Fast Falsification of Hybrid Systems Using Probabilistically Adaptive Input

Gidon Ernst[1(✉)], Sean Sedwards[2], Zhenya Zhang[3], and Ichiro Hasuo[3]

[1] Ludwig-Maximilians-University, Munich, Germany
gidon.ernst@lmu.de
[2] University of Waterloo, Waterloo, Canada
sean.sedwards@uwaterloo.ca
[3] National Institute of Informatics, Tokyo, Japan
{zhangzy,hasuo}@nii.ac.jp

Abstract. We present an algorithm that quickly finds falsifying inputs for hybrid systems, i.e., inputs that steer the system towards violation of a given temporal logic requirement. Our method is based on a probabilistically directed search of an increasingly fine grained spatial and temporal discretization of the input space. A key feature is that it adapts to the difficulty of a problem at hand, specifically to the local complexity of each input segment, as needed for falsification. In experiments with standard benchmarks, our approach shows comparable or better performance to existing techniques, while at the same time being relatively simple.

Keywords: Cyber-physical system · Falsification · Stochastic optimization · Temporal logic · Quantitative semantics · Las Vegas Tree Search

1 Introduction

The falsification problem we consider seeks a (time-bounded) input signal that causes a hybrid system model to violate a given temporal logic specification. A popular way to address this is to first construct a "score function" that quantifies how much of the specification has been satisfied during the course of an execution. The falsification can then be treated as an optimization problem, which can be solved using standard algorithms. This approach, especially using a quantitative "robustness" semantics [15] of requirements as the score function, has been successfully applied, resulting in a number of now mature tools [4,10] with practical applications as well as and friendly competitions [8,9,28].

Despite its apparent success, robustness is in general not a perfect optimization function, but only a heuristic score function [20] with respect to the falsification problem, as greedy hill climbing may lead to local optima. In practice, standard optimization algorithms overcome this limitation by including stochastic exploration. The most sophisticated of these can also model the dynamics of the system (e.g., [2]), in order to estimate the most productive direction of input signal space to explore. There is, however, "no free lunch" [27], and high performance general purpose optimization algorithms are not necessarily the best choice. For example, such algorithms often optimize with respect to the entire

D. Parker and V. Wolf (Eds.): QEST 2019, LNCS 11785, pp. 165–181, 2019.
https://doi.org/10.1007/978-3-030-30281-8_10

input trace, without exploiting the time causality of the problem, i.e., the fact that a good trace (one that eventually falsifies the property) may be dependent on a good trace prefix.

The contribution of this paper is a randomized falsification algorithm (Sect. 3.1) that exploits the time-causal structure of the problem and that adapts to local complexity. In common with alternative approaches, our algorithm searches a discretized space of input signals, but in our case the search space also includes multiple levels of spatial and temporal granularity (Sect. 3.2). The additional complexity is mitigated by an efficient tree search that probabilistically balances exploration and exploitation (Sect. 3.3).

The performance of our algorithm benefits from the heuristic idea to explore simple (coarse granularity) inputs first, then gradually switch to more complex inputs that include finer granularity. Importantly, the finer granularity tends only to be added where it is needed, thus avoiding the exponential penalty of searching the entire input space at the finer granularity. While it is always possible to construct pathological problem instances, we find that despite its simplicity, our approach is effective on benchmarks from the literature. Our experimental results (Sect. 4) demonstrate that our algorithm can achieve comparable or better performance than other methods, in terms of speed and reliability of finding a falsifying input.

2 Preliminaries

In this work we represent a deterministic black-box system model as an input/output function $\mathcal{M}\colon ([0,T] \to \mathbb{R}^n) \to ([0,T] \to \mathbb{R}^m)$. In general, $\mathcal{M}$ comprises continuous dynamics with discontinuities. $\mathcal{M}$ takes a time-bounded, real-valued input signal $\boldsymbol{u}\colon [0,T] \to \mathbb{R}^n$ of length $|\boldsymbol{u}| = T$ and transforms it to a time bounded output signal $\boldsymbol{y}\colon [0,T] \to \mathbb{R}^m$ of the same length, but potentially different dimensionality. The dimension n of the input indicates that at each moment $t \leq T$ within the time horizon T, the value $\boldsymbol{u}(t) \in \mathbb{R}^n$ of the input is an n-dimensional real vector (analogously for the output).

We denote by $\boldsymbol{u}_1\boldsymbol{u}_2\colon [0, T_1+T_2] \to \mathbb{R}^n$ the concatenation of signals $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ that have the same dimensions. Concatenation of more than two signals follows naturally and is denoted $\boldsymbol{u}_1\boldsymbol{u}_2\boldsymbol{u}_3\cdots$. A constant input signal segment is written (t, v), where t is a time duration and $v \in \mathbb{R}^n$ is a vector of input values. A piecewise constant input signal is the concatenation of such segments.

In this work we adopt the syntax and robustness semantics of STL defined in [12]. The syntax of an STL formula is thus given by

$$\varphi \ ::= \ \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathsf{U}_I\, \varphi \mid \Box_I \varphi \mid \Diamond_I \varphi \mid \mu, \tag{1}$$

where the logical connectives and temporal operators have their usual Boolean interpretations and equivalences, I is the interval of time over which the temporal operators range, and atomic formulas $\mu \equiv f(x_1, \ldots, x_m) > 0$ are predicates over the spatial dimensions of a trace. The robustness of trace $\boldsymbol{y}$ with respect to formula φ, denoted $\rho(\varphi, \boldsymbol{y})$, is calculated inductively according to the following robustness semantics, using the equivalence $\rho(\varphi, \boldsymbol{y}) \equiv \rho(\varphi, \boldsymbol{y}, 0)$.

