Published in the IEEE International Conference on Robotics and Automation. :4822-4828 (2012)

Accounting for Uncertainty in Simultaneous Task and Motion Planning Using Task Motion Multigraphs

Ioan A. Şucan, Lydia E. Kavraki

Abstract— **This paper describes an algorithm that considers uncertainty while solving the simultaneous task and motion planning (STAMP) problem. Information about uncertainty is transferred to the task planning level from the motion planning level using the concept of a task motion multigraph (TMM). TMMs were introduced in previous work to improve the efficiency of solving the STAMP problem for mobile manipulators. In this work, Markov Decision Processes are used in conjunction with TMMs to select sequences of actions that solve the STAMP problem such that the resulting solutions have higher probability of feasibility. Experimental evaluation indicates significantly improved probability of feasibility for solutions to the STAMP problem, compared to algorithms that ignore uncertainty information when selecting possible sequences of actions. At the same time, the efficiency due to TMMs is largely maintained.**

I. Introduction

Robotic devices often need to perform tasks that involve the execution of multiple actions. For example, opening a door may mean moving to the door, pressing the handle and then pushing the door. Executing such sequences of actions implies that robots need to plan at two levels: (1) the task planning level, where sequences of actions that take the robot to its goal are computed (a topic often studied in the artificial intelligence community) and (2) the motion planning level, where individual motion plans that implement the actions selected at the task planning level are computed (a topic often studied in the robotics community [1], [2]). The feasibility of motion plans cannot always be quickly determined at task planning level, and thus decoupling of task planning from motion planning leads to infeasible solutions when motion plans cannot be computed for proposed task plans. As a result, recent approaches consider task planning and motion planning simultaneously (e.g., [3]–[11]). For the remainder of the paper we will refer to this problem as the simultaneous task and motion planning (STAMP) problem. The input to the STAMP problem is in the form of task graphs. These are directed acyclic graphs that represent possible sequences of actions that take the robot to the goal. A path in the task graph that connects the robot's initial state to a goal state represents a possible task plan. If every edge along the task plan has a motion plan associated to it such that the motion plans can be executed in sequence, a solution to the STAMP problem has been found [12].

Physical systems are characterized by imperfect actuation, imperfect sensing and imperfect models, all of which lead to uncertainty. This work concerns itself with solving the STAMP problem while accounting for uncertainty in the following way: if some form of uncertainty is considered at the motion planning level, that uncertainty information is transferred over and used at the task planning level as well. Uncertainty at the motion planning level has been considered in previous work (e.g., [13]–[18]). In this paper we assume that motion planners capable of accounting for some form of uncertainty are employed in the computation of individual motion plans for the actions that the robot considers. Furthermore, we assume these motion planners are capable of reporting a probability of feasibility for the motion plans they compute. While this probability of feasibility may not be explicitly reported by the algorithms described in previous work, a routine that reports a probability of feasibility for a computed motion plan can easily be added. Such a routine would use information from the planner's internal data structures; for example, it could report the minimum probability of feasibility for the segments that make up the reported motion plan. We also assume motion planning is performed solely under geometric constraints (also known as path planning). This paper shows how to account for uncertainty at the task planning level by using information gathered by motion planners so that more robust task plans (plans with higher probability of feasibility) are proposed when solving the STAMP problem. Consideration of uncertainty is essential for most practical applications, especially for complex sensor-based systems that operate in human environments (e.g., mobile manipulators).

The work presented here relies on the concept of a Task Motion Multigraph (TMM), which we have introduced in recent work [10]. TMMs have been developed for use with complex robotic systems such as mobile manipulators, and TMM-based algorithms have been shown to solve the STAMP problem efficiently [11]. In this work we further develop our TMM-based algorithm so that it can account for uncertainty using a Markov Decision Process (MDP) [19] and a motion planner that can report a probability of feasibility for the plans it computes, as described above. The experiments we conduct show significantly improved probability of feasibility for the solutions to the STAMP problem when using the updated algorithm compared to the original one. At the same time, the computational performance gains offered by TMMs are largely maintained.

II. Background and Related Work

A. Task Motion Multigraphs

In recent work [10] we introduced the concept of a Task Motion Multigraph (TMM) to help solve the STAMP

I.A. Şucan and L.E. Kavraki are with Rice University, Houston, TX, 77005, {isucan, kavraki}@rice.edu

problem for mobile manipulators. TMMs lead to increased efficiency because they encode the information from the input task graph as well as information about the robot hardware. In particular, TMMs encode the possible choices of hardware components to use when performing an action (e.g., an arm, or more generally, any set of joints), which corresponds to different motion planning options (different state spaces to plan in). TMMs thus enable motion planners to compute solutions in lower dimensional spaces when possible, which is especially useful for complex robots such as mobile manipulators, and allow for the computation of task plans that consider the progress of motion planners. For convenience, we include a brief definition for TMMs [10] and an efficient TMM-based algorithm we developed [11] for solving the STAMP problem.

