Randomized Simulation of Hybrid Systems For Circuit Validation

Thao Dang and Tarik Nahhal

VERIMAG
2 avenue de Vignate
38610 Gières, France

Abstract. The paper proposes a simulation-based method for validating analog and mixed-signal circuits, using the hybrid systems methodology. This method builds upon RRT (Rapidly-exploring Random Trees), a probabilistic path/motion planning technique in robotics with a special property that allows to guarantee a good coverage quality. We focus on investigating conditions for preserving this coverage property and develop a variant of the classic RRTs which is more time-efficient. These results enabled us to implement a prototype tool that can handle high dimensional hybrid models.

1 Introduction

Due to an increasing utilization of embedded systems (systems in which the computer interacts with the physical world), there has been a dramatic rise in interest in analog and mixed-signal circuits. While digital circuit design can be done with performant CAD tools, analog and mixed signal design is still much less automated. This paper proposes a technique for validating these circuits, based on the hybrid systems methodology. Hybrid systems are systems which combine discrete event systems and continuous systems, and they can naturally describe the behaviors of such circuits. Recently, much effort has been devoted to the development of automatic analysis methods and tools for hybrid systems based on formal verification (see recent proceedings of the conferences HSCC *Hybrid System: Computation and Control*). This methodology has been successfully applied to some interesting examples of analog and mixed-signal circuits, such as in [8, 4, 7]. Nevertheless, its applicability is still limited to small size systems due to the complexity of exhaustive analysis. It is clear, however, that for large analog circuits, modeled at the transistor level (rather than at the functional level), one needs lighter methods based on simulation. On the other hand, simulation, which can be used for much larger size systems, is a standard validation method in industry, despite its limitations compared to formal verification (algorithmic or deductive). Indeed, (non-exhaustive) simulation can only reveal an error but does not permit proving the correctness of the system. The goal of this paper is to bridge the gap between these two approaches by investigating a simulation-based analysis method which can guarantee some level of confidence in the results.

We now explain our problem more formally. We want to develop an analysis method for hybrid systems using an available numerical simulator in combination with a search method. This method could be guided by some coverage criteria reflecting the simulation quality we want to achieve. For simplicity, we shall focus only on continuous systems defined by a differential algebraic equation DAE:

$$F(x(t), \dot{x}(t), u(t)) = 0 \tag{1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ denotes the state variables and u denotes the input variables (modelling external disturbances or input signals). The above DAE permits describing the behaviors of a large class of analog circuits. Note that the dependence of the dynamics on the circuit parameters here can be captured by the input variables (as in the example treated in Section 7). The set $\mathcal{X}$ is the state space of the system. We assume a set $\mathcal{U}$ of admissible piecewise continuous input functions $u : \mathbb{R}^+ \to U$ where the set $U \subset \mathbb{R}^m$ is bounded. We denote the initial condition[1] by x_0. We use $\phi(t, x_0, u(\cdot))$ to denote the value at time $t \geq 0$ of the solution of (1) with the initial condition $x(0) = x_0$ under the input function $u(\cdot) \in \mathcal{U}$. The *reachable set* from x_0 of the system (1) is defined as:

$$Reach = \{y \in \mathcal{X} \mid \exists t \geq 0 \; \exists u(\cdot) \in \mathcal{U} : y = \phi(t, x_0, u(\cdot))\}.$$

To simulate the system (1) by a deterministic numerical simulator, the user needs to provide a time step h and an input function $u(\cdot)$. The simulator then produces a sequence $\{\bar{x}^k \mid k = 0, 1, 2, \ldots\}$ of points, which we call a simulation trace. If there is no numerical simulation error, $\forall k > 0 : \bar{x}^k = \phi(kh, x_0, u(\cdot))$. In this paper, we assume that a reliable DAE integration tool is provided.

One major problem with this 'classic' simulation approach is the infiniteness of the input space and of the state space because in practice, it is only possible to simulate the system with a finite number of input functions and for a bounded time horizon. Furthermore, the results are only a finite number of finite sequences of points on some trajectories of the system. In other words, a numerical simulator cannot produce in practice the output signals which are functions from reals to reals but only their approximation in discrete time. In general, in order to verify the system using simulation, one first needs to fix an input signal and then check whether the simulation trace induced by this input signal is as expected. It is thus important to choose the inputs that lead to interesting scenarios (with respect to the property/functionality to check). To fulfill a desired analysis objective (such as to verify a safety property) as best

[1] The results can be straightforwardly extended to a set of initial conditions.

as possible, the arising question is thus how to choose appropriate simulation paremeters, such as a time step h and a set $\mathcal{U}_s$ of test input functions (stimuli). Simulation coverage criteria are indeed a way to evaluate the simulation quality, or the degree of fulfilling the desired analysis objective.

In this work, to solve the reachability problem of continuous and hybrid systems, we propose to apply RRT (Rapidly-exploring Random Trees) [20], a probabilistic path and motion planning technique in robotics with a special property that allows achieving a good coverage quality. On the other hand, simulation coverage criteria are indeed a way to evaluate the simulation quality, or the degree of fulfilling a desired validation objective. The essential ideas of our approach can be summarized as follows:

– It is based on the RRT (Rapidly-exploring Random Tree) algorithm introduced in [23, 22], a probabilistic motion planning technique in robotics. This algorithm has been successful in finding feasible trajectories in motion planning. The idea of applying RRTs to the verification of hybrid systems was previously explored in [6, 9, 14]. Their relationship with our work will be dicussed in Section 8.
– We focus on determining the conditions for preserving the coverage property of RRT in the context of reachability computation. This study enabled us to develop a variant of the classic RRTs, which has lower complexity with respect to the system dimension.
– A simulation coverage measure, defined in terms of the star discrepancy of the visited points, is used to guide the simulation process.

The rest of the paper is organized as follows. In Section 2, we explain a simulation algorithm based on RRT. We next prove its completeness property with respect to the computation of the reachable set. This proof is then used to derive a modified version of RRT, which is more time-efficient and still preserves the completeness of RRT. The next section is devoted to the simulation coverage measure and its estimation. We then present a variant of the RRT algorithm which is guided by this coverage measure. We call this variant *the gRRT algorithm*. In Section 7 we describe an implementation of these algorithms and some experimental results including an analog circuits example. Before concluding, we discuss related work.