$$\rho(\mu, \boldsymbol{y}, t) = f(x_1[t], \ldots, x_m[t]), \qquad \text{for } \mu \equiv f(x_1, \ldots, x_m) > 0$$
$$\rho(\neg\varphi, \boldsymbol{y}, t) = -\rho(\varphi, \boldsymbol{y}, t)$$
$$\rho(\varphi_1 \vee \varphi_2, \boldsymbol{y}, t) = \max(\rho(\varphi_1, \boldsymbol{y}, t), \rho(\varphi_2, \boldsymbol{y}, t)) \qquad \rho(\Diamond_I \varphi, \boldsymbol{y}, t) = \max_{t' \in t+I} (\rho(\varphi, \boldsymbol{y}, t'))$$
$$\rho(\varphi_1 \wedge \varphi_2, \boldsymbol{y}, t) = \min(\rho(\varphi_1, \boldsymbol{y}, t), \rho(\varphi_2, \boldsymbol{y}, t)) \qquad \rho(\Box_I \varphi, \boldsymbol{y}, t) = \min_{t' \in t+I} (\rho(\varphi, \boldsymbol{y}, t'))$$
$$\rho(\varphi_1 \mathsf{U}_I \varphi_2, \boldsymbol{y}, t) = \max_{t' \in t+I} \Big(\min_{t'' \in [t, t')} (\rho(\varphi_1, \boldsymbol{y}, t'')), \min(\rho(\varphi_2, \boldsymbol{y}, t')) \Big)$$

An important characteristic of the robustness semantics is that it is faithful to standard boolean satisfaction, such that

$$\rho(\varphi, \boldsymbol{y}) > 0 \implies \boldsymbol{y} \models \varphi \quad \text{and} \quad \rho(\varphi, \boldsymbol{y}) < 0 \implies \boldsymbol{y} \not\models \varphi. \tag{2}$$

Together, these equations justify using the robustness semantics $\rho(\varphi, \mathcal{M}(\boldsymbol{u}))$ to detect whether an input $\boldsymbol{u}$ corresponds to the violation of a requirement φ. This correspondence is exploited to find such falsifying inputs through global hill-climbing optimization:

$$\text{Find } \boldsymbol{u}^* = \underset{\boldsymbol{u} \in ([0,T] \to \mathbb{R}^n)}{\arg\min} \rho(\varphi, \mathcal{M}(\boldsymbol{u})) \text{ such that } \rho(\varphi, \mathcal{M}(\boldsymbol{u}^*)) < 0. \tag{3}$$

Of course, finding an adequate falsifying input $\boldsymbol{u}^*$ is generally hard and subject to the limitations of the specific optimization algorithm used.

Sound approximations of the lower and upper bounds of the robustness of a prefix $\boldsymbol{y}$ can sometimes be used to short-cut the search. We thus define lower and upper bounds in the following way.

$$\text{Lower: } \underline{\rho}(\varphi, \boldsymbol{y}) = \min_{\boldsymbol{y}'} \rho(\varphi, \boldsymbol{y}\boldsymbol{y}') \qquad \text{Upper: } \overline{\rho}(\varphi, \boldsymbol{y}) = \max_{\boldsymbol{y}'} \rho(\varphi, \boldsymbol{y}\boldsymbol{y}') \tag{4}$$

A lower bound $\underline{\rho}(\varphi, \mathcal{M}(\boldsymbol{u}))) > 0$ can be used to detect that a prefix cannot be extended to a falsifying trace (e.g., after the deadline for a harmful event has passed). An upper bound $\overline{\rho}(\varphi, \mathcal{M}(\boldsymbol{u})) < 0$ similarly implies $\mathcal{M}(\boldsymbol{u}\boldsymbol{u}') \not\models \varphi$ for all $\boldsymbol{u}'$, concluding that input $\boldsymbol{u}$ is already a witness for falsification (e.g., a limit is already exceeded). Robustness can be computed efficiently [11], as well as the respective upper and lower bounds [13].

3 Approach

We wish to solve the following falsification problem efficiently:

$$\text{Find } \boldsymbol{u}^* \text{ such that } \overline{\rho}(\varphi, \mathcal{M}(\boldsymbol{u}^*)) < 0. \tag{5}$$

Our approach is to repeatedly construct input signals $\boldsymbol{u} = \boldsymbol{u}_1 \boldsymbol{u}_2 \boldsymbol{u}_3 \cdots$, where $\boldsymbol{u}_i$ is drawn from a predetermined search space of candidate input segments, $\mathcal{A}$. The choice is probabilistic, according to a distribution $\mathcal{D}$ that determines the search strategy, i.e., which inputs are likely to be tried next given a partially explored search space. The construction of each input is done incrementally, to take advantage of the potential short cuts described at the end of Sect. 2.

Algorithm 1 "adaptive Las Vegas Tree Search" (aLVTS) codifies the high level functionality of this probabilistic approach, described in detail in Sect. 3.1.

The effectiveness of our algorithm in practice comes from the particular choices of $\mathcal{A}$ and $\mathcal{D}$, which let the search gradually *adapt* to the difficulty of the problem. The set $\mathcal{A}$ (defined in Sect. 3.2) contains input segments of diverse granularity, which intuitively corresponds to how precise the input must be in order to find a falsifying trace. The distribution $\mathcal{D}$ (defined in Sect. 3.3) initially assigns high probabilities to the "coarsest" input segments in $\mathcal{A}$. Coarse here means that the segments tend to be long in relation to the time horizon T and large in relation to the extrema of the input space. The algorithm probabilistically balances exploration and exploitation of segments, but as the coarser segments become fully explored, and the property has not been falsified, the algorithm gradually switches to finer-grained segments.

3.1 Algorithm

Algorithm 1 searches the space of input signals constructed from piecewise constant (over time) segments, which are chosen at random according to the distribution defined by $\mathcal{D}$ in line 6. This distribution is a function of the numbers of unexplored and explored edges at different levels of granularity, and thus defines the probabilities of exploration, exploitation and adaptation. The precise calculation made by $\mathcal{D}$ is described in Sect. 3.3.

As the search proceeds, the algorithm constructs a tree whose nodes each correspond to a unique input signal prefix. The edges of the tree correspond to the constant segments that make up the input signal. The root node corresponds to time 0 and the empty input signal (line 4).

