2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Adaptive Sample Bias for Rapidly-exploring Random Trees with Applications to Test Generation

Jongwoo Kim
GRASP Laboratory
University of Pennsylvania, PA, USA
jwk@grasp.cis.upenn.edu

Joel M. Esposito
Weapons and Systems Engineering
US Naval Academy, MD, USA
esposito@usna.edu

***Abstract*— We are developing a randomized approach to test generation for hybrid systems, and control systems in general, inspired by the Rapidly-exploring Random Trees (RRTs) technique from robotic motion planning which has proved successful in solving high dimensional nonlinear problems. The approach represents an automated analysis alternative for systems where computing the reachable set is intractable. The standard RRTs method creates a tree in the state space by uniformly generating random sampling point and trying to find inputs which connect them. In this paper we propose a novel adaptive sampling strategy. We initially bias the distribution so that states near the "unsafe" set are selected. We continually monitor the growth of the tree. As the growth rate of the tree declines we adjust the sampling distribution to be less biased. This adaptive search strategy varies bias between "greedy" and global, often finding test trajectories more quickly than the traditional algorithm.**

***Index Terms*— Motion Planning, Randomized Algorithms, Hybrid Systems, Test Generation, Adaptive Biasing.**

I. INTRODUCTION

Hybrid systems provide mathematical models of embedded or software controlled physical systems. It is well known that the interaction between discrete and continuous time dynamics of such systems can produce rich and often unexpected behavior. For this reason, as these systems grow in complexity and sophistication, the need for automated design tools increases. The focus to date in the literature has been on the formal verification of safe operation, via the solution of the reachability problem, initially through symbolic methods [21], [13] and later through numerical techniques [2], [1], [6], [19]. However, the class of hybrid systems for which the reachability problem is tractable is quite limited in both expressiveness and dimensionality.

Test generation – generation of the set of test inputs to identify system faults and confirm correct system behavior – is a relatively new approach to analyzing dynamic systems, whereas it is a well established concept for software design [22]. Rather than prove system safety exhaustively, our approach is to try to generate a set of test scenarios that cause the system to satisfy a *specification*. A specification is a certain property we intend to check. For example, a specification is an unsafe set in safety verification problems. We are developing a randomized approach to test generation for hybrid systems, and control systems in general, using techniques from robotic motion planning which have proved successful in solving high dimensional nonlinear problems. The merit of this approach as compared with exhaustively proving safety through reachability analysis is that decidability issues do not come into play because we are not attempting to represent the entire reachable set. The drawback of the approach is that it is a semi-decision method, meaning that we can only disprove system safety by counter example – safety cannot be proved. Despite this drawback randomized approaches hold great promise for addressing complex nonlinear real-world problems for which trial and error testing is not sufficient and formal analysis is intractable.

The basic algorithm introduced here is inspired by the Rapidly-exploring Random Trees (RRTs) [15], [16] technique from robotic motion planning. The standard RRT method creates a tree in the state space by uniformly generating random sampling points and trying to find inputs which connect them. Other sampling strategies which bias the samples in a region closer to the goal state have been tried in [17] and [5] with some success. However it is difficult to decide on the biasing *a priori*. In many problems a "greedy" goal biased approach will rapidly yield the most obvious solution. However, in problems with dynamics which are not small time locally controllable or state spaces which are not simply connected a more global sampling strategy is required to discover a solution trajectory.

The primary contribution of this paper is to propose a new adaptive sampling algorithm (Section IV and V) which adjusts the sampling distribution between heavily biased toward the "unsafe" region (*i.e.*, greedy) and the traditional uniform distribution (*i.e.*, global) depending on the growth rate of the tree. The outline is as follows. In Section II, we formally define the test generation problem for hybrid systems. In Section III we point out the connection to robot motion planning and review the RRT algorithm as well as our previous work in the area, especially with regard to estimating coverage. Section IV and V describe the new

0-7803-9098-9/05/$25.00 ©2005 AACC

adaptive sampling algorithm– the primary contribution of the paper. In Section VI some computational statistics for the algorithm are presented for two example problems: the thermostat problem from the hybrid system literature and a problem involving an unmanned aerial vehicle.