2 RRT-based Simulation

2.1 Abstract Algorithm

In path and motion planning, RRT are used to find feasible trajectories connecting a given set of points in a subset of the state space, which does not contain any obstacles and is thus called the free configuration space (see [20] and the references therein). Our abstract RRT-based simulation algorithm contains the same main steps as the classic RRT algorithms (see for example [10]). The free configuration space is indeed the state space $\mathcal{X}$. The reachable points are stored in a tree $\mathcal{T}$, the root of which corresponds to the initial state. In iteration k, a point x^k_{goal} in $\mathcal{X}$ is sampled. This point is called a goal point because it indicates the direction towards which the tree is expected to evolve. Note that in most RRT algorithms, the sampling distribution of x^k_{goal} is uniform over $\mathcal{X}$. To grow the tree towards x^k_{goal}, first an initial point x^k_{init} for the current integration step is determined. In the classic RRT algorithms, the point x^k_{init} is a nearest neighbor of x^k_{goal} (in the Euclidean distance): $x^k_{init} = argmin_{v \in V^{k-1}} ||x^k_{goal} - v||$ where V^k denotes the set of all vertices of the tree $\mathcal{T}$ at the end of iteration k. In our abstract algorithm, we do not specify how this initial point is computed, but its computation should satisfy some conditions which will be detailed later. Then, the procedure BestSuccessor tries to find the input so that the corresponding trajectory from x^k_{init} approaches x^k_{goal} as much as possible, and this results in a new point x^k_{new}. The main ingredients of the above abstract algorithm are:

Algorithm 1 RRT-based simulation algotihm

```
procedure RRT($x_0$, h)
    $\mathcal{T}^0.init(x_0)$, $k = 1$
    repeat
        $x^k_{goal}$ = Sampling($\mathcal{T}^k$)
        $x^k_{init}$ = InitialPoint($\mathcal{T}^k, x^k_{goal}$)
        $(u^k, x^k_{new})$ = BestSuccessor($x^k_{init}, h$)
        NewEdge($\mathcal{T}^k, x^k_{init}, x^k_{new}$), $k{+}{+}$
    until $k \geq K_{max}$
end procedure
```

sampling a goal point, finding an initial point, computing a best successor. So far, we did not detail how these functions are computed. In the next section, we study the conditions for preserving the completeness of RRT in the context of reachability computation.

2.2 Reachability completeness

Resolution completeness is an important property of RRT. It guarantees that for any point x in the free configuration space, the probability that the RRT tree $\mathcal{T}^k$ contains a vertex which is ε-close to x tends to 1 as the number k of iteration tends to infinity [10, 20]. This property thus makes RRT very suitable for solving safety verification problems. However, note that the proofs of the completeness in the path and motion planning context often assume that the whole free configuration space is 'controllable' in the sense that it is possible to reach any point in $\mathcal{X}$ from the initial point x_0 (see for example [10]). In our verification problem, not all the points in $\mathcal{X}$ are reachable from x_0. Indeed, if this were true, the verification problem would be solved. But we can still prove the resolution completeness with respect to the computation of the reachable set. We call this property *reachability completeness*. The proof of this result follows the idea of the proof in [2]. However, the lack of the above-mentioned controllability assumption makes the proof more complicated. We first introduce some definitions and intermediate results.

Definition 1. *For any set $S \subseteq \mathcal{X}$ with positive volume, if the probability that $x^k_{goal} \in S$ is strictly positive, then we say that the sampling process satisfies the* full coverage sampling property.

It is easy to see that the uniform sampling method satisfies this property. As we shall show later, it is a sufficient condition on the sampling process that guarantees the resolution completeness. In the remainder of the section, we assume that the sampling of goal points satisfies this property.

Given $x \in \mathbb{R}^n$ and $\varepsilon > 0$, $B(x, \varepsilon)$ is the ball with center x and radius ε. For a set V of points in $\mathbb{R}^n$, we denote the set $\bigcup_{x \in V} B(x, \varepsilon)$ by $N(V, \varepsilon)$.

Lemma 1. *Let $x \in Reach$ be a* **reachable** *point. Then, for any $\varepsilon > 0$ there exists a finite K such that $\exists v \in V^K : Pr[v \in B(x, \varepsilon)] > 0$ where V^K is the set of RRT vertices at iteration K.*

The proof of this lemma can be found in [5]. We point out that the proof uses the following important assumptions:

- (A1) There is a non-null probability that each vertex in V^k is selected to be x^k_{init}.
- (A2) If R_f is a set of reachable states with positive volume, then for all $k > 0$ $Pr[x^{k+1}_{new} \in R_f] > 0$. Intuitively, this assumption means that there is a non-null probability that 'each reachable direction' is selected.

Theorem 1. [Reachability completeness] *Given $\varepsilon > 0$ and a* **reachable** *point $x \in Reach$,*

$$lim_{k \to \infty} Pr[x \in N(V^k, \varepsilon)] = 1. \tag{2}$$

Proof. We first notice that the reachable set *Reach* is connected; therefore, for any $\varepsilon > 0$ the set $B_r(x) = Reach \cap B(x, \varepsilon)$ is always non-empty with strictly positive volume. Hence, using the full coverage sampling property, the probability $Pr[x^k_{goal} \in B_r(x)] > 0$ for all $k > 0$. We call $d^k(x) = min_{v \in V^k} ||x - v||$ the distance from x to V^k. Initially, $V^0 = \{x_0\}$, and hence $d^0(x) = ||x - x_0||$. If at iteration k, the tree already contains a vertex inside $B_r(x)$ implying that $x \in N(V^k, \varepsilon)$, then (2) is proved. It remains to prove (2) for the case where all the points in V^k are outside $B_r(x)$. We have seen that $Pr[x^k_{goal} \in B_r(x)] > 0$, and we suppose that $x^k_{goal} \in B_r(x)$. Because the whole set $B_r(x)$ is reachable, by Lemma 1, there exists a finite $k' > k$ such that

$$\exists v \in V^{k'} : Pr[v \in B_r(x)] > 0.$$

Note that $v \in B_r(x)$ implies $d^{k'}(x) < d^k(x)$. In addition, $d^k(x)$ is non-increasing with respect to k; therefore the expected value of the distance to x at iteration k' must be smaller than that at iteration k, that is $E(d^{k'}(x)) < E(d^k(x))$. Therefore, $lim_{k \to \infty} Pr[d^k(x) < \varepsilon] = 1$, which means that $lim_{k \to \infty} Pr[x \in N(V^k, \varepsilon)] = 1$ □

.

Remark. The validity of the proof of the reachability completeness requires the assumptions (A1) and (A2). These assumption guarantee that for any reachable point x there is a non-null probability that the new vertex x^{k+1}_{new} reduces the distance from x to the tree. In fact, the selection of x^k_{goal} controls the growth of the tree by determining both the initial point x^k_{init} and the direction of the expansion in each iteration. Consequently, to preserve the completeness it suffices to guarantee the satisfaction of the assumptions (A1) and (A2). The following lemma shows a sufficient condition for (A2) to be verified.

Lemma 2. *If the control set U is finite and for each $u \in U$ $Pr[u^k = u] > 0$, then the assumption (A2) is satisfied.*

The proof of this lemma can be found in [5]. In the classic RRT algorithms, the initial point for each iteration is a nearest neighbor of the goal point, and the new vertex is then computed by solving an optimal control problem (whose objective is to minimize the distance to the current goal point). These two problems are difficult, especially for non-linear systems in high dimensions. In the following, we shall exploit the above remark to derive a variant of the RRT algorithm which has lower complexity. Indeed, to determine the initial points we shall use approximate nearest neighbors.

Although this completeness property is mainly of theoretical interest, it is a way to explain the good space-covering property of the RRT algorithm, which makes it successful in solving robotic motion planning problems. This property also makes RRTs very suitable for our goal of developing a high-confidence simulation-based validation method. Indeed, we build on top of the RRT algorithm a guiding tool to bias the exploration in order to achieve a good coverage of the system's behaviors we want to check. To this end, we need a coverage measure, which is the topic of Section4.