To each node identified by an input signal prefix $\boldsymbol{u}$ is associated a set of unexplored edges, $\mathit{unexplored}(\boldsymbol{u}) \subseteq \mathcal{A}$, that correspond to unexplored input signal segments, and a set of explored edges, $\mathit{explored}(\boldsymbol{u}) \subseteq \mathcal{A}$, that remain inconclusive with respect to falsification. Initially, all edges are unexplored (line 1 and line 2). Once an edge has been chosen (line 6), the unique signal segment associated to the edge may be appended to the signal prefix associated to the node, to form an extended input signal. If the chosen edge is unexplored, it is removed from the set of unexplored edges (line 8) and the extended input signal $\boldsymbol{u}\boldsymbol{u}'$ is transformed by the system into an extended output signal (line 9). If the requirement is falsified by the output signal ($\boldsymbol{y}$ in line 10), the algorithm immediately quits and returns the falsifying input signal (line 11). If the requirement is satisfied, with no possibility of it being falsified by further extensions of the signal (12), the algorithm quits the current signal (line 13) and starts a new signal from the root node (line 4). This is the case, in particular, when the length of the signal exceeds the time horizon of the formula as a consequence of the definition of $\underline{\rho}$ in (4). If the requirement is neither falsified nor satisfied, the edge is added to the node's set of explored edges (line 14). Regardless of whether the chosen edge was previously explored or unexplored, if the signal remains inconclusive, the extended input signal becomes the focus of the next iterative step (line 15).

Algorithm 1: Adaptive Las Vegas Tree Search (aLVTS)

Input:
system model $\mathcal{M} : \boldsymbol{u} \to \boldsymbol{y}$, with $\boldsymbol{u} : [0, t] \to \mathbb{R}^n$ and $\boldsymbol{y} : [0, t] \to \mathbb{R}^m$,
time-bounded specification ϕ, set of all possible input trace segments $\mathcal{A}$

Output:
$\boldsymbol{u}$ such that $\mathcal{M}(\boldsymbol{u}\boldsymbol{u}') \not\models \phi$ for all $\boldsymbol{u}'$, or $\perp$ after timeout or maximum iterations

1 $\mathit{unexplored}(\boldsymbol{u}) \leftarrow \mathcal{A}$ for all $\boldsymbol{u}$
2 $\mathit{explored}(\boldsymbol{u}) \leftarrow \varnothing$ for all $\boldsymbol{u}$
3 **repeat**
4 $\boldsymbol{u} \leftarrow \langle\rangle$
5 **while** $\mathit{unexplored}(\boldsymbol{u}) \neq \varnothing \vee \mathit{explored}(\boldsymbol{u}) \neq \varnothing$ **do**
6 sample $\boldsymbol{u}' \sim \mathcal{D}(\boldsymbol{u})$
7 **if** $\boldsymbol{u}' \in \mathit{unexplored}(\boldsymbol{u})$ **then**
8 $\mathit{unexplored}(\boldsymbol{u}) \leftarrow \mathit{unexplored}(\boldsymbol{u}) \setminus \{\boldsymbol{u}'\}$
9 $\boldsymbol{y} \leftarrow \mathcal{M}(\boldsymbol{u}\boldsymbol{u}')$
10 **if** $\overline{\rho}(\phi, \boldsymbol{y}) < 0$ **then**
11 **return** $\boldsymbol{u}\boldsymbol{u}'$
12 **if** $\underline{\rho}(\phi, \boldsymbol{y}) > 0$ **then**
13 **continue** line 3
14 $\mathit{explored}(\boldsymbol{u}) \leftarrow \mathit{explored}(\boldsymbol{u}) \cup \{\boldsymbol{u}'\}$
15 $\boldsymbol{u} \leftarrow \boldsymbol{u}\boldsymbol{u}'$
16 **until** *timeout or maximum number of iterations*;
17 **return** $\perp$

While not explicit in the presentation of Algorithm 1, our approach is deliberately incremental in the evaluation of the system model. In particular, we can re-use partial simulations to take advantage of the fact that traces share common prefixes. Hence, for example, one can associate to every visited $\boldsymbol{u}$ the terminal state of the simulation that reached it, using this state to initialize a new simulation when subsequently exploring $\boldsymbol{u}\boldsymbol{u}'$. This idea also works for the calculation of robustness. We note, however, that incremental simulations may be impractical. For example, suspending and re-starting Simulink can be more expensive than performing an entire simulation from the start.

3.2 Definition of $\mathcal{A}$

The set $\mathcal{A}$ contains constant, n-dimensional input signal segments $\boldsymbol{u}'$ with values $(v_1, \ldots, v_n) \in \mathbb{R}^n$ and time duration t. Let $\underline{v}_i$ and $\overline{v}_i$ denote the minimum and maximum possible values, respectively, of dimension $i \in \{1, \ldots, n\}$. For each integer level $l \in \{0, \ldots, l_{\max}\}$, we define the set of possible proportions of the interval $[\underline{v}_i, \overline{v}_i]$ as

$$\mathbf{p}_l = \{(2j+1)/2^l \mid j \in \mathbb{N}_0 \leq (2^l - 1)/2\}.$$

The numerators of all elements are coprime with the denominator, 2^l, hence $\mathbf{p}_i \cap \mathbf{p}_j = \varnothing$ for all $i \neq j$. By definition, $\mathbf{p}_0$ also includes 0. Hence, $\mathbf{p}_0 = \{0, 1\}$, $\mathbf{p}_1 = \left\{\frac{1}{2}\right\}$ and $\mathbf{p}_2 = \left\{\frac{1}{4}, \frac{3}{4}\right\}$, etc. The set of possible values of dimension i at level l is thus given by

$$\mathbf{v}_{i,l} = \underline{v}_i + \mathbf{p}_l \times (\overline{v}_i - \underline{v}_i).$$

Rather than making the granularity of each dimension independent, we interpret the value of l as a granularity "budget" that must be distributed among the (non-temporal) dimensions of the input signal. The set of possible per-dimension budget allocations for level l is given by

$$\mathbf{b}_l = \{(b_1, ..., b_n) \in \mathbb{N}_0^n \mid b_1 + \cdots + b_n = l\}.$$

For example, with $n = 2$, $\mathbf{b}_3 = \{(0, 3), (1, 2), (2, 1), (3, 0)\}$. If we denote the set of possible time durations at level l by $\mathbf{t}_l$, then the set of possible input segments at level l is given by

$$\mathcal{A}_l = \bigcup_{(b_1,\ldots,b_n)\in\mathbf{b}_l} \mathbf{t}_l \times \mathbf{v}_{1,b_1} \times \cdots \times \mathbf{v}_{n,b_n}.$$

Note that while $\mathbf{t}_l$ is not required here to share the granularity budget, this remains a possibility. Our implementation actually specifies $\mathbf{t}_l$ by defining a fixed number of control points per level, $(k_0, \ldots, k_{l_{\max}})$, such that the $\mathbf{t}_l = \{T/k_l\}$ are singleton sets. The sizes of various $\mathcal{A}_l$ for different choices of n and l, assuming $|\mathbf{t}_l| = 1$, are given in Table 1.

