Hybrid System Falsification Under (In)equality Constraints via Search Space Transformation

Zhenya Zhang, Paolo Arcaini, and Ichiro Hasuo

***Abstract*—The verification of hybrid systems is intrinsically hard, due to the continuous dynamics that leads to infinite search spaces. Therefore, research attempts focused on hybrid system falsification of a black-box model, a technique that aims at finding an input signal violating the desired temporal specification. Main falsification approaches are based on stochastic hill-climbing optimization, that tries to minimize the degree of satisfaction of the temporal specification, given by its robust semantics. However, in the presence of constraints between the inputs, these methods become less effective. In this article, we solve this problem using a search space transformation that first maps points of the unconstrained search space to points of the constrained one, and then defines the fitness of the former ones based on the robustness values of the latter ones. Based on this search space transformation, we propose a falsification approach that performs the search over the unconstrained space, guided by the robustness of the mapped points in the constrained space. We introduce three versions of the proposed approach that differ in the way of selecting the mapped points. Experiments show that the proposed approach outperforms state-of-the-art constrained falsification approaches.**

***Index Terms*—(In)equality constraints, hybrid system falsification, search space transformation, signal temporal logic.**

I. Introduction

HYBRID Systems Falsification: Cyber-physical systems (CPSs) are *hybrid systems* combining physical and digital components. Quality assurance of CPS is a problem of great importance, however, automated formal verification of hybrid systems is almost impossible due to their physical components that lead to infinite search spaces.

Therefore, research has proposed the more feasible approach of *falsification* that tries to show the violation of a property rather than proving its satisfaction. Formally, given a *model* $\mathcal{M}$ that takes an input signal $\mathbf{u}$ and outputs a signal $\mathcal{M}(\mathbf{u})$, and a *specification* φ (a temporal formula), the falsification problem consists of finding a *falsifying input*, i.e., an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ. Classical approaches solve the falsification problem by turning it into an optimization one. To do this, they exploit the *robust semantics* of temporal formulas [1], [2]: instead of the classical Boolean satisfaction relation $\mathbf{v} \models \varphi$, robust semantics assigns a value $[\![\mathbf{v}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ (i.e., *robustness*) that assesses not only whether φ is satisfied or violated (by the sign) but also *how robustly* the formula is satisfied or violated. Negative robustness is an indicator of the violation of the system specification; therefore, the goal of falsification is to minimize the robustness to obtain a negative value. The task is particularly difficult due to the black box nature of the system and the expensiveness of the robustness computation which depends on system *simulations*. Therefore, optimization-based falsification algorithms employ stochastic optimization approaches, such as *hill-climbing* optimization, to generate inputs with the aim of decreasing robustness, and they terminate when they find an input with negative robustness (i.e., a *falsifying input*). Different optimization-based falsification algorithms have been proposed (see [3]). Also tools have been developed, as Breach [4] and S-TaLiRo [5].

Input Constraints: Falsification typically considers a hyperrectangle as the search space for input signals: this is the problem setting adopted in many works [6]–[12], and also in tools Breach [4] and S-TaLiRo [5]. However, in real hybrid systems, input signals are not free to assume any value within the hyperrectangle, because there exist some *constraints* ψ among them. In a vehicle, for example, the *throttle* and the *brake* cannot be pushed simultaneously. Descriptions of CPS usually report such constraints on the system inputs, e.g., [13] and [14]. In the presence of such constraints, inputs produced by falsification should be guaranteed to satisfy them; otherwise, the found inputs would not help the quality assurance efforts.

For black-box optimization, several techniques have been proposed for handling constraints, in particular, *penalty-based approaches*. For example, *death penalty* [15] is a method that rejects all the infeasible inputs, but it has been shown to be inefficient [16], [17]. Other penalty-based approaches define a *penalty factor* added to the objective function for those inputs violating the constraints; in some of these, the penalty factor is proportional to the degree of violation of the constraints, in others it increases over time. Application of penalty-based approaches in falsification has been studied in [18], where constraints are formulated as an STL formula and penalty is defined using the robustness.

Manuscript received April 17, 2020; revised June 17, 2020; accepted July 6, 2020. Date of publication October 2, 2020; date of current version October 27, 2020. This work was supported by ERATO HASUO Metamathematics for Systems Design Project under Grant JPMJER1603, JST. The work of Zhenya Zhang was supported by Grant-in-Aid for JSPS Fellows under Grant 19J15218. This article was presented in the International Conference on Embedded Software 2020 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Ichiro Hasuo.)*

Zhenya Zhang and Ichiro Hasuo are with the Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan, and also with the Department of Informatics, School of Multidisciplinary Sciences, Graduate University for Advanced Studies (SOKENDAI), Hayama 240-0193, Japan (e-mail: i.hasuo@acm.org).

Paolo Arcaini is with the Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan.

Digital Object Identifier 10.1109/TCAD.2020.3013073

Although penalty-based approaches are sound, they usually need to consider several infeasible samplings, before learning the feasible part of the search space. Moreover, penalty values that are added to the objective function can modify the *fitness landscape* (i.e., values of the objective function) in a way (e.g., a large flat plateau with a constant value) that the performance of the search algorithm (e.g., hill climbing) may be affected.

Another category of constraint handling methods is called *feasibility preservation*, which consists of *repairing* the infeasible inputs by moving them to the feasible space. The key of these methods is to define a proper repairing operator, but that task is technically tricky. For example, in [19], a mapping method is proposed. However, the performance of the method is highly dependent on the selection of some parameters, for which no good strategy is given; moreover, that technique is not able to handle equality constraints.

Contribution: In this article, we propose an approach for the *input-constrained falsification problem*, that aims at overcoming the limits of penalty-based approaches. The approach defines a *search space transformation* mapping each point $\overrightarrow{u}$ of an unconstrained search space to a point $\overrightarrow{\pi}$ of the constrained input space, and defines the fitness of $\overrightarrow{u}$ in terms of the robustness value of $\overrightarrow{\pi}$. In this way, the approach performs the falsification search over the unconstrained search space, without compromising the effectiveness of hill climbing (differently from what happens with the penalty-based approaches). When a sample $\overrightarrow{u}$ in the unconstrained search space is found with negative fitness, the corresponding point $\overrightarrow{\pi}$ in the constrained input space is guaranteed to be a feasible falsifying input. The main contributions of this article are as follows.

