Sampling-Based Algorithm for Testing and Validating Robot Controllers

Jongwoo Kim [a], Joel M. Esposito [b],
and Vijay Kumar [a]
[a] University of Pennsylvania, Philadelphia, PA
[b] US Naval Academy, Annapolis, MD
jwk@grasp.cis.upenn.edu, esposito@usna.edu,
kumar@grasp.cis.upenn.edu

August 23, 2006

Abstract

We address the problem of testing complex reactive control systems and validating the effectiveness of multi-agent controllers. Testing and validation involve searching for conditions that lead to system failure by exploring all adversarial inputs and disturbances for errant trajectories. This problem of *testing* is related to *motion planning*. In both cases, there is a goal or *specification set* consisting of a set of points in state space that is of interest, either for finding a plan, demonstrating failure or for validation. Unlike motion planning problems, the problem of testing generally involves systems that are not controllable with respect to disturbances or adversarial inputs and therefore, the reachable set of states is a small subset of the entire state space. We choose to apply sampling-based algorithms to the testing and validation problem. Our work is based on the Rapidly-exploring Random Trees (RRT) algorithm. First we analyse some of the factors that govern the exploration rate of the RRT algorithm. The analysis serves to motivate our enhancements. Then we propose three modifications to the original RRT algorithm, suited for use on uncontrollable systems. First, we introduce a new distance function which incorporates information about the system's dynamics to select nodes for extension. Second, we introduce a weighting to penalize nodes which are repeatedly selected but fail to extend. Third, we propose a scheme for adaptively modifying the sampling probability distribution, based on tree growth. We demonstrate the application of the algorithm via several examples and provide computational statistics to illustrate the effect of each modification. The final algorithm is demonstrated on a 25 state example and results in nearly an order of magnitude reduction in computation time when compared with the traditional RRT. Our algorithms are also applicable to motion planning for systems that are not small time locally controllable.

1 Introduction

As the use of logic-based or reactive control laws grows in both robotics and other fields, so does the need for automated design and analysis tools. The focus to date in the automated safety verification literature has been on the solution of the reachability problem, initially through symbolic methods (e.g., [1, 38]) and later through numerical techniques (e.g., [45, 18]). However, the class of systems for which the reachability problem is tractable is quite limited in both expressiveness and dimensionality. An alternative approach to exhaustively proving safety is to simply search for a single counter example – a series of inputs, disturbances or parameters that causes a system to fail. We term this semi-decision approach the *Testing Problem*.

Inspired by the connections between the Testing Problem for complex control systems and the Motion Planning problem, we have recently applied the Rapidly-exploring Random Tree (RRT) algorithm to the Testing Problem [21, 4] with considerable success. The RRT algorithm is an incremental, randomized search algorithm which explores state space fast and uniformly. However, the Motion Planning and Testing problems are different. Perhaps the most significant difference between the two lies in the nature of the system dynamics in each case. Robotic systems are almost always controllable (by design), so the reachable space is often the entire free space. With the exception of any workspace obstacles, whose configurations are known in advance, the tree can be expected to extend to fill the entire state space. On the other hand, when we test complex control systems, it is frequently with respect to disturbances or adversarial inputs. These systems are frequently *not* controllable with respect to disturbances or adversarial inputs — in fact, the reachable set is usually a tiny fraction of the entire state space.

In such systems, (1) the lack of an obvious metric to estimate connection time, (2) the natural Voronoi bias of the RRT algorithm and (3) the traditional uniform sampling strategy, lead to a slow exploration rate. With the goal of increasing the exploration rate for complex dynamic systems, we propose three modifications to the original RRT algorithm. First, we develop a new distance function which encodes local information about the system's dynamic constraints with a first order approximation (improved metric). Second, because the reachable state space is generally a small fraction of the total state space, we introduce a weighting factor which penalizes the repeated extension of boundary nodes (mitigate Voronoi bias). Finally, we propose a scheme for adaptively modifying the sampling probability distribution between the traditional uniform distribution and heavily biased toward the specification set based on tree growth (adaptive sampling).

The paper is organized as follows. In Section 2.1 we formally define the testing problem. Section 2.2 reviews the original RRT algorithm and the most relevant literature. Section 3 defines and analyzes the factors affecting the exploration rate. In light of this analysis, Section 4 examines three key features of the traditional RRT algorithm which are troublesome for testing problems; proposes methods to remedy them and presents simple illustrative examples,

complete with comparative computational statistics. A new algorithm unifying the enhancements is presented in Section 5.1. The algorithm is used to solve a multi-agent pursuit-evasion problem and performance statistics are discussed in Section 5.2. Concluding remarks follow in Section 6.

2 Background and Related Work

2.1 Problem Statement

Definition 2.1 *We define a* **Finite Time Control System** *as a tuple* $C = (X, U, T, Init, f)$ *where*

- $X \subset \mathbb{R}^n$ *is a set of* free *state variables;*
- $U \subset \mathbb{R}^m$ *is a compact set of input values;*
- $T = [t^0, t^f] \subset \mathbb{R}$ *is a compact time interval the system evolves over;*
- $Init \subset X$ *is a compact set of possible initial conditions; and*
- $f : X \times U \to \mathbb{R}^n$ *is a vector field which prescribes the time derivative of the state variables.*

We are generally interested in systems with collections of rigid bodies with very complicated dynamics, especially high-dimensional continuous systems or hybrid (discrete/continuous) and switched systems where f may be a non-smooth function of x. We do not impose any structure on the nature of the dynamics (except assuming that solutions exist in the sense of Filippov [48], which permits sliding modes). We use the term "input" in the most general sense in that it can include yet unspecified feedback control inputs, human-in-the-loop inputs, and disturbances.

Problem 2.2 Testing Problem*: Given a tuple* (C, x^0, S)*, where*

- $C = (X, U, T, Init, f)$ *is a finite time control system,*
- $x^0 \in Init$*, and*
- S *is a specification set,*

the goal is to determine an open loop control law $\mathcal{U} : T \to U$ *such that* $\exists t \in T$ *for which* $x(t) \in S$*.*

In other words, the goal is to determine a counter-example – an input sequence which will cause the system to fail by entering S – if one exists. However, in order to make the problem algorithmically tractable, instead of searching the set of all possible functions $\mathcal{U} : T \to U$, the search must be restricted to some subset of functions with finite dimensional parameterization [12].

We make three assumptions that simplify the presentation of the key ideas in the paper without restricting the scope in a significant way. First, we will

assume there is a simple test that can be applied to determine if a point in $\mathbb{R}^n$ is a member of $X \subset \mathbb{R}^n$. Second, assume the specification set S can be defined as the sub-level set of some function $S = \{x | x \in X, s(x) \leq 0\}$. Finally, we restrict our search over $\mathcal{U}$ to piecewise constant functions of time with k segments, each of time duration Δt. Thus, instead of the continuous map $\mathcal{U}$, we consider the search over $\bar{\mathcal{U}} : T \rightarrow U$, as the search for a k-vector of parameters. With $u^i \in U$

$$\bar{u} = [u^1, u^2, \ldots, u^k]^T$$

so the input $u(t)$ is given by

$$u(t) = u^i \in U \;\; if \;\; t^0 + (i-1)\Delta t \leq t < t^0 + (i)\Delta t$$

for $i = 1, \ldots, k$.

2.2 Related Work

While there has been a great deal of work on this problem, complete methods for robot motion planning have prohibitive complexity. It was shown that the generalized mover's problem is PSPACE-hard [47]. Explicit construction of configuration space is computationally impractical. This has motivated many researchers to focus on sampling-based randomized algorithms that can solve many challenging high-dimensional problems efficiently at the expense of completeness.