II. Problem Statement

Definition 2.1: We define a **Finite Time Hybrid Control System** (modified from the Hybrid Automata, see [18]) as a tuple $H = (X, Q, U, T, Init, f, Inv, E, G, R)$ where

- $X \subset \mathbb{R}^N$ is a set of continuous variables;
- $Q \subset \mathbb{N}$ is a set of discrete variables which index the system modes;
- $U \subset \mathbb{R}^m$ is a compact set of possible input values;
- $T = [t_0, t_f] \subset \mathbb{R}$ is a compact time interval the system evolves over;
- $Init \subset X \times Q$ is a set of possible initial conditions;
- $f : X \times Q \times U \to \mathbb{R}^N$ is a vector field which prescribes the time derivative of the continuous variables (*i.e.*, $\dot{x} = f(x, q, u)$);
- $Inv : Q \to 2^X$ assigns to each $q \in Q$ an invariant set;
- $E \subset Q \times Q$ is a collection of edges describing the possible discrete transition (a.k.a.- mode switches);
- $G : E \to 2^X$ assigns to each $e = (q, q') \in E$ a guard; and
- $R : E \times X \to 2^X$ assigns to each $e = (q, q') \in E$ a reset relation.

Throughout this paper we refer to (x, q) as the state of the hybrid system. Note that we use the term "input signal" in the most general sense in that it can include yet unspecified feedback control inputs, human in the loop type inputs, disturbances, etc.

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

- $H = (X, Q, U, T, Init, f, Inv, E, G, R)$ is a finite time hybrid control system,
- $x^0, q^0 \in Init$, 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 $(x(t), q) \in S$.

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 parametrization.

For the sake of convenience we make three additional assumptions. First, assume $X \times Q \subset \mathbb{R}^N \times \mathbb{N}$ is defined in such a way that a point in $\mathbb{R}^N \times \mathbb{N}$ can be easily tested for membership in $X \times Q$. Second, assume the specification set S can be defined as the sub-level set of some function $S = \{(x, q) | x \in X, q \in Q, s(x, q) \leq 0\}$. 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 $\bar{\mathcal{U}} : T \to U$, 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 $i = 1, \ldots, k$.

III. Background and related work

A. Motion planning and the RRT

Motivated by the similarities between the Testing Problem for hybrid systems introduced in Section II and the motion planning problem from the robotics literature (in particular see [7]), we apply the Rapidly-exploring Random Trees (RRTs) algorithm [15], [16] to the testing problem. Figure 1 illustrates the concepts and a very basic algorithm is given in Algorithms 1 and 2. Note that ρ is a suitable metric function; and the notation $(x, q) + \int^{\delta t} f(x, q, u)dt$ means: using (x, q) as an initial condition, simulate the evolution of the system for δt seconds using $u(t) \in U$ as the control input. Various versions of the algorithm can be generated using different metrics (ρ) or random distributions (referred to below as `pdf`).

The benefits of the RRT algorithm are as follows.

- Because the algorithm works directly in the space of admissible inputs, it is applicable to any type of control system, unlike other algorithms [21] which are only useful for systems with a specific structure.
- It has been shown [15] to be *probabilistically complete*, meaning that, if the problem is solvable, the probability of the algorithm finding a solution approaches one as the number of nodes approaches infinity.
- Unlike testing schemes which use pure randomization of the inputs the RRT algorithm exhibits a *Voronoi Bias* [16] – meaning that it quickly visits unexplored regions of the state space.

This algorithm has experienced widespread success in solving a variety of high dimensional and nonlinear problems in motion planning and has recently been applied to controller synthesis problems for hybrid systems.