3 Approximate RRTs

3.1 Approximating neighbors

In this section, we show the construction of our approximate RRTs. The coordinates of the tree vertices are stored in a data structure which is similar to a kd-tree. We assume that the state space $\mathcal{X}$ is a box $\mathcal{B}$. Each node of the tree

Algorithm 2 Compute the box that contains x

```
procedure ContainingLeaf($x$)
    $s = root(\mathcal{T})$, $H = \emptyset$
    while (!isLeaf($\mathcal{T}, s$)) do
        $k = s.axis()$, $d = s.pos()$
        if ($x[k] \geq d$) then
            $s = s$.rightChild(), $\sigma = -1$
        else
            $s = s$.leftChild(), $\sigma = 1$
        end if
        $H = H \cup \{\mathcal{H}(k, d, \sigma)\}$
    end while
    $\boldsymbol{b} = constructBox(H)$, $V_s = s.ptset()$
return ($s$, $\boldsymbol{b}$, $V$)
end procedure
```

has exactly two children. The information associated with a node s consists of a partitioning axis $k = s.axis()$ and a partitioning position $d = s.pos()$, which define a *partitioning plane* $x[k] = d$. The additional information associated with a leaf is a point set $V_s = s.ptset()$. Each node thus corresponds to a box, defined recursively as follows. The box of the root of the tree is $\mathcal{B}$. If the box at the node s is $\boldsymbol{b}$, and its left and right child nodes are respectively s_1 and s_2, then the boxes $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ at s_1 and s_2 are the results of dividing the box $\boldsymbol{b}$ by the partitioning plane of the parent node s. We now show how to perform two important operations on the aRTT tree: adding a new point and finding a neighbor. To add a new point x in the tree, we use the procedure ContainingLeaf in Algorithm 2, which traverses the tree from its root to a leaf whose box contains x; this box is thus called the *containing box* of x. This procedure also collects all the half-spaces defining the containing box $\boldsymbol{b}$ and the point set V at the leaf. Then, the new point x is added in V. In the algorithm, $\mathcal{H}(k, d, \sigma)$ denotes the half-space defined as $\{x \mid \sigma x[k] \leq \sigma d\}$. If the new point set needs to be split, the box $\boldsymbol{b}$ is partitioned into two sub-boxes and two new children of s are created to store the points inside each sub-box. It is easy to see that the containing box $\boldsymbol{b}$ of x does not necessarily contain a nearest neighbor of x, which may indeed be in a neighboring box. However, we restrict the search for a neighbor only within the contaning box. More concretely, let $(s, \boldsymbol{b}, V_s) = \textsc{ContainingLeaf}(x)$ where $x = x^k_{goal}$, then we compute a neighbor of x as:

$$x^k_{init} = argmin_{v \in V_s} ||x - v||. \tag{3}$$

It is important to note that this approximation is sufficient to preserve the resolution completeness. Indeed, for any *arbitrary* vertex $v \in V_s$, the Voronoi cell C_v of v with respect to $\boldsymbol{b}$ has positive volume. Due to the full coverage sampling property, the probability that the goal point is in $C_v \cap \boldsymbol{b}$ is positive and thus the probability that v is the initial point x^k_{init} as determined in (3) is also positive. The reason we use this approximation is that it has lower complexity with respect to dimension than the computation of exact nearest neighbors. It is important to note that although the resolution completeness is preserved, excessive error in this approximation might slow down significantly the convergence of the algorithm. Consequently, we control the error by fixing a maximum size of the containing boxes in the partition. An additional rule for partitioning is the maximal number of points in each box.

4 Coverage Measure

As mentioned earlier, simulation coverage is a way to evaluate the simulation quality. More precisely, it is a way to relate the number of simulations to carry out with the fraction of the system's behaviors effectively explored. The classic coverage notions mainly used in software testing, such as statement coverage and if-then-else branch coverage, path coverage (see for example [32, 28]), are not appropriate for the trajectories of continuous and hybrid systems defined by differential equations. However, geometric properties of the hybrid state space can be exploited to define a coverage measure which, on one hand, has a close relationship with the properties to verify and, on the other hand, can be efficiently computed or estimated. In this work, we are interested in point coverage and focus on a measure that describes how 'well' the explored points represent the reachable set of the system. This measure is the star discrepancy in statistics, which characterises the uniformity of the distribution of a point set within a region.

4.1 Star Discrepancy as Simulation Coverage

In this section, we present a brief introduction of the star discrepancy. The reader is referred to the excellent books on this topic, such as [19, 12, 25, 26].

The star discrepancy is an important notion in equidistribution theory as well as in quasi-Monte Carlo techniques (see for example [17]).

We assume that the state space $\mathcal{X}$ is a box $\mathcal{B} = [l_1, L_1] \times \ldots \times [l_n, L_n]$, called the bounding box. Given a set of k points $P = \{p^1, p^2, \ldots, p^k\}$ where each point p^i is in $\mathcal{B}$. The star discrepancy of P with respect to the box $\mathcal{B}$ is defined as:

$$D^*(P, \mathcal{B}) = sup_{J \in \Gamma} D(P, J)$$

where $D(P, J)$ is the local discrepancy with respect to J, a subbox of $\mathcal{B}$ of the form $J = \prod_{i=1}^{n} [l_i, \beta_i]$ with $\beta_i \in [l_i, L_i]$. The set Γ is the set of all such sub-boxes. The local discrepancy is defined as follows:

$$D(P, J) = |\frac{A(P, J)}{k} - \frac{\lambda(J)}{\lambda(\mathcal{B})}|$$

where $A(P, J)$ is the number of points of P that are inside J, and $\lambda(J)$ is the volume of the box J. Note that $0 < D^*(P, \mathcal{B}) \leq 1$.

Intuitively, the star discrepancy is a measure for the irregularity of a set of points. A large value $D^*(P, \mathcal{B})$ means that the points in P are not much equidistributed over $\mathcal{B}$.

Simulation Coverage. Let P be the set of all points explored by a simulation. The coverage of this simulation is defined as: $Cov(P) = 1 - D^*(P, \mathcal{B})$. This means that a large value of $Cov(P)$ indicates a good coverage quality.

4.2 Estimation of the Simulation Coverage

The computation of the star discrepancy is not easy (see for example [24, 16, 31]). Many theoretical results for one-dimensional point sets are not generalizable to higher dimensions, and among the fastest algorithms we can mention the one proposed in [16] of time complexity $\mathcal{O}(k^{1+d/2})$. In this work, we do not try to compute the star discrepancy but approximate it by estimating a lower and upper bound. These bounds are then used to decide whether the box $\boldsymbol{b}$ has been 'well explored' or it needs to be explored more. This estimation is based on the results published by Eric Thiémard [26, 27]. Let us briefly recall these results. Although in these results, the box $\mathcal{B}$ is $[0, 1]^n$, we have extended to the general case where $\mathcal{B}$ can be any full-dimensional box.

We define a box partition of $\mathcal{B}$ as a set of boxes $\Pi = \{\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m\}$ such that $\cup_{i=1}^{m} \boldsymbol{b}^i = \mathcal{B}$ and the interiors of the boxes $\boldsymbol{b}^i$ do not intersect. Each such box is called an *elementary box.* Given a box $\boldsymbol{b} = [\alpha_1, \beta_2] \times \ldots \times [\alpha_n, \beta_n] \in \Pi$, we define $\boldsymbol{b}^+ = [l_1, \beta_1] \times \ldots \times [l_n, \beta_n]$ and $\boldsymbol{b}^- = [l_1, \alpha_1] \times \ldots \times [l_n, \alpha_n]$. Recall that the bounding box is $\mathcal{B} = [l_1, L_1] \times \ldots \times [l_n, L_n]$ (see Figure 1 for an illustration).