Table 1. Size of $\mathcal{A}_l$ for input dimensionality n and level l given that $|\mathbf{t}_l| = 1$.

n	$l = 0$	1	2	3	4	5	6	7	8	9	10
2	4	4	9	20	44	96	208	448	960	2048	4352
3	8	12	30	73	174	408	944	2160	4896	11008	24576

In summary, an input segment $\boldsymbol{u}' = (t, v_1, \ldots, v_n) \in \mathcal{A}_l$ has $t \in \mathbf{t}_l$ and corresponding budget allocation $b_1 + \cdots + b_n = l$, with the value v_i for each dimension given by $v_i = \underline{v}_i + p_i(\overline{v}_i - \underline{v}_i)$, where $p_i = (2j_i + 1)/2^{b_i}$, for some j_i, defines the proportion between minimum $\underline{v}_i$ and maximum $\overline{v}_i$.

By construction, $\mathcal{A}_i \cap \mathcal{A}_j = \varnothing$, for all $i \neq j$. Hence, we define

$$\mathit{unexplored}_l(\boldsymbol{u}) = \mathit{unexplored}(\boldsymbol{u}) \cap \mathcal{A}_l \text{ and}$$
$$\mathit{explored}_l(\boldsymbol{u}) = \mathit{explored}(\boldsymbol{u}) \cap \mathcal{A}_l.$$

The set of all possible input signal segments is given by $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_{l_{\max}}$. Figure 1 depicts the construction of $\mathcal{A}$ for two dimensions. The majority of candidate input points is concentrated on the outer contour, corresponding to an extreme choice for one dimension and a fine-grained choice for the other dimension. While this bias appears extreme, as layers are exhausted, finer-grained choices become more likely, e.g., after two points from $\mathcal{A}_0$ have been tried, all remaining points from both levels in the second panel would be equally probable.

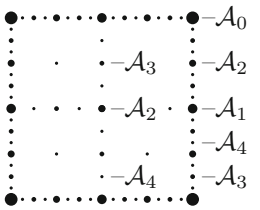


Fig. 1. Construction of $\mathcal{A}$ for $n = 2$ and $l_{\max} = 4$, with $\mathcal{A}_0$ representing the extremes of the two spatial dimensions. Showing $\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$. Larger points correspond to more likely values. Many points lie on the contour: the algorithm tends to combine finer choices in one dimension with coarse choices in the other dimension.

3.3 Definition of $\mathcal{D}$

The distribution $\mathcal{D}(\boldsymbol{u})$ is constructed implicitly. First, a granularity level $l \in \{0, \ldots, l_{\max}\}$ is chosen at random, with probability in proportion to the fraction of edges remaining at the level, multiplied by an exponentially decreasing scaling factor. Defining the overall weight of level l as

$$w_l = \frac{|\mathit{unexplored}_l(\boldsymbol{u})| + |\mathit{explored}_l(\boldsymbol{u})|}{2^l \cdot |\mathcal{A}_l|},$$

level l is chosen with probability $w_l / \sum_{i=0}^{l_{\max}} w_i$.

Having chosen l, one of the following strategies is chosen uniformly at random:

1. select $\boldsymbol{u}' \in \mathit{unexplored}_l(\boldsymbol{u})$, uniformly at random;
2. select $\boldsymbol{u}' \in \mathit{explored}_l(\boldsymbol{u})$, uniformly at random;
3. select $\boldsymbol{u}' \in \mathit{explored}_l(\boldsymbol{u})$, uniformly at random from those that minimise $\rho(\varphi, \mathcal{M}(\boldsymbol{u}\boldsymbol{u}'))$;
4. select $\boldsymbol{u}' \in \mathit{explored}_l(\boldsymbol{u})$, uniformly at random from those that minimise $\rho(\varphi, \mathcal{M}(\boldsymbol{u}\boldsymbol{u}'\boldsymbol{u}^*))$, where $\boldsymbol{u}^*$ denotes any, arbitrary length input signal suffix that has already been explored from $\boldsymbol{u}\boldsymbol{u}'$.

Strategy 1 can be considered pure exploration, while strategies 3–4 are three different sorts of exploitation. In the case that $\mathit{unexplored}_l(\boldsymbol{u})$ or $\mathit{explored}_l(\boldsymbol{u})$ are empty, their corresponding strategies are infeasible and a strategy is chosen uniformly at random from the feasible strategies. If for all $\boldsymbol{u}' \in \mathit{explored}(\boldsymbol{u})$, $\mathit{explored}(\boldsymbol{u}\boldsymbol{u}') = \varnothing$, then strategy 4 is equivalent to strategy 3, but it *is* feasible.

4 Evaluation

We evaluate our approach on a selection of standard driving-related automotive benchmarks, which are of particular interest to us and common in the falsification literature. Each benchmark has at least one time-varying input to discover (the motivation of aLVTS), in contrast to simply searching for an initial configuration. In addition to experiments conducted by us, we include some recent results from

the ARCH 2019 falsification competition [28]. Our selection is representative, but not exhaustive, and note that there are some benchmarks used in [28] where our algorithm does not work well. There are, however, benchmarks where our algorithm is significantly better than the alternatives, motivating its potential inclusion in an ensemble approach.