Definition: A *task motion multigraph* (TMM) is a directed acyclic multigraph $G_M = (V_M, E_M)$ such that:

• $V_M = \{v \mid Q(v) \subset \mathcal{X}\}$ is a finite set of vertices. Every vertex v is associated with a set of robot states $Q(v) \subseteq \mathcal{X}$, where $\mathcal{X}$ is the full state space of the robot; $Q(v)$ can be explicitly specified as a set of states or implicitly specified in a manner that allows sampling.

• E_M is a finite multiset of edges representing all the motion planning options between all pairs of nodes $(v_i, v_j) \in V_M \times V_M$. For a pair of nodes (v_i, v_j) there may be multiple edges $E_{i,j} = \{e \mid e \text{ connects } v_i \text{ to } v_j\} \subseteq E_M$ that represent planning options between v_i and v_j. The elements of $E_{i,j}$ differ by the state space along which motion plans are to be computed. Given an edge $e \in E_{i,j}$, the state space used for planning is denoted as $Space(e)$. Motion plans along an edge $e \in E_{i,j}$ must start at a state $x \in Q(v_i)$, end at a state $x' \in Q(v_j)$ and lie in $Space(e)$.

Algorithm: The concept of a TMM was utilized to develop an efficient algorithm (Algorithm 1) for solving the STAMP problem [11]. This algorithm consists of roughly two parts: (1) proposing a sequence of actions that can take the robot to its goal (the *SelectPossibleTaskPath()* function) [line 2] and (2) computing a motion plan for an action along the proposed sequence [lines 3,4], or, if such a motion plan is not found, using a randomized process to attempt the computation of a different motion plan [lines 6,7]. Δt is the amount of time spent computing a motion plan at each attempt (repeated attempts continue from where the planner has left off).

Algorithm 1 TMM-Computation($G_M = (V_M, E_M)$)

```
1:  while timeSpent < MaxT do
2:      P ← SelectPossibleTaskPath(G_M)
3:      edge ← SelectEdgeFromPath(P)
4:      (edge', sol) ← TMM-MotionPlan(edge, Δt)
5:      if sol = nil then
6:          nextEdge ← SelectEdge(E_M, edge)
7:          (edge', sol) ← TMM-MotionPlan(nextEdge, Δt)
8:      if sol ≠ nil then
9:          RecordSolution(edge', sol)
10:     if HaveSolution(G_M) then
11:         return ExtractSolution(G_M)
12: return nil
```

B. Considering Uncertainty in Motion Planning

Algorithms that consider uncertainty at the motion planning level can be separated into two categories: ones that plan in the state space and ones that plan in the belief space.

1) Planning in the State Space: Algorithms that plan in the state space build upon already proven sampling-based algorithms such as the Probabilistic Roadmap (PRM) [20], and estimate probabilities of feasibility for segments that make up solution paths. Techniques that plan in the state space can consider different types of uncertainty. For example, uncertainty in sensing (e.g., [13], [14]) and uncertainty in actuation (e.g., [15], [16]) can be considered separately. Nevertheless, considering both uncertainty in sensing and uncertainty in actuation is possible (e.g., [17]).

Missiuro and Roy present a modified version of PRM that can account for uncertainty in the representation of the environment [13]. It is assumed that the vertices that make up the sensed obstacles in the environment are represented using Gaussian probability distributions, and uncertainty in the robot state is ignored. Alterovitz et al. present another modification of PRM which they call the Stochastic Roadmap Method (SMR) [15]. Their approach assumes a stochastic model of motion and is applied for of a system that accepts a finite set of controls.

2) Planning in the Belief Space: Another approach for considering uncertainty at the motion planning level is to plan in the belief space. A point in the belief space consists of the parameters necessary to represent a probability distribution for a point in the state space. For example, Gaussian models of probability or sets of particles can be used to approximate the belief state.

An approach similar in implementation to planning in the state space is the Belief Roadmap [21], which can consider uncertainty in state information. The idea behind this approach is to run PRM in the belief space. Points in the belief space are Gaussian probability distributions. To construct a roadmap in this space, Prentice and Roy [21] sample the means of the belief states — which are in fact elements of the state space — and factorize the covariance matrices in a manner that affords computational savings.

C. Considering Uncertainty in Task and Motion Planning

The work closest related to ours is by Kaelbling and Lozano-Pérez [22], where both task planning and motion planning are performed, but uncertainty information is propagated as a belief at the task planning level only. Works that employ Partially Observable Markov Decision Processes (POMDPs) (e.g., [23]) are also related, in the sense that such approaches could also be used to model uncertainty at the task planning level.

