Task Planning with Continuous Actions and Nondeterministic Motion Planning Queries

Kris Hauser
School of Informatics and Computing
Indiana University
Bloomington, IN 47405

Abstract

We present a new probabilistic tree-of-roadmaps (PTR) planner that integrates discrete task planning and continuous motion planning using a probabilistic forward search. PTR is explicitly designed with two characteristics in mind: 1) it allows tasks to take continuously variable parameters as input, such as the location of contact points for a robot manipulator executing an object pick-up task; 2) it accounts for variability in the feasibility and difficulty of motion planning for each task. To do so, it uses a sample-based tree planner to explore tasks, and uses a probabilistic roadmap planner to explore the feasible space defined by each task. We prove that PTR is probabilistically complete under relatively weak conditions with a convergence rate that improves with the abundance of actions that admit easy motion planning queries. Because these conditions are not explicitly dependent on dimensionality, PTR scales well to high dimensional problems. We demonstrate the application of PTR to a manipulation task on the Honda ASIMO humanoid robot.

1. Introduction

Autonomous robots that perform multi-handed or multi-fingered manipulation must be able to break complex tasks (say, assembling furniture or preparing a meal) into discrete primitive actions that are realized using low-level continuous motions. Although symbolic task planners are adept at breaking complex tasks into primitive actions, task-and-motion planning requires solving expensive motion planning queries as a precondition of taking an action. Although fast geometric tests can prune out obviously infeasible actions (such as attempting to grasp an object far out of reach), feasibility in the general case cannot be easily inferred (see Figure 1). As a result, a seemingly sensible action at a high symbolic level can turn out to be infeasible upon performing a motion planning query

To plan motions in high dimensional configuration spaces under complex geometric constraints, the only practical solution is to use sample-based motion planners, e.g., Probabilistic Roadmaps (PRMs). We consider task-and-motion planning problems that contain a variety of easy, hard, and

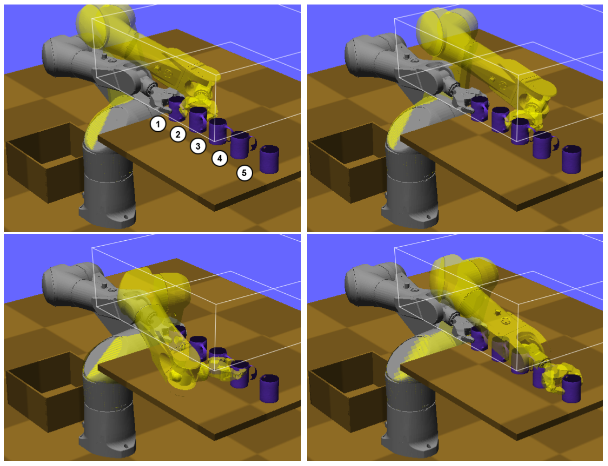

Figure 1: The existence of feasible motions depends on subtle geometric conditions that are expensive to test. Consider a manipulator reaching for cups in a cluttered environment (the upper obstacle is drawn in wireframe for better visibility). Without moving cup 1, there is no collision free path to grasp cup 2; without moving cup 2, there is no collision free path to grasp cup 3, and so on. Examples of infeasible configurations are drawn in yellow.

infeasible PRM planning queries. Such problems are commonplace in manipulation in cluttered household settings, dexterous manipulation, and legged locomotion on rough terrain. Because PRMs cannot answer that a query is infeasible in finite time, they are usually given a time limit after which failure is declared. If a query fails, it is difficult to tell whether it was merely hard or genuinely infeasible. A time limit that is set too low causes a task-and-motion planner to risk failing on hard queries for crucial actions. On the other hand, this time limit should not be too high, or else time will be wasted on infeasible queries.

We also consider the general setting where the planner can choose from a continuous infinity of actions, such as the contact points of a grasp, or the placement of an object moved aside. A planner may only attempt motion planning queries for a finite subset of these actions. For a difficult problem where few actions are feasible, a huge number of attempts may be needed before the planner makes progress toward a goal. But in easy problems the planner can make rapid progress by choosing actions almost arbitrarily. A balanced approach is desirable for most realistic problems (e.g., household manipulation); the planner should not expect an adversarial environment, but it should still be able to solve occasional hard problems.

To address these issues we present a new Probabilistic Tree-of-Roadmaps (PTR) task-and-motion planner that is designed to 1) handle continuously parameterized actions by growing a search tree that distributes states sparsely, and 2) plan quickly in easy problems but adaptively increase the time limit when a problem proves harder than expected. We prove that PTR is reliable in that it returns a sequence of feasible actions, and motions to achieve those actions, with probability approaching 1 as more time is spent planning. To motivate the development of programming support for task-and-motion planning research, we also present a hypothetical general-purpose problem definition language that supports continuous actions and explicit motion planning queries.

2. Related Work

2.1 Integrating Task, Contact, and Motion Planning

The importance of integrating planning at discrete and continuous levels has long been recognized when studying systems that make and break contact, such as manipulation (Alami, Laumond, and Siméon, 1995; Ferbach and Barraquand, 1997; Koga and Latombe, 1994; Sahbani, Cortés, and Siméon, 2002; Xu, Koo, and Li, 2005), industrial assembly (Wilson and Latombe, 1995), and legged locomotion (Boissonnat et al., 1995; Bretl et al., 2004; Chestnutt et al., 2003). This type of planning can be considered as a subset of task-and-motion planning as long as the notion of "subtask" is construed to subsume the act of achieving a certain contact state.