1) A general framework based on *search space transformation* for handling constraints in falsification; it consists of mapping points from an unconstrained search space to the constrained space, and associating their fitness.
2) A novel technique for search space transformation, named *proportional transformation*, that performs the mapping following an iterative process along space dimensions, by considering them singularly in a given order.
3) Three approaches that exploit the proportional transformation to solve the constrained falsification problem. They differ in the way of using the proportional transformation: the Fixed-Priority method maps each point following a predefined order of dimensions, the All-Priorities method considers all the possible orders, while the MAB-Priority instantiates a *multiarmed bandit* (MAB) model to *learn* the order that makes hill climbing more effective.

Experiments over benchmarks used in falsification show that the proposed approaches always outperform penalty-based approaches, and that MAB-Priority is the best one.

Organization: Section II introduces some background. Section III presents the problem and overviews the proposed approach, that is described in detail in Sections IV and V. Then, Section VI presents the experiments we performed, Section VII reviews some related work, and Section VIII concludes this article.

II. Preliminaries

In this section, we review the widely accepted method of hill-climbing optimization-based falsification. The core of making use of hill-climbing optimization is the introduction of *robust semantics* of temporal formulas.

A. Robust Semantics for STL

Our definitions here are taken from [1] and [2].

Definition 1 (Time-Bounded Signal): Let $T \in \mathbb{R}_+$ be a positive real. An *M-dimensional signal* with a time horizon T is a function $\mathbf{w} : [0, T] \to \mathbb{R}^M$.

We treat the system model as a black box, i.e., its behaviors are only observed from inputs and their corresponding outputs. So, we simply define the system model as a function.

Definition 2 (System Model $\mathcal{M}$): A *system model*, with M-dimensional input and N-dimensional output, is a function $\mathcal{M}$ that takes an input signal $\mathbf{u} : [0, T] \to \mathbb{R}^M$ and returns a signal $\mathcal{M}(\mathbf{u}) : [0, T] \to \mathbb{R}^N$. Here, the common time horizon $T \in \mathbb{R}_+$ is arbitrary.

Definition 3 (STL Syntax): We fix a set **Var** of variables. In STL, *atomic propositions* and *formulas* are defined as follows, respectively: $\alpha :: \equiv f(x_1, \ldots, x_N) > 0$, and $\varphi :: \equiv \alpha \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathcal{U}_I\, \varphi$. Here, f is an N-ary function $f : \mathbb{R}^N \to \mathbb{R}$, $x_1, \ldots, x_N \in \mathbf{Var}$, and I is a closed nonsingular interval in $\mathbb{R}_{\geq 0}$, i.e., $I = [a, b]$ or $[a, \infty)$ where $a, b \in \mathbb{R}$ and $a < b$.

We omit subscripts I for temporal operators if $I = [0, \infty)$. Other common connectives, such as $\vee$, $\rightarrow$, $\top$, $\Box_I$ (always) and $\Diamond_I$ (eventually), are introduced as abbreviations, e.g., $\Diamond_I \varphi \equiv \top\, \mathcal{U}_I\, \varphi$ and $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$. An atomic formula $f(\vec{x}) \leq c$, where $\vec{x} = (x_1, \ldots, x_N)$ and $c \in \mathbb{R}$, is accommodated using $\neg$ and the function $f'(\vec{x}) :\!= f(\vec{x}) - c$.

Definition 4 (Robust Semantics [2]): Let $\mathbf{w} : [0, T] \to \mathbb{R}^N$ be an N-dimensional signal, and $t \in [0, T)$. The *t-shift* of $\mathbf{w}$, denoted by $\mathbf{w}^t$, is the time-bounded signal $\mathbf{w}^t : [0, T - t] \to \mathbb{R}^N$ defined by $\mathbf{w}^t(t') \coloneqq \mathbf{w}(t + t')$. $\mathbf{w}(t)(x_i)$ indicates the value of dimension x_i of the signal $\mathbf{w}$ at time t.

Let $\mathbf{w} : [0, T] \to \mathbb{R}^{|\mathbf{Var}|}$ be a signal, and φ be an STL formula. We define the *robustness* $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ as follows, by induction on the construction of formulas. $\bigsqcap$ and $\bigsqcup$ denote the infimums and supremums of real numbers, respectively. Their binary versions $\sqcap$ and $\sqcup$ denote minimum and maximum

$$
\begin{aligned}
&[\![\mathbf{w}, f(x_1, \ldots, x_N) > 0]\!] \coloneqq f(\mathbf{w}(0)(x_1), \ldots, \mathbf{w}(0)(x_N)) \\
&[\![\mathbf{w}, \bot]\!] \coloneqq -\infty \quad [\![\mathbf{w}, \neg\varphi]\!] \coloneqq -[\![\mathbf{w}, \varphi]\!] \\
&[\![\mathbf{w}, \varphi_1 \wedge \varphi_2]\!] \coloneqq [\![\mathbf{w}, \varphi_1]\!] \sqcap [\![\mathbf{w}, \varphi_2]\!] \\
&[\![\mathbf{w}, \varphi_1\, \mathcal{U}_I\, \varphi_2]\!] \coloneqq \bigsqcup_{t \in I \cap [0,T]} \left([\![\mathbf{w}^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{w}^{t'}, \varphi_1]\!] \right).
\end{aligned}
$$

For atomic formulas, $[\![\mathbf{w}, f(\vec{x}) > c]\!]$ stands for the vertical margin $f(\vec{x}) - c$ for the signal $\mathbf{w}$ at time 0. A negative robustness value indicates how far the formula is from being true. It follows from the definition that the robustness for the eventually modality is given by: $[\![\mathbf{w}, \Diamond_{[a,b]}(x > 0)]\!] = \bigsqcup_{t \in [a,b] \cap [0,T]} \mathbf{w}(t)(x)$.

The above robustness notion taken from [2] is *spatial*. Other robustness notions take *temporal* aspects into account, such as "how long before the deadline the required event occurs" (see [2], [20]). Our choice of spatial robustness in this article is for the sake of simplicity, and is thus not essential.

