IN *Proc. Int. Symp. on Robotics Research, 2009*

Motion Planning under Uncertainty for Robotic Tasks with Long Time Horizons

Hanna Kurniawati, Yanzhu Du, David Hsu, and Wee Sun Lee

Abstract Partially observable Markov decision processes (POMDPs) are a principled mathematical framework for planning under uncertainty, a crucial capability for reliable operation of autonomous robots. By using probabilistic sampling, point-based POMDP solvers have drastically improved the speed of POMDP planning, enabling POMDPs to handle moderately complex robotic tasks. However, robot motion planning tasks with long time horizons remain a severe obstacle for even the fastest point-based POMDP solvers today. This paper proposes *Milestone Guided Sampling* (MiGS), a new point-based POMDP solver, which exploits state space information to reduce the effective planning horizon. MiGS samples a set of points, called *milestones*, from a robot's state space, uses them to construct a compact, sampled representation of the state space, and then uses this representation of the state space to guide sampling in the belief space. This strategy reduces the effective planning horizon, while still capturing the essential features of the belief space with a small number of sampled points. Preliminary results are very promising. We tested MiGS in simulation on several difficult POMDPs modeling distinct robotic tasks with long time horizons; they are impossible with the fastest point-based POMDP solvers today. MiGS solved them in a few minutes.

1 Introduction

Efficient motion planning with imperfect state information is an essential capability for autonomous robots to operate reliably in uncertain and dynamic environments. With imperfect state information, a robot cannot decide the best actions on the basis of a single known state; instead, the best actions depend on the set of all possible states consistent with the available information, resulting in much higher compu-

H. Kurniawati
Singapore–MIT Alliance for Research Technology, e-mail: hannakur@smart.mit.edu
Most of the work was done while the author was with the Department of Computer Science, National University of Singapore.

Y. Du, D. Hsu, and W. S. Lee
Department of Computer Science, National University of Singapore, e-mail: {duyanzhu, dyhsu, leews}@comp.nus.edu.sg

tational complexity for planning the best actions. Partially observable Markov decision processes (POMDPs) [7, 17] provide a general and principled mathematical framework for such planning tasks. In a POMDP, we represent a set of possible states as a *belief*, which is a probability distribution over a robot's state space. We systematically reason over the belief space $\mathcal{B}$, the space of all beliefs, by taking into account uncertainty in robot control, sensor measurements, and environment changes, in order to choose the best robot actions and achieve robust performance. By incorporating uncertainty into planning, the POMDP approach has led to improved performance in a number of robotic tasks, including localization, coastal navigation, grasping, and target tracking [4, 6, 12, 15].

Despite its solid mathematical foundation, POMDP planning faces two major computational challenges. The first one is the "curse of dimensionality": a complex robotic task typically generates a high-dimensional belief space. If a robotic task is modeled with a discrete state space, its belief space has dimensionality equal to the number of states. Thus a task with 1,000 states has a 1,000-dimensional belief space! In recent years, point-based POMDP solvers [12] have made dramatic progress in overcoming this challenge by sampling the belief space and computing approximate solutions. Today, the fastest point-based POMDP solvers, such as HSVI2 [19] and SARSOP [9], can handle moderately complex robotic tasks modeled as POMDPs with up to 100,000 states in reasonable time. The success of point-based solvers can be largely attributed to *probabilistic sampling*, which allows us to use a small number of sampled points as an approximate representation of a high-dimensional belief space. The approximate representation substantially reduces computational complexity. The same reason underlies the success of probabilistic sampling in other related problems and approaches, e.g., probabilistic roadmap (PRM) algorithms [2] for geometric motion planning (without uncertainty).

The second major challenge is the "curse of history". In a motion planning task, a robot often needs to take many actions to reach the goal, resulting in a long time horizon for planning. Unfortunately the complexity of planning grows exponentially with the time horizon. Together, a long time horizon and a high-dimensional belief space compound the difficulty of planning under uncertainty. For this reason, even the best point-based POMDP algorithms today have significant difficulty with robotic tasks requiring long planning horizons (see Section 6 for examples).

To overcome this second challenge and scale up POMDP solvers for realistic robot motion planning tasks, we have developed a new point-based POMDP solver called *Milestone Guided Sampling* (MiGS). It is known from earlier work on related problems that the most important component of a planning algorithm based on probabilistic sampling is the sampling strategy [5]. MiGS reduces the planning horizon by constructing a more effective sampling strategy. It samples a set of points, called *milestones*, from a robot's state space S, uses the milestones to construct a compact, sampled representation of S, and then uses this representation of S to guide sampling in the belief space $\mathcal{B}$. The intuition is that many paths in $\mathcal{B}$ are similar. Using the sampled representation of the state space, MiGS avoids exploring many of the similar belief space paths, which enables us to capture the essential features of $\mathcal{B}$ with a small number of sampled points from $\mathcal{B}$.

We tested MiGS in simulation on several difficult POMDPs modeling distinct robotic tasks with long time horizons, including navigation in 2D and 3D environments, and target finding. These tasks are impossible with the fastest point-based POMDP solvers today. MiGS solved them in a few minutes.

2 Background

2.1 Motion Planning under Uncertainty

Despite its importance and more than almost three decades of active research [10, 21], motion planning under uncertainty remains a challenge in robotics. Several recent successful algorithms are based on the probabilistic sampling approach. Stochastic Motion Roadmap [1] combines PRM with the Markov decision process (MDP) framework to handle uncertainty in robot control, but it does not take into account uncertainty in sensing. Another method, Belief Roadmap [14], handles uncertainty in both robot control and sensing, but one major limitation is the assumption that the uncertainty can be modeled as Gaussian distributions. Unimodal distributions such as the Gaussian distribution are inadequate when robots operate in complex geometric environments.