Some low-dimensional or geometrically tractable problems have been solved by searching a graph whose vertices enumerate the connected components of each motion planning subspace (Alami, Laumond, and Siméon, 1995; Boissonnat et al., 1995; Wilson and Latombe, 1995), but these approaches do not scale well to geometrically complex, high-dimensional problems. Other work uses sample-based planners that randomly sample control inputs and discrete actions (e.g., pp. 270–274 of LaValle (2006), Xu, Koo, and Li (2005)). However, in many applications certain (or most) actions can only be applied at particular states or submanifolds of state space, so this approach may fail to ever find a state that satisfy the preconditions of necessary actions.

A general forward search approach was introduced in a manipulation planning application (Alami, Laumond, and Siméon, 1995), where the vertices of the search graph identify states, and arcs identify either transit or transfer motions that connect states. The planner searches while explicitly constructing motions between neighboring states. This approach scales well to high-dimensional problems when sample-based techniques are used for motion planning queries (Bretl et al., 2004; Nielsen and Kavraki, 2000).

For problems with continuously parameterized actions, a search-based approach requires a discretization step must be performed. The choice of discretization is critical: too coarse, and the planner cannot find a path; too fine, and planning will be extremely slow. A major contribution of this work is to study the conditions necessary for the planner to find a path with a small number of sampled actions.

2.2 Completeness and Composition

In earlier work we devised a probabilistically complete general-purpose sample-based planner for problems with a finite number of actions (Hauser and Latombe, 2009). More similar to the present work, van den Berg et. al. proved probabilistic completeness of a planner for navigation among movable obstacles under a certain path clearance assumption (van den Berg et al., 2008). This planner discretizes a continuous action space by growing a tree at uniformly at random. They also assume that the connected components of the feasible space associated with an action can be computed exactly, which restricts the applicability of their work to low-dimensional problems. PTR also grows a tree at random, but 1) does not require complete motion planning queries, 2) uses weaker and more general assumptions, and 3) achieves a better exponential convergence bound using a density-weighted sampling strategy.

A common hierarchical approach to task-and-motion planning by first plans a discrete task-level path and then performs continuous motion planning to achieve those tasks (Casal, 2001; Casal and Yim, 1999). A common scheme in object manipulation takes the form of planning an object trajectory first, and then planning motions to make or break contact with the object (Cherif and Gupta, 1998; Koga and Latombe, 1994; Yashima and Yamaguchi, 2002). The motivation is to reduce the number of motion plans that must be constructed, but ultimately this approach is unreliable because the feasibility of lower-level plans cannot be guaranteed at the abstraction level of tasks. Our recent work we argued that discrete task-level subgoaling is better suited as heuristic information to help guide a lower-level, probabilistically complete motion planner (Hauser and Latombe, 2009). Plaku and Hager (2010) use symbolic knowledge to bias the search of a configuration space planner under symbolic and differential constraints.

2.3 Language Support for Task-and-Motion Planning

The work of (Cambon, Alami, and Gravot, 2009) also addresses unified task, contact, and motion planning in a common STRIPS-like definition language. In this language, symbolic propositions are decoupled from the continuous variables that manipulate them. Instead, our language tightly mixes discrete and continuous state variables in a more familiar imperative programming style. It is unclear at this time whether one language is more expressive than another.

3. A STRIPS-like Language for Integrated Task-And-Motion Planning

This section presents a hypothetical STRIPS-like language that is expressive enough to represent continuous state variables, to accept actions with continuous parameters, and to bind actions to complex subroutines like collision and motion planning queries. We also specify a realistic manipulation planning problem in that language. This language clarifies several of the major challenges for planning with continuous state variables and actions.

3.1 Language Description

A planning problem $\mathcal{P}$ is specified as a list of parameterized actions acting upon a particular state space $\mathcal{X}$. A state x of the world is defined using a collection of named variables, defined either over continuous or discrete domains. Actions and their parameters must be scheduled by the planner in order to move the state such that a desired endgame condition G holds. Actions may also require the planner to solve motion planning queries before they are executed. The planner is given some control over the execution of each query, but the query may fail nondeterministically, run for an undetermined amount of time, and produce unpredictable trajectories.

We define an action $A(u)$ as a parameter-dependent tuple $< P, Q, E, C >$ containing:

- Preconditions P: a list of boolean conditions that must be true of the current state x before an action can be executed. If they are true, we say $A(u)$ is *applicable* to x.
- Queries Q (optional): probabilistic roadmap motion planning queries that must be solved before the action is executed. Queries are written in the form "trajectory = Query(space,start,end)". The "space" variable defines the continuous subspace of $\mathcal{X}$ in which the query is allowed to move, as well as the operational constraints for the query (e.g., obstacle avoidance, differential constraints).
- Effects E: changes to the state of the world once the action is executed, written in the form "Variable=value". We assume that the effects of an action are invariant to the trajectory produced by a plan. Define the function $Appl(A(u), x)$ that applies the effects of $A(u)$ to the state x, and produces the resulting state.
- Commands C: a sequence of commands that should be sent to the robot upon executing this action.