Algorithm 1 Grow Test Set $\mathcal{T}$

```
Initialize RRT: T.addVertex(x^0, q^0)
while ∄(x, q) ∈ T such that s(x, q) ≤ 0 do
  Extend(T, pdf)
end while
```

B. Coverage measures

Because many testing problems have no solution we must decide when to terminate Algorithm 1. One way to decide when to terminate the algorithm is based on how well the tree covers the state space. We review the concept of coverage and developments from [8] here. The

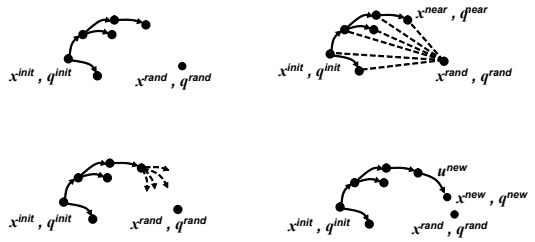

Fig. 1. The Testing algorithm (inspired by the RRT [15], [16]). The test set is represented as a tree $\mathcal{T}$ with nodes as states (x, q) and edges as inputs $u \in U$. First a new state is generated at random, (x^{rand}, q^{rand}) (*top left*) from a given probability density function (denote as `(pdf)`). The algorithm then determines the closest state, (x^{near}, q^{near}) in the tree to the random state (*top right*). It determines which $u \in U$ brings (x^{near}, q^{near}) closest to (x^{rand}, q^{rand}) (*bottom left*). n^{new} is applied for a duration δt and the new node (x^{new}, q^{new}) and edge n^{new} are added to the tree (*bottom right*).

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

$x^{rand}, q^{rand} \leftarrow$ random(pdf)
$x^{near}, q^{near} \leftarrow$ nearestNeighbor($\mathcal{T}$, (x^{rand}, q^{rand}))
$n^{new} = \arg\min_{u \in U}\{\rho(\ (x^{rand}, q^{rand}),\ (x^{near}, q^{near}) + \int^{\delta t} f(x, q, u)dt)\}$
$(x^{new}, q^{new}) = (x^{near}, q^{near}) + \int^{\delta t} f(x, q, n^{new}(t))dt$
$\mathcal{T}$.addVertex(x^{new}, q^{new})
$\mathcal{T}$.addEdge(n^{new}, $(x^{near}, q^{near}) \rightarrow (x^{new}, q^{new})$)

Discrepancy (a concept from the Monte Carlo literature) is mentioned in [14] but it is too expensive to compute online. Another appealing idea to measure the growth or coverage of the RRTs is to compute the volume of the *convex hull*. Unfortunately the convex hull is more indicative of the distribution of the points than it is of the coverage. In [17] a variant of the convex hull is explored. Rather than compute the hull of all tree vertices, vertices are grouped according to their depth from parent nodes. The union of these hulls clearly provides a better approximation. However the selection of the grouping is somewhat arbitrary. It is not clear how to relate the union to coverage due to possible overlaps. *Dispersion* measures the largest unexplored region (see [10] or more recently [14]), which was introduced in the Monte Carlo literature. Assuming we have a sample set $\bar{X}$, which contains N points, over the space X, it is defined as

$$\mu(\bar{X}, \rho) = \sup_{x \in X} \min_{\bar{x} \in \bar{X}} \rho(x, \bar{x}) \tag{III.1}$$

and can be thought of as the radius of the largest empty ball whose center lies in X. We reject it for computation on two grounds: (1) it is impractical to compute in high dimensions; and, (2) it is an overly conservative coverage measure because it only considers the *largest* ball. For example, in Figure 2 the left and right panels represent two sample sets with the same dispersion. Obviously the sample shown on the right covers the state better.