The original semantics of STL is Boolean, given as usual by a binary relation $\models$ between signals and formulas. The robust semantics refines the Boolean one in the following sense: $[\![\mathbf{w}, \varphi]\!] > 0$ implies $\mathbf{w} \models \varphi$, and $[\![\mathbf{w}, \varphi]\!] < 0$ implies $\mathbf{w} \not\models \varphi$, see [1, Proposition 16]. Optimization-based falsification via robust semantics hinges on this refinement.

B. Hill-Climbing-Guided Falsification

The falsification problem has attracted a lot of attention both in industry and academia. Hill-climbing optimization is the main adopted methodology (see [2], [3], [6], [7], [10], [11], [21], tools Breach [4], S-TaLiRo [5]). We formulate the problem and the methodology, for later use in our approach.

Definition 5 (Falsifying Input): Let $\mathcal{M}$ be a system model and φ be an STL formula. A signal $\mathbf{u} : [0, T] \to \mathbb{R}^M$ is a *falsifying input* if $[\![\mathcal{M}(\mathbf{u}), \varphi]\!] < 0$; the latter implies $\mathcal{M}(\mathbf{u}) \not\models \varphi$.

The input signal $\mathbf{u} : [0, T] \to \mathbb{R}^M$ (see Definition 1) to the system $\mathcal{M}$ usually comes with an M-dimensional hyperrectangle $\mathcal{D} \coloneqq \prod_{k=1}^{M} D_k$, where D_k is the interval of the kth dimension of $\mathbf{u}$, and such that for any $t \in [0, T]$, $\mathbf{u}(t) \in \mathcal{D}$.

As the time domain $[0, T]$ of $\mathbf{u}$ is a continuous space, it is unrealistic to synthesize a falsifying input by searching for values $\mathbf{u}(t)$ at each time point $t \in [0, T]$ and concatenating them. In practice, engineers usually discretize the time domain of $\mathbf{u}$, and interpolate based on some rules to simplify the problem (see [22], [23]). Signals thus can be classified by the interpolation methods. We introduce one typical interpolation, *piecewise constant*, as follows.

Definition 6 (Piecewise Constant Signal): Let c be a positive integer. A signal $\mathbf{u} : [0, T] \to \mathbb{R}^M$ is *piecewise constant* if for all $k \in \{0, \dots, \mathsf{c} - 1\}$, $\mathbf{u}(t)$ is a vector of M constant values in the time interval $t \in [k(T/\mathsf{c}), (k + 1)(T/\mathsf{c})]$. The parameter c is known as *control point*.

We denote the discretized representation of $\mathbf{u}$ as a vector $\overrightarrow{u} = (u_{1,1}, \dots, u_{1,M}, \dots, u_{\mathsf{c},1}, \dots, u_{\mathsf{c},M})$, and use it from now on to indicate the input signal.

The technique for solving a falsification problem is via transforming it into an optimization problem as follows:

$$\min_{\overrightarrow{u}} \ [\![\mathcal{M}(\overrightarrow{u}), \varphi]\!] \quad \text{s.t.} \quad \overrightarrow{u} \in \Omega \tag{1}$$

where $\Omega = \mathcal{D}^{\mathsf{c}}$ is an $M\mathsf{c}$-dimensional hyperrectangle. In the following, let $n = M\mathsf{c}$.

Assume the setting given by Definitions 5 and 6 and (1). For finding a falsifying input, the methodology of *hill-climbing-guided falsification* is presented in Algorithm 1.

Here, the function HILL-CLIMB (line 5) samples an input signal $\overrightarrow{u}_k$ in the space Ω, aiming at minimizing the robustness $\mathsf{rb}_k = [\![\mathcal{M}(\overrightarrow{u}_k), \varphi]\!]$, that acts as the *fitness function* in this context. It does so, learning from the previous observations $\big(\overrightarrow{u}_l, \mathsf{rb}_l\big)_{l \in \{1,\dots,k-1\}}$ of input signals $\overrightarrow{u}_1, \dots, \overrightarrow{u}_{k-1}$ and their corresponding robustness values. The algorithm records the $\overrightarrow{u}_k$ that has the best robustness over the sampling history (line 7), and finally either returns a falsifying input that has a negative robustness or reports a failure when no falsifying input is found within the time budget (line 8).

Algorithm 1 Hill-Climbing-Guided Falsification

Require: a system model $\mathcal{M}$, an STL formula φ, and a time budget K
1: **function** HILL-CLIMB-FALSIFY($\mathcal{M}, \varphi, K$)
2: $\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$
3: **while** $\mathsf{rb} \geq 0$ and within the time budget K **do**
4: $k \leftarrow k + 1$
5: $\overrightarrow{u}_k \leftarrow$ HILL-CLIMB$\Big(\big(\overrightarrow{u}_l, \mathsf{rb}_l\big)_{l \in \{1,\dots,k-1\}}\Big)$
▷ Hill climbing suggests $\overrightarrow{u}_k$ based on the sampling history
6: $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\overrightarrow{u}_k), \varphi]\!]$ ▷ Compute robustness
7: **if** $\mathsf{rb}_k < \mathsf{rb}$ **then** $\mathsf{rb} \leftarrow \mathsf{rb}_k$
▷ Update the best robustness so far if applicable
8: $\overrightarrow{u} \leftarrow \begin{cases} \overrightarrow{u}_k & \text{if } \mathsf{rb} < 0, \text{ that is, } \mathsf{rb}_k = [\![\mathcal{M}(\overrightarrow{u}_k), \varphi]\!] < 0 \\ \text{Failure} & \text{otherwise, that is, no falsifying input found} \end{cases}$
9: **return** $\overrightarrow{u}$

Hill climbing can be implemented by various stochastic optimization algorithms. Examples include CMA-ES (used in our experiments) [24], simulated annealing [25], etc.