By a *condition*, we mean a deterministic boolean function of one or more variables. These may be mathematical relations (e.g., equalities and inequalities) or subroutines that perform complex geometric and numerical computations (e.g, collision detection, stability testing).

3.2 A Blocks World Manipulation Planning Example

As an example, let us consider the specification of a realistic implementation of the traditional "blocks world" problem that is often used in discrete planning benchmarks. The planner should produce kinematic manipulation plans for a manipulator arm that can pick and place any of n rigid objects in a static environment. This is precisely the same problem considered by (Sahbani, Cortés, and Siméon, 2002). For simplicity, we will not allow objects to be stacked on top of one another.

Pickup(i, q_d, g)
P: GraspedObject= 0, Object$_i$ =Pose(i, q_d, g),
Grasp-Stable?(i, g)
Q: $p =$ Query($\mathcal{F}_{transit,x}$,Robot,q_d)
E: GraspedObject= i, Grasp= g, Robot= q_d
C: Move(p), Close-Gripper

Putdown(i, q_d)
P: GraspedObject= i, Object-Stable?(i,Pose$(i, q_d$,Grasp))
Q: $p =$ Query($\mathcal{F}_{transfer,x}$,Robot,q_d)
E: GraspedObject= 0, Object$_i$ =Pose(i, q_d,Grasp),
Robot= q_d
C: Move(p), Open-Gripper

Figure 2: The action list for a basic manipulation problem.

In this example $\mathcal{X}$ contains the following state variables:

- Robot: continuous robot configuration in $\mathcal{C}$.
- Object$_i$: ungrasped object pose in $SE(3)$, $i = 1, \ldots, n$.
- GraspedObject: the index of the grasped object.
- Grasp: a description of the current grasp, e.g. the gripper width and its transformation relative to the object.

Motion planning queries are needed to generate *transit* (no object grasped) and *transfer* (object grasped) paths for the robot arm. Transit paths occur in the subspace of the robot configuration only, but collision constraints are dependent on the pose of the objects in the current state. Thus we write the collision-free transit subspace as $\mathcal{F}_{transit,x}$. In transfer motions the currently grasped object is rigidly affixed to the robot's gripper, and thus the planner must perform collision detection between the grasped object against the environment and all other objects. We write this transit subspace as $\mathcal{F}_{transfer,x}$.

The problem is defined using two actions, listed in Figure 2, which make use of the following subroutines:

- Object-Stable?(i, p) returns true if object i is stably supported by the environment at pose p.
- Grasp-Stable?(i, g) returns true if object i can be stably grasped using grasp g (e.g., with a force closure grasp).
- Pose(i, q, g) produces the pose of object i if grasped by grasp g at configuration q.

3.3 Challenges of Forward Planning with Continuous States

Although specific task-and-motion planning problems can be solved in a diversity of ways, we consider a general forward search. Forward search grows a tree rooted at the start state x_0 by repeatedly applying actions to states in the tree

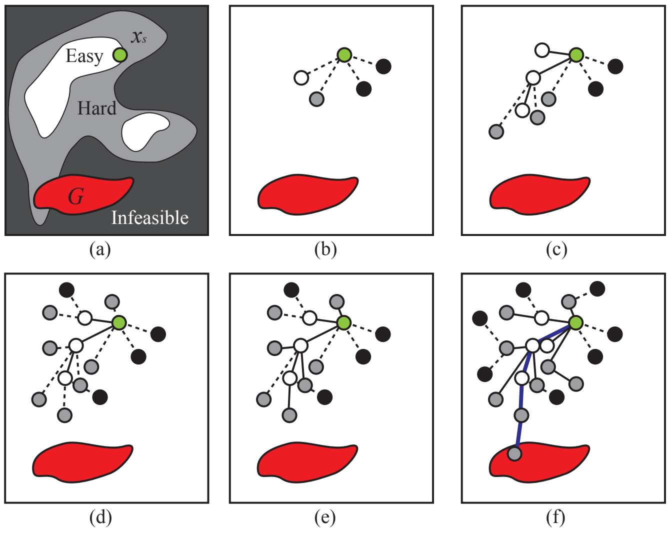

Figure 3: Illustrating the growth of the search tree in (a) an abstract state space with regions of easy, hard, and infeasible motion planning queries (white, grey, and black nodes). Active edges are dotted, Complete edges are solid. (b,c) After a few iterations, the easy portion of space is partially explored. (d) Continued exploration. (e) As the time cutoff for motion queries increases, more hard queries begin to be answered. (f) Reaching the goal.

until the endgame G is reached. Applying a continuously parameterized action $A(u)$ to the state x poses three major challenges:

1. There may be an uncountably infinite number of valid instantiations of u, so the planner must sample a finite number of parameter instantiations.
2. The planner may have zero probability of choosing valid u arbitrarily, because the set of valid parameters may form a lower dimensional subset of the parameter space.
3. There may be no feasible continuous path that achieves the action, and the planner cannot determine this fact without performing a continuous motion planning query.