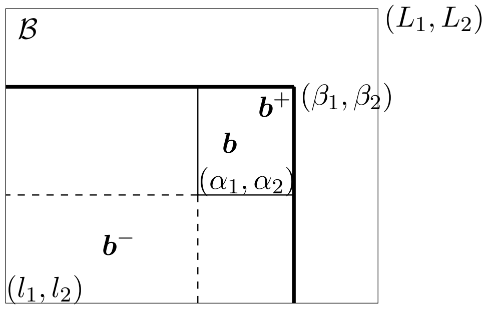

Fig. 1. Illustration of the star discrepancy notion.

For any finite box partition Π of $\mathcal{B}$, the star discrepancy $D(P, \mathcal{B})$ of the point set P with respect to $\mathcal{B}$ satisfies:

$$C(P, \Pi) \leq D(P, \mathcal{B}) \leq B(P, \Pi)$$

where the upper bound is:

$$B(P, \Pi) = \max_{\boldsymbol{b} \in \Pi} \max\{\frac{A(P, \boldsymbol{b}^+)}{k} - \frac{\lambda(\boldsymbol{b}^-)}{\lambda(\mathcal{B})}, \frac{\lambda(\boldsymbol{b}^+)}{\lambda(\mathcal{B})} - \frac{A(P, \boldsymbol{b}^-)}{k}\} \tag{4}$$

and the lower bound is:

$$C(P, \Pi) = \max_{\boldsymbol{b} \in \Pi} \max\{|\frac{A(P, \boldsymbol{b}^-)}{k} - \frac{\lambda(\boldsymbol{b}^-)}{\lambda(\mathcal{B})}|, |\frac{A(P, \boldsymbol{b}^+)}{k} - \frac{\lambda(\boldsymbol{b}^+)}{\lambda(\mathcal{B})}|\} \tag{5}$$

The imprecision of this approximation is the difference between the upper and lower bounds, which can be bounded as follows:

$$B(P, \Pi) - C(P, \Pi) \leq W(\Pi) = \max_{\boldsymbol{b} \in \Pi} (\lambda(\boldsymbol{b}^+) - \lambda(\boldsymbol{b}^-))/\lambda(\mathcal{B}) \tag{6}$$

Thus, one needs to find a partition Π such that this difference is small.

5 Discrepancy Guided Sampling

In this section, we use the definition and estimation of the star discrepancy to derive a simulation guiding strategy. Recall that our goal is to achieve a good simulation coverage quality, which is equivalent to a low level of star discrepancy of the explored points. More concretely, in each iteration of the RRT algorithm, the goal point sampling distribution is no longer uniform but biased according to the star discrepancy of the current set of explored points.

Let Π be a finite box partition of $\mathcal{B}$ that is used to estimate the star discrepancy. The sampling process consists of two steps:

– Step 1: sample an elementary box $\boldsymbol{b} \in \Pi$.
– Step 2: sample a point in $\boldsymbol{b}$.

The sampling in Step 2 is uniform. In the following, we show how to biase the elementary box sampling distribution in Step 1 in order to optimize the star discrepancy.

Let P be the set of the vertices of the tree after k iterations. We assume that the new point computed in each iteration is always added in the tree; hence, the number of points in P is exactly k. Indeed, in order to reduce the complexity of the RRT algorithm, it is sometimes preferable not to add a new point if it is too close to an existing point, since the new point does not significantly improve the coverage. We assume further that after the iteration k, no splitting is needed and the partition Π is thus unchanged.

The intuition behind our strategy is to favor the selection of an elementary box such that a new point x added in this box results in a smaller star discrepancy of the new point set $P \cup \{x\}$. The strategy is determined so as to reduce both the lower bound $C(P, \Pi)$ and the upper bound $B(P, \Pi)$.

5.1 Reducing the lower bound

We consider a set $\tilde{P}$ of k points in the box $\mathcal{B}$ such that for any box $\boldsymbol{b} \subseteq \Pi$, we have

$$\frac{\lambda(\boldsymbol{b})}{\lambda(\mathcal{B})} = \frac{A(\tilde{P}, \boldsymbol{b})}{k}$$

where $A(\tilde{P}, \boldsymbol{b})$ is the number of points inside $\boldsymbol{b} \cap \tilde{P}$. Note that P and $\tilde{P}$ have the same cardinality. We denote

$$\Delta_A(\boldsymbol{b}) = A(P, \boldsymbol{b}) - A(\tilde{P}, \boldsymbol{b}),$$

which is the difference in the number of points inside $\boldsymbol{b}$, when comparing P with $\tilde{P}$. Denote $c(\boldsymbol{b}) = \max\{|\Delta_A(\boldsymbol{b}^+)|, |\Delta_A(\boldsymbol{b}^-)|\}$, and the lower bound of the star discrepancy of the point set P over the bounding box $\mathcal{B}$ becomes:

$$C(P, \Pi) = \frac{1}{k} \max_{\boldsymbol{b} \in \Pi} \{c(\boldsymbol{b})\} \tag{7}$$

Note that in comparison with $\tilde{P}$, the negative (respectively positive) sign of $\Delta_A(\boldsymbol{b})$ indicates that in this box there is a lack (respectively an excess) of points; its absolute value indicates how significant the lack (or the excess) is. We observe that adding a point in $\boldsymbol{b}$ reduces $|\Delta_A(\boldsymbol{b}^+)|$ if $\Delta_A(\boldsymbol{b}^+) < 0$, and increases $|\Delta_A(\boldsymbol{b}^+)|$

otherwise. However, doing so does not affect $\varDelta_A(\boldsymbol{b}^-)$ (see Figure 1). Thus, we define a function reflecting the potential influence on the lower bound as follows:

$$\xi(\boldsymbol{b}) = \frac{1 - \varDelta_A(\boldsymbol{b}^+)/k}{1 - \varDelta_A(\boldsymbol{b}^-)/k}, \tag{8}$$

and we favor the selection of the box $\boldsymbol{b}$ if the value $\xi(\boldsymbol{b})$ is large. Note that: $1 - \varDelta_A(\boldsymbol{b})/k > 0$ for any box $\boldsymbol{b}$ inside $\mathcal{B}$. The intepretation of ξ is as follows. If $\varDelta_A(\boldsymbol{b}^+)$ is negative and its absolute value is large, the 'lack' of points in $\boldsymbol{b}^+$ is significant. In this case, $\xi(\boldsymbol{b})$ is large, meaning that the selection of $\boldsymbol{b}$ is favored. On the other hand, if $\varDelta_A(\boldsymbol{b}^-)$ is negative and its absolute value is large, then $\xi(\boldsymbol{b})$ is small, because it is preferable not to select $\boldsymbol{b}$ in order to increase the chance of adding new points in $\boldsymbol{b}^-$.