III. Considering Uncertainty in Task and Motion Planning using TMMs

The probability of a robot successfully executing a task depends on both the sequence of actions the robot performs and on the hardware components the robot uses for each action. For example, a robot executing actions that take it

through an empty environment may have more difficulty in localization when compared to a robot that executes actions that take it around corners and walls. At the same time, different actuators have different abilities in terms of following planned paths, and the set of actuators used determines the state spaces in which motions are planned. As such, the information that TMMs include, i.e., the possible sequences of actions that lead to the goal and the state spaces that could potentially be used to plan motions for those actions, directly affects a robot's capability to address uncertainty issues. For this reason, our previously introduced TMM-based framework is a natural choice for considering uncertainty at the task planning level. An added benefit of using TMMs is the increased efficiency for solving the STAMP problem, as demonstrated in our previous work [11].

As mentioned in Section II, Algorithm 1 consists of roughly two steps, the first of which proposes a possible sequence of actions that takes the robot to its goal (the *SelectPossibleTaskPath()* function) [line 2]. In our previous work, *SelectPossibleTaskPath()* was based on an assignment of costs to edges and Dijkstra's algorithm for shortest paths [10], [11]. In this work, the use of Dijkstra's algorithm is replaced by the use of Markov Decision Processes (MDPs) [19] and value iteration. MDPs provide a formal framework for considering probabilities that would otherwise be inserted in an ad-hoc way in the cost function used by Dijkstra's algorithm. Because we are using motion planners capable of reporting a probability of feasibility for their reported solutions, the implementation of *SelectPossibleTaskPath()* can use the value of those probabilities to propose sequences of actions that have a higher probability of feasibility. Furthermore, a user can additionally specify uncertainty information specific for edges of the TMM (e.g., the probability of a particular hardware component being able to follow a given path).

A. Markov Decision Processes

A Markov Decision Process (MDP) is a 4-tuple (S, A, T, R), where:

- S is the set of states the robot could be in.
- A is the set of actions the robot can perform in order to move between states.
- $T : S \times A \to 2^S$ is a transition function indicating the probabilities of transition between states; $T(s, a, s')$ is the probability of transitioning from state s to state s' under action a. Time is discretized, and at every step the robot takes, the transition function is evaluated to determine which state the robot reaches.
- $R : S \times A \to \mathbb{R}$ is a reward function; $R(s', a)$ is the reward the robot receives if it reaches state s' after performing action a. This reward is usually discounted over time using a parameter γ.

Often, the sets S and A are assumed to be finite. The solution to an MDP is a policy $\pi : S \to A$ which specifies what the optimal action is for every state the robot could be at. The notion of optimality is defined in terms of maximizing rewards:

$$\begin{aligned} \pi(s) &= \arg\max_a \sum_{s'} T(s, a, s')(R(s', a) + \gamma V(s')) \\ V(s) &= \sum_{s'} T(s, \pi(s), s')(R(s', \pi(s)) + \gamma V(s')), \end{aligned}$$

where γ is the discount factor for the MDP. This factor has the effect of diminishing the value of rewards the further they are in the future.

Typical algorithms that find the optimal policy for an MDP are policy iteration and value iteration [19].

B. Construction of an MDP for Finding Robust Task Plans

Given the information from a TMM $G_M = (V_M, E_M)$ at a particular time, an MDP (S, A, T, R) is constructed as follows:

- $S = V_M \cup \{Fail\}$; the vertices in the TMM become states in the MDP, and an additional state $(Fail)$ that corresponds to catastrophic failure is added.
- $A = \bigcup_{e \in E_M} Space(e)$; all the state spaces that could be used for planning are considered actions in the MDP.
- The MDP state transition function T is defined as follows. For every edge $e = (s, s') \in E_M$:

$$T(s, Space(e), s') = m(e) \cdot \begin{cases} p & \text{if } sol \\ h \cdot \frac{1}{1+t} & \text{if not } sol, \end{cases} \quad (1)$$

$$T(s, Space(e), Fail) = \sum_{s' \in \{s' | (s,s') \in E_M\}} (1 - T(s, Space(e), s')) \quad (2)$$

where $m(e)$ can be used to model uncertainty sources that are known before the computation of motion plans, p is the maximum probability of feasibility of a motion plan found while planning along e (in the state space $Space(e)$), t is the amount of time spent planning motions for edge e, and sol is a flag indicating whether any motion plans have been found for edge e. If sol is true, a value of p is available. If sol is false, we consider the probability of feasibility to be proportional to the amount of time already spent planning and a model specific constant h. At $t = 0$ (i.e., no planning was yet performed) $T(s, Space(e), s') = m(e) \cdot h$. In this work we consider even chances for finding a solution ($h = \frac{1}{2}$), and $m(e)$ was set in accordance with $Space(e)$: when $Space(e)$ corresponds to an arm, $m(e)$ is higher than when $Space(e)$ corresponds to a mobile base.