In the Probabilistic Roadmap (PRM) method and its variants [29, 31, 2, 32, 30, 10], a network of paths called a *roadmap* is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a local planner (preprocessing phase). After construction of the roadmap, the user-specified initial configuration and the goal configuration are connected to the roadmap and a solution path is obtained by a graph-search algorithm (query phase). Modifications to the sampling of random configurations have been suggested such as visibility-based PRM [49], Gaussian sampling [7], the bridge test [26] and the medial axis method [25, 52]. Single-query PRM, called Lazy PRM is proposed in [6]. In Lazy PRM, the collision check is conducted only in the query phase and the roadmap is updated as the planner searches a solution path to minimize the number of collision checks. Recently, the influence of the sampling measure and sampling source on the PRM planner's performance was presented in [28] showing the source has small impact on a planner's performance as compared to the measure. If we can design a local planner that connects pairs of configuration satisfying corresponding constraints, a feasible path is obtained by simple concatenation of the successive path segments. Therefore, the PRM method can be easily applied to virtually any type of holonomic robots. However, the connection problem can be as difficult as designing a nonlinear controller, particularly for complicated nonholonomic and dynamic systems.

The problem of finding a suitable trajectory and control inputs to drive a robot from an initial state to a goal state while satisfying physically-based

dynamic constraints has been an active research area in many fields. The decoupled approach, in which one solves a basic path planning problem followed by finding a trajectory and controller that satisfies the dynamics, has been applied successfully to a variety of problems. However, decoupled approaches often fail to find a feasible solution due to the system dynamics or constraints on control inputs. It is often the case that kinematic and dynamic constraints have to be taken into consideration simultaneously. This type of problem is known as kinodynamic motion planning [20]. For planning under differential constraints, expansive space trees (ESTs) [27] assign a weight to each node indicating how densely the neighborhood of the node has already been explored and choose a node to extend with probability inversely proportional to the weight. PDST-EXPLORE algorithm uses nonuniform subdivision of state space and sampling of paths rather than states [36, 37].

A Rapidly-exploring Random Tree (RRT) is a randomized algorithm that is designed for a broad class of motion planning problems [40, 41]. The advantage of RRT algorithm is that they work directly with the set of admissible inputs and are therefore directly applicable to systems with complex dynamics. It is well suited to the problem of quickly searching high-dimensional spaces that have both algebraic and differential constraints. The key idea is to bias the exploration toward unexplored portions of the space by randomly sampling points in the state space and incrementally pulling the search tree toward them, reducing the size of largest Voronoi regions as the tree grows. Therefore, the graph explores the state space uniformly and quickly. The RRT algorithm has experienced widespread success in solving a variety of high dimensional and nonlinear problems in motion planning [34, 14, 15].

We base our approach on the Rapidly-exploring Random Tree (RRT) algorithm. A very basic algorithm is given in Algorithms 1 and 2, where ρ is some suitable metric and *pdf* is a probability distribution. The RRT algorithm is attractive because it works directly in the space of admissible inputs making them suitable for systems with dynamic constraints and because it is *probabilistically complete* [40]. However, surprisingly little attention has been directed at predicting or measuring the rate at which the trees explore the space.

There exists much work on safety verification. Reachability analysis has been an active research area. Symbolic methods [1, 24, 38, 23] are found for very limited classes of hybrid systems. Approximate reachability computation relies on numerical methods such as Hamilton-Jacobi equation [45, 50, 44], flow-pipe approximation [16, 17, 18], ellipsoidal calculus [35, 8], and polyhedral approximation [19, 3]. The barrier certificates method, similar to Lyapunov stability results, is proposed in [46]. However, construction of barrier certificates is generally not easy and a computational technique is known only for polynomial systems. The class of hybrid systems for which the reachability problem is tractable is quite limited in both expressiveness and dimensionality. The primary reason for this limitation stems from the difficulties explicitly representing and manipulating the reachable set. Such an approach overcomes traditional limitations because it makes no attempt to explicitly represent the reachable set. The drawback however is that it is a semi-decision method – the test is

inconclusive if no counter example is found. However, for non-linear, high dimensional systems there is at this time no alternative method to verify safe operation.

The approach of using the RRT algorithm to analyze logic-based control systems is recent. In [22] RRT algorithm was used to design trajectories of hybrid systems. The first published work using the RRT algorithm for analyzing hybrid systems is [9, 42]. In a similar vein, a blimp system control law was validated under unpredictable but bounded disturbances [33]. In [5], the reachable set for aircraft collision avoidance problem was obtained and several extensions of the RRT approach were mentioned. We have applied a variant of this method [4] to testing hypotheses and establishing properties of biological networks.

Algorithm 1 Generate RRT: $\mathcal{T}$

Initialize RRT: $\mathcal{T}$.addVertex(x^0)
while $\nexists x \in \mathcal{T}$ such that $s(x) \leq 0$ **do**
  Extend($\mathcal{T}$)
end while

Algorithm 2 Extend($\mathcal{T}$)

$x^{rand} \in X \leftarrow$ pdf()
$x^{near} \leftarrow \arg\min_{x^j \in \mathcal{T}} \rho(x^j, x^{rand})$
$u^{new} = \arg\min_{u \in \bar{U}} \{\rho(x^{near} + \int^{\Delta t} f(x,u)dt, x^{rand})\}$
$x^{new} = x^{near} + \int^{\Delta t} f(x, u^{new}(t))dt$
$\mathcal{T}$.addVertex(x^{new})
$\mathcal{T}$.addEdge(u^{new}, $x^{near} \rightarrow x^{new}$)

There have been several enhancements to the basic RRT algorithm. In [13] a method for penalizing the repeated selection of collision prone nodes for extension is introduced. In [43] a node selection strategy is described which increases the natural Voronoi bias of the method for the purposes of dispersion reduction. In [51, 53] adaptive RRT methods for problems with complex obstacles are addressed. However, no approach is able to reliably reduce the dispersion (which must be measured within the reachable set) for uncontrollable systems with dynamic constraints. Biasing the sampling toward regions close to the goal state has been tried in [41], [42] and [9] with some success. However the sample bias factor is fixed *a priori* and it can lead to difficulties in non-convex systems because of the presence of local minima. In [33], a metric accounting for under-actuated dynamics is suggested but is specific to the aerial robots example considered there. In [21] the Rapidly-exploring Random Forest of Trees (RRFT) algorithm was introduced which searches over time invariant parameters by planting many RRTs at a sampling of parameter values. All the trees are grown simultaneously. Individual trees may be terminated if they fail to grow at a sufficient rate.

3 Exploration rate

The goal of the testing problem is to find a sequence of inputs from x^0 to S as quickly as possible. While easily measurable and of great practical importance, the required time to find a solution for a given system strongly depends on the location of S and x^0 in X. Alternatively, because a desirable feature of any sampling-based algorithm is to rapidly explore the state space, a measure of exploration rate would be more useful and perhaps more robust as a performance measure. In this section, we define what we mean by "exploring" the state space, and analyze the factors that influence the rate of exploration. The analysis points to some methods of improving the exploration rate.

We would like to determine what fraction of the reachable set a tree $\mathcal{T}$ has explored. However, because a tree node x^k is a point it has measure zero and, therefore, a collection of tree nodes can never fill the state space. Instead, define the set of states that could be reached from a node $x^k(t^k)$ in a single iteration with time step Δt as

$$R(x^k, t^k, \Delta t) = \{x \in X | \exists u \in U, x = x^k + \int_{t^k}^{t^k+\Delta t} f(x,u)dt\}. \tag{1}$$

Note that this set has a non-zero measure, allowing us to discuss its volume which can be related to the fraction of the space that can be explored from that node. This notation has many uses, as illustrated in Figure 1.

- The set of possible locations for x^{new} in a single iteration is $R(x^{near}, t^{near}, \Delta t)$ where x^{near} is selected to minimize some metric $\rho(x^{near}, x^{rand})$.
- The *reachable set*, $R(x^0, t^0, \Delta T)$, where $\Delta T = t^f - t^0$ is the set of all states that can possibly be explored from initial condition x^0 within the given time interval $T = [t^0, t^f]$.
- The *explored set* is defined by $R(\mathcal{T}_K, \Delta t) = \bigcup_{k=0}^{K-1} R(x^k, t^k, \Delta t)$. It is the set of states that the tree can possibly explore in the next iteration, from any of the K nodes in a tree. If a state within the specification set S (or goal) is located within $R(\mathcal{T}_K, \Delta t)$, the algorithm can terminate.