Challenge 1 is troublesome when a reasonable number of parameter samples is not known a priori; if a problem is difficult, a huge number of samples may be needed, but if a problem is easy, the huge branching factor would overwhelm the planner. Challenge 2 requires some special effort on the planner to choose samples wisely, but it can be handled by numerical techniques described in Section 4.2.Challenge 3 is troublesome when the planner must make sample-based motion planning queries that are infeasible or of varying difficulty. The technical contribution of PTR is to address all three challenges simultaneously.

4. Probabilistic Tree-of-Roadmaps Planner

Given a planning problem $\mathcal{P}$, the Probabilistic Tree-of-Roadmaps (PTR) algorithm performs forward search to build a tree T of feasible states, but uses 1) a density-weighting strategy to achieve an even coverage of state space, and 2) incrementally increases the time spent on individual motion planning queries so that hard queries will eventually be solved. PTR builds a tree T with the following properties:

- An edge $x \rightarrow x'$ is constructed by selecting an action $A(u)$ that is applicable to x, and determining the resulting state x'. We attach the action $A(u)$ to the edge, as well as an (initially) incomplete motion planning query.
- The status of an edge is marked as Complete or Active. Complete edges have finished a motion planning query, while motion planning queries are currently being processed at Active edges.
- If T contains a path of Complete edges leading to a goal state, then the concatenation of trajectories along this path is a solution to the planning problem.

Pseudocode for PTR is as follows:

```
PTR($x_0, G, N$)
1. $T$ = Init-Tree($x_0$)
2. Repeat for $i = 1, 2, \ldots, N$:
3.     Extend-Tree($T$)
4.     For all Active edges $e = x \rightarrow x'$:
5.         Plan-More($e, t(i, e)$)
6.         If planning is successful:
7.             If $x' \in G$ return Success
8.             Mark $e$ as Complete
9. Return Failure
```

The subroutine Plan-More(e, t) performs more computation on the motion planning query associated with the edge e, and halts when the total time spent on this query (accumulated from the moment of edge creation) exceeds t. We assume that motion planning queries can be paused and resumed at any time. Furthermore we assume that the queries use PRMs or any other probabilistically complete technique, so that the success of finding a feasible trajectory approaches 1 as more time is spent. So, the time PTR spends on each motion planning query should increase asymptotically; in particular we t as a linearly increasing function of the iteration count i. We shall see in Section 5that this choice makes PTR asymptotically complete.

4.1 Expanding the Tree With Uniform Density

The Extend-Tree subroutine attempts to find a connected state in T and an applicable action that extends the tree to a new state uniformly from its reachable set. This strategy was inspired by the EST motion planner (Hsu et al., Mar 2002), and is chosen such that the asymptotic distribution of states is uniform across $\mathcal{X}$.

An approximation to an "ideal" Extend-Tree uses a kernel density estimation and rejection sampling technique as follows:

```
Extend-Tree($T$)
1. $C = \{\}$
2. For all connected nodes $x$ in $T$:
3.     $x_1, \ldots, x_k$ =Sample-k-Successors($x$)
4.     Add the candidate edge $x \to x_i$ to $C$ for $i = 1, \ldots, k$
5. Pick an edge $x \to x'$ from $C$ proportionally to $w(x')$
6. Add $x \to x'$ to $T$, with the status Active.
```

The parameter k is chosen so that samples in C achieves a sufficient approximation to the reachable set. Here the weight $w(x)$ is defined as $1/\pi(x)$, where $\pi(x)$ is defined as the density of states in $T \cup C$ around x as measured by, say, a Gaussian kernel. The time complexity of a single evaluation of $w(x)$ is $O(k|T|)$, but a grid-based hashing approximation can reduce this to $O(1)$. A naive implementation that recomputes all candidate successors for every iteration of PTR is a rather expensive $O(k|T|)$, but caching can reduce this to a modest $O(k)$.

The uniform density objective is meant to encourage the tree to achieve a constant rate of exploration of the reachable portion of $\mathcal{X}$ at early stages of planning, and then to achieve a uniform refinement asymptotically. There are other ways of approximating the same objective, for example, an RRT-like strategy picks a state at random from $\mathcal{X}$ and expands the tree toward that state (LaValle and Kuffner, 1999). Uniformity is convenient for theoretical analysis, but nonuniformity may be preferred in the presence of prior information that could help guide exploration quickly to more promising areas of a large state space. Methods for improving coverage will be discussed in Section 4.3

4.2 Choosing Applicable Action Parameters

When an action's parameters are continuous, an extremely small subset may satisfy its preconditions. In fact, the subset will have zero volume if the preconditions define one or more functional equalities. Consider a Pickup action. The subset of parameters q_d and g that define a configuration that actually grasps the object with grasp g is lower-dimensional subset of all possible choices of q_d and g. So, the planner may not pick them arbitrarily (e.g., uniformly at random), but must rather choose from the subset directly.

To sample the space of valid parameters u for $A(u)$, the planner must explicitly consider all functional equalities encoded in the preconditions of $A(u)$. This requires preconditions to expose the underlying functional constraints. For example, Grasp-Stable? requires that the robot gripper and object geometry are touching at two or more points. Such a condition can be represented as a boolean formula of equalities and inequalities whose specific form depends on the underlying geometry (e.g, grasping a ball with center c and radius r could require "gripper-width $= 2r$" and "gripper-center $= c$"). A valid choice of u can be chosen by sampling a u at random, and then solving for the simultaneous equalities using numerical root-finding techniques (e.g., Newton-Raphson methods). Algebraic techniques could be practical if all equalities could be represented as low-order polynomials, but are prohibitively expensive when many high-degree polynomials are involved. General techniques for picking u are often slower than special-purpose subroutines (e.g., analytic inverse kinematics solvers).