Because the probabilities of outgoing transitions from a particular state under a particular action do not necessarily sum up to 1, the transition function needs to also be normalized before use, but this is automatically handled by our implementation of value iteration.

The transition function brings information from the motion planning level to the task planning level and thus allows Algorithm 1 to make use of it. Intuitively, for every edge in the TMM, there exist two corresponding transitions in the MDP. The first transition represents moving to the desired state and the second transition represents failure. The probability of the desired transition depends on the state space of the system

(given by $m(e)$) and the computed motion plan (given by p) when such a plan is found. If motion plans do not yet exist for the edge e, a probability of feasibility that decreases as computation time increases is assumed.

- The reward function R of the MDP is defined such that:

$$R(Fail, \cdot) = -10000 \quad \textit{// large negative value} \tag{3}$$

$$R(s,a) = \begin{cases} R^0(s,a) \cdot d & \text{if } sol \\ -\xi \cdot dim(a) & \text{if not } sol \end{cases} \tag{4}$$

$$d = \begin{cases} dim(\mathcal{X}) - dim(a) & \text{if } p > P_C \\ dim(a) & \text{otherwise} \end{cases}$$

$$R^0(s,a) = \beta \cdot \left(\frac{1}{1 + e^{-\alpha \cdot (p - P_C)}} - \frac{1}{2} \right),$$

where $\mathcal{X}$ is the full state space of the robotic system, α, β, ξ are scaling factors, $a = Space(e)$, p is the maximum probability of feasibility of a motion connecting s to s', sol is a flag that indicates whether any motion plan connecting s to s' (irrespective of the state space) has been found and P_C is a trust threshold. Explanations for these variables follow.

The intention of the reward function is to penalize actions that are not desirable and to reward ones that are. At first, no probabilities are known for any of the actions in the TMM. In this case, the only reasonable approach is to consider a small penalty for each action in the corresponding MDP, so that we avoid a bias towards long policies.

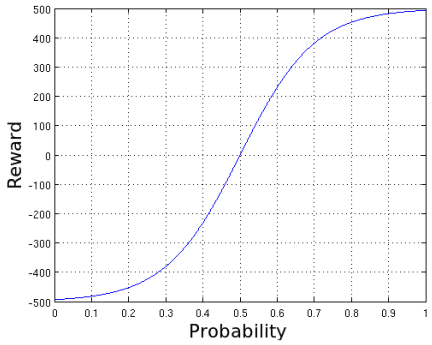

Fig. 1. Value of $R^0(\cdot,\cdot)$ function used in defining rewards for $P_C = 0.5$, $\xi = 0.05, \alpha = 10$ and $\beta = 1000$.

When a motion plan is found for a particular edge in the TMM, the probability of that motion plan can either be sufficiently high and then including the corresponding MDP transition in the followed task path is desirable, or it is a low probability and the corresponding MDP transition should be avoided. The distinction is made using the user defined trust threshold P_C: edges that have $p > P_C$ will yield a reward that is positive, while ones that have $p < P_C$ will yield a negative reward; α and β are positive factors. A representation of the R^0 function is shown in Figure 1. β sets the scale of the returned rewards: the value of R^0 is always in the range $(-\frac{\beta}{2}, \frac{\beta}{2})$. α regulates how fast the value of R^0 reaches its asymptotes. The higher the value of α, the closer R^0 is to a step function. P_C is a key parameter, as it determines whether the reward is positive or not. When a probability p is not available (because a solution has not yet been found), the reward achieved by the edge is considered to be a small negative value, and the ξ positive factor is used for scaling.

In this work, the discount factor γ for the constructed MDP is always set to 0.95. The constants for the definition of the reward functions are $P_C = 0.5, \xi = 0.05, \alpha = 10$ and $\beta = 1000$; $m(e)$ was set to 0.99 when $Space(e)$ corresponds to an arm, to 0.90 when $Space(e)$ corresponds to the base and to 0.75 otherwise.

The values mentioned here are by no means fixed. For practical applications, these values can be tuned. The intention in this work is only to show that TMMs can easily be used to consider uncertainty at task level. The values selected, as well as the MDP construction model, already lead to results supporting this point, without further tuning.