POMDPs are a general framework that can overcome the above limitations. By tackling the difficulty of long planning horizons, MiGS brings POMDPs a step closer to being practical for complex robotics tasks.

2.2 POMDPs

A POMDP models an agent taking a sequence of actions under uncertainty to maximize its reward. Formally, it is specified as a tuple $(S, A, O, T, Z, R, \gamma)$, where S is a set of states describing the agent and the environment, A is the set of actions that the agent may take, and O is the set of observations that the agent may receive.

At each time step, the agent lies in some state $s \in S$, takes some action $a \in A$, and moves from a start state s to an end state s'. Due to the uncertainty in action, the end state s' is modeled as a conditional probability function $T(s, a, s') = p(s'|s, a)$, which gives the probability that the agent lies in s', after taking action a in state s. The agent then receives an observation that provides information on its current state. Due to the uncertainty in observation, the observation result $o \in O$ is again modeled as a conditional probability function $Z(s, a, o) = p(o|s, a)$.

In each step, the agent receives a real-valued reward $R(s, a)$, if it takes action a in state s. The goal of the agent is to maximize its expected total reward by choosing a suitable sequence of actions. When the sequence of actions has infinite length, we typically specify a discount factor $\gamma \in (0, 1)$ so that the total reward is finite and the problem is well defined. In this case, the expected total reward is given by $\mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$, where s_t and a_t denote the agent's state and action at time t.

The solution to a POMDP is an *optimal policy* that maximizes the expected total reward. In a POMDP, the state is partially observable and not known exactly. So we rely on the concept of beliefs. A POMDP policy $\pi : \mathcal{B} \rightarrow A$ maps a belief $b \in \mathcal{B}$ to the prescribed action $a \in A$.

A policy π induces a value function $V_\pi(b) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | b, \pi\right]$ that specifies the expected total reward of executing policy π starting from b. It is known that V^*, the value function associated with the optimal policy π^*, can be approximated arbitrarily closely by a convex, piecewise-linear function,

$$V(b) = \max_{\alpha \in \Gamma} (\alpha \cdot b) \tag{1}$$

where Γ is a finite set of vectors called α-vectors and b is the discrete vector representation of a belief. Each α-vector is associated with an action. The policy can be executed by selecting the action corresponding to the best α-vector at the current belief. So a policy can be represented as a set of α-vectors.

Given a policy, represented as a set Γ of α-vectors, the control of the agent's actions, also called policy execution, is performed online in real time. It consists of two steps executed repeatedly. The first step is action selection. If the agent's current belief is b, it finds the action a that maximizes $V(b)$ by evaluating (1). The second step is belief update. After the agent takes an action a and receives an observation o, its new belief b' is given by

$$b'(s') = \tau(b, a, o) = \eta Z(s', a, o) \sum_s T(s, a, s') b(s) \tag{2}$$

where η is a normalization constant. The process then repeats.

2.3 Point-based POMDP Solvers

The adoption of POMDPs as a planning framework in robotics has been hindered by the high dimensional belief space and the long time horizon typical of many robotics tasks. Many approaches have been proposed to alleviate these difficulties [21]. Point-based solvers [9, 12, 19, 20] are currently the most successful approach. Using the idea of probabilistic sampling, they have made impressive progress in computing approximate solutions for POMDPs with large number of states and have been successfully applied to a variety of non-trivial robotic tasks, including coastal navigation, grasping, target tracking, and exploration [4, 11, 12, 13, 18]. Despite this impressive progress, even the best point-based POMDP solvers today have significant difficulty with robotic tasks that require long planning horizons.

MiGS follows the approach of point-based POMDP solvers, but aims at overcoming the difficulty of long horizons. Learning from the successful PRM approach for geometric motion planning [2], MiGS tries to construct a more effective strategy for sampling the belief space.

3 Milestone Guided Sampling

A key idea of point-based POMDP solvers is to sample a set of points from $\mathcal{B}$ and use it as an approximate representation of $\mathcal{B}$. Let $\mathcal{R} \subseteq \mathcal{B}$ be the set of points reachable from a given initial belief point $b_0 \in \mathcal{B}$ under arbitrary sequences of actions and observations. Most of the recent point-based POMDP algorithms sample from $\mathcal{R}$ instead of $\mathcal{B}$ for computational efficiency. The sampled points form a belief tree $\mathcal{T}$ (Fig 1). Each node of $\mathcal{T}$ represents a sampled point $b \in \mathcal{B}$. The root of $\mathcal{T}$ is the initial belief point b_0. To sample a new point b', we pick a node b from $\mathcal{T}$ as well as an action $a \in A$ and an observation $o \in O$ according to suitable probability distributions or heuristics. We then compute $b' = \tau(b, a, o)$ using (2) and insert b' into $\mathcal{T}$ as a child of b. If a POMDP requires an effective planning horizon of h actions and observations, $\mathcal{T}$ may contain $\Theta((|A||O|)^h)$ nodes in the worst case, where $|A|$ is the number of actions and $|O|$ is the number of observations for the POMDP. Thus any point-based solvers trying to construct $\mathcal{T}$ exhaustively must have running time exponential in h and suffer from the "curse of history".

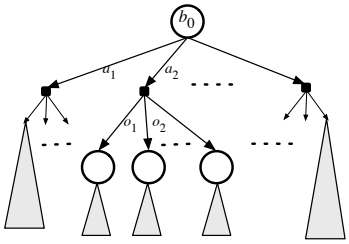


Fig. 1 The belief tree rooted at b_0.

To overcome this difficulty, let us consider the space from which we must sample. If the effective planning horizon is h, we must sample from a subset of $\mathcal{R}$ that contains $\mathcal{R}_h$, the set of belief points reachable from b_0 with at most h actions and observations. Our difficulty is that the size of $\mathcal{R}_h$ grows exponentially with h. The basic idea of MiGS is to sample $\mathcal{R}_h$ hierarchically at multiple resolutions and avoid exhaustively sampling $\mathcal{R}_h$ unless necessary.