In [8] we introduced a new coverage measure which is used in this paper as termination criteria for the Testing

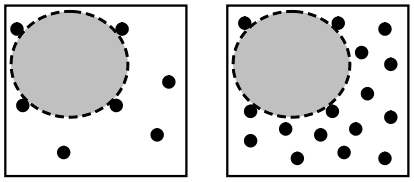

Fig. 2. Two sample sets which have the same dispersion (the radius of the largest empty ball drawn with dashed line). The set on the left clearly has inferior coverage to the set on right.

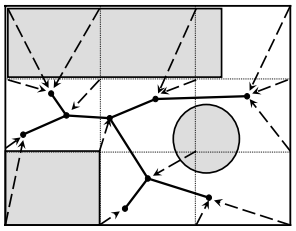

Fig. 3. A grid is superimposed on the state space. The shaded regions indicate unreachable sets. The distances from the grid points to the closest nodes are d_j (shown as dashed arrows) and the grid spacing is δ.

Algorithm. We begin by overlaying a grid containing n_g points with spacing δ on the state space (see Fig. 3). We calculate the minimum distance from each grid point j to the set of nodes in the tree, d_j. The quantity $\min(d_j, \delta)$ may be thought of as the radius of the largest ball centered at each grid point which does not contain a tree node or another grid point. Given a tree $\mathcal{T}$ we define its coverage at the k^{th} iteration of Algorithm 1 as

$$c(\mathcal{T}_k) = \frac{1}{\delta} \sum_{j=1}^{n_g} \frac{\min(d_j, \delta)}{n_g} \tag{III.2}$$

which is the average of all the node distances, normalized by the grid spacing. Our measure is similar to an approximation of an "average" dispersion (see equation (III.1)), but far less conservative and faster to compute. We can set a threshold, such that if $c(\mathcal{T}_k) \leq \bar{c}$ the algorithm should terminate. If there are unreachable regions larger than a grid cell, $c(\mathcal{T})$ will approach a non-zero constant as $k \rightarrow \infty$. In such a case we look at the change in c over the previous n_c nodes

$$\Delta c_k = -(c(\mathcal{T}_k) - c(\mathcal{T}_{k-n_c}))/n_c. \tag{III.3}$$

And define a threshold such that if $\Delta c_k \leq \bar{\Delta}$ the algorithm should terminate. An exact relationships between c and the true coverage is unknown at present; but it would likely require computing the reachable set which is, of course, assumed to be unknown. Overall one of the advantages of this measure is that, the grid size can be as fine or coarse as one chooses. Finer grids will tend to require more distance queries but are more accurate indications of coverage. Of course grids can be generated in the "output" or specification space to measure coverage there as well.

C. Related work

Aside from general work on hybrid system verification and software testing mentioned earlier, the idea of using test generation, especially RRTs, to analyze hybrid systems is new. In [20] a genetic algorithm approach is used. The first published work using RRTs for analyzing hybrid systems is [5], [17]. In a similar vein, a blimp system control law was validated under unpredictable but bounded disturbances [12]. In [4], the reachable set for aircraft collision avoidance problem was obtained and several extensions of the RRT approach were mentioned. [3] applies the method to biological networks and [8] is the first work to consider synthesizing time invariant parameters in addition to inputs. It also is the first work to address the issue of estimating coverage and termination criteria. However all of these works use a fixed sample distribution.

IV. Sample Biasing

Traditionally randomized motion planning algorithms use a uniform sample distribution to generate x^{rand} which tends to grow the tree in such a way that the entire state space is covered. In robot motion planning applications, the presence of obstacles often precludes "greedy" solutions which involve the robot proceeding directly to the goal. The use of the uniform distribution increases the algorithm's robustness when solving problem in which the state space is not convex or the system's dynamics are not small time locally controllable (STLC). However, when the state space *is* convex and the system *is* small time locally controllable, using a uniform distribution entails much wasted computation because the "obvious" solution is not explored immediately.