C. Using MDPs in the TMM Algorithm

Given an MDP constructed as shown above, value iteration is executed to find the optimal policy. Let $\pi : S \to A$ be the optimal policy. The optimality in this case is with respect to the arbitrary reward model described above. As such, the use of value iteration could be replaced by faster approximating algorithms. The sequence of states $s_0, s_1, \ldots, s_k$ is extracted from the MDP such that s_0 is the MDP state that corresponds to the root of the TMM,

$$s_{i+1} = \max_{s' \in S} \left(T(s_i, \pi(s_i), s') \cdot R(s', \pi(s_i)) \right),$$

and s_k corresponds to a goal state in the TMM (a goal is always reachable because the TMM is acyclic). The sequence of states above is used to produce a sequence of possible actions that take the robot from the root of the TMM to one of the goals. This computation replaces the call to Dijkstra's algorithm in Algorithm 1 [line 2].

IV. Experimental Results

We compare the original version of Algorithm 1 [11] to the version that uses MDPs described in this work. We use a simulation of the Willow Garage PR2 (a mobile manipulator) as the robotic system for testing our approach. For the construction of TMMs from input task graphs we assume every motion planning action can be performed using any combination of the state spaces that correspond to the base, the left arm and the right arm of the PR2.

A. Source of Uncertainty

To be able to construct the MDP as described in Section III-B, a probability of feasibility needs to be determined for every motion plan computed as part of solving the STAMP problem. Ideally, we should use a motion planner that considers some form of uncertainty and reports the probability of feasibility for computed plans, as described in Section I. To simplify our implementation, we used a sampling-based planner that was readily available but does not report uncertainty, and we assigned a probability of feasibility to the computed motion plan in a subsequent step.

For the experiments shown in this section, a simple model of uncertainty was assumed: the probability of feasibility of a particular robot state depends on the robot's position in the

ground plane at that state. The probability of feasibility is depicted in Figures 2, 3, 4 and 5 as levels of gray that vary from 0.1 (almost black, high uncertainty) to 1.0 (white, no uncertainty). The value of the intensity of the gray area that corresponds to a particular robot state is considered to be the probability of feasibility for that state. For Figures 2, 3 and 5, the gray levels depend on the distance to walls, thus mimicking a simplistic model of uncertainty in localization. This source of uncertainty is merely an example and many different sources of uncertainty could be used. For example, the environment in Figure 4 shows an arbitrary distribution of areas with high uncertainty.

When a motion planner computes a plan, that plan is projected to the ground plane and the intensity of the darkest spot the plan touches is considered to be the probability of feasibility of that plan. The values computed for this source of uncertainty are not intended to be realistic. Their purpose is to serve as proof of concept and emphasize the fact that using information about uncertainty at the task planning level improves the probability of feasibility of the final solution.

The probability of feasibility for a solution to the STAMP problem is defined to be:

$$prob = \frac{\sum_u Prob(u) \cdot Length(u)}{\sum_u Length(u)},$$

where u stands for the individual motion plans that make up the STAMP problem solution, $Prob(u)$ represents the probability of feasibility reported by the motion planner for plan u and $Length(u)$ represents the length of that plan.

B. Experimental Setup

Algorithm 1 is updated as indicated in Section III and executed on the environments shown in Figures 2, 3, 4 and 5. Each figure consists of two sides: the left side is a representation of the environment and the right side is a representation of the input task graph [10]. The black dot at the bottom-left of each picture represents the scale of the robot within the environment. The environment in Figure 2 is very small and very simple; it is a motivating example for considering uncertainty at the task level as well. The environment in Figure 3 has a large open area that does not provide good localization information. The same environment is presented in Figure 4, but with a different distribution of areas with high uncertainty. In both Figure 3 and Figure 4, the shorter sequence of actions takes the robot through areas with low probabilities of feasibility. This environment is chosen to show that the algorithm presented in this work can detect this fact and choose the longer route, which has higher probability of feasibility. All individual motion plans are fast to compute. For comparison, we include the environment in Figure 5, which is fairly cluttered and individual motion plans are significantly slower to compute, but good localization information is available throughout the environment.

For all environments, the vertices of the task graphs require motion planning for both the robot's base and its arms, as we had done in previous work [11]. However, the model of uncertainty we assumed does not affect arm motions. As such, the results presented reflect the value of accounting for uncertainty in base motion only. The difference in probability of success would be larger if considering uncertainty in arm motion as well.

For experiments presented in this section, all values are averaged over 30 runs, and the motion planner used is RRT-Connect [24] from OMPL [25] (although different types of motion planners could be used for each edge in the TMM). The results are shown in Tables I, II, III and IV. Each table shows the amount of time spent planning motions ("time"), the amount of memory consumed by exploration data-structures ("mem"), the length of the produced solution ("length"), the amount of time spent in calls to *SelectPossibleTaskPath()* ("pct") and the probability of feasibility for the complete task plan ("prob"). These values are shown for our previously described algorithm [11] (rows marked by "Dij.") and for the new algorithm (rows marked by "MDP"). For each of the reported measurements, the method that performed better is shown in bold face. The results are shown for different parameters Δt (used in Algorithm 1).