To do so, MiGS builds a *roadmap* graph G in a robot's *state space* S. The nodes of G are states sampled from S and are called *milestones*. An edge e between two milestones s and s' of G is annotated with a sequence of actions $(a_1, a_2, \ldots, a_\ell)$ that can bring the robot from s to s'. The edge e is also annotated with a sequence of states $(s_0, s_1, \ldots, s_\ell)$ that the robot traverses under the actions $(a_1, a_2, \ldots, a_\ell)$, with $s_0 = s$ and $s_\ell = s'$. If we think of G as a collection of edges, each representing a sequence of actions, we can then use such sequences of actions to construct the belief tree $\mathcal{T}$. At a node b, we apply a *sequence of actions* associated with a selected edge of G, instead of a single action, to derive a child node b'. Suppose, for example, that G has maximum degree d and each edge of G contains an action sequence of length ℓ. Then, for a POMDP with time horizon h, $\mathcal{T}$ contains at most $\mathcal{O}((d\,|O|^\ell)^{h/\ell}) = \mathcal{O}(d^{h/\ell}|O|^h)$ nodes. This indicates that the action sequences encoded in G help in reducing the effect of long planning horizons due to actions, but not necessarily observations. Since the size of $\mathcal{T}$ grows exponentially with h, the reduction is nevertheless significant.

To sample at multiple resolutions, we start with a roadmap with a large ℓ value. In other words, we sample S coarsely and connect the milestones with long sequences of actions. We then refine the sampling of S and gradually reduce the ℓ value.

Now it should be clear that MiGS is indeed faster, as the belief tree $\mathcal{T}$ is smaller. However, a more fundamental question remains: since the roadmap G contains only a subset of sampled states and not all states in the state space S, do the belief points sampled with the help of G cover the entire reachable belief space well and likely lead to a good approximation to the optimal value function and the optimal policy? The answer is yes, if we sample S adequately in a sense which we now explain.

Denote by Γ^* a set of α-vectors representing an optimal value function. Given a constant $\epsilon > 0$, we partition the state space S into a collection of disjoint subsets so that for any $\alpha \in \Gamma^*$ and any two states s and s' in the same subset, $|\alpha(s) - \alpha(s')| \leq \epsilon$. Intuitively, the partitioning condition means that any two states in the same subset are similar in terms of their significance in the optimal value function. The constant ϵ controls the resolution of partitioning. We call such a partitioning of S an ϵ-partitioning and denote it by $\mathcal{K}$. The partitioning $\mathcal{K}$ induces a distance metric on the belief space $\mathcal{B}$:

Definition 1. Let $\mathcal{K}$ be an ϵ-partitioning of the state space S. The distance between any two beliefs b and b' in $\mathcal{B}$ with respect to $\mathcal{K}$ is

$$d_{\mathcal{K}}(b, b') = \sum_{K \in \mathcal{K}} \left| \sum_{s \in K} b(s) - \sum_{s \in K} b'(s) \right|. \tag{3}$$

This new metric is more lenient than the usual L_1 metric and is upper-bounded by the L_1 metric. It measures the difference in probability mass for subsets of states rather than individual states. This is desirable, because the states within a subset $K \in \mathcal{K}$ are similar under our assumption and there is no need to distinguish them. Using $d_{\mathcal{K}}$, we can derive a Lipschitz condition on the optimal value function $V^*(b)$:

Theorem 1. *Let $\mathcal{K}$ be an ϵ-partitioning of the state space S. For any b and b' in the corresponding belief space $\mathcal{B}$, if $d_{\mathcal{K}}(b, b') \leq \delta$, then $|V^*(b) - V^*(b')| \leq \frac{R_{\max}}{1-\gamma}\delta + 2\epsilon$, where $R_{\max} = \max_{s \in \mathcal{S}, a \in \mathcal{A}} |R(s, a)|$.*

The proof is given in the appendix. Theorem 1 provides a sampling criterion for approximating V^* well. Suppose that B is a set of sampled beliefs that *covers* the belief space $\mathcal{B}$: for any $b \in \mathcal{B}$, there is a point b' in B with $d_{\mathcal{K}}(b, b') \leq \delta$, where δ is some positive constant. Theorem 1 implies that the values of V^* at the points in B serve as a good (sampled) approximation to V^*. Furthermore, to estimate these values, we do not need to consider all the states in S, because $d_{\mathcal{K}}$ does not distinguish states within the same subset $K \in \mathcal{K}$; it is sufficient to have one representative state from each subset. This justifies MiGS' sampling of the state space during the roadmap construction.

Of course, MiGS does not know the partitioning $\mathcal{K}$ in advance. To sample S adequately, one way is to use uniform random sampling. If each subset $K \in \mathcal{K}$ is sufficiently large, then we can guarantee that uniform sampling generates at least one sampled state from each $K \in \mathcal{K}$ with high probability. To improve efficiency, our implementation of MiGS uses a heuristic to sample S. See Section 4 for details.

We now give a sketch of the overall algorithm. MiGS iterates over two stages. In the first state, we sample a set of new milestones from S, and then use it to construct

Algorithm 1 Perform α-vector backup at a belief b.