A further difficulty is that if actions are not defined carefully, a parameter choice may cause inconsistencies several actions in the future. For example, let $A_1(u)$ set $x_1 = u, ...,$ $A_n(u)$ set $x_n = u$, and the preconditions of action B require $(x_1, ..., x_n)$ to be set to a specific value. Then, n levels of lookahead are needed to have any chance of choosing a consistent value for x_1. For the purpose of this paper we will assume that actions are defined so that no lookahead is necessary. If this does not hold the planner may need to consider aggregate actions and/or better subgoal regression techniques. We comment on some of these possible extensions in Section 6.

4.3 Tuning the Expansion and Motion Planning Strategies

We will show that the basic PTR technique has asymptotic reliability guarantees, but its practical performance can be tuned. To better exploit problem structure, ExpandTree may be guided into picking better actions using subgoaling heuristics that are constructed without taking continuous variables into account (Plaku and Hager, 2010). Such heuristics can have the effect of tuning the exploration weight $w(x)$. The rejection rate of ExpandTree could also be improved by sampling large sequences of actions using nonlinear optimization techniques to enforce subgoal consistency.

To help avoid infeasible motion planning queries, computer algebra techniques can prove infeasibility by manipulating the algebraic relations encoded in the action's preconditions. This approach was applied to a climbing robot in (Bretl et al., 2004). Another possible approach is to learn a model of feasible queries in precomputation and then to use the model to bias the growth of the tree (Hauser, Bretl, and Latombe, 2005). Biases in the exploring motion planning queries can be implemented by tuning the query time limit $t(i, e)$.

A straightforward improvement is that several actions originating from a given state may require motion planning in the same configuration space, only with different destinations. Using a separate motion planning query for each action duplicates effort, so a great deal of time could be saved by using a single probabilistic roadmap to answer all queries.

On the other hand, if roadmaps cannot be reused for multiple queries, then PTR should use tree-based motion planners that are better suited for single-query operation than standard PRMs. Such motion planners are also more convenient for planning under differential constraints. PTR's theoretical guarantees should hold under any probabilistically complete sample-based motion planner.

5. Analysis of Asymptotic Completeness

This section proves that PTR is probabilistically complete for task-and-motion planning problems under relatively weak assumptions. Specifically, we show that the probability that PTR fails to find a feasible path, if one exists, decreases nearly exponentially in running time. This bound implies that expected running time and variance in running time are finite. Furthermore, the convergence rate is not explicitly dependent on the dimensionality of continuous planning problems, and depends instead on the abundance of action sequences that yield "easy" motion planning queries.

The strategy of the proof is to first define a certain state space *expansiveness* property that characterizes the reachability of the goal under a constant time cutoff for motion planning queries. The next step proves that if this cutoff is sufficiently high, then PTR is probabilistically complete. In most problems, this is a strong assumption because the level of "sufficiently high" is unknown. An increasing threshold strategy achieves probabilistic completeness under weaker assumptions, because it will eventually reach such a threshold as long as one exists. This latter strategy expends a larger computational cost per iteration, leading to a convergence rate that is slightly subexponential in running time.

5.1 Conditional Completeness of Fixed-Cutoff Motion Planning

Let us define Cutoff-k-PTR as the implementation of PTR with a constant cutoff $t(e,i) = k$. To prove that Cutoff-k-PTR is probabilistically complete with sufficiently high k, we need the state space to satisfy a notion of *expansiveness* which is analogous to the concept of configuration space expansiveness defined in (Hsu et al., Mar 2002). Broadly speaking, a state space is expansive if the Cutoff-k-PTR has a significant chance of picking an action that significantly expands the set of states reachable from the tree.

We first require some preliminary definitions. Let $L(x, A(u))$ denote the motion planning query associated with applying $A(u)$ to x. Define $L(x, A(u))$ as (k,p)-*feasible* if the query succeeds with probability at least p within time cutoff k. Let a state x' be (k,p)-*reachable* from x if x' is the result of applying some applicable $A(u)$ to x and $L(x, A(u))$ is (k,p)-feasible. Define the *locally hypothetically reachable* set $\mathcal{V}_L(X)$ of $X \subset \mathcal{X}$ as the set of states $x' \notin X$ that result from applying some action $A(u)$ to a state $x \in X$. Define the *locally reachable* set $\mathcal{V}_{L,k,p}(X)$ of $X \subset \mathcal{X}$ as the set of states $x' \notin X$ such that x' is reachable by a single (k,p)-feasible motion planning queries from some state $x \in X$. Similarly define the *globally reachable* set $\mathcal{V}_{k,p}(X)$ as the set of states reachable by any finite number of (k,p)-feasible queries. We are now ready to define expansiveness and state our theorem.