C. Discussion of Results

Figure 2 shows a very simple problem where the robot has two options: move directly to the goal (from "ROOT" to "r2") in one single action, or take two additional actions, so that dark areas are avoided. In the process of the computation using Dijkstra's algorithm, the shortest path is the only one considered, the motion plan between "ROOT" and "r2" is trivial to find, and the problem is solved. When using MDPs, the direct solution is computed at first. However, the probability of feasibility for the motion plan connecting "ROOT" to "r2" is computed to be 0.1, which is less than P_C. This makes the reward of moving to "r2" directly very low, so the subsequently proposed sequences of actions extracted using MDPs are via regions "r0" and "r1". The probabilities of feasibility for the corresponding motion plans are higher. Table I shows the values of the collected measurements. Rows indicated by "Dij." correspond to the use of Dijkstra's algorithm in Algorithm 1 and rows indicated by "MDP" correspond to the use of MDPs, as explained in Section III. The probability of feasibility when using MDPs is much larger than when using Dijkstra's algorithm. Additionally, the execution of Dijkstra's algorithm is much faster than value iteration (the "pct" column), and the memory consumption is slightly increased when using MDPs. All these results are expected, just as is the fact that the length of the computed task solutions is higher for MDPs. The differences are small in this particular experiment because of the small scale of the environment.

The "Office1" environment shows the benefit of using MDPs. As shown in Table II, the probability of feasibility when using MDPs is significantly higher, as the algorithm avoids the darker areas and follows the longer task path (via "r7"). The actual value for the probability of feasibility is still low even for the experiments using MDPs. The reason for these low values is the non-physical model of uncertainty

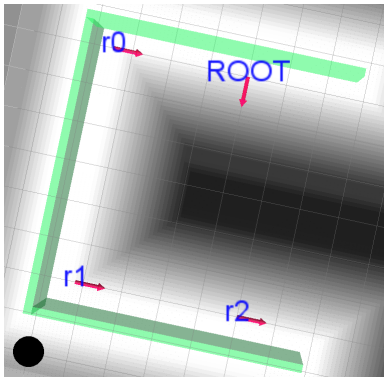

Fig. 2. Motivating example for considering uncertainty. The obvious shorter path is to move from "ROOT" to "r2", crossing the dark area. Considering uncertainty produces the longer solution, via the "r0" and "r1" regions, thus avoiding dark areas.

Δt (s)		time (s)	mem (MB)	pct (ms)	length	prob
0.10	Dij.	**0.04**	**0.00**	**0.2**	**12.66**	0.07
	MDP	0.10	0.01	77.2	13.35	**0.58**
0.50	Dij.	**0.04**	**0.00**	**0.2**	**12.52**	0.07
	MDP	0.10	0.01	84.5	13.38	**0.57**
1.00	Dij.	**0.04**	**0.00**	**0.2**	**12.53**	0.07
	MDP	0.10	0.01	81.0	13.37	**0.56**
2.00	Dij.	**0.04**	**0.00**	**0.2**	**12.72**	0.07
	MDP	0.11	0.01	62.9	13.43	**0.56**

TABLE I

EXPERIMENTAL RESULTS FOR THE MOTIVATING EXAMPLE (FIGURE 2).

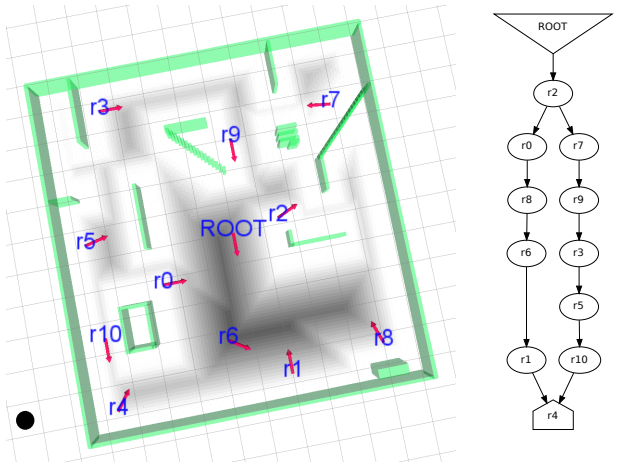

Fig. 3. Left: The "Office1" environment with uncertainty map for localization. Darker areas cause poor localization. Right: The task graph for the task to solve.