Using this notation we define the *explored volume fraction* of a tree with K nodes as

$$C_K = \frac{\mu(R(\mathcal{T}_K, \Delta t))}{\mu(R(x^0, t^0, \Delta T))}, \tag{2}$$

the ratio of the volume of the explored set to that of the reachable set, where $\mu(*)$ is the measure (volume) of the set.

When a new node, x^K is added to the tree $\mathcal{T}_K$ we can define the change in the explored volume fraction as the *growth of the explored volume fraction*

$$g_{K+1} = \frac{\mu\left(R(x^K, t^K, \Delta t) - R(\mathcal{T}_K, \Delta t)\right)}{\mu(R(x^0, t^0, \Delta T))}. \tag{3}$$

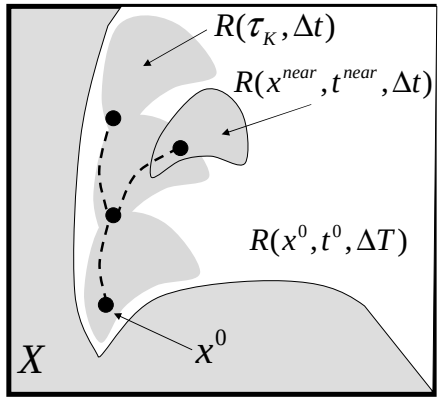

Figure 1: An illustration of $R(x^0, t^0, \Delta T)$, $R(x^{near}, t^{near}, \Delta t)$ and $R(\mathcal{T}_K, \Delta t)$
.

Since, with the exception of x^0, the location of each new node is a random variable, it is meaningful to discuss the change in the *expected value* of the explored volume fraction, $\hat{C}$ or the growth of the explored volume fraction $\hat{g}$:

$$\hat{C}_{K+1} = \hat{C}_K + \hat{g}_{K+1} = C_1 + \sum_{k=2}^{K+1} \hat{g}_k \tag{4}$$

In order to understand the dynamics of $\hat{C}$, we need to know how g_{K+1} varies with x^{rand} (illustrated in Figure 2). To that end we partition the state space into three regions. Let $int(\mathcal{T}_K)$ be the set of all nodes that do not support the boundary of $R(\mathcal{T}_K, \Delta t)$.

1. *Interior*: $x^{rand} \in \bigcup_{k \in int(\mathcal{T}_K)} R(x^k, t^k, \Delta t)$. In this case x^{rand} is selected in the interior of previously explored area. No new growth results ($g_{K+1} = g^{int} = 0$).

2. *Exterior*: $x^{rand} \notin R(\mathcal{T}_K, \Delta t)$. In this case x^{rand} is selected outside of the previously explored region and $g_{K+1} = g^{ext} \in [0, g^{\max}]$, depending on the accuracy of the metric in selecting the x^{near} with the most potential for extension.

3. *Boundary*: $x^{rand} \in R(\mathcal{T}_K, \Delta t) - \bigcup_{k \in int(\mathcal{T}_K)} R(x^k, t^k, \Delta t)$. This case represents x^{rand} in a boundary layer of the previously explored region. As in the previous case $g_{K+1} \in [0, g^{\max}]$, except the value depends on the location of x^{rand} as well as x^{near}.

It is difficult the extent to which boundary effects mentioned in Case 3 affect the value of g_{K+1} because it is a function of the current tree shape. In the interest

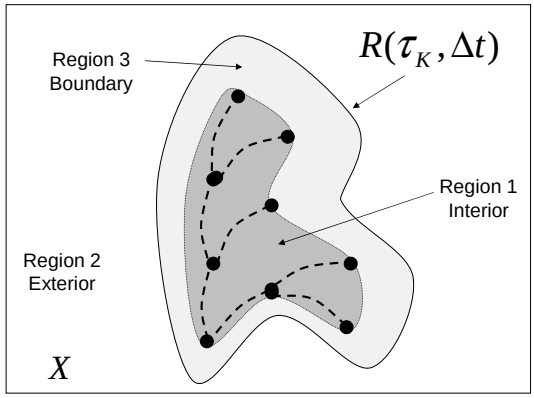

Figure 2: Three possible cases for g_{K+1} depending on the region in which x^{rand} is generated.

of streamlining the analysis, let us assume the boundary region is small. Then:

$$\hat{g}_{K+1} = P^{ext} \cdot g^{ext} \tag{5}$$

where P^{ext} is the probability x^{rand} is selected in the exterior of the explored region; and g^{ext} is the expected growth in that case. Therefore, in order to improve $\hat{g}_{K+1}$ our algorithmic enhancements will be targeted at:

1. increasing g^{ext} by using a metric, suitable for systems with complex dynamics, which is more apt in selecting nodes, x^{near} with more potential for growth (Section 4.1 and 4.2); or

2. increasing P^{ext} by altering the probability distribution to generate x^{rand} in unexplored regions of the state space (Section 4.3).

Finally, we need a method to experimentally measure the fraction of the explored set to evaluate the effect of our enhancements. Because the area of the reachable set is unknown *a priori*, from a practical point of view it is sometimes useful to define explored volume fraction with respect to the entire set X rather than the reachable set in X.

$$C_K^X = \frac{\mu(R(\mathcal{T}_K, \Delta t))}{\mu(X)}. \tag{6}$$

Even still, measuring $R(x^k, t^k, \Delta t)$ in high dimensions and accounting for the overlaps in computing the union is frequently impossible. For complex dynamic systems, the volume of the explored set is often difficult to calculate. Typically *dispersion* is used [39] which is loosely defined as the radius of the largest ball in X which does not contain a tree node. Unfortunately, it is difficult to compute and because, by only focusing on the largest such ball, it yields an overly

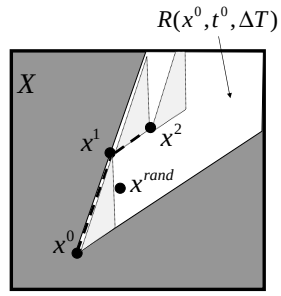

Figure 3: The white triangle is $R(x^0, t^0, \Delta T)$ – the total reachable set. The light shaded regions constitute $R(\mathcal{T}_K, \Delta t)$. Nodes selected for extension on the basis of their distance from x^{rand} (such as x^1) may be not be ideal to grow $R(\mathcal{T}_K, \Delta t)$ when the system is not small time locally controllable. x^0 is a better candidate for extension.

conservative estimate of explored set. Instead, we introduce a *dispersion-based explored fraction*. Given a tree $\mathcal{T}_K$, a set of grid points $G \subset X$ with spacing δx, and some suitable metric ρ

$$\bar{C}_K^X(\mathcal{T}_K, G) = 1 - \frac{1}{|G| \cdot \delta x} \sum_{x^g \in G} \min(\rho(x^g, \mathcal{T}_K), \delta x). \tag{7}$$

We use the dispersion-based explored fraction for the purposes of monitoring tree growth for complex dynamic systems.

4 Enhancements to the RRT Algorithm

In this section, we propose three modifications to the original RRT algorithm, all designed to improve the expected value of the growth of the explored volume fraction $\hat{g}_{K+1}$. Referring to Equation (5), the first two enhancements (Section 4.1 and 4.2) modify the metric used to select an x^{near} in an attempt to improve the growth per iteration when x^{rand} is outside the explored region, g^{ext}. This is especially challenging for systems with complex dynamics, since there are no obvious metrics to establish temporal proximity relationships and the reachable set is not known *a priori*. The third enhancement (Section 4.3), attempts to increase the probability x^{rand} is selected in an unexplored region of the state space, P^{ext}. While there exist many distributions that can accomplish this, we choose to bias our samples near the un-reached specification (goal) set when advantageous.