5.2 Reducing the upper bound

The upper bound in (4) can be rewritten as

$$B(P, \varPi) = \max_{\boldsymbol{b} \in \varPi} f_m(\boldsymbol{b}) \tag{9}$$

where $f_m(\boldsymbol{b}) = \max\{f_c(\boldsymbol{b}), f_o(\boldsymbol{b})\}$ and

$$f_c(\boldsymbol{b}) = \frac{A(P, \boldsymbol{b}^+) - A(\tilde{P}, \boldsymbol{b}^-)}{k}$$
$$f_o(\boldsymbol{b}) = \frac{A(\tilde{P}, \boldsymbol{b}^+) - A(P, \boldsymbol{b}^-)}{k}$$

Since the value of f_m is determined by the comparison between f_c and f_o. After straightforward calculations, the inequality $f_c(\boldsymbol{b}) - f_o(\boldsymbol{b}) \leq 0$ is equivalent to $f_c(\boldsymbol{b}) - f_o(\boldsymbol{b}) = \frac{1}{k}(\varDelta_A(P, \boldsymbol{b}^+) + \varDelta_A(P, \boldsymbol{b}^-)) \leq 0$. Therefore,

$$f_m(\boldsymbol{b}) = \begin{cases} f_o(\boldsymbol{b}) & \text{if } \varDelta_A(\boldsymbol{b}^+) + \varDelta_A(\boldsymbol{b}^-) \leq 0, \\ f_c(\boldsymbol{b}) & \text{otherwise.} \end{cases} \tag{10}$$

Again, note that adding a point in $\boldsymbol{b}$ increases $f_c(\boldsymbol{b})$, but this does not affect $f_o(\boldsymbol{b})$. To reduce $f_o(\boldsymbol{b})$ we need to add points in $\boldsymbol{b}^-$. Hence, if $\boldsymbol{b}$ is a box in $\varPi$ that maximizes f_m in (9), it is preferable not to add more points in $\boldsymbol{b}$ but in the boxes where the values of f_m are much lower than the current value of $B(P, \varPi)$ (in particular those inside $\boldsymbol{b}^-$). Using the same reasoning for each box $\boldsymbol{b}$ locally, the smaller $|\varDelta_A(P, \boldsymbol{b}^+) + \varDelta_A(P, \boldsymbol{b}^-)|$ is, the smaller sampling probability we give to $\boldsymbol{b}$. Indeed, as mentioned earlier, if $f_m(\boldsymbol{b}) = f_c(\boldsymbol{b})$, increasing $f_c(\boldsymbol{b})$ directly increases $f_m(\boldsymbol{b})$. On the other hand, if $f_m(\boldsymbol{b}) = f_o(\boldsymbol{b})$, increasing $f_c(\boldsymbol{b})$ may make it greater than $f_o(\boldsymbol{b})$ and thus increase $f_m(\boldsymbol{b})$, because small $|\varDelta_A(P, \boldsymbol{b}^+) + \varDelta_A(P, \boldsymbol{b}^-)|$ implies that $f_c(\boldsymbol{b})$ is close to $f_o(\boldsymbol{b})$.

We define two functions reflecting the global and local potential influences on the upper bound:

$$\beta_g(\boldsymbol{b}) = B(P, \Pi) - f_m(\boldsymbol{b}) \tag{11}$$

and

$$\beta_l(\boldsymbol{b}) = |\Delta_A(P, \boldsymbol{b}^+) + \Delta_A(P, \boldsymbol{b}^-)|/k \tag{12}$$

We can verify that $\beta_g(\boldsymbol{b})$ and $\beta_l(\boldsymbol{b})$ are always positive.

Now, combining the above functions with the function ξ in (8) that describes the potential influence on the lower bound, we define:

$$\kappa(\boldsymbol{b}) = \gamma_\xi \xi(\boldsymbol{b}) + \gamma_g \beta_g(\boldsymbol{b}) + \gamma_l \beta_l(\boldsymbol{b})$$

where γ_ξ, γ_g, and γ_l are non-negative weights that can be user-defined parameters. Then, at the iteration $k+1$, the probability of choose the box $\boldsymbol{b}$ can be defined as follows:

$$Pr(\boldsymbol{b}) = \frac{\kappa(\boldsymbol{b})}{\sum_{\boldsymbol{b} \in \Pi} \kappa(\boldsymbol{b})} \tag{13}$$

6 Guided RRT algorithm

In order to guide the simulation by the star discrepancy, we need to perform the operatio of updating the discrepancy estimation.

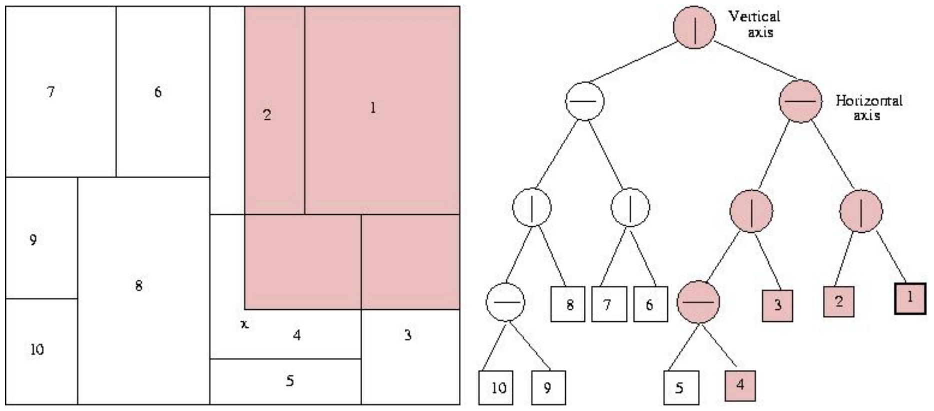

Fig. 2. Illustration of the update of the star discrepancy estimation.

Update the discrepancy estimation. When a new point x is added in the tree, the estimation of the star discrepancy needs to be updated. More concretely, we need to find all the elementary boxes $\boldsymbol{b}$ such that the new point has increased the number of points in the corresponding $\boldsymbol{b}^-$ and $\boldsymbol{b}^+$. These boxes are indeed those which intersect with the box $B_x = [x_1, L_1] \times \ldots \times [x_n, L_n]$. In addition, if $\boldsymbol{b}$ is a subset of B_x, the numbers of points in both the boxes $\boldsymbol{b}^+$ and $\boldsymbol{b}^-$ need to be incremented; if $\boldsymbol{b}$ intersects with B_x but is not entirely inside B_x, only the number of points in $\boldsymbol{b}^+$ needs to be incremented.

Searching for all the elementary boxes that are affected by x can be done by traversing the tree from the root and visiting all the nodes the boxes of which intersect with B_x. In the example of Figure 2, the box B_x is the dark rectangle, and the nodes of the trees visited in this search are drawn as dark cirles.

6.1 Preservation of the completeness property

The proofs of the completeness of RRTs are often established for the algorithms where the goal point sampling distribution is uniform and other operations are exactly computed (see for example [10]). We identify the following sampling condition: the probability that each point in the current tree is selected to be the initial point x^k_{init} is strictly positive. We can prove that this condition is sufficient for the completeness proofs to remain valid, even when the sampling distribution is non-uniform [5]. Note that to show that this sampling condition is satisfied, it suffices to prove that: for any set $S \subseteq \mathcal{X}$ with positive volume, the probability that $x^k_{goal} \in S$ is strictly positive. It is easy to see that the uniform sampling method satisfies this condition. We now give a sketch of proof that our guided sampling method and nearest neighbor approximation also satisfy it.