4.1 Benchmarks

Automatic Transmission. This benchmark was proposed in [16] and consists of a Simulink model and its related requirements. The model has two inputs: *throttle* and *brake*. Depending on the speed and engine load, an appropriate gear is automatically selected. Besides the current gear g, the model outputs the car's speed v and engine rotations ω. We consider the following requirements:

$$\begin{aligned}
&\mathrm{AT1}_t = \Box_{[0,t]}\ v < 120 \qquad \mathrm{AT2} = \Box_{[0,10]}\omega < 4750\\
&\mathrm{AT5}_i = \Box_{[0,30]}((g \neq i \wedge \circ\, g = i) \implies \circ\, \Box_{[0,2.5]} g = i)\\
&\mathrm{AT6}_{t,\overline{v},\underline{\omega}} = \big(\Box_{[0,10]}\ v < \overline{v}\big) \vee \big(\Diamond_{[0,t]}\ \underline{\omega} < \omega\big)\\
&\mathrm{ATX1}_i = \Box_{[0,30]}\ \big(g = i \implies v > 10 \cdot i\big),\ i \in \{3,4\}\\
&\mathrm{ATX2} = \neg\big(\Box_{[10,30]} v \in [50,60]\big)
\end{aligned}$$

AT* are from [16], with AT5 here subsuming AT3 and AT4 of [16]. ATX* are additional requirements with interesting characteristics. The syntax $\circ\ \phi$ denotes $\Diamond_{[0.001,0.1]}\ \phi$. Note that for falsification, AT1 and AT2 require extreme inputs, whereas AT6 and ATX2 require fine-grained inputs. The robustness scores of AT5 and ATX1 can be $\pm\infty$ and are discontinuous at gear changes.

The input signal for the benchmarks is piecewise constant with 4 control points for random sampling/Breach/S-TaLiRo, which is sufficient to falsify all requirements. We choose 6 levels, with 2, 2, 3, 3, 3, 4 control points, respectively, corresponding to a time granularity of input segment durations between 15 ($= \frac{30}{2}$, coarsest) to 7.5 ($= \frac{30}{4}$, finest).

Powertrain Control. The benchmark was proposed for hybrid systems verification and falsification in [21]. Falsification tries to detect amplitude and duration of spikes in the air-to-fuel AF ratio with respect to a reference value AF_{ref}. Those occur as a response to changes in the throttle θ. The input $\theta \in [0, 62.1)$ varies throughout the trace, whereas $\omega \in [900, 1100]$ is constant. The abbreviation $\mu = |AF - AF_{\mathrm{ref}}|/AF_{\mathrm{ref}}$ denotes the normalized deviation from the reference.

We consider requirements 27 from [21], which states that after a falling or rising edge of the throttle, μ should return to the reference value within 1 time unit and stay close to it for some time. We also consider requirement 29, which expresses an absolute error bound.

$$\mathrm{AFC27} = \Box_{[11,50]}\ (\mathit{rise} \vee \mathit{fall}) \implies \Box_{[1,5]}\ |\mu| < \beta \quad \mathrm{AFC29} = \Box_{[11,50]}|\mu| < \gamma$$

where rising and falling edges of θ are detected by $\mathit{rise} = \theta < 8.8 \wedge \Diamond_{[0,\epsilon]}\ 40.0 < \theta$ and $\mathit{fall} = 40.0 < \theta \wedge \Diamond_{[0,\epsilon]}\ \theta < 8.8$ for $\epsilon = 0.1$. The concrete bounds $\beta = 0.008$

and $\gamma = 0.007$ are chosen from [8,9,28] as a balance between difficulty of the problem and ability to find falsifying traces.

The input signal is piecewise constant with 10 control points for all falsification methods, specifically 5 levels with 10 control points each for aLVTS.

Chasing Cars. In the model from [17] five cars drive in sequence. The time varying input controls the throttle and brake of the front car (each in $[0, 1]$), while the other cars simply react to maintain a certain distance. We consider five artificial requirements from [28], where y_i is the position of the ith car. FalStar uses the same input parameterization as for the AT benchmarks.

$$\text{CC1} = \Box_{[0,100]} y_5 - y_4 \leq 40 \qquad \text{CC2} = \Box_{[0,70]} \Diamond_{[0,30]} y_5 - y_4 \geq 15$$
$$\text{CC3} = \Box_{[0,80]}((\Box_{[0,20]} y_2 - y_1 \leq 20) \vee (\Diamond_{[0,20]} y_5 - y_4 \geq 40))$$
$$\text{CC4} = \Box_{[0,65]} \Diamond_{[0,30]} \Box_{[0,20]} y_5 - y_4 \geq 8$$
$$\text{CC5} = \Box_{[0,72]} \Diamond_{[0,8]}((\Box_{[0,5]} y_2 - y_1 \geq 9) \rightarrow (\Box_{[5,20]} y_5 - y_4 \geq 9))$$

4.2 Experimental Results

We have implemented Algorithm 1 in the prototype tool FalStar, which is publicly available on github, including repeatability instructions.[1] In our own experiments, we compare the performance of aLVTS with uniform random sampling (both implemented in FalStar) and with the state-of-the-art stochastic global optimization algorithm CMA-ES [18] implemented in the falsification tool Breach.[2] The machine and software configuration was: CPU Intel i7-3770, 3.40 GHz, 8 cores, 8 Gb RAM, 64-bit Ubuntu 16.04 kernel 4.4.0, MATLAB R2018a, Scala 2.12.6, Java 1.8.

We compare two performance metrics: *success rate* (how many falsification trials were successful in finding a falsifying input) and the *number of iterations* made, which corresponds to the number of simulations required and thus indicates time. To account for the stochastic nature of the algorithms, the experiments were repeated for 50 trials. For a meaningful comparison of the number of iterations until falsification, we tried to maximize the falsification rate for a limit of 300 iterations per trial.

The number of iterations of the top-level loop in Algorithm 1 in our implementation corresponds exactly to one complete Simulink simulation up to the time horizon. For random sampling and CMA-ES, the number of iterations likewise corresponds to samples taken by running exactly one simulations each. Hence the comparison is fair and, as the overhead is dominated by simulation time, the numbers are roughly proportional to wall-clock times.

[1] https://github.com/ERATOMMSD/falstar.
[2] https://github.com/decyphir/breach release version 1.2.9.

Table 2. Successes in 50 trials ("succ.", higher is better) and number of iterations averaged over successful trials ("iter.", lower is better) of uniform random sampling, Breach/CMA-ES, and FalStar/ Algorithm 1 for a maximum of 300 iterations per trial. est results for each requirement are highlighted. $AT6_a$: $\overline{v} = 80, \underline{\omega} = 4500$. $AT6_b$: $\overline{v} = 50, \underline{\omega} = 2700$. $t = 30$ in both cases.