Δt (s)		time (s)	mem (MB)	pct (ms)	length	prob
0.10	Dij.	**0.47**	0.04	**1.6**	**51.36**	0.38
	MDP	0.53	0.04	2011.1	56.21	**0.46**
0.50	Dij.	**0.47**	0.04	**2.0**	**49.25**	0.34
	MDP	0.55	0.04	2007.4	56.18	**0.46**
1.00	Dij.	**0.51**	0.04	**1.7**	**47.72**	0.31
	MDP	0.54	0.04	2022.7	57.28	**0.47**
2.00	Dij.	**0.49**	0.04	**1.8**	**46.99**	0.32
	MDP	0.55	0.04	2080.9	55.20	**0.45**

TABLE II

EXPERIMENTAL RESULTS FOR THE "OFFICE1" PROBLEM (FIGURE 3).

that was used and the fact that in this work, motion planners ignored the uncertainty information. Motion planners that incorporate uncertainty do exist but were not used here. The time spent computing motion plans (the "time" column) shows similar values for the compared approaches. However, the time spent proposing task plans is much higher for MDPs (the "pct" column).

The observations made for the "Office1" environment hold true for the "Office1*" environment as well, although the distribution of uncertainty is very different. In fact, the difference in terms of probability of feasibility between the previous variant using Dijkstra's algorithm and the proposed variant using MDPs is much more significant.

"Office2" is a version of "Office1" that includes fewer dark areas. As expected, the probability of feasibility does not really vary when using MDPs rather than Dijkstra's algorithm, as shown in Table IV. More time is spent motion planning due to the difficulty of the individual planning problems. The length of the produced solution paths varies because the approach of selecting paths using MDPs also considers probability of feasibility, as opposed to the approach using Dijkstra's algorithm. As a result, the MDP-based approach has a much lower chance of considering both possible task paths. Since the motion plans that make up the solution for a particular path in the task graph are different, the lengths of the the solutions produced with Dijkstra's algorithm and ones produced with MDPs will differ.

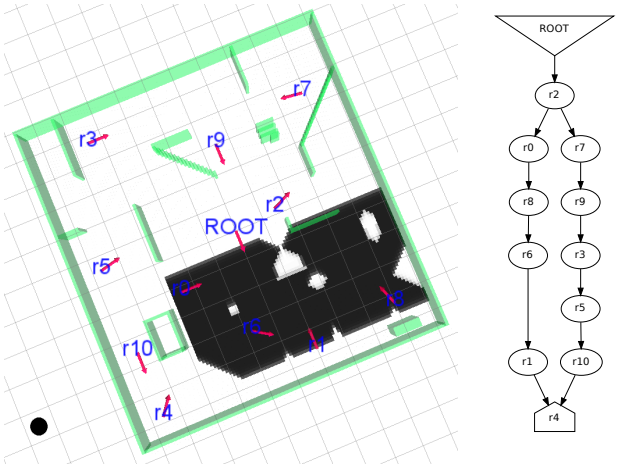

Fig. 4. Left: The "Office1*" environment with an arbitrary distribution of areas that mark low probability of feasibility. Right: The task graph for the task to solve.

Δt (s)		time (s)	mem (MB)	pct (ms)	length	prob
0.10	Dij.	**0.46**	**0.04**	**2.0**	**51.97**	0.50
	MDP	0.58	0.05	1991.9	57.99	**0.74**
0.50	Dij.	**0.52**	0.04	**2.1**	**49.82**	0.37
	MDP	0.56	0.04	2057.8	56.91	**0.75**
1.00	Dij.	**0.53**	0.04	**1.9**	**47.12**	0.26
	MDP	0.62	0.04	2061.9	58.04	**0.75**
2.00	Dij.	**0.53**	0.04	**1.8**	**49.50**	0.37
	MDP	0.58	0.04	2083.4	57.29	**0.74**

TABLE III

EXPERIMENTAL RESULTS FOR THE "OFFICE1*" PROBLEM (FIGURE 4).

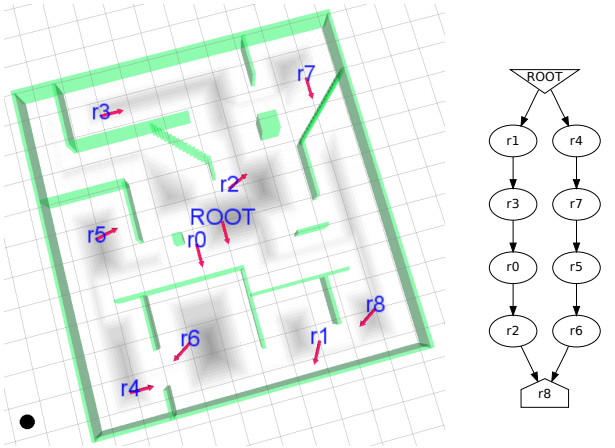

Fig. 5. Left: The "Office2" environment with uncertainty map for localization. Darker areas cause poor localization. Right: The task graph for the task to solve.