III. Problem Definition and Overview of the Proposed Approach

As seen in (1), in falsification, an input signal $\overrightarrow{u}$ of a model is usually given in a reasonable interval, e.g., the pedal force of *throttle* should be within [0, 100]. However, due to the existence of *constraints* among inputs, it is not always true that any value of $\overrightarrow{u}$ can be taken at any time, e.g., when *brake* > 0, *throttle* should be exactly 0 rather than a value in [0, 100]. Not considering the input constraints could result in obtaining falsifying inputs that are meaningless, as they identify situations that cannot happen in reality.

In this article, we consider the *input-constrained falsification problem*, formally defined as follows.

Definition 7 (Input-Constrained Falsification Problem): The *input-constrained falsification problem* is defined as

$$\min_{\overrightarrow{u}} \ [\![\mathcal{M}(\overrightarrow{u}), \varphi]\!] \quad \text{s.t.} \quad \overrightarrow{u} \models \psi, \overrightarrow{u} \in \Omega$$

where ψ is a constraint on the input signal $\overrightarrow{u}$. The goal of the problem is to find an input signal $\overrightarrow{u}$ such that $\overrightarrow{u} \models \psi$, $\overrightarrow{u} \in \Omega$, and $[\![\mathcal{M}(\overrightarrow{u}), \varphi]\!] < 0$.

In this article, as ψ, we consider logical combinations of *linear constraints* (both equalities and inequalities). We now give the syntax of the supported constraints. Without loss of generality, we assume the logical constraints to be in disjunctive normal form (DNF). In this article, for the sake of presentation, constraints are not given in DNF, but of course they can be transformed to it.

Definition 8 (Syntax of Constraints): We define an n-ary constraint ψ as follows:

$$\psi \ ::\equiv \psi \vee \psi \mid \gamma \quad \gamma ::\equiv \gamma \wedge \gamma \mid \xi$$
$$\xi \ ::\equiv \sum_{k=1}^{n} a_k x_k + a_{n+1} = 0 \mid \sum_{k=1}^{n} a_k x_k + a_{n+1} < 0 \mid \bot \mid \neg\xi.$$

Here, $a_1, \dots, a_{n+1} \in \mathbb{R}$ are coefficients, and $x_1, \dots, x_n$ variables, each one x_i defined over a domain D_i.

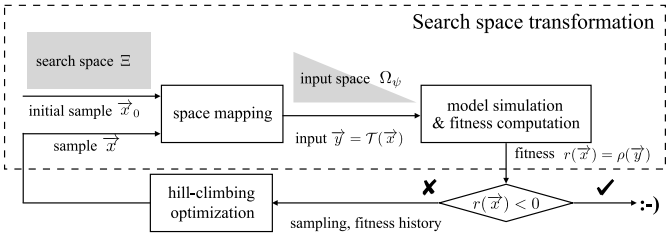

Fig. 1. Proposed constrained falsification approach.

In our context, the number of variables is $n = M\mathsf{c}$, because the input signal is a piecewise constant signal composed of M inputs having c control points.

Example 1: The aforementioned example of constraint that *throttle and brake cannot be positive simultaneously* can be expressed as $\bigwedge_{k=1}^{\mathsf{c}} (\text{throttle}_k = 0 \vee \text{brake}_k = 0)$.

In Definition 7, the input space is given by the application of ψ to Ω. We denote this *constrained space* as Ω_ψ, which contains all the points in Ω satisfying ψ, i.e., $\Omega_\psi \coloneqq \{\vec{x} \in \Omega \mid \vec{x} \models \psi\}$.

The input-constrained falsification problem in Definition 7 poses many challenges. Observe in Algorithm 1 the way a hill-climbing optimization algorithm works: it comes up with new samplings stochastically depending on the history of previous samplings (toward more promising regions). Since the classical setting is only bounded by a hyperrectangle Ω [as in (1)], sampling feasible points are easy because the boundary of the feasible area can be precisely determined without computational cost. Instead, if the search space was more irregular, computing the boundaries of the feasible area would be much more difficult, and the functionality of generating samplings will be harmed due to the limitations on the feasible sampling region.

A. Proposed Approach

In this article, we propose an approach for the input-constrained falsification problem. The workflow is shown in Fig. 1.

Let Ξ be an arbitrary hyperrectangle with the same number of dimensions as Ω_ψ. In the following, we name Ξ as a *search space* and Ω_ψ as an *input space*. The approach allows to perform the falsification search over the search space Ξ (weakly bounded, so not harming the effectiveness of hill climbing) but, at the same time, it minimizes the fitness computed based on the input space Ω_ψ. To do this, the fitness function r of Ξ is defined in terms of the fitness (robustness) distribution ρ in Ω_ψ. More precisely, it employs a *search space transformation* that firstly maps a point $\vec{x}$ of the search space Ξ into a point $\vec{y}$ of the input space Ω_ψ through a space mapping $\mathcal{T}$ (i.e., $\vec{y} = \mathcal{T}(\vec{x})$), and then defines the fitness accordingly (i.e., $r(\vec{x}) = \rho(\vec{y})$). In this way, the constrained falsification problem is turned into an unconstrained optimization problem:

$$\min_{\vec{x}} \; r(\vec{x}) \quad \text{s.t.} \quad \vec{x} \in \Xi.$$

Once a point $\vec{x}$ with negative fitness in Ξ is found, the mapped point $\vec{y} = \mathcal{T}(\vec{x})$ in Ω_ψ will be returned as falsifying input. Formally, the process of search space transformation is defined by the two following definitions.

Definition 9 (Space Mapping): Let Ξ be the search space, and Ω_ψ be the input space. We define a *space mapping* function $\mathcal{T} : \Xi \to \Omega_\psi$ as a *total surjective function* from Ξ to Ω_ψ.

We also define the fitness function of the points of the search space Ξ on the base of the fitness of the input space Ω_ψ.

Definition 10 (Fitness Function in Ξ): Let Ω_ψ be the input space, and $\rho : \Omega_\psi \to \mathbb{R}$ be a fitness function for Ω_ψ. Let $\mathcal{T} : \Xi \to \Omega_\psi$ be a space mapping from the search space Ξ to Ω_ψ. The fitness function $r : \Xi \to \mathbb{R}$ in the search space Ξ is defined as $r(\vec{x}) \coloneqq \rho(\mathcal{T}(\vec{x}))$.

The search space transformation guarantees two properties necessary in our approach.