Definition 1. *A problem $\mathcal{P}$ is* (k,p,α,β)-expansive *if there exists constants k, p, α, and β such that for* any *subset A of the reachable space, at least an α fraction of $\mathcal{V}_{k,p}(A)$ is (k,p)-reachable from a β fraction of A.*

Theorem 1. *If there exists positive constants k, p, α, and β such that:*

- *$\mathcal{P}$ is (k,p,α,β)-expansive,*
- *The volume $\gamma = \mu(E \cap \mathcal{V}_{k,p}(x_s))$ of the reachable portion of the endgame region satisfies $\gamma > 0$,*
- *Each iteration of PTR expands the tree by drawing uniformly from $\mathcal{V}_L(X)$,*

then after n iterations, the probability that Cutoff-k-PTR fails to find a path is no more than $c\exp(-dn)$, for some positive constants c and d that are elementary functions of p, α, β, and γ.

The proof is a relatively straightforward adaptation of the proof of the probabilistic completeness of the EST motion planner given in (Hsu et al., Mar 2002). The major difference is that EST relies on a deterministic local planner, and here we must treat motion planning queries as stochastic local planners. The derivation is given in the appendix.

It is important to note that α, β, and p are *properties* dependent on k, rather than parameters. For a given k, there exists a family of bounds that trade-off between increasing p and reducing α, β, and γ, so the running time of Cutoff-k-PTR will be bounded by the best of those exponentially convergent bounds. As k increases, the planner will be able to solve more queries for a given p, increasing the reachable portion of the endgame. If the cutoff k is not high enough for the difficulty of the problem, there may be no values of α, β, and γ such that expansiveness condition holds.

5.2 Completeness with an Increasing Cutoff

Cutoff-k-PTR suffers from the difficulty of determining a cutoff k for which the problem is (k,α,β,p)-expansive with $\alpha,\beta,p > 0$. This may be remedied by increasing the cutoff incrementally. A simple proof, which we omit due to space constraints, demonstrates that the following theorem holds:

Theorem 2. *If:*

- *There exists some (unknown) finite k such that Theorem 1 holds,*
- *PTR grows the time cutoff monotonically without bound at each iteration, that is $t(e,i+1) > t(e,i)$ and $\lim_{i\to\infty} t(e,i) = \infty$.*

then after n iterations, the probability that PTR fails to find a path is no more than $c\exp(-d(n-k))$, for the same constants c and d that are defined in Theorem 1.

This strategy allows weaker assumptions than 1 at a greater computational cost per iteration. For example, if $t(e,i) = O(i)$, the running time increases with the square of the number of iterations n, and so the convergence rate is exponentially decreasing in the square root of the running time T. In general, if $t(e,i) = O(i^{\epsilon/(1-\epsilon)})$, then the the probability that PTR fails is bounded by a subexponential function $c\exp(-dT^{1-\epsilon})$ (where c and d are nontrivial but well-behaved functions of ϵ).

6. Discussion and Future Work

Preliminary experimental results suggest that our work may provide a sound theoretical foundation for practical task-and-motion planning across a wide variety of problems. In prior work we applied a variant of PTR to enable the Honda ASIMO robot to push an object on a horizontal table (Figure 4). The problem was defined with seven continuously-parameterized actions and has a state space with 16 continuous dimensions. For a problems that require pushing the object around the edges of an obstacle-free table, the planner

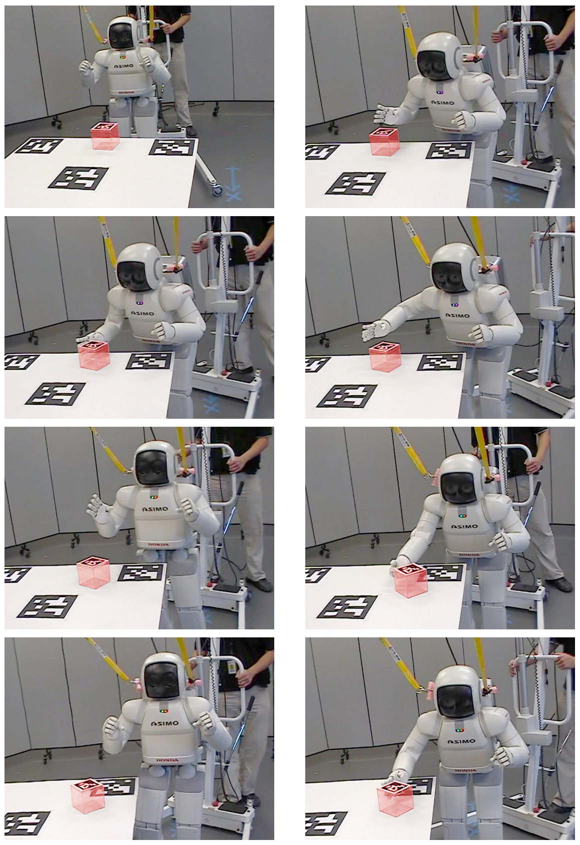

Figure 4: Pushing a block with the Asimo robot, switching between walking, reaching, and pushing modes. The block is shaded red for contrast.

produced solutions approximately 1 minute. With complex obstacles, the planner took a few minutes.

Although PTR is asymptotically complete, it does not follow the major strategy used by discrete task planners to make planning fast in large state spaces: exploit the structure encoded in the language. One problem is that subgoal regression in continuous spaces is much harder than in discrete spaces because subgoals can become intractably complex subsets. This may be addressed by embedding computer algebra systems that derive approximate subgoal representations on the fly. Hierarchical planning techniques could be implemented in a PTR framework without much difficulty, and may achieve faster coverage of large spaces. Future research in task-and-motion planning should seek to define common benchmark problems on which various planning techniques can be compared.

