic tasks with mixed observability. *Int'l Journal of Robotics Research* 29(8):1053–1068.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of operations research* 12(3):441–450.

Prentice, S., and Roy, N. 2009. The belief roadmap: Efficient planning in belief space by factoring the covariance. *Int'l Journal of Robotics Research* 8(11-12):1448–1465.

Shewchuk, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry* 22(1-3):21–74.

Simmons, R., and Koenig, S. 1995. Probabilistic robot navigation in partially observable environments. In *Int'l Joint Conf. on Artificial Intelligence*, 1080–1087.

Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. In *IEEE Int'l. Conference on Robotics and Automation*, 3310–3317.

Stentz, A. 1995. The focussed D* algorithm for real-time replanning. In *Int'l Joint Conf. on Artificial Intelligence*, 1652–1659.

Thrun, S.; Burgard, W.; Fox, D.; et al. 2005. *Probabilistic robotics*, volume 1. MIT press Cambridge.

Toit, N. E. D., and Burdick, J. W. 2012. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics* 28(1):101–115.