4.1 Dynamics-based selection of proximal node

Example 4.1 *Consider the trivial example*

$$\dot{x}_1 = 2, \quad \dot{x}_2 = u, \tag{8}$$

where $u \in U = [1, 2]$.

The reachable set $R(x^0, t^0, \Delta T)$, which is normally unknown can easily be computed by hand in this case, and is shown as the white region in Figure 3. A state x^{rand} is generated and the planner must select the "closest" tree node, x^{near} to attempt to connect from. Line 2 of Algorithm 2 (traditional RRT) selects $x^{near} \leftarrow x^1$ for extension based on proximity to x^{rand}, as determined by a distance metric ρ that is implicitly assumed to be a Euclidean metric. However, none of the possible velocity vectors at that state are able to proceed in the required direction. The closest x^{new} that can be generated from x^1 is a state that has already been visited x^2. This results in $g = 0$. Despite the fact that $\rho(x^0, x^{rand}) > \rho(x^1, x^{rand})$, x^0 is actually more suited to grow $R(\mathcal{T}_K, \Delta t)$ toward x^{rand} because the possible velocity vectors include a direction that moves toward x^{rand} resulting in $g = g^{\max}$. In addition to testing problems, this situation arises in a variety of robotic applications where the system is nonholonomic (*e.g.*, wheeled carts), and particularly in systems with unilateral constraints on velocities (*e.g.*, unmanned aerial vehicles). Ideally both distance and velocity constraints should be used to estimate a "time to go".

Since, a node, x^{near} can successfully connect to $x^{rand} \in R(x^{near}, t^{near}, \Delta t)$ it is useful to estimate the time required to go from a possible x^{near} to x^{rand} as compared with Δt. Therefore, we propose replacing $\rho(x^j, x^{rand})$ in Line 2 of Algorithm 2 with a local first order approximation of the "time-to-go"

$$t_{2go}(x^j, x^{rand}) = \begin{cases} \rho(x^j, x^{rand})/v & if \;\; v > 0 \\ \infty & if \;\; v \leq 0 \end{cases} \tag{9}$$

where v represents the instantaneous speed with which x^{rand} can be approached

$$v = \max_{u \in U} \left[-\frac{\partial \rho(x, x^{rand})}{\partial x} f(x, u)|_{x=x^j} \right].$$

Intuitively t_{2go} computes the distance from x^j to x^{rand} and divides by a first order approximation of the speed with which the distance can be decreased, giving t_{2go} units of time. Note that a negative value of v implies that the distance is actually increasing, which can be interpreted as infinite "time-to-go" (to first order). In a given iteration if none of the existing nodes have a finite value for t_{2go}, one can be chosen at random or based on some secondary criteria (such as distance as determined by ρ).

From a computational point of view, the maximization may be done by exhaustive search or by exploiting some problem dependent feature. For example if $f(x, u)$ is an affine function of u and the set U is the Cartesian product of rectangles, the maximization is a linear program in n dimensions which can be

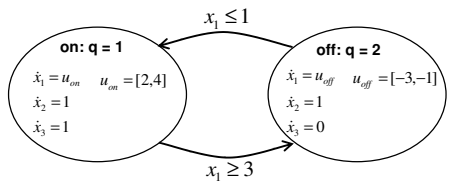

Figure 4: The system dynamics of the thermostat.

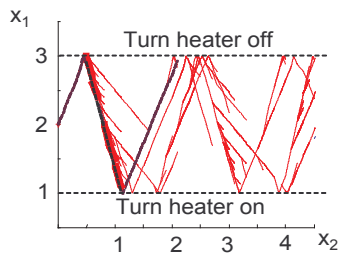

Figure 5: The solution of the thermostat counter example via the RRT using the dynamics-based selection of proximal node (temperature vs. time).

solved efficiently. If no efficient methods exist to compute this quantity, evaluating every node via this method can be intensive. In such a case, t_{2go} can be used as a secondary criterion to select x^{near} among the, for example, 10 closest nodes according to the Euclidean metric.

We next consider an example that is from the verification community. Although it is not central to robotics, it has many of the properties that are central to multi-agent robotic systems.

Example 4.2 *The hybrid automata model of a thermostat has been a popular example in the verification literature [24]. Figure 4 shows the system model.* $x = (x_1, x_2, x_3) \in X \subset \mathbb{R}^3$ *where* x_1 *is the temperature in the room,* x_2 *is the elapsed time, and* x_3 *is a timer that measures the cumulative amount of time the heater has been on for. The dynamics have two modes which denote the heater being "on" or "off". U consists of* $u_{on} = [2, 4]$*; and* $u_{off} = [-3, -1]$*. The values* u_{on} *and* u_{off} *represent the possible heating and cooling rates in the two modes. The conditions* $x_1 \le 1$ *and* $x_1 \ge 3$ *enable the mode switches* $off \rightarrow on$ *and* $on \rightarrow off$ *respectively. In [24] a symbolic verification tool is used to answer the question: "After an initialization period of two minutes, is it possible for the heater to be on for more than two thirds of the total time at any point during the first hour of operation?" Such a question may be important from an energy consumption point of view. The specification set is*

$$S = \{x \in X | 2/3x_2 - x_3 \le 0 \wedge -x_2 + 2 \le 0\}.$$

The initial conditions were mode = "on", and $x^o = [2\ 0\ 0]^T$.

Metric	No. of Nodes	Computation Time (sec)
Euclidean	2284	376.4
t_{2go}	1627	231

Table 1: Thermostat Example: A comparison of the use of the Euclidean metric and t_{2go} introduced in Section 4.1, averaged over 10 trials on a 1GHz PC.

Aside from being a classical verification example, the scenario is interesting in its own right. First, the system has quite nontrivial dynamics, since the control inputs do not appear in the right hand side of two of the state equations, or the specification equations. This, together with the narrow range of U, makes the reachable set a small subset of X. The set of possible velocity vectors at every point is very limited making this an ideal example to demonstrate the Dynamics-based selection of proximal node.

First the problem was solved 10 times selecting proximal nodes based on the Euclidean metric ρ; then 10 times with the Dynamics-based selection function t_{2go}. In all cases, the algorithm successfully computed a counter example as seen in Figure 5. Table 1 shows the computational statistics for two algorithms.

4.2 History-based selection of proximal node

A second situation is shown in Figure 6 where the traditional RRT is applied to the system and, after 3 iterations, the resulting tree is shown using dark circles and dashed line segments. The reachable set is the white region. Because the reachable set is so small, nodes on the boundary such as x^0, x^1,and x^2 will tend to maintain disproportionately large Voronoi regions causing them to be repeatedly selected as x^{near}. Initially, the boundary region is explored quickly. But then, as in the figure, when $x^{rand} \notin R(x^0, t^0, \Delta T)$, then $x^{near} \leftarrow x^1$ and $x^{new} = x^2$. But since x^{new} is already a node in the tree, the algorithm fails to extend the volume of the explored set $R(\mathcal{T}_K, \Delta t)$ (*i.e.* $g = 0$). Repeated failure to extend the boundary nodes prevents the interior nodes from extending within the reachable set.