Δt (s)		time (s)	mem (MB)	pct (ms)	length	prob
0.10	Dij.	**4.88**	**0.34**	**1.9**	**62.15**	0.39
	MDP	7.72	0.51	1448.0	79.50	**0.40**
0.50	Dij.	**4.06**	**0.27**	**1.0**	**50.92**	**0.42**
	MDP	7.00	0.46	911.8	71.62	0.39
1.00	Dij.	**4.75**	**0.32**	**0.9**	**52.83**	0.43
	MDP	7.01	0.44	931.3	59.32	**0.44**
2.00	Dij.	**5.78**	**0.37**	**1.2**	**56.69**	0.42
	MDP	7.25	0.45	912.2	61.16	**0.43**

TABLE IV

EXPERIMENTAL RESULTS FOR THE "OFFICE2" PROBLEM (FIGURE 5).

V. DISCUSSION

The use of TMMs and MDPs as shown in this work provides a simple approach for considering uncertainty at the task planning and the motion planning levels. Our experiments indicate that the probability of feasibility of the computed solutions to the STAMP problem can be significantly increased. Using motion planners that account for uncertainty as shown in previous work can further increase the probability of feasibility of solutions when solving the STAMP problem. Extensions to the proposed method so that cycles in TMMs can be considered would make our approach more general. Furthermore, different methods of constructing MDPs from TMMs and different approaches for computing MDP policies could be evaluated.

ACKNOWLEDGEMENTS

Work on this paper by I. Şucan and L. Kavraki was supported in part by NSF IIS 0713623, NSF DUE 0920721, NSF CCF 1018798, the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-09-1-0383, a Sloan Fellowship to LK and Rice University funds. The experimental part of this work was supported in part by the Shared University Grid at Rice funded by NSF under Grant EIA-0216467, and a partnership between Rice University, Sun Microsystems, and Sigma Solutions, Inc.

REFERENCES

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

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[3] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *International Journal of Robotics Research*, vol. 17, pp. 315–337, 1998.

[4] C. L. Nielsen and L. E. Kavraki, "A two level Fuzzy PRM for manipulation planning," in *International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 2000, pp. 1716–1722.

[5] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, pp. 729–746, 2004.

[6] K. Hauser and J.-C. Latombe, "Integrating task and PRM motion plannning," in *International Conference on Automated Planning and Scheduling*, 2009, workshop on Bridging the Gap between Task and Motion Planning.

[7] S. Cambon, R. Alami, and F. Gavrot, "A hybrid approach to intricate motion, manipulation and task planning," *International Journal of Robotics Research*, vol. 28, pp. 104–126, 2009.

[8] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation," in *International Conference on Automated Planning and Scheduling*, 2010, pp. 254–258.

[9] L. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[10] I. A. Şucan and L. E. Kavraki, "Mobile manipulation: Encoding motion planning options using task motion multigraphs," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 5492–5498.

[11] ——, "On the advantages of task motion multigraphs for efficient mobile manipulation," in *International Conference on Intelligent Robots and Systems*, 2011, pp. 4621–4626.

[12] I. A. Şucan, "Task and motion planning for mobile manipulators," Ph.D. dissertation, Rice University, 2011.

[13] P. E. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006, pp. 1261–1267.

[14] Y. Huang and K. Gupta, "Collision-probability constrained PRM for a manipulator with base pose uncertainty," in *International Conference on Intelligent Robots and Systems*, St. Louis, Missouri, October 2009, pp. 1426–1432.

[15] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, W. Burgard, O. Brock, and C. Stachniss, Eds. Atlanta, Georgia: MIT Press, June 2007.

[16] S. Chakravorty and S. Kumar, "Generalized sampling based motion planners with application to nonholonomic systems," in *IEEE International Conference on Systems, Man and Cybernetics*, October 2009, pp. 4077–4082.

[17] J. van den Berg, S. Patil, R. Alterovitz, P. Abbeel, and K. Goldberg, "LQG-based planning, sensing, and control of steerable needles," in *International Workshop on the Algorithmic Foundations of Robotics*, Singapore, December 2010.

[18] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," *Journal of Artificial Intelligence Research*, vol. 40, pp. 523–570, 2011.

[19] M. L. Puterman, *Markov Decision Processes*, 2nd ed. Wiley, 2005.

[20] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[21] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance," *International Journal of Robotics Research*, vol. 28, pp. 1448–1465, 2009.

[22] L. P. Kaelbling and T. Lozano-Pérez, "Planning in the know: Hierarchical belief-space task and motion planning," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, Mobile Manipulation Workshop.

[23] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, Zurich, June 2008.

[24] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, San Francisco, April 2000, pp. 995–1001.

[25] "The Open Motion Planning Library." [Online]. Available: http://ompl.kavrakilab.org