Hybrid Approximate Gradient and Stochastic Descent for Falsification of Nonlinear Systems

Shakiba Yaghoubi and Georgios Fainekos

***Abstract*— Studying transient properties of nonlinear systems is an important problem for safety applications. Computationally, it is a very challenging problem to verify that a nonlinear system satisfies a safety specification if not impossible. Therefore in many cases engineers try to solve a related problem, i.e. they try to find a system behavior that does not satisfy a given specification. This problem is called specification falsification. Optimization has been shown to be very effective in addressing the falsification problem. In this paper, we provide effective and practical local and global optimization strategies to falsify a nonlinear system of arbitrary complexity.**

I. INTRODUCTION

Every system should be designed so that it satisfies a set of requirements. Formal or semi-formal verification methods can be used in order to guarantee that the system behavior is appropriate with respect to sequencing of events, conditional reachability, invariants, and real-time constraints [1], [2], [3]. Most of these behavioral properties can be captured using Metric or Signal Temporal Logic (MTL [4] or STL [5]- see also [6] for a discussion on MTL and STL which will be referred to as temporal logics (TL) from this point on.)

However, formal verification of non-linear systems is a challenging problem (see [7] for an overview). As a result, semi-formal methods are used to study system properties. In a class of semi-formal methods, the system is driven to exhibit its inappropriate behavior – if any. This information is then used for further improvement of the system design. This process is called *Falsification*. Recently, a lot of effort has been invested in this approach for discovering unsafe behaviors of systems [8], [9], [10], [11]. S-TALIRO [12] and BREACH [13] are two software toolboxes that can be used for falsification of TL specifications.

In search based falsification, the system will be driven such that its trajectories get closer to invalidating a correctness requirement in TL. In particular, the space of operating conditions and/or inputs is searched in order to find an initial condition, a parameter and an input signal, that will force the system to exhibit an unsafe behavior. This problem can be converted into an optimization problem where the goal is to minimize the robustness of the specification (see [9] and for a more general discussion [7]).

Consider for instance the case where the correctness requirement is the avoidance of an unsafe set. If we try to minimize the distance of the system's trajectories to this set, then we may finally succeed in hitting the set which falsifies the problem and reveals an incorrect behavior. For this problem, a robustness value is defined for each trajectory and it shows how far the system is from being falsified, i.e., how far the trajectory is from entering the unsafe set.

The aforementioned approach highly depends on accurate simulations that require long simulation times. As a result, decreasing the total number of simulations is critical for all simulation-based methods. Considering this fact, the verification problem has been solved in [14] using the least possible number of simulations for linear time-varying systems.

The falsification problem for nonlinear systems using simulations is more challenging. In [15], the authors present a solution for autonomous systems. Later in [16], the problem was extended to non-autonomous systems by analytically computing a descent direction for the robustness valuation. Starting from an initial condition and a trivial input, the paper computes a descent direction that leads to another initial condition and input whose respected trajectory has a smaller minimum distance from the unsafe set. Gradient descent is a local search method that makes use of the structural information of the nonlinear model. This information, in general, may or may not be available. As a result, we need to consider some alternative methods for dealing with black-box or gray-box systems. Hence, in this paper, we present a method that approximates the decent directions without requiring such structural information.

While gradient descent is a local method, simulated Annealing (SA) is a global heuristic method to minimize the cost function. This method has been used in [9] to solve the falsification problem. Although [9] does not require the structural information about the system's model, it usually has a low convergence speed and requires a large number of simulations. As a result, a method that possibly brings together the speed of descent search and the globality of stochastic search may be preferable. Falsification for linear hybrid systems using local search has been studied in [17] and it has been shown to improve the falsification process.

In summary, in this paper, we make the following contributions. We first provide an alternative method for locally minimizing the robustness value using approximate descent directions, which do not require structural information about the system's model. Then, we show that this approximation can get arbitrarily close to the analytically computed descent directions. As a next step, we combine the gradient descent method with SA to bring together the advantages of both methods. Finally, using experimental results we show the effectiveness of our method.

S. Yaghoubi and G. Fainekos are with the Schools of Engineering at Arizona State University, Tempe, AZ, E-mail: {syaghoubs,fainekos}@asu.edu

This work was partially supported by the NSF awards CNS-1319560 and IIP-1361926, and the NSF I/UCRC Center for Embedded Systems.