	Random		**Breach: CMA-ES**		**FalStar: aLVTS**	
Req.	succ. /50	iter. mean	succ. /50	iter. mean	succ. /50	iter. mean
$AT1_{30}$	43	106.6	50	39.7	50	8.5
$ATX1_3$	50	41.0	50	13.2	50	33.4
$ATX1_4$	49	67.0	6	17.8	50	23.4
ATX2	19	151.1	50	145.2	50	86.3
$AT6_a$	36	117.3	50	97.0	50	22.8
$AT6_b$	2	117.7	49	46.7	50	47.6
AFC27	15	129.1	41	121.0	50	3.9

Table 3. Results from the ARCH competition. $AT6_a$: $t = 35, \overline{v} = 80,$. $AT6_b$: $t = 50, \overline{v} = 50$. $AT6_c$: $t = 65, \overline{v} = 50$. $\underline{\omega} = 3000$ in all three cases.

	S-TaLiRo: SOAR		**Breach: GNM**		**FalStar: aLVTS**	
Req.	succ. /50	iter. mean	succ. /50	iter. mean	succ. /50	iter. mean
$AT1_{20}$	50	118.8	50	11.0	50	33.0
AT2	50	23.9	50	2.0	50	4.3
$AT5_1$	50	26.7	41	74.6	50	69.5
$AT5_2$	50	4.1	49	72.0	26	125.3
$AT5_3$	50	3.4	49	74.5	50	70.8
$AT5_4$	50	10.5	21	84.9	50	71.1
$AT6_a$	49	78.4	50	97.9	50	76.1
$AT6_b$	33	132.6	49	112.9	50	82.4
$AT6_c$	47	61.3	50	94.1	0	–
AFC27	50	70.3	50	3.0	50	3.9
AFC29	50	13.5	50	3.0	50	1.2
CC1	50	9.4	50	3.0	50	4.1
CC2	50	6.0	50	1.0	50	4.0
CC3	50	19.9	50	3.0	50	6.9
CC4	20	188.0	0	–	2	52.0
CC5	50	42.9	49	26.1	46	91.2

Table 2 summarizes our results in terms of success rate, and mean number of iterations of successful trials. The unambiguously (possibly equal) best results are highlighted in blue. Where the lowest average number of iterations was achieved without finding a falsifying input for every trial, we highlight in grey the lowest average number of iterations for 100% success. We thus observe that aLVTS achieves the best performance in all but one case, $ATX1_3$. Importantly, within the budget of 300 iterations per trial, aLVTS achieves a perfect success rate. CMA-ES is successful in 296 trials out of the total 350, with sub maximal success for ATX_4 and AFC27. In comparison, random sampling succeeds in only 214 trials, with sub maximal success in all but ATX1.

The number of iterations required for falsification varies significantly between the algorithms and between the benchmarks. For the automatic transmission benchmarks, as an approximate indication of relative performance, CMA-ES requires about 50% more iterations than aLVTS, and random sampling requires again twice as many as CMA-ES. For the powertrain model (AFC27), the performance of aLVTS is more than an order of magnitude better: 3.9 iterations on average, compared to 121 for CMA-ES.

Figure 2 compares all trial runs for AFC27, ordered by the number of iterations required for falsification. Similar plots for the automatic transmission benchmarks are shown in Fig. 3. The shape of each curve gives an intuition of the performance and consistency of its corresponding algorithm. In general, fewer iterations and more successful results are better, so it is clear that aLVTS performs better than random sampling and CMA-ES.

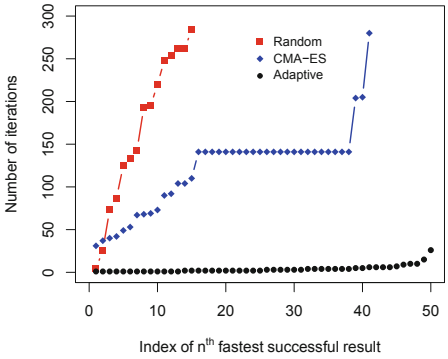


Fig. 2. Relative performance of falsification algorithms with AFC27 powertrain requirement: fewer iterations and more successful results are better.

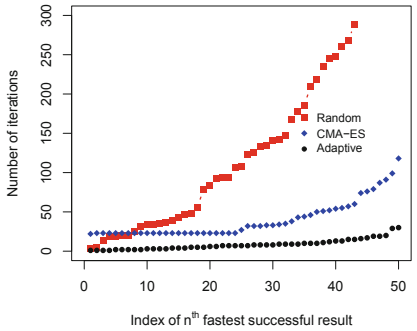
To reinforce that the performance of aLVTS is at least comparable to other approaches, Table 3 presents some results of the recent ARCH competition [28]. The values for S-TaLiro and Breach (in different configurations to our experiments) were provided by the respective participants.

4.3 Discussion

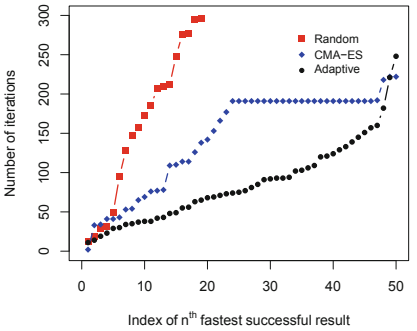
For AT1, aLVTS quickly finds the falsifying input signal, as the required throttle of 100 and brake of 0 are contained in level 0 and are very likely to be tried early on. In contrast, even though this is a problem that is well-suited to hill-climbing, CMA-ES has some overhead to sample its initial population, cf. Fig. 3(a).

While CMA-ES deals very well with ATX1 for $i = 3$, it struggles to find falsifying inputs for $i = 4$ (cf. Figs. 3(c) and (d)). We attribute this to the fact that reaching gear 4 by chance occurs rarely in the exploration of CMA-ES when the robustness score is uninformative. aLVTS not only explores the spatial dimensions, but takes opportunistic jumps to later time points, which increases the probability of discovering a trace (prefix) where the gear is reached.