Proposition 1 (Soundness and Completeness of the Search Space Transformation): Any falsification algorithm that samples over the search space Ξ using the fitness function r as guidance is guaranteed to be *sound* and *complete*.

Soundness: If a sample $\vec{x}$ with negative fitness ($r(\vec{x}) < 0$) is found in the search space Ξ, the corresponding input $\vec{y} = \mathcal{T}(\vec{x})$ in the input space Ω_ψ is guaranteed to be a falsifying input ($\rho(\vec{y}) < 0$). As soon as such an $\vec{x}$ is found, the falsification process can stop and $\vec{y}$ can be returned as the witness of the falsification.

Completeness: For each falsifying input $\vec{y}$ in the input space Ω_ψ, there is a sample $\vec{x}$ in the search space Ξ that maps to it, i.e., $\vec{y} = \mathcal{T}(\vec{x})$. This guarantees that the search over Ξ can find all the falsifying inputs (if any).

The soundness comes from the definition of r (see Definition 10): once $r(\vec{x}) < 0$, it means that $\rho(\mathcal{T}(\vec{x})) < 0$ and thus $\rho(\vec{y}) < 0$. The completeness is from the surjectiveness of $\mathcal{T}$ (Definition 9).

Remark 1: Proposition 1 states that *any* space mapping guarantees soundness and completeness of the approach. However, these are not the only desired properties. We would also like that the implemented space mapping does not harm the effectiveness of hill climbing. For guaranteeing this, hill climbing in the search space Ξ should get a *faithful* representation of the fitness landscape of the input space Ω_ψ.

Continuity of a space mapping, i.e., mapping points in proximity again to proximity, is a good criterion. We shall propose a specific class of continuous space mappings. It is called the *proportional transformation*.

IV. Proportional Transformation

Different search space transformations can be identified, that differ in the way they implement the space mapping (see Definition 9) and could lead to different performances. In this section, we propose the *proportional transformation* $\mathcal{T}$ that maps each point of the search space Ξ into a point of the input space Ω_ψ, by *proportionally* scaling the value of each dimension of Ξ. Note that such mapping does not favor any particular part of the input space.

We call a set of nonoverlapping intervals as *interval sequence*. Formally, an *interval sequence* over the real domain is defined as $R := (I_1, \ldots, I_q)$, where 1) each $I_j = [I_j^L, I_j^U]$ is a continuous interval with lower bound $I_j^L \in \mathbb{R}$ and upper bound $I_j^U \in \mathbb{R}$, such that $I_j^L \leq I_j^U$ and 2) $I_j^U < I_{j+1}^L$ for each $j = 1, \ldots, q-1$.

We denote the length of an interval I_j as $|I_j| = I_j^U - I_j^L$, and the *accumulated length* of all the intervals in R as $\texttt{accLen}(R) = \sum_{j=1}^{q} |I_j|$.

We now provide a definition for determining the bounds of the constrained space identified by the constraints.

Definition 11 (Feasible Interval Sequence): Let Ξ be the search space, and $\psi = \vee_{i=1}^{t} \psi_i$ an n-ary constraint in DNF defined over variables $\overrightarrow{x} = (x_1, \ldots, x_n)$, where each ψ_i is a conjunction of equalities and/or inequalities. Given a dimension $d \in \{1, \ldots, n\}$, we can identify the *bounds* of ψ over d as follows. For each conjunction ψ_i, we identify the minimum Γ_i^L and the maximum Γ_i^U of its feasible area, by solving two linear programming problems[1] (note that ψ_i only contains equalities and inequalities)

$$\begin{aligned} &\min\ x_d \text{ s.t.} \quad \overrightarrow{x} \models \psi_i, \overrightarrow{x} \in \Omega \\ &\max\ x_d \text{ s.t.} \quad \overrightarrow{x} \models \psi_i, \overrightarrow{x} \in \Omega. \end{aligned}$$

Then, the feasible interval sequence $\mathbf{R}_d$ of ψ on the dth dimension is computed as follows: $\mathbf{R}_d := \bigcup_{i=1}^{t} [\Gamma_i^L, \Gamma_i^U]$.

In the next definition, we show how a value belonging to a continuous interval can be mapped to an interval sequence.

Definition 12 (Proportional Position): Let $A = [A^L, A^U]$ be a continuous interval and $v \in A$. The proportional position $\Theta(v, A, R)$ of v in an interval sequence $R = (I_1, \ldots, I_q)$ is defined as follows:

$$\Theta(v, A, R) := p \cdot \texttt{accLen}(R) - \sum_{j=1}^{e} |I_j| + I_{e+1}^L$$

where:

$p = \frac{v - A^L}{A^U - A^L}$ is the proportional value of v in A;

$e \in \{1, \ldots, q\}$ is the maximum index that satisfies $p \cdot \texttt{accLen}(R) - \sum_{j=1}^{e} |I_j| > 0$.[2]

Definition 13 (Constraint Reduction): Let ψ be an n-ary constraint. Given a search space Ξ and a point $\overrightarrow{u} \in \Xi$, we compute the proportional position $\pi_d = \Theta(u_d, D_d, \mathbf{R}_d)$ over the dth dimension of $\overrightarrow{u}$ (i.e., u_d). Then, the function $\texttt{Reduce}(\psi, \pi_d, d) := \psi[x_d \mapsto \pi_d]$ is used to *reduce* ψ to an $(n-1)$-ary constraint.

We define S as a permutation of the set $\{1 \ldots, n\}$ of the dimensions of Ξ. S identifies the order in which dimensions are considered, and so it will be called *priority* in the following.

The proposed *proportional transformation* $\mathcal{T}$ maps a point $\overrightarrow{u}$ of Ξ into a point $\overrightarrow{\pi}$ of Ω_ψ (having the same number of dimensions), such that $\overrightarrow{\pi}$ satisfies the constraint ψ. To do this, it iteratively computes feasible interval sequences (Definition 11), identifies the proportional position (Definition 12), and performs constraint reduction (Definition 13), following a given priority order S, until all values on different dimensions of $\overrightarrow{u}$ have been mapped to their corresponding proportional positions. Algorithm 2 shows the computation of the proportional transformation.