II. Problem Description

Consider the following non-autonomous dynamical system, where $x \in X$ are the system states with $X \subseteq \mathbb{R}^n$ and $u \in U^{\mathbb{R}}$ are the input signals where U is a bounded set.

$$\Sigma: \quad \dot{x} = F(x, t, u) \tag{1}$$

It is assumed that (1) F is differentiable; (2) the initial state x_0 will reside in a bounded and closed set X_0 and (3) given a specific initial condition and a finite time $T > 0$, a unique solution $s_t(x_0, u)$ for the system exists. Let $\mathcal{U} \subset X$ denote the set of unsafe system states. If a solution s that enters $\mathcal{U}$ is found, then the falsification problem is solved.

Definition II.1. *Let $d_{\mathcal{U}}(x) = \inf_{y \in \mathcal{U}} ||x - y||$, then the robustness of a trajectory s with respect to $\mathcal{U}$ is [15]:*

$$f(x_0, u) = \min_{0 \le t \le T} d_{\mathcal{U}}(s_t(x_0, u)) \tag{2}$$

The notion of robustness with respect to an unsafe set $\mathcal{U}$ can be extended to more general requirements expressed with TL. Details on how the robustness of a TL formula relates to the robustness of an unsafe set can be found in [15].

If we consider f to be the robustness valuation of the system, our general problem is:

Problem 1. *Minimize $f(x_0, u)$ for all $x_0 \in X_0,\ u \in U^{[0,T]}$*

If the value becomes zero, then the system is falsified.

III. Optimization for Falsification

In the following, we provide local and global optimization methods to solve Problem 1.

A. Gradient Descent (GD) Computation

In order to introduce our solution, first we will review some previous results from [16]: Given $T > 0$ and starting from some x_0 and $u \in \bar{U} \subset U^{[0,T]}$, the goal is to find descent vectors dx and du for which there exist $\bar{h}_x$ and $\bar{h}_u$ such that for all $h = (h_x, h_u) \in [0, \bar{h}_x] \times [0, \bar{h}_u]$

$$f(x_0 + h_x dx, u + h_u du) < f(x_0, u) \tag{3}$$

This property enables us to shrink the descent vectors to fit the new initial state and input in the intended sets of X_0 and U and still have a smaller cost function.

Based on the results in [16] and [18], it can be shown that the descent directions dx and du for the system (1) can be computed using the following equations:

$$du(\tau) = n_{s_{t^*}(x_0,u)}(t^*) p_u(t^*, \tau), \tag{4}$$

$$dx_0 = n_{s_{t^*}(x_0,u)}(t^*) p_{x_0}(t^*). \tag{5}$$

where $t^* = \text{argmin}_{\,0<t<T} d_{\mathcal{U}}(s_t(x_0, u))$, $n_{s_t^*(x,u)}(t^*) = -\frac{\partial G}{\partial x}|_{s_t^*(x,u)}$, and p_u and p_{x_0} are sensitivity related matrices calculated using the following initial value problems:

$$\begin{aligned}
\frac{d}{dt} p_{x_0} &= A(s_t(x_0, u)).p_{x_0}(t), \quad p_{x_0}(0) = \mathbf{I}_{n \times n}, \\
\frac{d}{dt} p_u(t, \tau) &= A(s_t(x_0, u)).p_u(t, \tau), \\
p_u(\tau, \tau) &= B(s_\tau(x_0, u)), \\
A(s_t(x_0, u)) &= \left.\frac{\partial F}{\partial x}\right|_{s_t(x_0,u)}, \\
B(s_\tau(x_0, u)) &= \left.\frac{\partial F}{\partial u}\right|_{s_\tau(x_0,u)}.
\end{aligned} \tag{6}$$

Note that in order to overcome the problem of infinite dimensionality of the trajectory optimization problem in the presence of inputs, we limit our search over the inputs which can be represented using a linear combination of basis functions. Then the search would be over the weights of the basis functions. For instance, τ in the above equations is the starting time of each time interval in the case of piecewise constant inputs.

In this method, if at some point we could not decrease the robustness value along the descent direction, then first we decrease the step size h in order to see whether we have crossed the basin of some local minimum. If we could not find a better answer after some refinement, we terminate and decide that this point is probably a local minimum.