To counter balance this Voronoi bias, we propose a weighting to prevent the repeated failure of extension. If a node is selected as x^{near} in Line 2 of Algorithm 2 and the minimization in Line 3 produces an input u^{new} which has been applied previously, the resulting x^{new} is already an element of $\mathcal{T}$. When this happens we say the node has "failed to extend"; and determine the next best u^{new} which extends explored region (suggested in [13]). For each $x^j \in \mathcal{T}$ we propose storing the number of times the node has failed to extend n^j. This value can be used to compute a penalty weight to discourage the repeated selection of boundary nodes which fail to extend. Let $n_{\min}$ and $n_{\max}$ be the least and greatest values of n^j in the tree at a given iteration. The *History-based weighting* is defined as

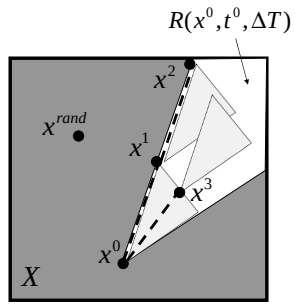

Figure 6: The reachable space is shown as the white triangular region, and bold circles and dashed lines are the RRT. Nodes on the boundary of the reachable space have disproportionately large Voronoi regions, causing them to repeatedly be selected as x^{near}.

$$H(x^j, x^{rand}; \rho) = \frac{\rho(x^j, x^{rand}) - \rho_{\min}}{\rho_{\max} - \rho_{\min}} + \frac{n^j - n_{\min}}{n_{\max} - n_{\min}} \tag{10}$$

where $\rho_{\min} = \min_{x^i \in \mathcal{T}} \rho(x^i, x^{rand})$ and $\rho_{\max}$ is defined in a similar manner. These bounds are used to normalize the distances so that the impact of the second term is not problem dependent. Note that any distance function, including t_{2go} can be substituted for ρ.

Example 4.3 *The RRT algorithm is used to find trajectories of the linear dynamic system with bounded control inputs in the form of*

$$\dot{x} = Ax + Bu + b \tag{11}$$

where $x \in X = [-200,\ 200] \times [-200,\ 200]$ *and* $u \in U = [-10,\ 10] \times [-10,\ 10]$.

Figure 7 shows trajectories generated by the RRT algorithm using the Euclidean metric (*left*) and using the History-based weighting described above (*right*). Note that reachable set is small fraction of the environment. The interior of the reachable region with the History-based selection of proximal node method is much more densely covered than when using the Euclidean metric. Figure 8 shows n^j for each node in $\mathcal{T}$. Nodes are sorted in descending order to facilitate visualization. In the conventional RRT algorithm, a smaller portion of nodes have disproportionately high values of n^j (ones on the boundary of the reachable set).

4.3 Adaptively biased sample generation

Both previous enhancements sought to maximize the growth of the explored region by increasing g when x^{rand} is generated in the unexplored region. But Equation (5) indicates another possibility is to select a probability distribution

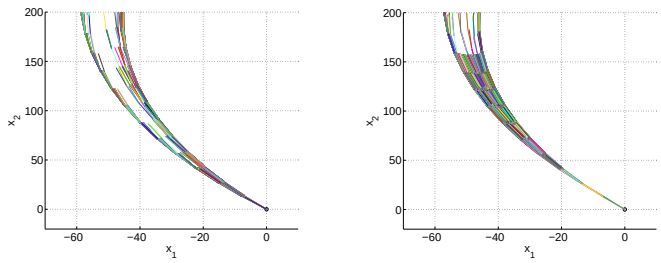

Figure 7: RRT for a linear system using the Euclidean metric (*left*) vs. a History-based selection of proximal node (*right*). After 5000 nodes the explored region of the reachable space is much more densely covered when using the weighting.

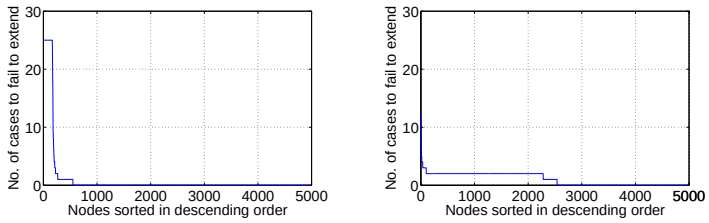

Figure 8: Value of n^j for each node (sorted in descending order) using the unweighted Euclidean metric (*left*) and History-based weighting (*right*).

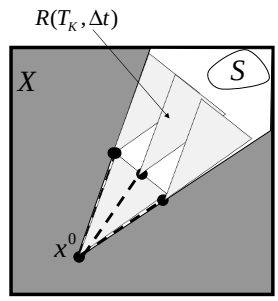

Figure 9: The reachable space is shown as the white triangular region, bold circles and dashed lines are the RRT. The algorithm terminates when x^{new} lies in the specification set S.

that would increase P^{ext} – the probability a node is selected in the exterior of the explored set $X - R(\mathcal{T}_K, \Delta t)$. However for complex systems there are two main challenges to this: (1) computing the explored set online is difficult; and (2) not all areas of the unexplored set are reachable.

Issue 1 can be overcome by realizing that, until the algorithm terminates, the specification (or goal) set S lies in the unexplored region by definition and that S has a convenient representation. Since we are seeking a trajectory to connect x^0 to S it makes sense to bias our sampling distribution inside S, as indicated in Figure 9. Assuming we follow this strategy, Issue 2 presents a greater challenge. Complex system dynamics can render S unreachable in which case the problem does not have a solution. However, a potential danger is that S is reachable but the tree fails to reach it using a biased sampling approach due to a non-convex state space or a lack of small time local controllability. Adaptive biasing circumvents this problem. Intuitively, biasing the sampling distribution for x^{rand} to generate a disproportionate number of samples inside the set S is effective when the system is easily steered toward S (*i.e.* the system is output controllable with respect to $s(x)$). To estimate this we update the amount of biasing for every N_s iterations of the RRT algorithm, where N_s is user defined number. If in a given iteration $\rho(x^{near}, x^{rand}) > \rho(x^{new}, x^{rand})$, where ρ is a metric function, we call such an iteration *successful* because the tree has grown toward x^{rand}. We count the number of successful iterations n_s, out of the n_β iterations where random states were generated inside the set defined by $s(x) \leq 0$ and compute

$$\beta = \frac{n_s}{n_\beta}. \tag{12}$$

Values of β close to unity indicate biasing sample generation inside S has been beneficial.

Our proposed probability density function $B(x; \mu, \beta)$, to be used in Line 1

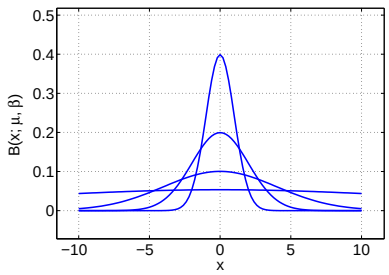

Figure 10: The distribution $B(x;\mu,\beta)$ with $\mu = 0$ and various values of β.

of Algorithm 2, resembles a Gaussian over some compact set, $a \leq x \leq b$

$$B(x;\mu,\beta) = \begin{cases} N(x;\mu,\sigma(\beta)) + C_t/(b-a), & a \leq x \leq b \\ 0 & otherwise \end{cases} \tag{13}$$

where $N(x;\mu,\sigma(\beta))$ is the Gaussian distribution with mean μ and standard deviation $\sigma(\beta)$. The last term, $C_t/(b-a)$, is added to ensure that the area under the curve is equal to one. C_t represents the area of the truncated portions above $x = b$ and below $x = a$

$$C_t = \int_{-\infty}^{a} N(x;\mu,\sigma)dx + \int_{b}^{\infty} N(x;\mu,\sigma)dx.$$

Obviously μ should be selected so that $s(\mu) \leq 0$. The standard deviation of $N(x;\mu,\sigma(\beta))$ effectively determines the bias and should be computed using $\beta \in [0,1]$

$$\sigma(\beta) = (1-\beta)(\sigma_{\max} - \sigma_{\min}) + \sigma_{\min}, \tag{14}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are user-defined values of the maximum and minimum standard deviations.

Figure 10 illustrates the shape of $B(x;\mu,\beta)$ with different values of β. Distribution (13) can be easily implemented using any random normal generator and rejecting points generated outside the compact domain.