Since the goal in testing is to find trajectories which enter the unsafe region and, unlike the motion planning problem, we assume the state space is simply connected, another option would be to use a distribution which was biased in such a way to generate the majority of its samples in the regions defined by $s(x,q) \leq 0$ (in robotics this would be akin to the goal region). The possible drawback of a biased search scheme (or any greedy search strategy) is that it can get stuck in "local minima" for non-small time locally controllable systems. Since the notion of small time local controllability may be state, mode, or system dependent and is difficult to assess the utility of biasing *a priori*, we elect to use a parameterized distribution whose bias can be easily altered. Our probability density function $B(x;\mu,\beta)$, resembles a Gaussian over a compact domain, $a \leq x \leq b$. Define $N(x;\mu,\sigma(\beta))$ as the Gaussian distribution with mean μ and standard deviation σ. Define the probability density function

$$B(x;\mu,\beta) = \begin{cases} N(x;\mu,\sigma(\beta)) + \\ C_t/(b-a), & a \leq x \leq b \\ 0 & else \end{cases} \qquad \text{(IV.1)}$$

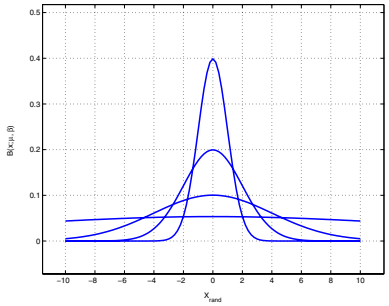

Fig. 4. The distribution $B(x;\mu,\beta)$ with $\mu = 0$ and various values of σ. Note that the function has compact support.

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 $s(\mu,q) \leq 0$. In Section V we discuss how to select σ. Note that by changing $0 \leq \sigma < \infty$ we can effectively alter the distribution to be completely deterministic ($x^{rand} = \mu$), completely uniform, or somewhere in between. Figure 4 illustrates the shape of $B(x;\mu,\beta)$ with different values of σ. Random states can be generated according to equation (IV.1) by rejecting random points generated outside the compact domain in conjunction with any standard random normal generator in each dimension. Note that q^{rand} can be generated using a similar distribution and rounding the result to the nearest integer.

V. Adaptive Sampling Algorithm

An adaptive sampling algorithm should begin with a biased search scheme and then, if the tree is growing rapidly, maintain or increase the bias; if the growth rate slows, a more uniform sampling strategy should be used. To adjust the bias we alter σ in equation (IV.1) using

$$\sigma = (1-\beta)(\sigma_{\max} - \sigma_{\min}) + \sigma_{\min}, \qquad \text{(V.1)}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are user-defined values of the maximum and minimum standard deviations and $\beta \in [0,1]$ is the biasing factor which we define below.

Intuitively, a large bias is effective in the RRT algorithm when it is possible to steer the system to the set such that $s(x,q) \leq 0$ while low bias is more advantageous when it is difficult to do so. So we must gauge the ease with which the system can be steered to a given state. We measure how well biasing works by monitoring the growth of tree when the random state is in the set satisfying $s(x,q) \leq 0$. Consider Figure 5. The vector $(x^{rand} - x^{near})$ represents the desired direction of tree growth in a given iteration of Algorithm 1, while $(x^{new} - x^{near})$ is the actual direction of growth. Therefore if the absolute value of the angle between these two vectors, $|\theta|$ is small the system is easily steered to the random state; while $|\theta| \geq \pi/2$ means that the tree is not

growing in the desired manner. This quantity is convenient because it is naturally bounded, $[0, \pi]$, and unitless. We update the amount of biasing for every N_s iterations of the RRT algorithm, where N_s is user defined number. For each of N_s iterations, we can compute the average of the absolute value of θ over the n_β iterations where random states are generated inside the set defined by $s(x, q) \leq 0$ and use this to compute the bias

$$\beta = \frac{\pi/2 - \min((1/n_\beta)\sum_{i=1}^{n_\beta} |\theta_i|, \pi/2)}{\pi/2} \tag{V.2}$$