We first observe that, in Step 1 of the sampling method, the elementary box sampling distribution guarantees that any box has non-null probability of being selected. We consider only the case where the elementary box $\boldsymbol{b}$ where we search for a neighbor of x is also the one that contains x (the other case can be handled similarly). Let $Pr(\boldsymbol{b})$ be the set of explored points that are inside $\boldsymbol{b}$. Let $\mathcal{V}_{\boldsymbol{b}}$ be the Voronoi diagram of $Pr(\boldsymbol{b})$ restricted to $\boldsymbol{b}$ and $\mathcal{C}_p$ the corresponding Voronoi cell of an explored point p. Recall that the Voronoi cell of a point p is the set of all points that are closer to p than to any other point. We can prove that the volume of $\mathcal{C}_p$ is strictly positive. Since the sampling distribution within $\boldsymbol{b}$ in Step 2 is uniform, the probability that p is the approximate neighbor is also positive. It then follows that any point in the tree has positive probability of being selected to be the initial point x^k_{init}.

7 Experimental results

We have implemented the above-described gRRT algorithm in a prototype tool which allows us to evaluate its practical performance. We first show the result on two non-linear systems, one of which is a simple analog circuit. Then, to evaluate the scalability of the method, we test it on a set of linear systems in various dimensions (up to 100). The results reported here were obtained by running the tool on a 1.4 GHz Pentium III under Linux.

Tunnel Diode Circuit. We use a tunnel diode circuit [3] to illustrate the application of our approach to analyze the circuit behavior under parameter and input variations. The state variables $(x_1, x_2) = (I, V_d)$ where I is the current through the inductor and V_d is the voltage across the diode (see Figure 3). The differential equations describing the behavior of the circuit are:

$$\begin{aligned} \dot{x}_1 &= \tfrac{1}{C}(-\iota(x_2) + x_1) \\ \dot{x}_2 &= \tfrac{1}{L}(E - Rx_1 - x_2) \end{aligned}$$

where $C = 2pF$, $L = 5nH$, $E = 1.2V$, $R = 1.5k\Omega$, the non-linear current-voltage characteristics of the tunnel diode is defined as

$$I_d = \iota(V_d) = 17.76V_d - 103.79V_d^2 + 229.62V_d^3 - 226.31V_d^4 + 83.72V_d^5.$$

We study the behavior of the circuit under two types of variations: the variation

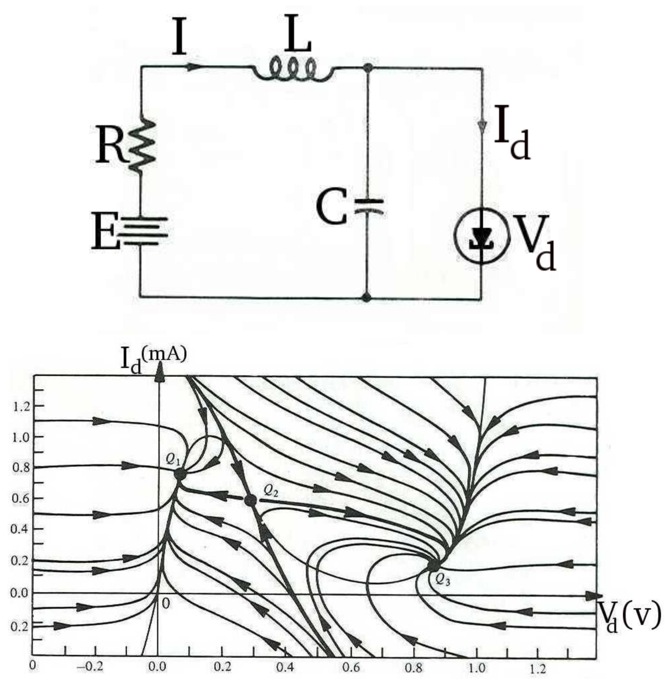

Fig. 3. The tunnel diode circuit and its phase portrait

on the diode characteristic modeled by $I_d = \iota(V_d) + \Delta_\iota$ and the source voltage variation Δ_E. These variations (Δ_ι, Δ_E) can be considered as disturbance input variables the range of which in this experiment is $[-0.12, -0.12] \times [-0.1, 0.1]$. The initial point is the unstable equilibrium state $(0.29, 0.6)$ and the randomly goal point is generated inside the rectangle $[-0.2, -0.2] \times [1.25, 1.25]$. Figure 4 shows the simulation result (after 30s), which is consistent with the phase portrait and has a high coverage quality.

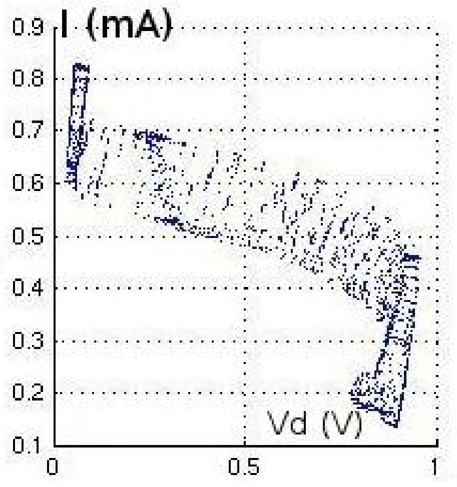

Fig. 4. Simulation results for the tunnel diode circuit

Competing specie model. The populations of two competing species can be described by the following equations:

$$\dot{x} = 2x(1 - x/2) - xy + u_1$$
$$\dot{y} = 3y(1 - y/3) - 2xy + u_2$$

We let their dynamics be slightly perturbed by an additive input u. The phase portrait of the dynamics is shown in Figure 5. The first two pictures show the simulation results after 50000 iterations, using the basic RRT and gRRT algorithms. The basic RRT algorithm we implemented uses the uniform sampling and exact nearest neighbors in the Euclidian distance, and we did not include any improvements proposed in the RRT literature.

The algorithms were run with the same initial points (100 points taken from a grid in the box $[0.2, 0.5] \times [0.2, 0.5]$, with the same integration step $h = 0.002$ and with the same set of discrete input values u (100 values from a grid in $[-0.05, 0.05] \times [0.05, 0.05]$). The run time of the RRT algorithm is 50 seconds and that of the gRRT algorithm is 1.2 minutes. From Figure 6 we can see that the coverage of the gRRT algorithm is better. Indeed, the better coverage result of the gRRT algorithm can be explained as follows. Due to the uniform sampling of goal points, the basic RRT exploration is biased by the Voronoi diagram of the vertices of the tree. More precisely, if the volume of the Voronoi cell of a node has a large volume, the node has a high probability of being selected to be the initial point x^k_{init}. If the actual reachable set is only a small fraction of the state space, the uniform sampling over the whole state space leads to a strong bias in selection of the points on the boundary of the tree, and the interior of the reachable set can only be explored after a large number of iterations. Indeed, if the reachable was known, sampling within the reachable set would produce better coverage results. In the gRRT algorithm, in each iteration, the sampling distribution is guided towards optimizing the coverage, using the discrepancy information of the current set of explored points.