Example 4.4 *We consider a hovercraft in constant altitude flight with 6 states, $x = (x_1, x_2, \theta, v_1, v_2, \omega)$. The dynamic equations are*

$$\begin{aligned} m\dot{v}_1 &= (f_1 + f_2)\cos(\theta) + f_{x_1 air}(x, v_{air}(x)) \\ m\dot{v}_2 &= (f_1 + f_2)\sin(\theta) + f_{x_2 air}(x, v_{air}(x)) \\ J\dot{\omega} &= (f_2 - f_1)l + \tau_{air}(x, v_{air}(x)) \end{aligned}$$

The control inputs are $u = [f_1\ f_2]^T$ (forward actuating forces) and $U = [-10,10]\times[-10,10]$. Forces due to wind disturbances in the x_1, x_2 and θ directions are $f_{x_1 air}$, $f_{x_2 air}$, and τ_{air} whose exact expressions are omitted for brevity but are quadratic in the difference between the craft's velocity and the wind velocity v_{air}

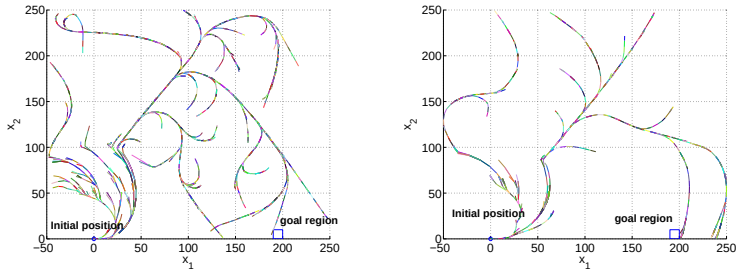

Figure 11: RRTs of the hovercraft problem with uniform sampling (*left*) and with adaptive bias (*right*).

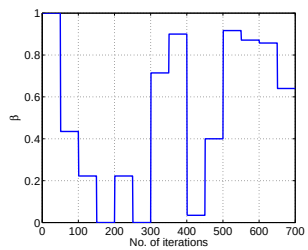

Figure 12: The evolution of the biasing factor β for the hovercraft problem.

and vary with the orientation of the craft. Note that the state is partitioned into two regions (indoor and outdoor) which define the wind velocity differently:

$$v_{air} = \begin{cases} [-\alpha_v x_2 \ \beta_v x_1]^T, & \sqrt{(x_1)^2 + (x_2)^2} \leq 100 \\ [0 \ 0]^T, & \sqrt{(x_1)^2 + (x_2)^2} > 100 \end{cases}.$$

We would like to determine if a hovercraft under these wind conditions can reach some goal zone, $S = \{(x_1, x_2) \in [190, 200] \times [0, 10]\}$. *Note that when outdoors, the wind forces are significantly greater in magnitude than the control inputs, making the system uncontrollable. The initial state is* $x^0 = [0 \ 0 \ 0 \ 0 \ 0 \ 0]^T$.

The distribution (13) was used to solve the problem 10 times on a 1GHz PC. Figure 11 shows the solutions of the problem with the uniform sampling distribution and adaptive bias. Figure 12 shows how β changes as the algorithm evolves. The adaptive algorithm is able to exploit the situations in which biasing is effective. As shown in Table 2, the adaptive biasing algorithm improves the efficiency of RRT method compared to other fixed bias strategies rather dramatically.

Sampling Method	No. of Nodes	Computation Time (sec)
Uniform	3556	1753.5
Med. Bias	1017	490.2
Heavy Bias	912	408.3
Adaptive Bias	678	342.5

Table 2: Hovercraft Example: A comparison of the sampling strategy introduced here (Adaptive Bias) to fixed-bias sampling strategies, averaged over 10 trials on a 1GHz PC.

5 Unified Algorithm

In this section we introduce a unified algorithm incorporating the three enhancements presented earlier and validate the new algorithm on a large scale example problem.

5.1 Unified algorithm

Algorithms 3 and 4 present the unification of the enhancements presented in the previous section. Note that, since most robotic problems are controllable, Algorithm 1 can terminate when a solution is found. In our case, it is a distinct possibility that no solution exists so we impose a secondary termination criterion. We can use the dispersion-based explored fraction defined in Section 3 to monitor tree growth. The change in $\bar{C}_K^X$ over the trailing N iterations $\Delta\bar{C}_K^X$, measures the growth of the tree. If $\Delta\bar{C}_K^X$ drops below some user-defined $\Delta\bar{C}_{K\,\min}^X$ we terminate the search. Before running simulations, the user needs to set the user-defined values such as $\bar{U}, \Delta\bar{C}_{K\,\min}^X, N, \delta x, N_s, \sigma_{min}$ and σ_{max} and initialize $\beta = 1$.

Algorithm 3 Generate enhanced-RRT: $\mathcal{T}$

Initialize RRT: $\mathcal{T}$.addVertex($x^0 \leftarrow x^{init}, n^0 \leftarrow 0$)
Global: $\beta = 1$
while ($\nexists x \in \mathcal{T}$ such that $s(x) \leq 0$) AND $\Delta\bar{C}_K^X \geq \Delta\bar{C}_{K\,\min}^X$ **do**
  enhanced-Extend($\mathcal{T}$)
end while

5.2 A Multiagent Problem

We consider a problem where multiple autonomous vehicles must guard against an intruder entering a designated area. This scenario has applications in games such as "capture the flag". It has applications in homeland security where autonomous vehicles (boats, airplanes, ground robots) can be deployed to detect unidentified vehicles entering a cordoned-off area or an exclusion zone.

In this example, we examine a circular area, S_E guarded by 4 robots. Each robot has sensor foot prints which are assumed to be circular with radius R_d

Algorithm 4 enhanced-Extend($\mathcal{T}$)

$\sigma = (1-\beta)(\sigma_{\max} - \sigma_{\min}) + \sigma_{\min}$
$x^{rand} \in X \leftarrow B(x; \mu, \beta)$ (see Equation (13))
$x^{near} \leftarrow \arg\min_{x^j \in \mathcal{T}} [H(x^j, x^{rand}; t_{2go})]$ (see Equation (9),(10))
$u^{new} = \arg\min_{u \in \bar{U}} [t_{2go}(x^{near}, x^{rand})]$
$x^{new} = x^{near} + \int^{\Delta t} f(x, u^{new}(t))dt$
if $x^{new} = x^j \in \mathcal{T}$ **then**
  n^j++
  while $x^{new} = x^j \in \mathcal{T}$ **do**
    $\bar{U} \leftarrow \bar{U} - u^{new}$
    $u^{new} = \arg\min_{u \in \bar{U}} [t_{2go}(x^{near}, x^{rand})]$
    $x^{new} = x^{near} + \int^{\Delta t} f(x, u^{new}(t))dt$
  end while
end if
$\mathcal{T}$.addVertex(x^{new}, $n^{new} = 0$)
$\mathcal{T}$.addEdge(u^{new}, $x^{near} \rightarrow x^{new}$)
reset $\bar{U}$
if N_s iterations **then**
  $\beta = \frac{n_s}{n_\beta}$
end if

for detection and R_c for capture, as shown in Figure 13.We assume each of the intruder and guard robots has 5 states, $x^i = (x_1^i, x_2^i, \theta^i, v^i, \omega^i)$ and 2 control inputs, $u^i = (u_1^i, u_2^i)$ where x^1 and u^1 indicate states and input of the intruder. The dynamics with nonholonomic constraints are given by:

$$\begin{aligned} &\dot{x}_1^i = v^i cos(\theta^i), \;\; \dot{x}_2^i = v^i sin(\theta^i), \;\; \dot{\theta}^i = \omega^i \\ &\dot{v^i} = u_1^i, \;\; \dot{\omega^i} = u_2^i. \end{aligned} \tag{15}$$

We can define the free space $X = X^1 \times X^2 \times \cdots \times X^5 \backslash \bigcup_{i=2}^{5} B(x^i(t), R_c) \subset \mathbb{R}^{25}$ where

$$\begin{aligned} X^i &= \{(x_1^i, x_2^i, \theta^i, v^i, \omega^i) \in \mathbb{R}^5 | (x_1^i)^2 + (x_2^i)^2 \leq R_I^2\} \\ B(x^i(t), R_c) &= \{(x_1^i, x_2^i) | (x_1^1 - x_1^i)^2 + (x_2^1 - x_2^i)^2 \leq R_c^2\}. \end{aligned}$$

Then the specification set S is defined by

$$S = \{\in X | (x_1^1)^2 + (x_2^1)^2 < R_s^2\}$$

where R_s is the radius of the circle C_E.