BACKUP(b, Γ)

1: For all $a \in A$, $o \in O$, $\alpha_{a,o} \leftarrow \text{argmax}_{\alpha \in \Gamma}(\alpha \cdot \tau(b, a, o))$.
2: For all $a \in A$, $s \in S$, $\alpha_a(s) \leftarrow R(s, a) + \gamma \sum_{o,s'} T(s, a, s')Z(s', a, o)\alpha_{a,o}(s')$.
3: $\alpha' \leftarrow \text{argmax}_{a \in \mathcal{A}}(\alpha_a \cdot b)$
4: Insert α' into Γ.

or refine a roadmap G. In the second stage, we follow the approach of point-based POMDP solvers and perform *value iteration* [16] on a set Γ of α-vectors, which represents a piecewise-linear lower-bound approximation to the optimal value function V^*. Exploiting the fact that V^* must satisfy the Bellman equation, value iteration starts with an initial approximation to V^* and performs backup operations on the approximation by iterating on the Bellman equation until the iteration converges. What is different in value iteration for point-based POMDP solvers is that backup operations are performed only at a set of sampled points from $\mathcal{B}$ rather than the entire $\mathcal{B}$. In MiGS, we sample incrementally a set of points from $\mathcal{B}$ by constructing a belief tree $\mathcal{T}$ rooted at an initial belief point b_0. To add a new node to $\mathcal{T}$, we first choose an existing node b in $\mathcal{T}$ in the least densely sampled region of $\mathcal{B}$, as this likely leads to sampled beliefs that cover $\mathcal{B}$ well. We then choose a suitable edge e from the roadmap G and use the associated action sequence $(a_1, a_2, \ldots, a_\ell)$ and state sequence $(s_0, s_1, s_2, \ldots, s_\ell)$ to generate a new node. Specifically, we first generate an observation sequence $(o_1, o_2, \ldots, o_\ell)$ so that each o_i is consistent with s_i and a_i, to be precise, $Z(s_i, a_i, o_i) = p(o_i|s_i, a_i) > 0$, for $1 \leq i \leq \ell$. We then start at b and apply the action-observation sequence $(a_1, o_1, a_2, o_2, \ldots, a_\ell, o_\ell)$ to generate a sequence of new beliefs $(b_1, b_2, \ldots, b_\ell)$, where $b_1 = \tau(b, a_1, o_1)$ and $b_i = \tau(b_{i-1}, a_{i-1}, o_{i-1})$ for $2 \leq i \leq \ell$. Finally, b_ℓ is inserted to $\mathcal{T}$ as a child node of b, while $(b_1, b_2, \ldots, b_{\ell-1})$ is associated with the edge from b to b_ℓ for backup operations. After creating the new node b_ℓ, we perform backup operations for every belief associated with the nodes and edges of $\mathcal{T}$ along the path from b_ℓ to the root b_0. A backup operation at b improves the approximation of $V^*(b)$ by looking ahead one step. We perform the standard α-vector backup (Algorithm 1). Each backup operation creates a new α-vector, which is added to Γ to improve the approximation of V^*. We repeat the sampling and backup processes until a sufficiently large number of new sampled beliefs are obtained. If necessary, we repeat the two stages to refine the roadmap G and sample additional new beliefs. The details for these two stages are reported in Sections 4 and 5, respectively.

4 State space sampling and roadmap construction

MiGS is designed for general motion planning problem where the robot's goal is to reach one of the possible goal state(s). It assumes that positive reward is given only when the robot reaches the goal state.

4.1 Sampling the milestones

MiGS samples the milestones from the state space without replacement. It biases sampling towards states that are more likely to improve $V(b_0)$ significantly, e.g., states that are useful for either reducing uncertainty or gaining high reward. In particular, MiGS samples a state $s \in S$ using the probability $P(s)$ as follows,

$$P(s) \propto K^{I(s)} \tag{4}$$

where K is a constant and $I(s)$ indicates the importance of visiting s in reducing uncertainty and gaining reward. The importance function we use, $I : S \rightarrow \mathbb{R}$, is a weighted sum of an expected reward function and a localization function, $avgReward(s) + \lambda \times locAbility(s)$, where the weight λ determines the importance of localization relative to the reward. Since localization depends on both the action performed and the observation perceived, we compute $locAbility(s)$ as an expected value over all possible actions and observations, assuming an action is selected uniformly at random and the observation is perceived according to the observation distribution function. More precisely, $locAbility(s) = \frac{1}{|A|}\sum_{a\in A}\sum_{o\in O} P(o|s,a) \cdot usefulness(s,o,a)$. To compute how useful an observation is towards localizing a state, we compute the posterior probability of being in the state after the observation is received, assuming a uniform prior on the states, $usefulness(s,a,o) = P(o|s,a)/\sum_{s\in S} P(o|s,a)$. The reward component is $avgReward(s) = \frac{1}{|A|}\sum_{a\in A} R(s,a)$.

4.2 Partitioning the state space

Once a set of milestones has been sampled, MiGS partitions the state space based on a notion of "distance" to the milestones. To help establish the notion of distance, MiGS constructs the state graph $\mathcal{S}$, a weighted multi-digraph where the vertices are states in S. We will refer to the vertices of $\mathcal{S}$ and their corresponding states interchangeably, as there is no confusion. An edge from $s \in S$ to $s' \in S$, labeled with action $a \in A$, exists in $\mathcal{S}$ whenever $T(s,a,s') > 0$. The weight of the edges act as the distance for partitioning. We define the weight $w((\overline{ss'},a))$ of an edge $(\overline{ss'},a)$ as the total regret of performing action a, and continuing from s' even when the robot may be at a state other than s' after performing a from s. More precisely,

$$w((\overline{ss'},a)) = c(s,a) + \sum_{s''\in S} T(s,a,s'') \cdot r(s',s'') \tag{5}$$

where $c(s,a)$ is the cost of performing action a from s. For computational efficiency, we would like the weight to always be positive. Therefore, we set $c(s,a) = -R(s,a)$ if $R(s,a) < 0$ and $c(s,a) = 0$ otherwise. The second component indicates how different the future total reward can be if we continue from s'. The function $r(s',s'')$ can be defined in many ways, including based on MDP value, i.e., $|V^*_{MDP}(s') - V^*_{MDP}(s'')|$. For simplicity, MiGS sets $r(s',s'')$ as a constant positive value whenever $s' \neq s''$ and 0 otherwise.