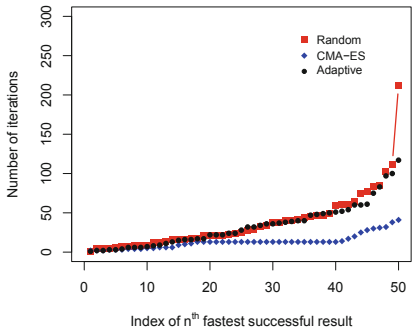
A priori, one would expect CMA-ES to perform well with ATX2 and AT6, exploiting its continuous optimization to fine tune inputs between conflicting requirements. E.g., ATX2 requires that v is both above 50 and below 60; AT4 requires that v is high while maintaining low ω, which is proportional to v. One would similarly expect the limited discrete choices made by aLVTS to hinder its ability to find falsifying inputs. Despite these expectations, our results



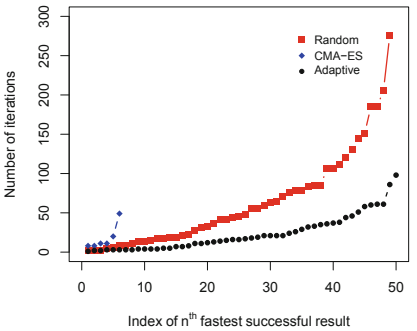
(a) Performance plot for AT1



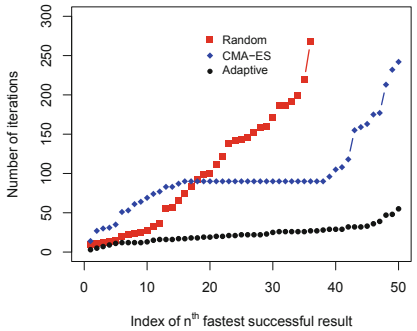
(b) Performance plot for ATX2



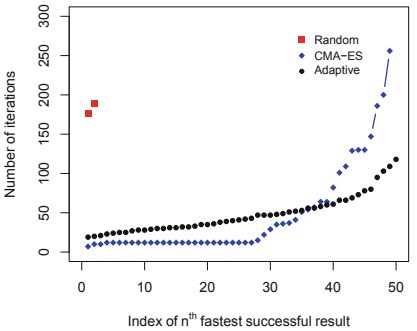
(c) Performace plot for ATX1_3



(d) Performance plot for ATX1_4



(e) Plot for AT6 ($\overline{v} = 80, \underline{\omega} = 4500$)



(f) Plot for AT6 ($\overline{v} = 50, \underline{\omega} = 2700$)

Fig. 3. Performance comparison for the automatic transmission benchmark.

demonstrate that in most situations aLVTS converges to a falsifying input more consistently and with fewer iterations than CMA-ES. We speculate that this is because CMA-ES is too slow to reach the "sweet spots" in the input space, where its optimization is efficient.

For ATX2, there are a few instances where aLVTS does not quickly find a good prefix towards the corridor $v \in [50, 60]$ at time 10 (the rightmost points in Fig. 3(b)), which can be explained by the probabilistic nature of the search.

Regarding two instances of AT6 in Figs. 3(e) and (f), the graph for aLVTS is generally smoother and shallower, whereas CMA-ES shows consistent performance for only some of the trials but takes significantly more time on the worst 10 trials. We remark that the two parameter settings seem to pose opposite difficulty for the two algorithms, as CMA-ES is significantly quicker for two thirds of the trials for the second instance. It is unclear what causes this variation in performance of CMA-ES.

The plateaux apparent in some of the results using CMA-ES are difficult to explain, but suggest some kind of procedural logic or counter to decide termination. In contrast, the curves for random sampling and aLVTS are relatively smooth, reflecting their purely probabilistic natures.

For the results in Table 3, Breach was configured to use the GNM algorithm, which has a phase of sampling extreme and random values, thus sharing some of the characeristics of the aLVTS algorithm. As a consequence, many results of these two approaches are quite similar. A take-away is that algorithms that use random sampling in a disciplined way but are otherwise fairly simple work well on many falsification problem. There is no best overall tool: S-TaLiRo is quickest on the AT5 requirements involving discrete gear changes, whereas GNM yields the best results for the chasing cars model.

The AT and CC results for aLVTS show its capability to adapt to the (informal) difficulty of the problem, where the number of iterations increases but the falsification rate stays high. For $AT5_2$ and $AT6_b$ in Table 3 we conjecture that the available granularities of the search space are misaligned with the actual values to find a violation. Precisely, the required values are not contained in a set $\mathcal{A}_i$ that is sampled within the budget of 300; i is too large.

5 Related Work

The idea to find falsifying inputs using robustness as an optimization function originates from [15] and has since been extended to the parameter synthesis problem (e.g., [22]). Approaches to make the robustness semantics more informative include [3,14], which use integrals instead of min/max in the semantics of temporal operators. Two mature implementations in MATLAB are S-Taliro [4] and Breach [10], which have come to define the benchmark in this field. Users of S-Taliro and Breach can select from a range of optimization algorithms, including Uniform Random, Nelder-Mead, Simulated Annealing, Cross-Entropy and CMA-ES. These cover a variety of trade-offs between exploration of the search space and exploitation of known good intermediate results.

Underminer [5] is a recent falsification tool that learns the (non-) convergence of a system to direct falsification and parameter mining. It supports STL formulas, SVMs, neural nets, and Lyapunov-like functions as classifiers. Other global approaches include [1], which partitions the input space into sub-regions from

which falsification trials are run selectively. This method uses coverage metrics to balance exploration and exploitation. Comprehensive surveys of simulation based methods for the analysis of hybrid systems are given in [6,23].

The characteristic of our approach to explore the search space incrementally is shared with *rapidly-exploring random trees* (RRTs). The so-called star discrepancy metric guides the search towards unexplored regions and a local planner extends the tree at an existing node with a trajectory segment that closely reaches the target point. RRTs have been used successfully in robotics [24] and also in falsification [13]. On the other hand, the characteristic of our approach taking opportunistic coarse jumps in time is reminiscent of stochastic local search [7] and multiple-shooting [31].

Monte Carlo tree search (MCTS) has been applied to a model of aircraft collisions in [25], and more recently in a falsification context to guide global optimization [30], building on the previous idea of time-staging [29]. That work noted the strong similarities between falsification using MCTS and statistical model checking (SMC) using *importance splitting* [19]. The robustness semantics of STL, used in [29,30] and the present approach to guide exploration, can be seen as a "heuristic score function" [20] in the context of importance splitting. All these approaches construct trees from traces that share common prefixes deemed good according to some heuristic. The principal difference is that importance splitting aims to construct a diverse set of randomly-generated traces that all satisfy a property (equivalently, falsify a negated property), while falsification seeks a single falsifying input. The current work can be distinguished from standard MCTS and reinforcement learning [26] for similar reasons. These techniques tend to seek optimal policies that make good decisions in *all* situations, unnecessarily (in the present context) covering the entire search space.