Algorithm 2 Proportional Transformation

Require: a search space $\Xi = \prod_{k=1}^{n} D_k$, a sampled point $\overrightarrow{u} \in \Xi$, a constraint $\psi = \vee_{i=1}^{t} \psi_i$ in DNF, a priority (permutation) S of the dimensions $\{1 \ldots, n\}$.

```
1:  function MAP-POINT(Ξ, ψ, S, u⃗)
2:      π⃗ = (π_1, ..., π_n) ← (0, ..., 0)            ▷ Initialize π⃗
3:      MAP-DIMENSION(Ξ, ψ, S, u⃗, π⃗)
4:      return π⃗
5:  procedure MAP-DIMENSION(Ξ, ψ, S, u⃗, π⃗)
6:      if length(S) > 0 then
7:          s ← S.head                    ▷ Obtain the first dimension in S
8:          R_s ← ⋃_{i=1}^{t} [Γ_i^L, Γ_i^U]          ▷ Feasible interval sequence
9:          π_s ← Θ(u_s, D_s, R_s)                    ▷ Obtain proportional position
10:         ψ' ← Reduce(ψ, π_s, s)                    ▷ Constraint reduction
11:         S' ← remove s from S
12:         MAP-DIMENSION(Ξ, ψ', S', u⃗, π⃗)           ▷ Recursive call
```

The algorithm starts by initializing an n-dimensional point $\overrightarrow{\pi}$ (line 2), and invoking the procedure MAP-DIMENSION using as arguments the search space Ξ, the constraint ψ, and the priority S (line 3). The procedure MAP-DIMENSION also receives the point $\overrightarrow{\pi}$, and iteratively modifies its value on each dimension. In each loop, the procedure obtains the first element s of the priority S, and determines the *feasible interval sequence* $\mathbf{R}_s$ of s dimension, following the rules in Definition 11 (line 8). Then, the proportional position $\Theta(u_s, D_s, \mathbf{R}_s)$ of $\overrightarrow{u}$ on s dimension is computed according to Definition 12 (line 9), and the constraint ψ is reduced over s dimension following Definition 13 (line 10). Finally, the priority S is updated by removing the first element s (line 11), and the procedure MAP-DIMENSION is invoked again to reduce the remaining arguments (line 12). The recursive call terminates when all dimensions have been mapped. At the end, $\overrightarrow{\pi}$ is returned (line 4) as the mapped point in Ω_ψ that satisfies the constraint ψ.

Proposition 2: The proportional transformation $\mathcal{T}$ is a space mapping.

It is easy to see that $\mathcal{T}$ is a total surjection as required by Definition 9. Indeed, one can follow its definition and construct, in a step-by-step manner, a right inverse g of $\mathcal{T}$ (i.e., $\mathcal{T} \circ g = \text{id}$). Existence of such g witnesses the surjectiveness of $\mathcal{T}$. Continuity of $\mathcal{T}$ is easily established, too.

Example 2: We use a simple example to explain our approach. We consider $\Xi = [0, 10] \times [0, 10]$ as search space, and $\psi = (x_1 + x_2 - 5 < 0)$ as constraint defining the input space Ω_ψ, as shown in Fig. 2(a).

Let us consider a point $\overrightarrow{u} = (8, 8) \in \Xi$. Let us call x_1 and x_2 the two dimensions of the search space. Using the priority $S^1 = (x_1, x_2)$, $\overrightarrow{u}$ is mapped to point $\overrightarrow{\pi}_1 = (4, 0.8)$; instead, using the priority $S^2 = (x_2, x_1)$, it is mapped to point $\overrightarrow{\pi}_2 = (0.8, 4)$. Fig. 2(b) shows the two transformations.

Remark 2: Note that the general transformation process is not specialized to linear constraints, and can be adapted for any type of constraints. What needs to be adapted is Definition 11 to find the bounds over a given dimension: different types of constraints need different solvers (for linear constraints, we

[1] They can be easily computed with any linear programming solver.

[2] Note that $\texttt{accLen}(R)$ can be 0. The implementation handles these cases.

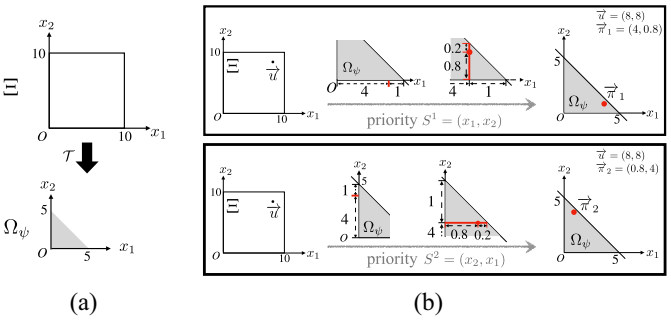

Fig. 2. Running example—proportional transformation. (a) Ξ and Ω_ψ. (b) Two proportional transformations.

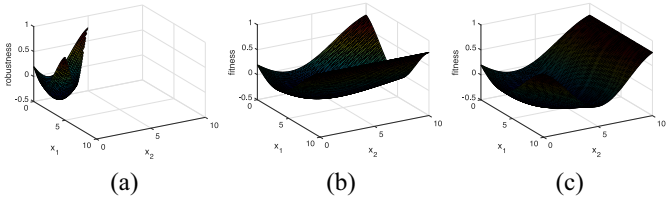

Fig. 3. Fitness landscape of Ω_ψ and transformed fitness landscapes in Ξ. (a) In Ω_ψ. (b) In Ξ (priority S^1). (c) In Ξ (priority S^2).

use a linear programming solver). In this article, we focus and perform experiments on linear constraints. Extending the approach to nonlinear constraints is left as future work.