where $\theta_1, ..., \theta_{n_\beta}$ are angles measured at each iteration. However, for systems whose state does not evolve over the Euclidean space, there is no meaningful way to compute such an angle. A more general notion of measuring the ease of steering the system to a given state is depicted in Figure 6. 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 and compute

$$\beta = \frac{n_s}{n_\beta} \tag{V.3}$$

If x^{near} is unsuccessful or the best candidate for x^{new} from x^{near} is already in a tree in the above test, we eliminate the x^{near} from consideration as x^{near} for the testing in future iterations to prevent it from being chosen repeatedly.

The appropriate definition of β can be used in Algorithm 3 to adaptively alter the bias.

Algorithm 3 Grow Test Set $\mathcal{T}$

```
Initialize RRT: T.addVertex(x^0, q^0)
c = ∞, Δc = ∞, β = 1
while ( ∄(x, q) ∈ T s.t. s(x, q) ≤ 0) ∧ c > c̄ ∧ Δc > Δ̄ do
  σ = (1 − β)(σ_max − σ_min) + σ_min,
  pdf = B(x; μ, β)
  Extend(T, pdf)
  Compute c, Δc
  if new iteration set then
    Update β
  end if
end while
Return T
```

VI. Examples

In this section we solve two example problems and examine the computational efficiency of the adaptive sampling algorithm compared to fixed biasing approaches. Specifically each example problem is solved 10 times using Algorithm 3. For comparison purposes the same problems are also solved 10 times using a variety of traditional, fixed sampling distribution RRT methods such as the one in Algorithm 1. The fixed bias solutions are computed using the distribution

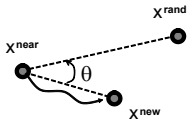

Fig. 5. If $|\theta|$ is small on average (close to zero) it is an indication that the system can be steered in the desired direction and a biasing strategy is appropriate. If $|\theta| \geq \pi/2$ it is an indication that the system cannot be steered in the desired direction and a more uniform sampling strategy is appropriate.

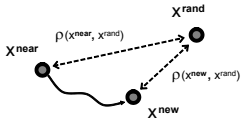

Fig. 6. A generalization of Figure 5 involves comparing two distances to determine if the tree is successfully growing. If $\rho(x^{near}, x^{rand}) > \rho(x^{new}, x^{rand})$ the iteration is considered successful.

in equation (IV.1), with $\sigma = \{1(b-a), 3(b-a), 6(b-a)\}$ – heavy bias, medium bias, and a nearly uniform distribution. The fixed bias approach is similar to [17]. The average number of tree nodes and total computation time on a $1GHz$ PC are tabulated. In each problem, we set μ so that it was in the center of the region S, defined by $s(x, q) \leq 0$. U was discretized into 100 possible values and the n^{new} was computed by exhaustive search. The algorithm in [9] was used to detect states in S. All trials converged to a solution.

A. Example 1: the thermostat

The hybrid automata model of a thermostat has been a popular example in the verification literature [11]. Figure 7 shows the system model. The discrete state space consists of two modes $Q = \{1, 2\}$ which denote the heater being "on" or "off" respectively. $X \subset \mathbb{R}^3$ 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. 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. They are unknown due to unmodeled disturbances. The guards $g_{e_1} : x_1 \leq 1$ and $g_{e_2} : x_1 \geq 3$ are associated with the edges $e_1 : 2 \rightarrow 1$ and $e_2 : 1 \rightarrow 2$ respectively. They represent the temperature at which the heater should turn on or turn off. In [11] 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. Therefore the specification set S, defined by $s(x, q) \leq 0$, is

$$S = \{s_1 = x \in X | -2/3x_2 + x_3 \geq 0 \wedge s_2 = x_2 - 2 \geq 0\}.$$

Aside from being a classical verification example, the scenario is interesting for two reasons. First, the system

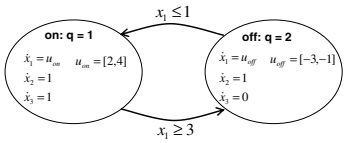

Fig. 7. The thermostat hybrid automata.