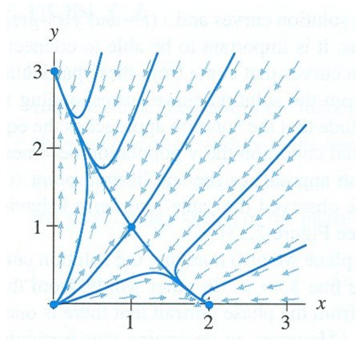

Fig. 5. Competing species model: phase portrait.

Linear Systems. We implemented the test generation algorithm using C++ in a prototype tool, and the results reported here were obtained by running the tool on a 1.4 GHz Pentium III. First, to demonstrate the performance of our algorithm, we use a set of examples of linear systems $\dot{x} = Ax + u$ in various dimensions. In this experiment, we did not exploit the linearity of the dynamics and the tested systems were randomly generated: the matrix A is in Jordan canonical form, each diagonal value of which is randomly chosen from $[-3, 3]$ and the input set U contains 100 values randomly chosen from $[-0.5, 0.5]^n$. We fix a maximal number $K_{max} = 50000$ of visited states. In terms of coverage, the star discrepancy of the results obtained by our algorithm and the classic RRT algorithm are shown in Table 1 (left), which indicates that our algorithm has better coverage quality. These discrepancy values were computed for the final set of visited states, using a partition optimal w.r.t. to the imprecision bound in (6). Note that in each iteration of our test generation algorithm we do not compute such a partition because it is very expensive. The results obtained on

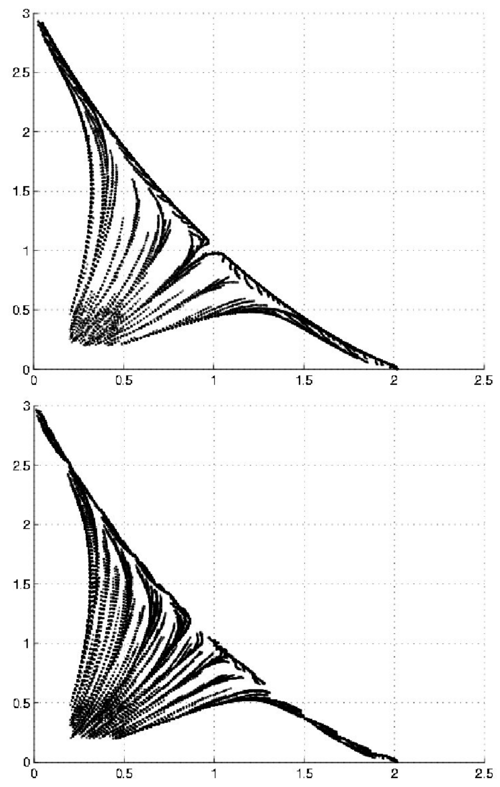

Fig. 6. Competing species model: results using the basic RRT algorithm (above) and results obtained using the gRRT algorithm (below).

a 2-dimensional system are visualized in Figure **??**. Table 1 (right) shows the time efficiency of our algorithm for linear systems of dimensions up to 100.

dim n	Lower bound		Upper bound	
	Algo 1	RRT	Algo 1	RRT
3	0.451	0.546	0.457	0.555
5	0.462	0.650	0.531	0.742
10	0.540	0.780	0.696	0.904

dim n	Time (min)
5	1
10	3.5
20	7.3
50	24
100	71

Table 1. Discrepancy results and computation time for some linear systems.

Hybrid Systems. To illustrate the application of our algorithm to hybrid systems, we use the well-known aircraft collision avoidance problem [30]. The dynamics of each aircraft is as follows: $\dot{x_i} = vcos(\theta_i) + d_1 sin(\theta_i) + d_2 cos(\theta_2)$, $\dot{y_i} = vsin(\theta_i) - d_1 cos(\theta_i) + d_2 sin(\theta_2)$, $\dot{\theta_i} = \omega$ where x_i, y_i describe the position and θ_i is the relative heading. The continuous inputs are d_1 and d_2 describing the external disturbances on the aircrafts and $-\delta \leq d_1, d_2 \leq \delta$. There are three discrete modes. At first, each aircraft begins in straight flight with a fixed heading (mode 1). Then, as soon as two aircrafts are within the distance between each other, they enter mode 2, at which point each makes an instantaneous heading change of 90 degrees, and begins a circular flight for π time units. After that, they switch to mode 3 and make another instantaneous heading change of 90 degrees and resume their original headings from mode 1. Thus for N aircrafts, the system has $3N + 1$ continuous variables (one for modeling a clock). The result for $N = 2$ aircrafts with the disturbance bound $\delta = 0.06$ is shown in Figure 7. In this example, the collision distance is 5 and no colission was detected after visiting 10000 states. The computation time was 0.9 min. For the same example with $N = 10$ aircrafts (see Figure 8), the computation time was 10 min and a collision was detected after visiting 50000 states.

8 Related work

The RRT algorithm have been used to solve a variety of reachability-related problems such as hybrid systems planning, control, verification and testing (see for example [14, 15, 6] and references therein). In this section, we only discuss a comparison of our approach with some existing RRT-based approaches for the validation of continuous and hybrid systems.

The problem of defining a coverage measure was investigated in [15], where the authors proposed a discretized version of dispersion, defined over a set of

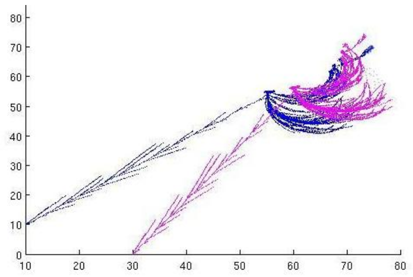

Fig. 7. Two-aircraft collision avoidance (10000 visited states, computation time: 0.9 min).

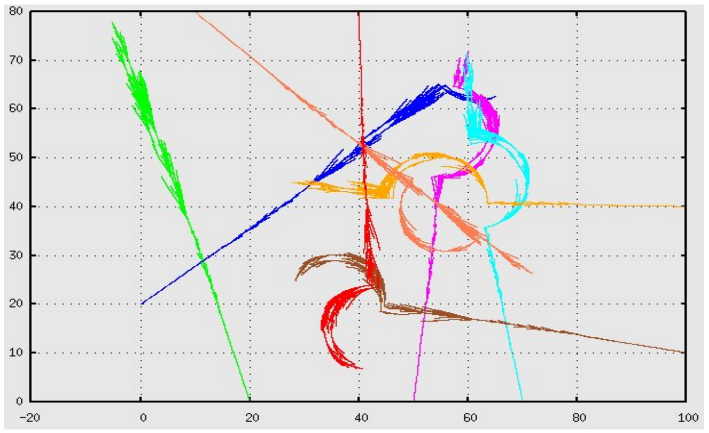

Fig. 8. Eight-aircraft collision avoidance (50000 visited states, computation time: 10 min).

grid points with a fixed size δ. The spacing s_g of a grid point g is the distance from g to the tree if it is smaller than δ, and $s_g = \delta$ otherwise. Let S be the sum of the spacings of all the grid points. This means that the value of S is the largest when the tree is empty. Then, the coverage measure is defined in terms of how much the vertices of the tree reduce the value of S. While in our work, the coverage measure is used to guide the simulation, in [15] it is used as a termination criterion. On the other hand, the star discrepancy was also used in a number of deterministic variants of the probabilistic path and motion planning algorithms (see for example [13]). This is to enhance the uniformity of sampled goal points over the state space, compared to the methods based on pseudo-random number generations. In our work, we use this notion to describe the actual coverage of the explored points.