Example 1. *In the following GD has been used to decrease the robustness value of the following benchmark with two input signals:*

$$\dot{x} = \begin{bmatrix} x_1(t) - x_2(t) + 0.1t + u_1(t) \\ x_2(t)\cos(2\pi x_2(t)) - x1(t)\sin(2\pi x_1(t)) + 0.1t + u_2(t) \end{bmatrix} \tag{7}$$

where $X_0 = [0, 1]^2$ and $\mathcal{U} = [1.5, 1.7] \times [0.9, 1.1]$. Starting from the initial condition $x_0 = (0.3, 0.8)$ and zero input signals, the robustness value decreased from 1.2042 to 0.032.

B. Local Optimization: Approximate Descent (AD)

While sensitivity analysis is inevitable for descent direction calculation, it is a hard task, in general, and impossible when we deal with black-box systems where no analytical information about function F is available. For example, in many practical applications, we deal with complicated models (like complex Simulink models) where sensitivity analysis is either impossible or computationally expensive. However, limited information about the system, e.g., linearization of the model at operating points, is usually available. In this section, we show that a number of system linearizations along the trajectory will help us approximate the descent directions.

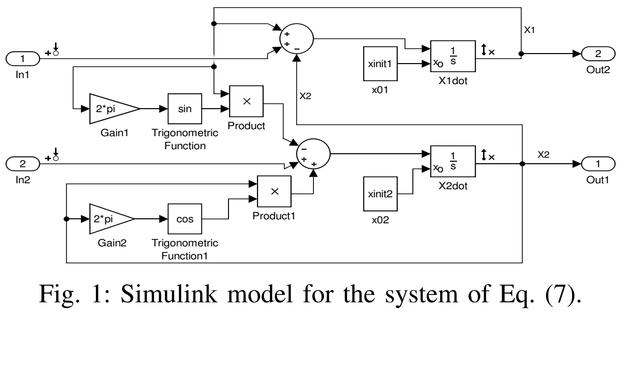

Fig. 1: Simulink model for the system of Eq. (7).

Assume that we have simulated the system of Eq. 1 starting from the nominal initial condition x_0, and $s_{t^*}(x_0, u)$ is the closest point on the trajectory to the unsafe set. We choose N samples x_0, $x_1 = s_{t_1}(x_0, u)$, ..., $x_N = s_{t^*}(x_0, u)$ on the nominal trajectory where $0 = t_0 < t_1 < ... < t_N = t^*$. Equation 1 can be approximated in the interval $[t_k, t_{k+1}]$ using the following piecewise affine system:

$$\Delta\dot{x}(t) = A_k \Delta x(t) + B_k \Delta u \tag{8}$$

where $\Delta x(t) = x(t) - x_k$, $\Delta u = u(t) - u(t_k)$, $A_k = \frac{\partial F}{\partial x}|_{x_k}$ and $B_k = \frac{\partial F}{\partial u}|_{x_k}$. Notice from Eq. (6) that $A(s_{t_k}(x_0, u)) = \frac{\partial F}{\partial x}|_{x_k} = A_k$ and $B(s_{t_k}(x_0, u)) = \frac{\partial F}{\partial u}|_{x_k} = B_k$ which are simply the linearized matrices from Eq. (8). Let us approximate the system of Eq. (6) at the interval $[t_k, t_{k+1}]$ with the following series of equations:

$$\frac{d}{dt}\tilde{p}_u(t, \tau) = A_k.\tilde{p}_u(t, \tau), \tag{9}$$
$$\frac{d}{dt}\tilde{p}_{x_0} = A_k.\tilde{p}_{x_0}(t),$$

The initial condition for the first interval is $\tilde{p}_u(t_0) = B_0$, $\tilde{p}_{x_0}(t_0) = \mathbf{I}_{n\times n}$ and the initial conditions for the other intervals are taken on the trajectory. As a result, there are two errors. The One Step Error (OSE) which is caused by approximating the linear time varying system of Eq. (6) with the LTI system of Eq. (9) on the interval $[t_k, t_{k+1}]$ assuming that initial conditions are the same. The other error is the Accumulated Error (AE) which is accumulated over time because the initial conditions actually mismatch.