To partition S, MiGS uses inward Voronoi partition [3] on the state graph $\mathcal{S}$ with M as the Voronoi sites. It partitions the vertices of $\mathcal{S}$ into a set of Voronoi sets $\{Vor(m)|m \in M\} \bigcup U$, where the Voronoi set $Vor(m)$ of a milestone m is the set of vertices whose distance to m is less than the distance to any other milestone in M, and U is the set of vertices that can not reach any milestone in M.

4.3 Inserting the edges

MiGS constructs the roadmap G on top of the state graph $\mathcal{S}$. The vertices are the milestones and an edge $\overline{mm'}$ is inserted to G whenever there is a path from m to m' in the graph induced by $Vor(m) \cup Vor(m')$. MiGS will then annotate each edge $\overline{mm'}$ in G with a sequence of actions that can bring the robot from m to m' and a sequence of states that the robot traverses under the sequence of actions that annotates $\overline{mm'}$. The two sequences that annotate $\overline{mm'}$ are constructed based on the shortest path $\Pi(m, m')$ from m to m' in the graph induced by $Vor(m) \cup Vor(m')$. The sequence of actions is the action labels in the sequence of edges in $\Pi(m, m')$, while the sequence of states is the sequence of vertices in $\Pi(m, m')$. The weight of $\overline{mm'}$ is then the total weight of $\Pi(m, m')$.

4.4 Roadmap refinement

Similar to most probabilistic roadmap, MiGS refines the roadmap by sampling additional milestones and reconstructing the roadmap, taking into account the newly added milestones. The main question is when should MiGS refines the roadmap. Refining the roadmap too often is costly, while refining the roadmap too seldom may slow down MiGS in covering the belief space well. In the current implementation, MiGS uses a heuristic. It refines the roadmap when the approximation of $V^*(b_0)$ does not improve after a consecutive pre-specified number of beliefs are expanded and the corresponding backups are performed.

Now, since the milestones are sampled without replacement from S, after awhile, there will be no more milestones that can be sampled. When this happens, although all vertices of $\mathcal{S}$ have become milestones, the sampling guides that correspond to some edges of $\mathcal{S}$ may not be in G yet. To refine G further, MiGS inserts additional edges to G, such that each edge, including self-loops, of $\mathcal{S}$ corresponds to an edge of G. This refinement strategy ensures that given enough time, MiGS generates all possible action-observation combinations for sampling beliefs reachable from b_0. And therefore, MiGS is probabilistically complete in the sense that given enough time, MiGS is guaranteed to sample the set of beliefs representative enough to generate an optimal policy.

5 Belief space sampling

So far we have discussed the roadmap G, a more compact representation of the state space. Now, the question is how to use G to guide sampling in the belief space.

To favor sampling beliefs that are more likely to improve covering, MiGS favors expanding nodes of $\mathcal{T}$ that lies far from most of the other beliefs. For this, MiGS

sets the weight $w(b)$ of a node b in $\mathcal{T}$, to be the number of nodes in $\mathcal{T}$ that lie within a small pre-specified distance from b. MiGS will then select a node b for expansion based on the probability $P(b) \sim \frac{1}{w(b)}$.

To enable sampling different beliefs that lie within the planning horizon fast, we would like to expand b_0 using roadmap paths from a state in the support of b_0 to a possible goal state. However, these long roadmap paths may be misleading, in the sense that the path turns out to generate beliefs that lie in a very localized region of the belief space. Therefore, to alleviate wasting a lot of computational resources, MiGS expands beliefs using *edges* of G, but maintains a memory of which path is being used for expanding a particular branch of $\mathcal{T}$. For instance, if b' is a node of $\mathcal{T}$ generated by expanding b using edge $\overline{mm'}$ of G, then b' is annotated with m'. The next time b' is selected for expansion, MiGS chooses an out-edge of m' in G according to an arbitrary ordering in a circular fashion, returning to the first out-edge after the last out-edge is used. The idea here is to iteratively expand $\mathcal{T}$ using shorter partial paths, and then uses the generated belief to predict which path is more likely to significantly improve belief-space covering.

6 Experimental Setup and Results

The purpose of our experiment is two folds. One (Section 6.1) is to compare the performance of MiGS with the best point-based POMDP solvers and other alternatives to motion planning with uncertainty. The other (Section 6.2) is to test the robustness of the generated policy.

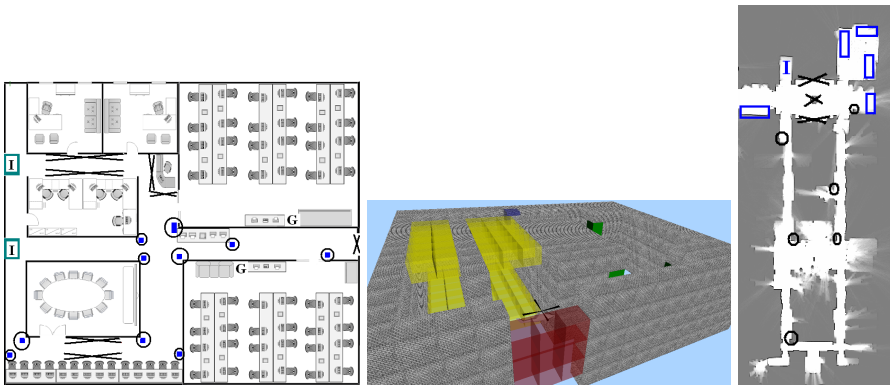
6.1 Comparison with Other Planners

We tested MiGS in several complex realistic robotics scenarios that require long planning horizon. The scenarios are presented in Section 6.1.1. The experimental setup and results are presented in Section 6.1.2 and Section 6.1.3, respectively.