V. Falsification Based on the Proportional Transformation

In this section, we describe how we use the proportional transformation presented in Section IV to implement a falsification algorithm that considers the constraints existing among the inputs. Namely, we adapt the hill-climbing-guided falsification approach described in Section II-B. The approach performs classical hill-climbing optimization over the search space Ξ; sampled points $\vec{u}_k$ are mapped to points $\vec{\pi}_k$ of the input space Ω_ψ, using the proportional transformation presented in Section IV. In this context, the fitness function ρ of Ω_ψ is given by the robustness value of the mapped input $\vec{\pi}_k$ for the specification φ, i.e., $\rho(\vec{\pi}_k) = [\![\mathcal{M}(\vec{\pi}_k), \varphi]\!]$. See the whole workflow in Fig. 1. We can notice that the hill-climbing algorithm (performed over the whole search space Ξ) has a *deformed* view of the fitness landscape of the input space Ω_ψ. The obtained deformed landscape depends on the priority used in the proportional transformation [see Section IV, Algorithm 2, and the two proportional transformations in Fig. 2(b)].

Example 3: Let us consider Example 2. The fitness landscape of the input space Ω_ψ (produced by a given objective function) is as shown in Fig. 3(a). By applying the proportional transformation using the two priorities S^1 and S^2 [as shown in Fig. 2(b)], we obtain the fitness landscapes in Fig. 3(b) and (c).

As we will show in the experiments in Section VI, the chosen priority can greatly affect the performance of the falsification. In the following sections, we consider three methods for selecting the priority: selecting one priority, considering all the priorities, or *learning* which priority is better.

Algorithm 3 Fixed-Priority Approach

Require: a system model $\mathcal{M}$, an STL formula φ, a constraint ψ, a priority of dimensions S, and a time budget K

1: **function** FALS-FIXED-PRIORITY($\mathcal{M}, \varphi, K, \psi, S$)
2: $\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$
3: **while** $\mathsf{rb} \geq 0$ and within the time budget K **do**
4: $k \leftarrow k + 1$
5: $\vec{u}_k \leftarrow$ HILL-CLIMB$\Big((\vec{u}_l, \mathsf{rb}_l)_{l\in\{1,\dots,k-1\}} \Big)$
6: $\vec{\pi}_k \leftarrow$ MAP-POINT($\Xi, \psi, S, \vec{u}_k$) ▷ Proportional transformation
7: $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\vec{\pi}_k), \varphi]\!]$ ▷ Robustness value of the mapped point
8: **if** $\mathsf{rb}_k < \mathsf{rb}$ **then** $\mathsf{rb} \leftarrow \mathsf{rb}_k$
9: $\vec{\pi} \leftarrow \begin{cases} \vec{\pi}_k & \text{if } \mathsf{rb} < 0, \text{ that is, } \mathsf{rb}_k = [\![\mathcal{M}(\vec{\pi}_k), \varphi]\!] < 0 \\ \text{Failure} & \text{otherwise, that is, no falsifying input found} \end{cases}$
10: **return** $\vec{\pi}$

Algorithm 4 All-Priorities Approach

Require: a system model $\mathcal{M}$, an STL formula φ, constraint ψ, priorities *Prior*, and a time budget K

1: **function** FALS-ALL-PRIORITIES($\mathcal{M}, \varphi, K, \psi, Prior$)
2: $\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$
3: **while** $\mathsf{rb} \geq 0$ and within the time budget K **do**
4: $k \leftarrow k + 1$
5: $\vec{u}_k \leftarrow$ HILL-CLIMB$\Big((\vec{u}_l, \mathsf{rb}_l)_{l\in\{1,\dots,k-1\}} \Big)$
6: $\Pi \leftarrow \{$MAP-POINT($\Xi, \psi, S, \vec{u}_k$) $\mid S \in Prior\}$ ▷ Prop. trans.
7: $\vec{\pi}_k \leftarrow \arg\min_{\vec{\pi}\in\Pi} [\![\mathcal{M}(\vec{\pi}), \varphi]\!]$ ▷ Selection of best mapped point
8: $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\vec{\pi}_k), \varphi]\!]$ ▷ Robustness computation
9: **if** $\mathsf{rb}_k < \mathsf{rb}$ **then** $\mathsf{rb} \leftarrow \mathsf{rb}_k$
10: $\vec{\pi} \leftarrow \begin{cases} \vec{\pi}_k & \text{if } \mathsf{rb} < 0, \text{ that is, } \mathsf{rb}_k = [\![\mathcal{M}(\vec{\pi}_k), \varphi]\!] < 0 \\ \text{Failure} & \text{otherwise, that is, no falsifying input found} \end{cases}$
11: **return** $\vec{\pi}$

A. Method 1: Fixed-Priority

In this approach, the user must give a priority order S. Algorithm 3 shows how the hill-climbing-guided falsification is modified to implement the proportional transformation with Fixed-Priority.

There are two differences: first, the algorithm now also considers a constraint ψ and, for computing the fitness of an input $\vec{u}_k$ sampled in the search space Ξ, it first maps it to a point $\vec{\pi}_k$ in the input space Ω_ψ using the proportional transformation (line 6), and then uses the robustness of $\vec{\pi}_k$ as fitness for $\vec{u}_k$ (line 7); second, the final falsifying input (if any) is a point $\vec{\pi}_k$ of the input space (line 9).

B. Method 2: All-Priorities

Although the proportional transformation is surjective, it changes the distribution of fitness on the base of the selected priority, and so it influences the performance of hill-climbing optimization. Therefore, different priorities lead to different falsification performance and different results. The current method is based on this observation, and so it considers *all* the priorities. Algorithm 4 shows the implementation of the All-Priorities approach.

At line 6, the approach now generates *all* the mapped points Π of $\vec{u}_k$, using the priorities contained in set *Prior* given as input.

The set *Prior* is built as follows. Initially, all the permutations of the dimensions $\{1 \dots, n\}$ are added to the set. However, some priorities are guaranteed to map the points in the same way. Therefore, such *equivalent priorities* are identified with these two rules.

1) Only the variables contained in the constraint ψ affect the result of the priority application. Given two priorities S^1 and S^2, if the relative order of variables contained in ψ is the same in S^1 and S^2, then they are equivalent.
2) Two variables are *independent* if they occur in the same conjunction ψ_i of ψ (recall that ψ is in DNF) and they are not in the same atomic proposition. Given two priorities S^1 and S^2, if they only differ in the relative order of independent variables, then they are equivalent.