OSE: Consider $p(t) = [p_{x_0}(t), p_u(t)]^T$ and $\tilde{p}(t) = [\tilde{p}_{x_0}(t), \tilde{p}_u(t)]^T$. Let us define an error variable $e_{OSE} = p(t) - \tilde{p}(t)$ whose dynamics are:

$$\dot{e}_{OSE}(t) = A(s_t(x_0, u)).p(t) - A_k.\tilde{p}(t)$$
$$e_{OSE}(0) = 0 \tag{10}$$

We define $r_k(t) = (A(s_t(x_0, u)) - A_k)p(t)$. As a result:

$$\dot{e}_{OSE}(t) = A_k e_{OSE}(t) + r_k(t) \qquad e_{OSE}(0) = 0 \tag{11}$$

Without loss of generality, let us consider the error at the interval $[t_0, t_1]$ and define $\Delta t_k = t_{k+1} - t_k$ and $d_k = max_{t\in[t_k,t_{k+1}]}\|r_k(t)\|$, the following theorem establishes an upper bound on $\|e_{OSE}\|$ for all $t \in [t_0, t_1]$.

Theorem III.1. *For e_{OSE} defined in (5), $\forall t \in [t_0, t_1]$: $\|e_{OSE}\| \leq \psi_0(A_0, \Delta t_0)d_0\Delta t_0$; where $\psi_0(A, t) = \sup_{s\in[0,t]} \|e^{At}\|$.*

AE: This is the error that accumulates over time. Since at the start of each interval $[t_k, t_{k+1}]$, $\tilde{p}(t_k)$ is not exactly equal to $p(t_k)$, consequently $e_{AE}(t_k) = p(t_k) - \tilde{p}(t_k) \neq 0$. As a result, the dynamics for e_{AE} at this interval is like those of e_{OSE} with a different initial condition:

$$\dot{e}_{AE}(t) = A_k e_{AE}(t) + r_k(t) \quad \|e_{AE}(t_k)\| \leq \epsilon_k \tag{12}$$

The following theorem establishes an upper bound on $\|e_{AE}\|$ for all $t \in [t_k, t_{k+1}]$.

Theorem III.2. *For e_{AE} defined in Eq. (12) $\forall t \in [t_k, t_{k+1}]$: $\|e_{AE}\| \leq \psi_0(A_k, \Delta t_k).(d_k\Delta t_k + \epsilon_k)$.*

Proofs of Theorems (III-B) and (III.2) are similar to the proofs in [19].

Using the above theorem, we know that there is an upper bound for $\|p(t) - \tilde{p}(t)\|$. Using appropriate time steps h_k one can decrease the error. As a result, we can use the system's linearized matrices to approximate sensitivity and calculate the descent direction approximately by replacing p_{x_0}, p_u with $\tilde{p}_{x_0}$, $\tilde{p}_u$ in Eq. (4) and (5). This is important because in many cases, we work with black-box systems and while there is no analytical information about the function F, there exist powerful tools to extract linearization matrices at operating points. One of these tools is the "`linearize`" Simulink command that provides linearized system matrices at operating points.

Remark. *Although accurately calculating the upper bound for the error is not possible (since d_k depends on F which is assumed to be unavailable), we can change Δt_k adaptively to achieve the desired accuracy. That is, if we cannot descend using the AD, then we increase the number of samples to decrease Δt_k and consequently decrease the error.*

To see how this method works we implemented the system of Eq. (7) in Simulink (shown in Fig. 1). For AD, the only information that was used was the linearized matrices returned by the Simulink "`linearize`" command on 10 samples taken on the trajectory. The final input signals are shown in Fig. 2 where we used piecewise constant signals as the basis functions for the inputs. As shown in Fig. 3, the robustness valuation has been reduced using AD from 1.2042 to 0.1976. Note that using 10 samples on the trajectory the AD directions stay close to the GD directions, and the more samples we use the better approximation we achieve.