6.1.1 Scenarios

In our experiment, we use the three scenarios below.

(a) ***2D-Navigation.*** In this problem, a 2-DOFs mobile robot navigates in a research lab (Fig 6.1.1(a)). The robot's position is represented as a uniform grid of size 60×70. The robot needs to navigate from the entrance of the lab (marked with "I") to one of the goal states (marked with "G"), while avoiding obstacles. The robot never knows its exact position, but it can localize well at some parts of the environment, marked with circles. At each step, the robot performs an action to move to one of its eight adjacent cells. However, due to control error, the robot will only reach its intended destination 90% of the time. For the rest of the time, it may remain in its cell or drift to the left or to the right of its intended destination. Moreover, some parts (marked with crosses) of the lab are occupied by hostile people, that would "abuse" the robot once they see it. Despite the imperfect information about its position and its control uncertainty, the robot needs to decide which way to go such that it can reach the goal as fast as possible while avoiding both obstacles and dangerous places in the environment.



(a) *(b)* *(c)*

Fig. 2 Experimental scenarios. (a) *2D-Navigation.* (b) *3D-Navigation.* (c) *Target Finding.*

(b) ***3D-Navigation.*** In this problem, a 5-DOFs unmanned aerial vehicle (UAV) navigates in a tunnel where GPS signal is not available. The robot's configuration is represented as $(x, y, z, \theta_p, \theta_y)$, where the first three DOFs are the robot's position in a 3D environment, θ_p is the pitch angle, and θ_y is the yaw angle. The configuration space is discretized into grids. The position dimensions are represented as a 3D uniform grid of 5 levels and 18×14 positions at each level (Fig 6.1.1(b)). The pitch angle ranges from -45^0 to 45^0 and is discretized into three equal size cells, while the yaw angle ranges from 0^0 to 360^0 and is discretized into eight equal size cells. The robot needs to navigate from the entrance of the tunnel (colored red) to a goal position (colored blue). The robot never knows its exact configuration. It can localize by observing the landmarks (colored green) in the environment. However, due to limited sensing ability, it can only observe the landmarks when the robot is in front of the landmark and is heading towards the landmark. At each time step, the robot can either rotate in place or move forward to one of its adjacent cells according to its heading. However, when the robot moves forward, 10% of the time, the robot fails to reach its destination state, instead it drifts to its left or right or remains at the same cell. Moreover, the robot needs to be careful, as the ceiling and floor of some passages of the tunnel have become bat nests (colored yellow) due to long abandonment. If the robot hits a bat nest, the bats would damage the robot and the robot would stop functioning. The main problem in this task is similar to that of the 2D-Navigation scenario, but the state space is much larger. Despite the imperfect information about its position and its control uncertainty, the robot needs to decide which way to go such that it can reach the goal as fast as possible while avoiding both obstacles and dangerous places in the environment.

(c) ***Target Finding.*** In this problem, a 2-DOFs mobile robot needs to find a moving target in an environment (Fig 6.1.1(c), courtesy of the Radish data set). The state space is represented as (r, t), where r is the robot's position and t is the target's position. These positions are represented as a uniform grid of size 49×29. The target may be in one of the places marked with rectangles. It can move within the rectangle, but it can not move to different rectangles. The target motion behavior is entirely

unknown. The robot starts from an initial position marked with "I" and needs to find the target by exploring the possible places of the target. The robot never knows its exact position, but it can localize itself at places marked with circles. Furthermore, due to sensing limitation, the robot will only know the position of its target, and hence accomplish the task, when they are in the same grid cell. At each step, the robot performs an action to move to one of its eight adjacent cells. However, due to control error, the robot will only reach its intended destination 85% of the time. For the rest of the time, it may remain in its cell or drift to the left or to the right of its intended destination. Furthermore, some parts (marked with crosses) of the environment are too dangerous for the robot to pass. Therefore, despite of the imperfect information about its position and its control uncertainty, the robot needs to decide an exploration strategy that finds the target as fast as possible while avoiding both obstacles and dangerous places in the environment.

6.1.2 Experimental Setup

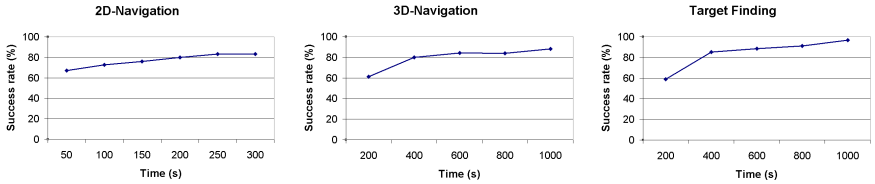
We implemented MiGS in C++ on top of the software package APPL v0.2 [9]. We tested MiGS on the three tasks above and compared the results with PRM [8], a successful motion planner that does not take uncertainty into account, with QMDP [21] which is an approximate POMDP solver well-known in robotics, and with the fastest point-based POMDP solver today, HSVI2 [19]. PRM is implemented in C++. QMDP is implemented on top of the software package APPL v0.2. For HSVI2, we used the newest software released by their original authors, ZMDP v1.1.5. All the experiments were performed on a 2.66GHz Intel processor PC and 2GB memory.

For each task and each method, we performed preliminary runs to determine the suitable parameters, and used the best parameters for generating the results. For MiGS and HSVI2, we used the best parameters to run each method on each scenario for at most 2 hours.

For each task, we compared the success rate, i.e., the percentage that the robot accomplishes the given task successfully within a pre-specified time limit. For MiGS and PRM, we generate 30 different POMDP policies and roadmaps for each task and average the results, as these methods use randomization. For each task, each method, and each policy/roadmap, we ran 100 simulation trial runs to test how well the robot that uses a particular policy/roadmap performs in solving the given task.