6 Conclusion

The falsification problem is inherently hard (no theoretically best solution for all examples can exist), but our simple approach can provide useful results in isolation or as part of an ensemble. We have demonstrated this by matching and outperforming existing state-of-the-art methods on a representative selection of standard benchmarks. We hypothesize the reason our approach works well stems from the fact that there tends to be a significant mass of simple falsifying inputs for common benchmarks. As future work we will test this hypothesis (and the limits of our approach) by applying our algorithm to a wider range of benchmarks. In addition, we propose to fine-tune the probabilities of exploration vs. exploitation, and find better inputs by interpolating from previously seen traces, in a manner reminiscent of the linear combinations computed by the Nelder-Mead algorithm.

Acknowledgement. This work is supported by the ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST; and Grants-in-Aid No. 15KT0012, JSPS.

References

1. Adimoolam, A., Dang, T., Donzé, A., Kapinski, J., Jin, X.: Classification and coverage-based falsification for embedded control systems. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 483–503. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_24
2. Akazaki, T.: falsification of conditional safety properties for cyber-physical systems with gaussian process regression. In: Falcone, Y., Sánchez, C. (eds.) RV 2016. LNCS, vol. 10012, pp. 439–446. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46982-9_27
3. Akazaki, T., Hasuo, I.: Time robustness in MTL and expressivity in hybrid system falsification. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 356–374. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21668-3_21
4. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: a tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 254–257. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19835-9_21
5. Balkan, A., Tabuada, P., Deshmukh, J.V., Jin, X., Kapinski, J.: Underminer: a framework for automatically identifying nonconverging behaviors in black-box system models. ACM Trans. Embed. Comput. Syst. **17**(1), 1–28 (2017)
6. Bartocci, E., et al.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification. LNCS, vol. 10457, pp. 135–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_5
7. Deshmukh, J., Jin, X., Kapinski, J., Maler, O.: Stochastic local search for falsification of hybrid systems. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 500–517. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_35
8. Dokhanchi, A., Yaghoubi, S., Hoxha, B., Fainekos, G.E.: ARCH-COMP17 category report: preliminary results on the falsification benchmarks. In: Frehse, G., Althoff, M. (eds.) Applied Verification of Continuous and Hybrid Systems (ARCH). EPiC Series in Computing, vol. 48, pp. 170–174. EasyChair (2017)
9. Dokhanchi, A., et al.: ARCH-COMP18 category report: results on the falsification benchmarks. In: Frehse, G. (ed.) Applied Verification of Continuous and Hybrid Systems (ARCH). EPiC Series in Computing, vol. 54, pp. 104–109. EasyChair (2019)
10. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 167–170. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_17
11. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 264–279. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_19

12. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15297-9_9
13. Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., Deshmukh, J.V.: Efficient guiding strategies for testing of temporal properties of hybrid systems. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NFM 2015. LNCS, vol. 9058, pp. 127–142. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17524-9_10
14. Eddeland, J., Miremadi, S., Fabian, M., Åkesson, K.: Objective functions for falsification of signal temporal logic properties in cyber-physical systems. In: Conference on Automation Science and Engineering (CASE), pp. 1326–1331. IEEE (2017)
15. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comp. Sci. **410**(42), 4262–4291 (2009)
16. Hoxha, B., Abbas, H., Fainekos, G.E.: Benchmarks for temporal logic requirements for automotive systems. In: Frehse, G., Althoff, M. (eds.) Applied veRification for Continuous and Hybrid Systems (ARCH). EPiC Series in Computing, vol. 34, pp. 25–30. EasyChair (2014)
17. Hu, J., Lygeros, J., Sastry, S.: Towards a theory of stochastic hybrid systems. In: Lynch, N., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 160–173. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46430-1_16
18. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evol. Comput. **15**(1), 1–28 (2007)
19. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_38
20. Jegourel, C., Legay, A., Sedwards, S.: An effective heuristic for adaptive importance splitting in statistical model checking. In: Margaria, T., Steffen, B. (eds.) ISoLA 2014. LNCS, vol. 8803, pp. 143–159. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45231-8_11
21. Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., Butts, K.R.: Powertrain control verification benchmark. In: Fränzle, M., Lygeros, J. (eds.) Hybrid Systems: Computation and Control (HSCC), pp. 253–262. ACM (2014)
22. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(11), 1704–1717 (2015)
23. Kapinski, J., Deshmukh, J.V., Jin, X., Ito, H., Butts, K.: Simulation-based approaches for verification of embedded control systems: an overview of traditional and advanced modeling, testing, and verification techniques. IEEE Control Syst. Mag. **36**(6), 45–64 (2016)
24. LaValle, S.M., Kuffner Jr., J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. (IJRR) **20**(5), 378–400 (2001)
25. Lee, R., Kochenderfer, M.J., Mengshoel, O.J., Brat, G.P., Owen, M.P.: Adaptive stress testing of airborne collision avoidance systems. In: IEEE/AIAA 34th Digital Avionics Systems Conference (DASC 2015), pp. 6C2:1–6C2:13 (2015)
26. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT press, Cambridge (2018)
27. Wolpert, D., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
28. Yaghoubi, S., et al.: ARCH-COMP19 category report: results on the falsification benchmarks. In: Frehse, G. (ed.) Applied Verification of Continuous and Hybrid Systems (ARCH). EPiC Series in Computing. EasyChair (2019)

29. Zhang, Z., Ernst, G., Hasuo, I., Sedwards, S.: Time-staging enhancement of hybrid system falsification. In: 2018 IEEE Workshop on Monitoring and Testing of Cyber-Physical Systems (MT-CPS 2018), pp. 3–4. IEEE, April 2018
30. Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., Hasuo, I.: Two-layered falsification of hybrid systems guided by Monte Carlo tree search. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD 2018) (2018)
31. Zutshi, A., Deshmukh, J.V., Sankaranarayanan, S., Kapinski, J.: Multiple shooting, CEGAR-based falsification for hybrid systems. In: Embedded Software (EMSOFT), pp. 5:1–5:10 (2014)