The guarding scheme is shown in Figure 14. Initially, the guard robots distribute evenly along the perimeter of the exclusion zone. If the intruder enters the detection range of a guard robot, the robot pursues the intruder and other robots redistribute evenly along the circle C_E. If the intruder escapes the detection range of the pursuing robot, the robot returns to the perimeter

and all robots redistribute evenly. To evenly distribute guard robots along the perimeter, we use the algorithm proposed in [11]. Each guard robot is subject to the force

$$\tau^j = -k\nabla\psi^2(q^j) - c\dot{q}^j + \sum_{k\in N_j} F_r(q^j, q^k) \tag{16}$$

where $q^j = (x_1^j, x_2^j) \in \mathbb{R}^2$ is the position of robot j, $\psi : \mathbb{R}^2 \to \mathbb{R}$ is an implicit function description of the perimeter of the exclusion zone that must be guarded and N_j is the set of robots neighboring robot j. F_r is a Coulomb-like repulsive force that ensures that the robots do not cluster together, while c is a constant which provides a viscous damping term. The force is applied to a point that is at a finite distance away from a robot to address nonholonomic constraints. A detailed description of the control law including a proof of convergence to different shapes is provided in [11]. However, the analysis in that paper cannot be used to predict the transients as each guard robot moves toward the perimeter.

The question we wish to answer is as follows. If an intruder or an adversary is allowed to start anywhere in a specified region S_I, and the guard robots, employing the control law described above, are initially evenly distributed on the circle C_E, can the intruder enter the exclusion zone (S_E) uncaptured? The answer to our question can only be found by searching for an initial condition and a control input function for the intruder which drives it into the exclusion zone without crossing any of the capture ranges. Note that the reachable set of states in X is a small subset of the entire state due to the fact that the system is uncontrollable and U is bounded. Finally, note that the intruder can start anywhere in the set S_I. In other words, the initial condition for the intruder must be chosen from this finite set, each condition leading to a RRT.

We apply the RRFT algorithm [21] with enhancements suggested in Section 5.1 to the problem. The control inputs are $u = (u_1^1, u_2^1) \in U = [-5,\ 5] \times [-\pi/15,\ \pi/15]$ with $R_I = 300m$, $R_s = 100m$, $R_d = 110m$ and $R_c = 40m$. Figure 15 shows the forest of trees where a solution trajectory is found for the algorithm with the "history-based selection of proximal node", visualizing the position of the intruder. Eight initial conditions are generated and a forest starts to grow until a solution is found. Table 3 shows the statistics obtained for this example with all the enhancements. The second column shows the average number of nodes used to find a solution trajectory for the intruder robot (one such trajectory is shown in Figure 15). The third column shows the computation time with different options. The first main point to note from these two columns is that the standard algorithm takes eight times as long requiring eight times the number of nodes to find the same solution. The second main point in this example is that adaptive biasing allows the most improvement in efficiency among the three enhancements. This is because, while the probability distribution used does increase P^{ext}, the primary motivation in selecting it was to grow the tree toward the goal, as fast as possible. Figure 16 shows the change of β as the algorithm with "adaptively biased sample generation" evolves.

Figure 17 shows the comparison of the dispersion-based explored fraction of the original algorithm compared with the history-based and dynamics-based

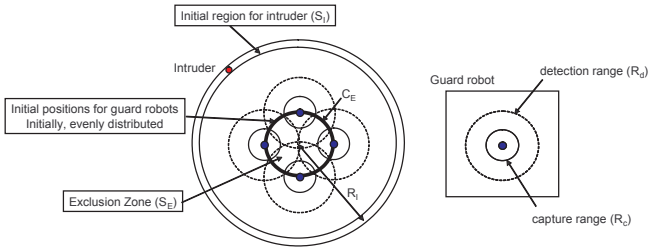

Figure 13: Initial conditions for guard robots and intruder. Each robot has a detection range R_d within which the intruder is detected, and capture range R_c within which the intruder is captured.

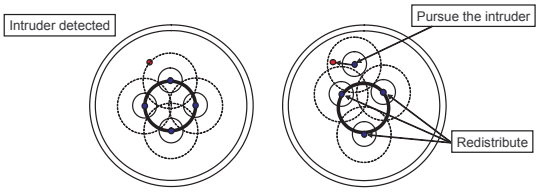

Figure 14: Guarding scheme of the robots. Distribute until the intruder is detected (*left*); and pursue if the intruder is within the detection range of a guard robot (*right*).

selection of proximal node. As expected the dynamics-based and history-based algorithms provide better tree growth.

Enhancement Method	No. of Nodes	Computation Time (sec)
No Enhancement	25251	14978.8
Dynamics-based	14944	7549.4
History-based	12896	6387.7
Adaptively biased	4924	3396.9
Three Enhancements	3352	1976.3

Table 3: Guard-intruder Example: A comparison of the algorithm, with and without enhancements averaged over 10 trials on a 3GHz PC.

6 Conclusion

The RRT algorithm has been successful in solving complex motion planning problems. We explore the application of this algorithm and its variants to the problem of testing complex reactive control systems and validating the effec-

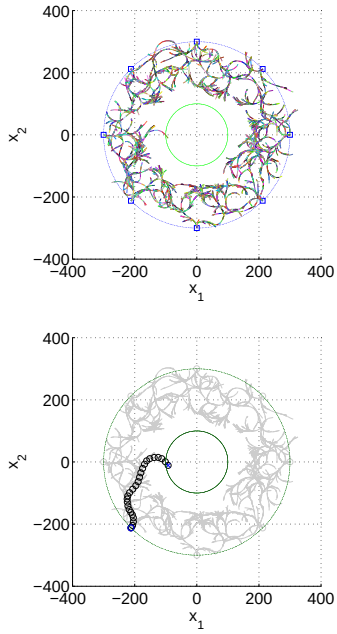

Figure 15: The forest of RRTs with 8 different initial conditions. A solution trajectory of the intruder is highlighted on the bottom.

tiveness of multi-agent controllers. Testing and validation involve searching for conditions that lead to system failure by exploring all adversarial inputs and disturbances for errant trajectories. Unlike motion planning problems, the systems may not be controllable with respect to disturbances or adversarial inputs and the reachable set of states is generally a small subset of the entire state space.

Because of the differences between testing and motion planning, we propose three modifications to the original RRT algorithm. First, we develop a new distance function which encodes local information about the system's dynamics with a first order approximation. Second, because the reachable state space is generally a small fraction of the total state space, we modify the node selection strategy to discourage the repeated selection of boundary nodes. Finally, we propose a scheme for adaptively modifying the sampling probability distribution based on tree growth. We demonstrate the application of the algorithm via three simple examples and one large scale (25 dimensions) multi-agent pursuit-evasion example problem and provide computational statistics demonstrating a reduction of computation time by a factor of eight. To demonstrate improvements in the rate at which the state space is explored with the dynamics-based and history-based selection of proximal node algorithms, we derived a dispersion-based measure of the explored set. Growth rates of the complex dynamic systems are qualitatively compared with and without enhancements of

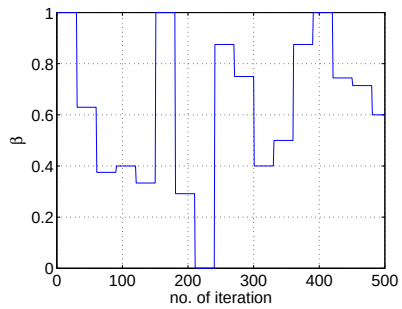

Figure 16: The evolution of the biasing factor β for the guard-intruder example.