C. Global Optimization: Simulated Annealing combined with gradient descent (SA+GD)

Simulated Annealing (SA) is a stochastic global search method. Starting from an initial point and input, it chooses another random sample inside the set of initial states and input range. The method accepts all the new samples that have less robustness values. Samples corresponding to greater robustness values can also be accepted in SA with respect to a probability function whose value decrease over time. More information about this method can be found in [20]. This method was successfully used for falsification in [9]. There are many different ways to combine SA with GD. One possibility is that each time we get a sample using SA, we descend from that sample for a few iterations or until we reach a point that seems to be a local minimizer. Then, with respect to that point, we sample a new point using SA. Another possible approach to combine these two methods is to use SA until it fails to find a better point for some number of iterations and, then, the falsification procedure switches to GD to descend from that SA sample. Throughout this paper, when we talk about SA+GD we use the first strategy which is described in Alg. 1. Notice that while using the function

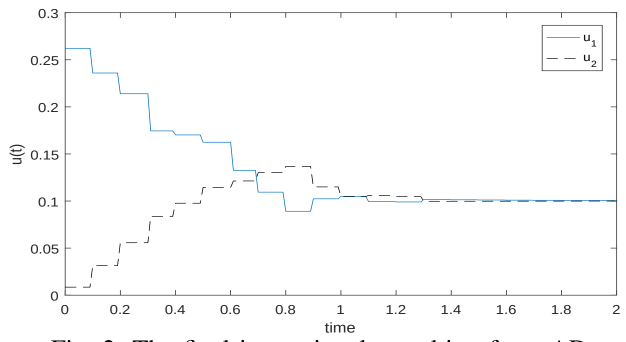

Fig. 2: The final input signals resulting from AD

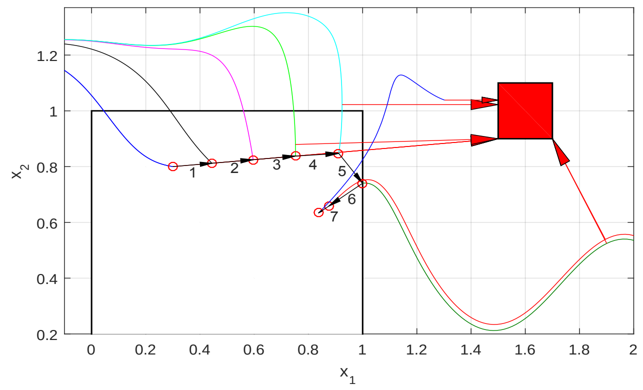

Fig. 3: AD for the System of Eq. (7): the red (gray) box is $\mathcal{U}$; and the white box is X_0; red (gray) arrows show the shortest distance from the trajectories to $\mathcal{U}$; and black arrows point to the next initial state taken by GD.

APPLY_DESCENT at step 11, if structural information about F is not available, we can use AD instead of the analytically computed descent. This step is the only step that requires structural information about F. In all other steps, we only require simulations of the system trajectories.

Remark. *For the falsification problem, an optimization method that can falsify the problem with fewer tests/simulations is preferable. The effectiveness of each algorithm in solving the falsification problem with fewer iterations and smaller robustness value depends on the problem. If the problem is easy to falsify, then the simpler the algorithm the better since even a few random samples may solve the problem without requiring descent direction computation. But, if the system is large scale or if the problem is hard to falsify, then we should consider more efficient methods like SA+GD.*

Figure 4 represents a sample execution of Alg. 1 with $k_1 = 5$ on the benchmark NONLIN2D(1) which is the autonomous (without input) version of Eq. (7) with $X_0 = [-1, 1]^2$ and $\mathcal{U} = [1.4, 1.8] \times [1, 1.4]$. As shown in Fig 4, while GD helps in decreasing the robustness levels locally, SA converges to a global minimum. Starting from a random point (A) chosen by SA, Alg. (1) takes $k_1 = 5$ GD steps to reach the point (B); then switches to SA and samples another point (C). This process continues until the total number of iterations becomes equal to N or the problem is falsified.

Algorithm 1 SA+GD algorithm where we descend from all samples for finding the minimum robustness valuation

Input: System $\Sigma : \dot{x} = F(x, u, t)$, set of initial states X_0, input range U, system specification φ, final time T, total number of iterations N, step size h, maximum number of iterations to descend from SA samples, k_1 and maximum number of iterations we decrease step size k_2.