Appendix

The appendix will prove Theorem 1 that states that Cutoff-k-PTR is probabilistically complete. First we will define expansiveness more rigorously. In this appendix we fix a constant k and $p > 0$.

Definition 2. *For any set $X \in \mathcal{S}$ and constant $\beta > 0$, let (β, k, p)-Lookout(X) be the subset*

$$\{x \in X | \mu(\mathcal{V}_{L,k,p}(x) \setminus X) \geq \beta \mu(\mathcal{V}_{k,p}(X))\}. \tag{1}$$

Definition 3. *$\mathcal{P}$ is (k,p,α,β)-expansive if for any $S \subseteq \mathcal{X}$, there exist positive constants α, β such that $\mu((\beta,k,p)$-Lookout$(S)) \geq \alpha \mu(S)$.*

Roughly speaking, a lookout is the subset of X that can reach an significant fraction of the locally reachable set of X. Expansiveness requires that each set in the state space have a significant lookout.

The theorem will also make use of the elementary fact that if events Y_1 and Y_2 imply X, then

$$\begin{aligned} Pr(\neg X) \leq & Pr(\neg Y_1) + Pr(\neg Y_2 | Y_1) Pr(Y_1) \\ \leq & Pr(\neg Y_1) + Pr(\neg Y_2 | Y_1). \end{aligned} \tag{2}$$

So, if $Pr(\neg Y_1) \leq c_1 e^{-d_1 n}$ and $Pr(\neg Y_2 | Y_1) \leq c_2 e^{-d_2 n}$, then

$$Pr(\neg X) \leq (c_1 + c_2) e^{-n \min(d_1, d_2)}. \tag{3}$$

Claim 1. *Theorem 1 holds.*

Proof. The proof is a relatively straightforward adaptation of the argument in Hsu et al. (Mar 2002). Consider the locally reachable set of the search tree at step n, $\mathcal{V}_{L,k,p}(\mathcal{T}_k)$. We will examine how the volume of the lookout set $\mu_n \equiv \mu(\mathcal{V}_{L,k,p}(\mathcal{T}_k))$ converges toward the volume of the globally reachable set, $\mu(\mathcal{V}_{k,p}(x_s))$, as a function of n. For simplicity, assume the measure is normalized such that $\mu(\mathcal{V}_{k,p}(x_s)) = 1$

By assumption, the destination y of a random extension is drawn randomly from $\mathcal{V}_L(\mathcal{T}_n)$. By the expansiveness condition, there is at least a α probability of y lying in the lookout set (β, k, p)-Lookout$(\mathcal{V}_{L,k,p}(\mathcal{T}_n))$. There is also at least a p probability that the motion planning query succeeds.

If y lies in the lookout set and the query succeeds, then the extended tree $\mathcal{T}_{n+1}$ has a significantly larger locally reachable set. Specifically,

$$\mathcal{V}_{L,k,p}(\mathcal{T}_{n+1}) = \mathcal{V}_{L,k,p}(\mathcal{T}_n) \cup \mathcal{V}_{L,k,p}(y), \tag{4}$$

so,

$$\begin{aligned} \mu_{n+1} &= \mu_n + \mu(\mathcal{V}_{L,k,p}(x') \setminus \mathcal{V}_{L,k,p}(\mathcal{T}_n)) \\ &\geq \mu_n + \beta \mu(\mathcal{V}_{k,p}(\mathcal{V}_{L,k,p}(\mathcal{T}_n))) \end{aligned} \tag{5}$$

Since $\mathcal{V}_{k,p}(\mathcal{V}_{L,k,p}(\mathcal{T}_n)) = \mathcal{V}_{k,p}(x_s) \setminus \mathcal{V}_{L,k,p}(\mathcal{T}_n)$, we have

$$\mu_{n+1} \geq \mu_n + \beta(1 - \mu_n). \tag{6}$$

If, on the other hand, y does not lie in the lookout set, we trivially have $\mu_{n+1} \geq \mu_n$.

If after n iterations of Random-MMP, m expansions have lied in the lookout set and have yielded successful motion planning queries, then

$$\mu_n \geq \sum_{i=0}^{m-1} (1-\beta)^i \beta = 1 - (1-\beta)^m \tag{7}$$

Now consider the event that m is high enough such that the locally reachable set of $\mathcal{T}_n$ overlaps the endgame region by volume $\gamma/2$ (or some other significant fraction). Using some algebraic manipulation of (7), we see that this is guaranteed to occur when $m \geq \ln(\gamma/2)/\ln(1-\beta)$. Let $m^\star = \lceil \ln(\gamma/2)/\ln(1-\beta) \rceil$ be the number of expansions needed to satisfy the condition. Once $m^\star$ of these expansions occur, then each subsequent iteration of Cutoff-k-PTR samples an expansion into the endgame region with probability at least $\gamma/2$.

Finally, consider the event that Random-MMP succeeds after n iterations. Success is implied if (A) the first $n/2$ iterations of Random-MMP draws at least $m^\star$ lookout expansions, and then (B) the last $n/2$ iterations successfully samples an expansion into the endgame region.