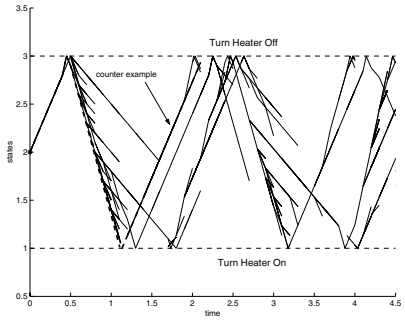

Fig. 8. The solution of the thermostat counter example via the randomized algorithm.

has quite nontrivial dynamics, since the discrete dynamics play a significant role in system behavior. For example the choice of u_{on} or u_{off} only influences x_1 which does not explicitly appear in S. It is only by influencing the switching behavior that the inputs change S. Second, the set of all "bad" trajectories only intersects the failure region by a very small margin, making the hunt for a counter example quite difficult.

Note that the true solution lies in the function space of piecewise constant inputs, so our approximation of U in no way impacts the solution. The test generator algorithm successfully computed a counter example as seen in Figure 8. The initial conditions were $q^0 = 1$ (on), $x^0 = [2\ 0\ 0]^T$. We use $N_s = 30$ and $n_c = 30$ to ensure that the means and differences were statistically significant, with $\bar{c} = 0.01$ and $\bar{\Delta} = 0.01$. The Euclidian distance was used for metric. β is computed via (V.2). Table I shows the computational statistics for the various distributions. The adaptive strategy is slightly better than the fixed heavy bias and far superior to the other fixed bias distributions by a factor of two or more. It is important to stress that there is no way to know which bias level is most effective *a priori* for a given problem so the adaptive method is most appropriate.

B. Example 2: hovercraft

For our second example we consider a scenario in which we would like to determine if a hovercraft (see Figure 9) under high wind conditions reaches some "bad" region such as a turbulent area. We denote it by goal region. We assume constant altitude flight so the craft has 6 states, $x = (x_1, x_2, \theta, v_1, v_2, \omega)$ and the dynamic equations are

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

Sampling Method	No. of Nodes	Computation time (sec)
Uniform	356	126.4
Med. Bias	164	67.3
Heavy Bias	112	40.4
Adaptive Bias	88	37.1

TABLE I

THERMOSTAT EXAMPLE: A COMPARISON OF THE SAMPLING STRATEGY INTRODUCED HERE (ADAPTIVE BIAS) TO OTHER FIXED-BIAS SAMPLING STRATEGIES, AVERAGED OVER 10 TRIALS ON A 1GHZ PC.

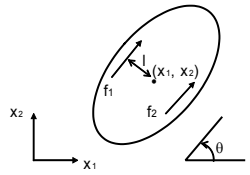

Fig. 9. Hovercraft with two thrusts

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} 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:

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

The biasing factor β is computed via (V.3). The initial state is $[0\ 0\ 0\ 0\ 0\ 0]^T$ and the goal region, which is the specification set, is $x_1 \in [190,\ 200]$, $x_2 \in [0,\ 10]$. $N_s = 50$ and $n_c = 50$ are used with $\bar{c} = 0.01$ and $\bar{\Delta} = 0.01$.

Figure 10 shows the solutions of the problem with the uniform sampling distribution and adaptive algorithm. Figure 11 shows how β changes as the algorithm evolves. Intuitively biasing is not effective in mode 1 when the craft is subject to high wind forces but it is very effective when there is no wind in mode 2. The adaptive algorithm is able to exploit the situations in which biasing is effective. As shown in Table II, adaptive biasing algorithm improves the efficiency of RRT method compared to other 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 II

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.