Output: optimal initial condition x_0^*, optimal input u^* and the related optimal robustness value r^*.

```
 1: $i \leftarrow 1$ and $(x_0, u) \leftarrow$ GET_INITIAL_SAMPLE$(\Sigma, X_0)$.
 2: $r^* \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0, u, \Sigma, T, \varphi)$.
 3: while $i \leq N$ do
 4:     $(x_0', u') \leftarrow$ GET_NEW_SAMPLE$(x_0, u, \Sigma, X_0)$.
 5:     $r \leftarrow$ SIMULATE_FIND_ROBUSTVAL$(x_0, u, \Sigma, T, \varphi)$.
 6:     $Bool \leftarrow$ ACCEPT?$(r, BestR)$.
 7:     $i \leftarrow i + 1$
 8:     if $Bool = 1$ then
 9:         $SP \leftarrow (X_0, U_0, \varphi, k_1, k_2, h, F)$
10:         $(x_0, u, r, i') \leftarrow$ APPLY_DESCENT$(x_0', u', SP)$
11:         $i \leftarrow i + i'$
12:         if $r \leq r^*$ then
13:             $(x_0^*, u^*) \leftarrow (x_0, u)$ and $r^* \leftarrow r$
14:         end if
15:     end if
16: end while
```

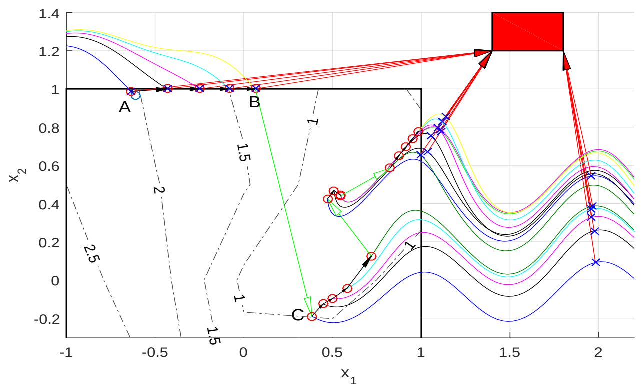

Fig. 4: SA+GD for NONLIN2D(1): the red (gray) box is $\mathcal{U}$, the white box is X_0, red (solid gray) arrows show the shortest distance from the trajectories to $\mathcal{U}$, black arrows point to the next samples taken by GD and green (light gray) arrows show the descent direction by SA.

IV. Experimental Results

For the purpose of implementation and comparison, MATLAB 2015b has been used. The processor was an Intel(R) Core(TM) i7-4790 CPU @3.6 GHZ with 16 GB memory and the operating system was Windows Server 2012 R2 Standard. The following Matlab toolboxes are necessary for simulation and GD computation: Simulink, S-TaLiRo [21], SUNDIALS [22], MATISSE, and MPT [23].

A. Approximate Descent

As mentioned earlier by increasing the number of samples taken on the trajectory, AD directions can get closer to the analytically computed ones. However, since taking more samples on the trajectory increases the complexity, we

TABLE I: AD becomes more accurate by taking more samples on the trajectory

number of samples	5	10	20	50	100
mean(α)	3.347	1.477	0.7329	0.3069	0.1667

should keep a balance between the number of samples and the required accuracy. Generally, AD directions[1] are good enough if they satisfy Eq. (3).

To study the effect of the number of samples taken on the trajectory, we designed the following experiment: For the system NONLIN2D(1), we took 100 random points and calculated the analytical descent directions and AD directions with respect to these points using different number of samples. The result is shown in Table I where the mean value has been calculated for the parameter $\alpha = ||dx - dx^l||/||dx||$, where dx is the actual descent direction and dx^l is the AD direction. Clearly, by increasing the number of samples, dx^l gets closer to dx.

B. *Comparison of SA with SA+GD*

To examine how much GD can help improve SA for the falsification problem, we studied the experimental performance of the two methods. In summary, if the problem is easy to falsify, using GD will probably decrease the speed of falsification, in general. In fact, if finding a trajectory that can falsify the problem is easy, then GD is too conservative for the procedure. But if the problem is hard to falsify, GD will help SA in falsification as long as we strike a balance between the number of samples for SA (globality) and the number of times we descend using GD (local search).