6.1.3 Results

The results show that MiGS significantly out-performs other methods for planning with uncertainty, as well as the fastest POMDP solvers today. It is interesting to notice that in 3D-Navigation, PRM that does not take uncertainty into consideration performs better than QMDP. The reason is that the successful runs of PRM are due to luck. On lucky runs, the shortest path from a possible initial state reaches the goal state. However, due to uncertainty, most of the time, the shortest path heuristic moves the robot to a danger zone and prevents it from reaching the goal. QMDP is able to realize that the shortest path strategy has a very high risk, and therefore tries to avoid it. However, QMDP's one lookahead is not sufficient to generate an alternative policy that reaches the goal. By performing farther lookahead, HSVI2



MiGS' results.

2D-Navigation	% success
PRM	0.0
QMDP	0.0
HSVI2	0.0
MiGS	**83.3**

3D-Navigation	% success
PRM	14.4
QMDP	0.1
HSVI2	24.0
MiGS	**88.2**

Target finding	% success
PRM	–
QMDP	14.5
HSVI2	17
MiGS	**96.7**

Fig. 3 Experimental results. The time for MiGS includes all initialization and pre-processing needed to construct the roadmap. The results in the table for MiGS is after 300 seconds of runs for 2D-Navigation, and after 1,000 seconds of runs for 3D-Navigation and Target Finding. The results for HSVI2 are the results after 2 hours of runs. We do not apply PRM to the target finding task, as it is unreasonable. The goal (i.e., target) in this problem is moving dynamically, while PRM executes a pre-computed path blindly.

performs better than QMDP. In fact in general, HSVI2 performs much better than QMDP or other POMDP solvers today. However, due to the long planning horizon required to generate a good policy in the above problems, HSVI2 is still unable to perform well. The main reason for the poor performance of HSVI2 is that they over-commit to a single heuristic, i.e., the MDP heuristic, to guide its belief-space sampling. This strategy significantly alleviates the difficulty of planning in a high dimensional belief space, but at the cost of significant reduction in belief space exploration ability, which is important for performing well when the heuristic is misleading. MiGS alleviates this problem by using a more compact representation of the state space to guide belief-space sampling. This strategy enables MiGS to explore significantly different parts of the belief space fast.

A video demo of the policy generated for the above scenarios can be seen in http://bigbird.comp.nus.edu.sg/∼hannakur/migs.html. We have also tested MiGS on 2D navigation with uncertain map. The results are similar to the above results and we do not reported it here due to space limitation.

6.2 Robustness of The Generated Plan

Fig. 4 Simple navigation scenario.

In many robotics scenarios, the uncertainty model that one has for planning may not be accurate. Therefore, a desired property of motion planning with uncertainty is to generate motion strategies that are robust enough against slight distortion in the uncertainty model used for planning.

In this set of experiments, we use a simple scenario to test the robustness of the policy generated by MiGS. The scenario involves a 2-DOFs mobile robot navigating in a simple environment (Fig 4), represented as a uniform grid of size 42×20. In this scenario, the robot needs to navigate from the initial position (marked with "I") to the goal position (marked with "G"), while avoiding

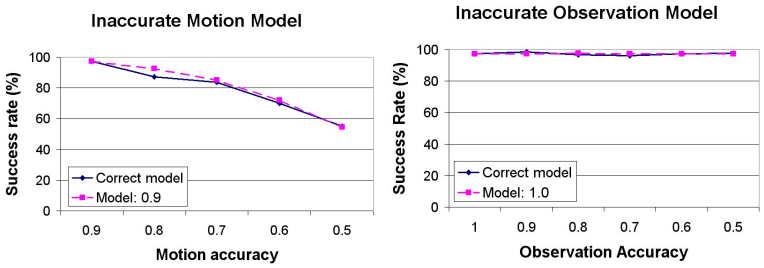


Fig. 5 The results of generating the policy in 3s. Similar trend holds for the policies generated at different time. The success rate of the "Correct model" is the benchmark. The policy used by the robot is generated based on the correct motion and observation models. (a) "Model: 0.9" means that the policy used by the robot is generated based on motion accuracy 0.9. (b) "Model: 1.0" means that the policy used by the robot is generated based on perfect observation accuracy.

obstacles and dangerous regions (marked with crosses). Due to limited sensing, the robot can only localize at places marked with circles. There are two landmarks in this environment, and we use the sensing model to encode how well the robot is able to differentiate these two landmarks. At each step, the robot performs an action to move to one of its eight adjacent cells. However, due to control error, the robot will not always reach its intended destination. To model the uncertainty of the system, we need to decide the motion accuracy of the robot and how well its sensing is in differentiating the two landmarks.

To test the robustness of the policy generated by MiGS, we first generate the policies for the above scenario with a particular motion and sensing model of the robot, and then use the generated policies for robots with different motion and sensing uncertainty to accomplish the same task. Since MiGS uses randomization, we generate 30 policies for a particular robot motion and sensing uncertainty. These policies are generated for robots with 0.9 motion accuracy and perfect sensing ability to differentiate which landmark it sees. We then use each of the 30 policies on robots with motion accuracy ranging from 0.8 to 0.5, and sensing accuracy ranging from 0.9 to 0.5. The average success rates are shown in Fig 5.

The results show that the policy generated by MiGS based on inaccurate motion or observation model can still be used by the robot to accomplish the given task well. The reason is that knowing a rough idea of a good motion strategy is often sufficient to accomplish the task. For instance, in this environment, as long as the robot knows to stay away from the passages containing dangerous regions, regardless of the exact motion it performs, the robot would reach the goal with high probability. This result corroborates the intuition that many paths in the belief space generate policies with similar quality.