Because each lookout expansion occurs with probability at least αp, Hoeffding's inequality Hoeffding (1996) shows that the probability that A is false is no more than

$$Pr(\neg A) \leq \exp(-\frac{(\alpha p n - 2m^\star)^2}{n}). \tag{8}$$

Once A is true, then the probability of reaching the endgame region is a coin flip with probability at least $\gamma p/2$. So, we have

$$Pr(\neg B|A) \leq (1-\gamma p/2)^{n/2} \leq e^{-n\gamma p/4} \tag{9}$$

where the latter inequality uses the fact that $(1-x)^y \leq e^{-xy}$ for $x \in [0,1]$ and $y > 0$. Substituting these two equations in (3) and performing a few substitutions, we have the probability that Random-MMP fails in n iteration is no more than

$$\begin{gathered} Pr(\neg A) + Pr(\neg B|A) \leq \\ (\exp(4\alpha p m^\star - m^{\star 2}) + 1)\exp(-n\min(\alpha^2 p^2, \gamma p/4)). \end{gathered} \tag{10}$$

This equation gives exponentially decreasing bound $c\exp(-nd)$ with constants $c = \exp(4\alpha p m^\star - m^{\star 2}) + 1$ and $d = \min(\alpha^2 p^2, \gamma p/4)$. □

References

Alami, R.; Laumond, J.-P.; and Siméon, T. 1995. Two manipulation planning algorithms. In Goldberg, K.; Halperin, D.; Latombe, J.-C.; and Wilson, R., eds., *Alg. Found. Rob.* Wellesley, MA: A K Peters. 109–125.

Boissonnat, J.-D.; Devillers, O.; Donati, L.; and Preparata, F. 1995. Motion planning of legged robots: The spider robot problem. *Int. J. of Computational Geometry and Applications* 5(1-2):3–20.

Bretl, T.; Lall, S.; Latombe, J.-C.; and Rock, S. 2004. Multistep motion planning for free-climbing robots. In *WAFR*.

Cambon, S.; Alami, R.; and Gravot, F. 2009. A hybrid approach to intricate motion, manipulation and task planning. *Intl. J. of Robotics Research* 28(104).

Casal, A., and Yim, M. 1999. Self-reconfiguration planning for a class of modular robots. In *SPIE II*, 246–257.

Casal, A. 2001. *Reconfiguration Planning for Modular Self-Reconfigurable Robots*. Ph.D. Dissertation, Stanford University, Stanford, CA.

Cherif, M., and Gupta, K. 1998. Planning for in-hand dextrous manipulation. In Agarwal, P.; Kavraki, L.; and Mason, M., eds., *Robotics: The Algorithmic Perspective*. AK Peters, Ltd. 103–117.

Chestnutt, J.; Kuffner, J.; Nishiwaki, K.; and Kagami, S. 2003. Planning biped navigation strategies in complex environments. In *IEEE Int. Conf. Hum. Rob.*

Ferbach, P., and Barraquand, J. 1997. A method of progressive constraints for manipulation planning. *IEEE Trans. Robot. and Autom.* 13(4):473–485.

Hauser, K., and Latombe, J.-C. 2009. Multi-modal planning in non-expansive spaces. In *IEEE J. of Robotics Research*.

Hauser, K.; Bretl, T.; and Latombe, J.-C. 2005. Learning-assisted multi-step planning. In *IEEE Int. Conf. Rob. Aut.*

Hoeffding, W. 1996. Probability inequalities for sums of bounded random variables. *J. of the American Statistical Association* 58(301):13–30.

Hsu, D.; Kindel, R.; Latombe, J.-C.; and Rock, S. Mar 2002. Kinodynamic motion planning amidst moving obstacles. *Int. J. Rob. Res.* 21(3):233–255.

Koga, Y., and Latombe, J.-C. 1994. On multi-arm manipulation planning. In *IEEE Int. Conf. Rob. Aut.*, 945–952.

LaValle, S., and Kuffner, J. 1999. Randomized kinodynamic planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, 473–479.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University Press.

Nielsen, C. L., and Kavraki, L. E. 2000. A two level fuzzy prm for manipulation planning. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, 1716–1721.

Plaku, E., and Hager, G. 2010. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *In IEEE International Conference on Robotics and Automation*.

Sahbani, A.; Cortés, J.; and Siméon, T. 2002. A probabilistic algorithm for manipulation planning under continuous grasps and placements. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, 1560–1565.

van den Berg, J.; Stilman, M.; Kuffner, J.; Lin, M.; and Manocha, D. 2008. Path planning among movable obstacles: a probabilistically complete approach. In *Workshop on the Algorithmic Foundations of Robotics*.

Wilson, R., and Latombe, J. 1995. Geometric reasoning about mechanical assembly. *Artificial Intelligence* 71(2):371–396.

Xu, J.; Koo, T. J.; and Li, Z. 2005. Finger gaits planning for multifingered manipulation. In *IEEE Conf. on Intel. Rob. and Sys.*

Yashima, M., and Yamaguchi, H. 2002. Dynamic motion planning whole arm grasp systems based on switching contact modes. In *IEEE Int. Conf. Rob. Aut.*