For the experiments, we utilized the following strategies:

1) Comparison on achieved falsification: Continue the simulation until the problem is falsified (robustness value zero) and study the statistics on the number of tests/simulations required for falsification.
2) Comparison using an equal number of total simulations: Use the same number of tests/simulations for both methods and compute statistics on the minimum robustness values of pure SA and SA+GD to see which method has less mean robustness value.

We use the second strategy whenever the system simulation is very time consuming (because of the complexity of the model) or achieving falsification is very hard and requires a large number of simulations. Different parameters can affect the statistical properties of the algorithms:

1) System dynamics: it is usually harder to deal with nonlinear systems. We achieve better results for SA+GD as the system complexity increases.
2) How hard the problem is to falsify: for the same system dynamics the location of the initial conditions and the unsafe set can change the computational hardness of the problem. We require more complex methods for harder problems, so we experimentally confirmed that SA+GD works better for hard problems.
3) System dimension: the number of choices for SA grows exponentially with the system dimension. We expect that moving toward a descent direction that will be guaranteed to decrease the robustness regardless of dimensions might help to falsify the problem faster.
4) Number of times we descend from each SA sample (parameter k_1 in Alg. 1): experimental results showed that using a large number of GD for each sample is not helpful. In fact, this number should be chosen such that a balance is kept between globality of SA and local search of GD.
5) Step size h in Alg. 1: smaller step sizes reduce the speed of convergence and large step sizes might end up passing some smaller valued level sets.
6) Number of times we decrease the step size h (parameter k_2 in Alg. 1): if we could not find a better robustness value, then we need to decrease the step size and try again the same descent direction.

[1]We remark that multiple directions can satisfy Eq. (3) and that, in practice we would be interested in the steepest descent.

TABLE II: Comparison between number of simulations necessary for achieving falsification using SA+GD as opposed to SA for NONLIN2D(1).

	mean	N-mean	min	max
SA	210.87	2−N-mean(SA+GD)	5	1303
SA+GD ($k_1 = 2$)	129.57	0.7612	8	816
SA+GD ($k_1 = 3$)	85.55	0.5772	4	379
SA+GD ($k_1 = 5$)	86.2	0.5803	5	384
SA+GD ($k_1 = 10$)	138.03	0.7912	6	680

All the properties are problem dependent, except the last 3 ones which can be tuned to provide a good performance.

For each experiment, we collected statistics over 100 runs to examine the usefulness of combining SA and GD for 6 different autonomous and non-autonomous benchmarks which can be found in S-TaLiRo. In all the experiments, the total number of tests is 100 and the initial step size is $h = (0.1, 0.1)$. In tables II to IV, the comparison is based on the number of necessary simulations for achieved falsification. In order to study the effect of the number of descents per sample, we studied the statistics of these three benchmarks using different k_1 values. For NONLIN2D(1) in Table II, the best results are related to $k_1 = 3$. For NONLIN2D(2) in Table III, $k_1 = 5$ is the best parameter value. For TCP in Table IV, the best k_1 value is 20. In all the tables you can see that using very small and very large values for k_1 can decrease the effectiveness of the method. In the last three tables, the comparison is based on the equal total number of simulations. As it is evident from the tables, in all the benchmarks the mean, maximum and minimum robustness values are better for SA+GD than pure SA. Also notice that based on Table VI, SA+GD has been able to falsify the problem in some of the tests while SA has not.

In the presented results, the smaller normalized mean (*N-mean*) values show that SA+GD is more effective for that problem. Consider that *N-mean(method)= mean(method)/(mean(SA)+mean(SA+GD))*.

V. Conclusion

We presented a method for approximating descent directions towards a set without requiring structural information about the system model. This is important when working

TABLE III: Comparison between number of simulations necessary for achieving falsification using SA+GD as opposed to SA for NONLIN2D(2) (a linear parameter varying autonomous 2D system).

	mean	N-mean	min	max
SA	1584.3	2−N-mean(SA+GD)	10	7615
SA+GD ($k_1 = 3$)	1306.4	0.9039	35	9567
SA+GD ($k_1 = 5$)	883.24	0.7159	51	3457
SA+GD ($k_1 = 10$)	960.94	0.7551	15	5250

TABLE IV: Comparison between number of simulations necessary for achieving falsification using SA+GD as opposed to SA for TCP (a linear autonomous 4 dimensional system).