7 Conclusion & Future Work

We have proposed *Milestone Guided Sampling* (*MiGS*), a new point-based POMDP solver that uses a set of sampled states, called milestones, to guide belief-space sam-

pling. MiGS uses the milestones to construct a more compact representation of the state space, and then uses this more compact representation of the state space to guide belief-space sampling. Reasoning using a more compact representation of the state space significantly reduces the planning horizons while still capturing most of the useful beliefs. Preliminary experimental results are very promising. They indicate that long planning horizons problems that are impossible to solve using the fastest POMDP solvers today, can be solved by MiGS in just a few minutes.

Two main challenges for enabling POMDP to be practical for realistic robotics problem are the large number of states and the long planning horizon typical of robotics problems. The recently introduced point-based algorithms have shown impressive progress in solving POMDP problems with very large number of states. However, the performance of point-based POMDP degrades significantly when the required planning horizon is long. By alleviating the difficulty of solving problems that require long planning horizon, we hope our work would bring POMDP a step closer to becoming a practical tool for robot motion planning in uncertain and dynamic environment.

Acknowledgements We thank Sylvie Ong and Shao Wei Png for reading the first draft of this paper and helping with scripting a POMDP model. This work is supported in part by AcRF grant R-252-000-327-112 from the Ministry of Education of Singapore.

Appendix 1 Proof of Theorem 1

Proof. The optimal value function V^* can be approximated arbitrarily closely by a piecewise-linear convex function and represented as $V^*(b) = \max_{\alpha \in \Gamma}(\alpha \cdot b)$ for a suitable set Γ of α-vectors. Let α and α' be the maximizing α-vectors at b and b', respectively. Without loss of generality, assume $V^*(b) \geq V^*(b)$. Thus $V^*(b) - V^*(b') \geq 0$. Since α' is a maximizer at b', we have $\alpha' \cdot b' \geq \alpha \cdot b'$ and $V^*(b) - V^*(b') = \alpha \cdot b - \alpha' \cdot b' \leq \alpha \cdot b - \alpha \cdot b' \leq \alpha \cdot (b - b')$. It then follows that

$$|V^*(b) - V^*(b')| \leq |\alpha \cdot (b - b')|.$$

Next, we calculate the inner product over the partitioned state space:

$$|V^*(b) - V^*(b')| \leq \Big|\sum_{s \in S} \alpha(s)(b(s) - b'(s))\Big| \leq \Big|\sum_{K \in \mathcal{K}} \sum_{s \in K} \alpha(s)(b(s) - b'(s))\Big|$$

Let s_K denote any state in the subset $K \in \mathcal{K}$. We have

$$\begin{aligned} & |V^*(b) - V^*(b')| \\ \leq & \Big|\sum_{K \in \mathcal{K}} \sum_{s \in K} (\alpha(s) - \alpha(s_K) + \alpha(s_K))(b(s) - b'(s))\Big| \\ \leq & \Big|\sum_{K \in \mathcal{K}} \sum_{s \in K} (\alpha(s) - \alpha(s_K))(b(s) - b'(s))\Big| + \Big|\sum_{K \in \mathcal{K}} \sum_{s \in K} \alpha(s_K)(b(s) - b'(s))\Big|. \end{aligned} \tag{6}$$

Let e_1 and e_2 denote the two terms in (6), respectively. We now bound e_1 and e_2 separately. By the definition of ϵ-partitioning, $|\alpha(s) - \alpha(s_K)| \leq \epsilon$ for all s and s_K in $K \in \mathcal{K}$. Thus,

$$e_1 \leq \sum_{K \in \mathcal{K}} \sum_{s \in K} \epsilon |b(s) - b'(s)| = \epsilon \sum_{s \in S} |b(s) - b'(s)| \leq 2\epsilon, \tag{7}$$

where the last inequality holds because the L_1 distance between any two beliefs is no greater than 2. Let us now consider e_2. Since the absolute values of α-vector coefficients are no more than $R_{\max}/(1-\gamma)$, it follows that

$$e_2 \leq \sum_{K \in \mathcal{K}} \Big|\alpha(s_K)\Big|\Big|\sum_{s \in K}(b(s) - b'(s))\Big| \leq \sum_{K \in \mathcal{K}} \frac{R_{\max}}{1-\gamma}\Big|\sum_{s \in K} b(s) - b'(s)\Big|.$$

Using the condition $d_{\mathcal{K}}(b, b') \leq \delta$, we get $e_2 \leq \frac{R_{\max}}{1-\gamma}\delta$. Combining this with (6) and (7) gives the desired result. □

References

1. R. Alterovitz, T. Simeon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *Proc. Robotics: Science and Systems*, 2007.
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*. The MIT Press, 2005.
3. M. Erwig. The graph voronoi diagram with applications. *Networks*, 36(3):156–163, 2000.
4. Kaijen Hsiao, L.P. Kaelbling, and T. Lozano-Perez. Grasping POMDPs. In *Proc. IEEE International Conference on Robotics & Automation*, pages 4685–4692, 2007.
5. D. Hsu, J.C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *International Journal of Robotics Research*, 25(7):627–643, 2006.
6. D. Hsu, W.S. Lee, and N. Rong. A point-based POMDP planner for target tracking. In *Proc. IEEE International Conference on Robotics & Automation*, pages 2644–2650, 2008.
7. L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
8. L.E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, 1996.
9. H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.
10. J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
11. J. Pineau and G. Gordon. POMDP planning for Robust Robot Control. In *Proc. International Symposium on Robotics Research*, 2005.
12. J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conferences on Artificial Intelligence*, pages 1025–1032, August 2003.
13. J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and result. *Robotics and Autonomous Systems*, 42(3–4):271–281, 2003.
14. Sam Prentice and Nicholas Roy. The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance. In *Proc. International Symposium on Robotics Research*, 2007.
15. N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.
16. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2002.
17. R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.
18. T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. Uncertainty in Artificial Intelligence*, 2004.
19. T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Uncertainty in Artificial Intelligence*, July 2005.
20. M.T.J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.
21. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.