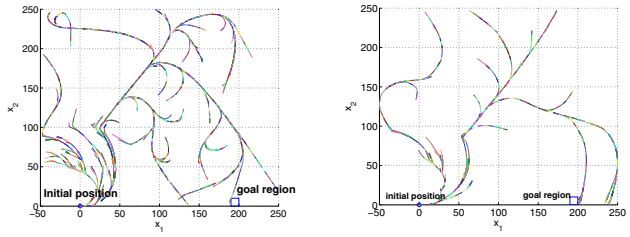

Fig. 10. RRTs of the hovercraft problem with uniform sampling (*left*) and with adaptive bias (*right*).

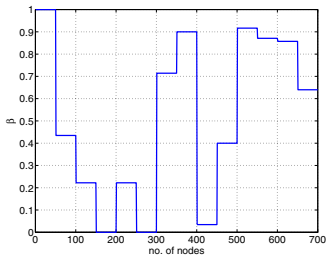

Fig. 11. The evolution of the bias β for the hovercraft problem.

VII. CONCLUSION

In this paper we introduce the idea of test generation for hybrid systems, and control systems in general, as a practical alternative for scenarios in which verification via reachability is not practical. The goal is to search for a single unsafe trajectory – a counter example to the safety or performance specification. The Rapidly-exploring Random Trees (RRTs), a well established algorithm in robot motion planning, was applied to a classical verification problem with considerable success. Two novel refinements to the traditional incarnation of the RRT algorithm are introduced to adapt the method to the unique features of the problem of test generation. First we introduced a novel on-line method of computing the coverage of the state space by the random tree. The method closely approximates known coverage measures and is efficient to compute. Because the algorithm is a semi-decision process the coverage measure is useful as termination criteria. In addition we proposed a new measure of tree growth which is used to alter the search process. Traditionally RRT methods generate samples according to a uniform distribution. We use the growth measures to adjust the distribution adaptively. As the growth rate of the tree declines we adjust the sampling distribution to be less biased. This adaptive search strategy varies bias between "greedy" and global, often finding test trajectories more quickly than the traditional algorithm.

REFERENCES

[1] Thomas A.Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 130–144. Springer Verlag, 2000.

[2] E. Asarin, T. Dang, and O. Maler. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 365–370. Springer Verlag, Copenhagen, 2002.

[3] Calin Belta, Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Computational techniques for anaylsis of genetic network dynamics. To appear in International Journal of Robotics Research.

[4] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2004.

[5] Michael S. Branicky, Michael M. Curtiss, Joshua Levine, and Stuart Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. December 2003.

[6] Oleg Butchkarev and Stavros Tripakis. Verification of hyrbid 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. Springer Verlag, 2000.

[7] 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.

[8] Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *International Workshop on the Algorithmic Foundatiosn of Robotics 2004*, Zeist, Netherlands, 2004.

[9] Joel M. Esposito, George J. Pappas, and Vijay Kumar. Accurate event detection for simulating hybrid systems. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 204–217. Springer Verlag, 2001.

[10] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.*, (2):84–90, 1960.

[11] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[12] Jongwoo Kim, Jim Keller, and Vijay Kumar. Design and verification of controllers for airships. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2003.

[13] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32:231–253, 2001.

[14] Steven M. LaValle and Michael S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Workshop on the Algorithmic Foundation of Robotics (WAFR)*. December 2002.

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

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

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

[18] John Lygeros and George Pappas. A tutorial on hybrid systems: Modeling, analysis and control. 14th IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, September 1999.

[19] Ian Mitchell and Claire 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.

[20] Q.Zhao, B.H.Krogh, and P.Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine*, 23(4):49–57, 2003.

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

[22] Hong Zhu, Patrick A. V. Hall, and John H.R. May. Software unit test coverage. *ACM Computing Surveys*, 29(4):366–427, 1997.