In the definition and computation of a nearest neighbor, besides the use of different distances, controllability can be taken into account by exploiting the particularity of the dynamics and the exploration history [14, 15, 6]. Our nearest neighbor approximation can be thought of as a special definition of nearest neighbors. We remark that, in this paper we did not yet exploit the controllability information.

Finally, our idea of guiding the simulation via the sampling process has some similarity with the sampling domain control [29]. As mentioned earlier, the RRT exploration is biased by the Voronoi diagram of the vertices of the tree. If there are obstacles around such vertices, the expansion from them is limited and choosing them frequently can slow down the exploration. In the dynamic-domain RRT algorithm, the domains over which the goal points are sampled need to reflect the geometric and differential constraints of the system, and more generally, the controllability of the system. A similar idea was used in [15] where the number of successful iterations is used to define an adaptive biased sampling. Thus, the difference which is also the novelty in our guiding method is that we use the information about the current coverage of the explored points in order to improve it. However, this can be combined with controllability information to achieve more efficient guiding strategies.

9 Conclusion

In this paper we described a simulation-based approach for the validation of continuous and hybrid systems. This approach is built upon the RRT algorithm, a robotic motion planning technique. The contribution of our paper is a way to guide the simulation by a coverage measure defined in terms of the star discrepancy of the explored points. The final result of the paper is gRRT, a guided version of the RRT algorithm. The experimental results obtained using an implementation of the gRTT algorithm show its scalability to high dimensional systems and an improvement in simulation coverage quality. A number of directions for future research can be identified. One direction is to extend the approach to hybrid systems. Convergence rate of the gRRT algorithm is another interesting theoretical problem to tackle. This problem is particular hard especially in the verification context where the system is subject to uncontrollable inputs. We are also interested in defining a measure for trace coverage. Finally, we intend to apply the results of this research to develop a simulation-based tool specialized for analog and mixed-signal circuits, a domain where simulation is a widely used technique.

Although the paper focused on analog systems, the results can be straightforwardly extended to hybrid systems by using a hybrid systems numerical simulator, such as Simulink/Matlab. We intend to continue this work in a number of directions. On one hand, we are currently working on a coverage measure allowing a better bias towards the behaviors interesting with respect to the property to prove. This measure should also take into account discrete transitions in hybrid models. On the other hand, in this paper we assumed a reliable simulator which accurately computes new successor points. However, there are delicate numerical analysis problems associated with high index differential algebraic equations of a number of practical circuits which, if not handled carefully, may lead to unreliable simulation results or to prohibitively-slow computation. We thus intend to combine our RRT-based method with a recent numerical integration technique based on *the non-smooth approach* [1], developed at INRIA Rhônes-Alpes.

References

1. V. Acary and F. Pérignon. SICONOS: A software platform for modeling, simulation, analysis and control of Non Smooth Dynamical system. *Proc. of MATHMOD*, ARGSIM Verlag, 2006.
2. P. Cheng and S. M. LaValle, Resolution complete rapidly-exploring random trees, In *Proc. IEEE Int'l Conference on Robotics and Automation*, pages 267–272, 2002.
3. L. O. Chua, C. A. Desoer, and E. S. Kuh. Linear and nonlinear circuits. McGraw-Hill Book Company, 1987.
4. T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *FMCAD*, LNCS, Springer, 2004.
5. T. Dang and T. Nahhal. Randomized simulation of hybrid systems. Technical report, Verimag, IMAG, May 2006.
6. A. Bhatia and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. HSCC, LNCS 2993, pages 142-156, Springer, 2004.
7. Proceeding of Workshop on Formal Verification of Analog Circuits (ETAPS Satellite Event), Edinburgh, ENTCS 153(2), 2005.
8. W. Hartong, L. Hedrich, and E. Barke. On discrete modelling and model checking for nonlinear analog systems. In *Computer Aided Verification*, LNCS 2404, 401–413, Springer, 2002.
9. J. Kim, J. M. Esposito, and V. Kumar. An RRT-based Algorithm for Testing and Validating Multi-Robot Controllers. In *Robotics: Science and Systems*, Cambridge, MA, 2005.

10. J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 995–1001, 2000.
11. S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.
12. J. Beck and W. W. L. Chen. Irregularities of Distribution. Cambridge University Press, 1987.
13. S.M. LaValle, M.S. Branicky, and S.R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. Intl. Journal of Robotics Research. 23(7-8):673-692, August 2004.
14. M. S. Branicky, M. M. Curtiss, J. Levine, and Stuart Morgan. Sampling-based reachability algorithms for control and verification of complex systems. *Proc. Thirteenth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, 30 May-1 June 2005.
15. J.M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Int. Workshop on the Algorithmic Founddations of Robotics* , Zeist, Netherlands, 2004.
16. D. Dobkin and D. Eppstein. Computing the discrepancy. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 47-52, 1993.
17. Entacher, K. Discrepancy Estimates Based on Haar Functions. *Math. Computers in Simulation* 55, 49-57, 2001.
18. B. Krogh, J. Kapinski, O. Maler, and O. Stursberg. On systematic simulation of continuous systems. In *Hybrid Systems: Computation and Control HSCC'03*, LNCS 2623, pages 283-297, 2003.
19. L. Kuipers and H. Niederreiter. Uniform Distribution of Sequences. John Wiley, New York, 1974.
20. S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.
21. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Int. Journal of Robotic Research* 20, 378-400, 2001.
22. S. M. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Proc. of the 1999 IEEE Int. Conf. on Robotics and Automation*, 1999.
23. S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technival report 98-11, Iowa State University, Ames, IA, 1998.
24. H. Niederreiter. Discrepancy and convex programming. Ann. Mat. Pura Appl. 93, pp 89-97, 1972.
25. J. Matousek. Geometric Discrepancy. Springer-Verlag, 1999.
26. E. Thiémard. Computing bounds for the star discrepancy. Computing 65(2):169-186, 2000.
27. E. Thiémard. An algorithm to compute bounds for the star discrepancy. J. Complexity 17(4):850, Dec 2001.
28. J. Tretmans. Testing Concurrent Systems: A Formal Approach. In *Int. Conference on Concurrency Theory CONCUR*, LNCS 1664, Springer, 1999.
29. A. Yershova, L. Jaillet, T. Simeon, and S.M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proc. IEEE International Conference on Robotics and Automation*, 2005.
30. I. Mitchell and C. Tomlin. Level Set Methods for Computation in Hybrid Systems HSCC, LNCS 1790, Springer, 2000.

31. P. Winker and K.-T. Fang. Application of threshold accepting to the evaluation of the discrepancy of a set of points. SIAM J. Numer. Anal. 34, pp 20282042, 1997.
32. H. Zhu, P.A.V. Hall, and J.H.R. May. Software Unit Test Coverage and Adequacy. In *ACM Computing Surveys*, 29, 4. Dec. 1997.