	mean	N-mean	min	max
SA	2662	2−N-mean(SA+GD)	135	9824
SA+GD($k_1 = 5$)	894.97	0.5031	152	2528
SA+GD($k_1 = 10$)	462.22	0.2959	184	920
SA+GD($k_1 = 20$)	455.45	0.2922	170	660
SA+GD($k_1 = 50$)	456.81	0.2929	198	702

TABLE V: Comparison between robustness values after 100 simulations computed using SA+GD as opposed to SA with $k_1 = 5$ and equal number of total simulations for VFLuminescence (a nonlinear autonomous 9D system).

	mean	N-mean	min	max
SA	6453.9	1.198	2143.3	14302.6
SA+GD	4322.8	0.802	102.83	25321.1

TABLE VI: Comparison between robustness values computed after 100 simulations using SA+GD as opposed to SA with $k_1 = 5$ and equal number of total simulations for RLC (a linear non-autonomous 81D system).

	mean	N-mean	min	max
SA	1.5440	1.1222	0.4431	2.5035
SA+GD	1.2077	0.8778	-0.0104	2.1435

TABLE VII: Comparison between robustness values computed after 100 simulations using SA+GD as opposed to SA with $k_1 = 5$ and equal number of total simulations for Insulin (a highly nonlinear non-autonomous 6D system).

	mean	N-mean	min	max
SA	3.442	1.19	1.8411	5.6373
SA+GD	2.3430	0.81	1.6508	3.4994

with complex or black box models where linearizations are available. We combine Gradient Descent (GD) (or equivalently its approximation) and Simulated Annealing (SA) optimization methods to get the advantages of both methods. With SA we maintain the global search capabilities, while with GD we find local minima faster. The experimental results showed that in general the hybrid method (SA+GD) can be very helpful, especially, if we are working with high dimensional systems or systems that are hard to falsify.

REFERENCES

[1] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel, "Verification of automotive control applications using s-taliro," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 3567–3572.

[2] A. Donzé, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui, "Robustness analysis and behavior discrimination in enzymatic reaction networks," *PloS one*, vol. 6, no. 9, p. e24246, 2011.

[3] A. Jones, U. Ali, and M. Egerstedt, "Optimal pesticide scheduling in precision agriculture," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–8.

[4] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[5] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[6] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Metric interval temporal logic specification elicitation and debugging," in *Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on*. IEEE, 2015, pp. 70–79.

[7] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems*, vol. 36, no. 6, pp. 45–64, 2016.

[8] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of ltl safety properties in hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2009, pp. 368–382.

[9] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, p. 95, 2013.

[10] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski, "Multiple shooting, cegar-based falsification for hybrid systems," in *Proceedings of the 14th International Conference on Embedded Software*. ACM, 2014, p. 5.

[11] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, J. Kapinski, and X. Jin, "Falsification of safety properties for closed loop control systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 299–300.

[12] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 254–257.

[13] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.

[14] P. S. Duggirala and M. Viswanathan, "Parsimonious, simulation based verification of linear systems," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 477–494.

[15] H. Abbas and G. Fainekos, "Computing descent direction of mtl robustness for non-linear systems," in *2013 American Control Conference*. IEEE, 2013, pp. 4405–4410.

[16] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius, "Functional gradient descent method for metric temporal logic specifications," in *2014 American Control Conference*. IEEE, 2014, pp. 2312–2317.

[17] H. Abbas and G. Fainekos, "Linear hybrid system falsification through local search," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2011, pp. 503–510.

[18] A. K. Winn and A. A. Julius, "Optimization of human generated trajectories for safety controller synthesis," in *2013 American Control Conference*. IEEE, 2013, pp. 4374–4379.

[19] Z. Han and B. H. Krogh, "Reachability analysis of nonlinear systems using trajectory piecewise linearized models," in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.

[20] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of statistical physics*, vol. 34, no. 5-6, pp. 975–986, 1984.

[21] S-TaLiRo: Temporal Logic Falsification Of Cyber-Physical Systems, https://sites.google.com/a/asu.edu/s-taliro/s-taliro, 2013, [Online; accessed April-2014].

[22] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "Sundials: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.

[23] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, "Multi-parametric toolbox (mpt)," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2004, pp. 448–462.