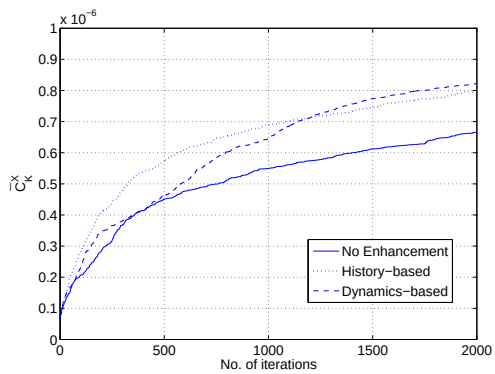

Figure 17: A comparison of the dispersion-based explored fraction.

the algorithm and the result shows distinct improvements.

Note that the enhancements introduced here are targeted at systems with complex (non-linear, non-holonomic, or non-smooth) dynamics which cause large portions of the state space to be unreachable. In contrast, motion planning problems with simple dynamics (holonomic) but geometrically complex spaces (e.g. narrow passages), pose a different set of challenges. The motivation for the History-based and Dynamics-based selection techniques here is an underlying assumption that the connection attempt (*i.e.* generating x^{new}) is (1) computationally expensive and (2) unlikely to actually reach x^{rand} due to small Δt. Therefore metrics are used as less than ideal proxies to guide the selection of which node to expend. In contrast, for systems with simple dynamics, connecting a random node to the existing search graph can be trivial and can be possibly be tested exhaustively. A more daunting issue for geometrically complex problems has to do with generating a sufficient number of samples in "critical" regions of the configuration space like narrow passages. To that end the concept of biasing the distribution adaptively can be useful.

Based on our experience we find that the relative impact each enhancement

has on algorithmic efficiency is problem dependent. For example, the history-based selection method yields significant improvements when the volume of the reachable set is a *very* small fraction of state space (Example 4.3), and the goal set lies in the interior of the reachable set. Fortunately it requires little computational overhead so, even when little is known about the system, there is little drawback to using it. In general, biasing the sample distribution yields dramatic improvements in regions of the state space where the system is small time locally controllable or the configuration space is simply connected. For systems in which this is not true, the adaptive approach takes slightly longer to find a solution compared with a uniform distribution. However, by virtue of being adaptive, after a brief transient the distribution converges to near uniform. Again, little overhead is required to implement this enhancement so there is little risk in including it. The dynamics-based selection method is very effective for systems that are not small time locally controllable; particularly those with unilateral constraints on velocities (i.e. a aircraft that moves forward only). However, the computational overhead to compute Equation (9) can vary considerably. If the input range is discretized into N_u values and an exhaustive search is used to compute the best value of u at all K nodes in the tree, it needs $K \cdot N_u$ evaluation of $f(x, u)$ and inner products at each iteration. If the tree has many nodes, this overhead will outweigh the benefits of the enhancement. An efficient method of selecting u to connect to a state is required. In such a case, t_{2go} can be used as a secondary criterion to select x^{near} as mentioned in Section 4.1. It has been our experience that the dynamics-based selection method provides considerable computational improvement even with the overhead for many examples.

Future work is being directed to a systematic approach to selecting the user-defined parameters of the algorithms proposed in this paper. For example, as thresholds for stopping the growth of a tree, we use significantly small constant. However, we believe there exists a mathematically meaningful way to set the threshold on an acceptable growth rate. Such thresholds will be of importance since they are directly related to the termination criteria for the algorithm.

Acknowledgements

This work was supported by NSF grant IIS-0413138, ONR grant FA8650-04-C-7133 and NSF grant CNS-0410514.

References

[1] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[2] N. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 113–120, 1996.

[3] E. Asarin, T. Dang, and O. Maler. d/dt: A tool for reachability analysis of continuous and hybrid systems. In *5th IFAC Symposium on Nonlinear Control Systems*, St. Petersburg, Russia, 2001.

[4] C. Belta, J. M. Esposito, J. Kim, and V. Kumar. Computational techniques for analysis of genetic network dynamics. *International Journal of Robotics Research*, 24(2-3):219–235, 2005.

[5] A. Bhatia and E. Frazzoli, editors. *Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems*, volume 2993 of *Lecture Notes in Computer Science*. Springer, 2004.

[6] R. Bohlin and L. E. Kavraki. Path planning using lazy PRM. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 521–528, 2000.

[7] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1018–1023, 1999.

[8] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 73–88, 2000.

[9] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. In *IEEE Conference on Decision and Control*, 2003.

[10] B. Burns and O. Brock. Toward optimal configuration space sampling. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[11] L. Chaimowicz, N. Michael, and V. Kumar. Controlling swarms of robots using interpolated implicit functions. In *IEEE International Conference on Robotics and Automation*, 2005.

[12] P. Cheng. *Sampling-Based Motion Planning with Differential Constraints*. PhD thesis, University of Illinois, Urbana, Illinois, 2005.

[13] P. Cheng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 43–48, 2001.

[14] P. Cheng, Z. Shen, and S. M. LaValle. Using randomization to find and optimize feasible trajectories for nonlinear systems. *Proceedings of 38th Annual Allerton Conference on Communication, Control, and Computing*, 2000.

[15] P. Cheng, Z. Shen, and S. M. LaValle. RRT-Based trajectory design for autonomous vehicles and spacecraft. *Archives of control sciences*, 11(3-4):167 –194, 2001.

[16] A. Chutinan and B. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37th International Conference on Decision and Control*, 1998.

[17] A. Chutinan and B. H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid Systems : Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 76–90, 1999.

[18] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.

[19] T. Dang and O. Maler. Reachability analysis via face lifting. In *Hybrid Systems : Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109, 1998.

[20] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the Acm*, 40(5), November 1993.

[21] J. M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *International Workshop on the Algorithmic Foundations of Robotics*, Zeist, Netherlands, 2004.

[22] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *AIAA Conference on guidance, navigation and control*, August 2000.

[23] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939, pages 225–238, Liege, Belgium, 1995. Springer Verlag.

[24] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[25] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1408–1413, 2000.

[26] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[27] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[28] D. Hsu, J.-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. In *Proceedings of the International Symposium on Robotics Research*, 2005.

[29] L. E. Kavraki. *Random networks in configuration space for fast path planning*. PhD thesis, Stanford University, 1995.

[30] L. E Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2138–2145, 1994.

[31] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362, Las Vegas, NV, 1995.

[32] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmar. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *International Transactions on Robotics and Automation: 12(4):566-580*, 1996.

[33] J. Kim, J. Keller, and V. Kumar. Design and verification of controllers for airships. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2003.

[34] J. Kim and J. P. Ostrowski. Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints. In *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[35] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 202–214, 2000.

[36] A. M. Ladd and L. E. Kavraki. Fast tree-based exploration of state space for robots with dynamics. In *International Workshop on the Algorithmic Foundations of Robotics*, Zeist, Netherlands, 2004.

[37] A. M. Ladd and L. E. Kavraki. Motion planning in the presence of drift, underactuation and discrete system changes. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[38] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32:231–253, 2001.

[39] S. M. LaValle and M. S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2002.

[40] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[41] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. Lynch, and D. Rus, editors, *Algorithmic and computational robotics: new directions*, pages 293–308. A.K. Peters, Wellesley, MA, 2001.

[42] J. A. Levine. Sampling-based planning for hybrid systems. Master's thesis, Case Western Reserve University, September 2003.

[43] S. R. Lindermann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[44] I. Mitchell. *Application of level set methods to control and reachability problems in continuous and hybrid systems*. PhD thesis, Stanford University, 2002.

[45] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 310–323. Springer Verlag, 2000.

[46] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems : Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492, 2004.

[47] J. H. Reif. Complexity of the mover's problem and generalizations. In *20th IEEE Symposium on Foundation of Computer Science*, 1979.

[48] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, 1999.

[49] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. In *Proceedings of the IEEE International Conference on Intelligent Robotics and Systems*, 1999.

[50] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi. Computational techniques for the verification and control of hybrid systems. *Proceedings of IEEE*, 91(7):986–1001, July 2003.

[51] C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *IEEE/RSJ IROS 2003*, October 2003.

[52] Y. Yang and O. Brock. Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[53] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.