So, *Prior* is the set of all nonequivalent permutations of $\{1 \dots, n\}$. Note that even if two priorities are not equivalent, a point can still be mapped to the same point under both priorities: hence, at line 6, duplicated points are removed. At lines 7 and 8, it determines the point $\overrightarrow{\pi}_k$ in Π having the minimum robustness rb_k. Finally, at lines 10 and 11, it returns a falsifying input $\overrightarrow{\pi}$ in the input space or reports a failure.

Algorithm 5 MAB-Priority Approach—Difference w.r.t. Algorithm 1

Require: a system model $\mathcal{M}$, an STL formula φ, constraint ψ, permutations $Prior = \{S^1, \dots, S^m\}$ of dimensions $\{1, \dots, n\}$, and a time budget K

1: **function** FALS-MAB-PRIORITY($\mathcal{M}, \varphi, K, \psi, Prior$)
2: $\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$
3: **while** $\mathsf{rb} \geq 0$ and within the time budget K **do**
4: $k \leftarrow k + 1$
5: $i_k \leftarrow \text{MAB}(Prior, \langle (S^{i_1} \dots S^{i_{k-1}}), (\mathsf{rb}_1 \dots \mathsf{rb}_{k-1}) \rangle)$ ▷ Selection of the priority
6: $\overrightarrow{u}_k \leftarrow \text{HILL-CLIMB}\Big(\big((\overrightarrow{u}_l, \mathsf{rb}_l) \big)_{l \in \{1,\dots,k-1\} \text{ such that } i_l = i_k} \Big)$ ▷ Hill climbing suggests $\overrightarrow{u}_k$ based on sampling history of S^{i_k}
7: $\overrightarrow{\pi}_k \leftarrow \text{MAP-POINT}(\Xi, \psi, S^{i_k}, \overrightarrow{u}_k)$ ▷ Proportional transformation
8: $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\overrightarrow{\pi}_k), \varphi]\!]$ ▷ Robustness computation
9: **if** $\mathsf{rb}_k < \mathsf{rb}$ **then** $\mathsf{rb} \leftarrow \mathsf{rb}_k$
10: $\overrightarrow{\pi} \leftarrow \begin{cases} \overrightarrow{\pi}_k & \text{if } \mathsf{rb} < 0, \text{ that is, } \mathsf{rb}_k = [\![\mathcal{M}(\overrightarrow{\pi}_k), \varphi]\!] < 0 \\ \text{Failure} & \text{otherwise, that is, no falsifying input found} \end{cases}$
11: **return** $\overrightarrow{\pi}$

12: **function** MAB($Prior, \langle (S^{i_1} \dots S^{i_{k-1}}), (\mathsf{rb}_1 \dots \mathsf{rb}_{k-1}) \rangle$)
13: $i_k \leftarrow \underset{z \in \{1,\dots,|Prior|\}}{\arg\max} \left(\texttt{Rew}(z, k-1) + c \sqrt{\frac{2 \ln(k-1)}{N(z, k-1)}} \right)$
14: **return** i_k

C. Method 3: MAB-Priority

Although the All-Priorities approach guarantees to find the *best* priority (i.e., the one mapping to points with minimum robustness), it is computationally expensive, as it requires to simulate all the mapped points. In this section, we propose a method that tries to *learn* the best priority during execution: it is based on the MAB problem, that has proven to be effective in other contexts for falsification [26].

We first provide an introduction to the MAB problem, and then we describe how we apply it to our context.

Multiarmed Bandit Problem: The MAB problem describes the situation where a gambler sits in front of a row $A_1, \dots, A_m$ of slot machines, each one giving, when its arm is played (i.e., in each attempt), a reward according to a prescribed (but unknown) probability distribution μ_i. The goal is to maximize the cumulative reward after a number of attempts, playing a suitable arm in each attempt. The best strategy of course is to keep playing the best arm $A_{\max}$, i.e., the one whose average reward $\mathsf{avg}(\mu_{\max})$ is the greatest. However, this best strategy is infeasible, because the distributions $\mu_1, \dots, \mu_m$ are initially unknown. Therefore, the gambler must learn about $\mu_1, \dots, \mu_m$ through attempts. A formal definition of the MAB problem is as follows.

Definition 14 (MAB Problem):

Input: Arms $(A_1, \dots, A_m)$, the associated probability distributions $\mu_1, \dots, \mu_m$ over $\mathbb{R}$, and a time horizon $H_T \in \mathbb{N} \cup \{\infty\}$.

Goal: Synthesize a sequence $A_{i_1} A_{i_2} \dots A_{i_H}$, so that the cumulative reward $\sum_{k=1}^{H_T} \mathsf{rew}_k$ is maximized. Here, the reward rew_k of the kth attempt is sampled from the distribution μ_{i_k} associated with the arm A_{i_k} played at the kth attempt.

Let $\langle (A_{i_1} \dots A_{i_k}), (\mathsf{rew}_1 \dots \mathsf{rew}_k) \rangle$ be a *history*, i.e., the sequence of arms played so far (here $i_1, \dots, i_k \in \{1, \dots, m\}$), and the sequence of rewards obtained by those attempts (rew_l is sampled from μ_{i_l}).

MAB-Based Falsification: In our context, an arm is a priority S. The reward of a priority S is based on the minimum robustness value observed when using S for mapping the sampled point (i.e., playing that arm). The hill-climbing algorithm implementing the MAB approach is shown in Algorithm 5 [differences with respect to (w.r.t.) Algorithm 1].

In line 5, an MAB algorithm is run to decide which priority S, taken from a set of priorities *Prior* (see Section V-B), must be executed in the kth attempt.

The function MAB takes as inputs: 1) priorities $Prior = \{S^1, \dots, S^m\}$ (i.e., the arms); 2) the history $(S^{i_1}, \dots, S^{i_{k-1}})$ of previously played arms; and 3) the history of robustness values $(\mathsf{rb}_1, \dots, \mathsf{rb}_{k-1})$ of the previously selected inputs.

Our MAB algorithm is based on the UCB1 (*upper confidence bound*) algorithm. UCB1 algorithm exemplifies the *exploitation* and *exploration* tradeoff over the set of arms. It is instantiated at line 12 of Algorithm 5: it returns the index i_k of the priority S^{i_k} that has the largest sum of *exploitation* and *exploration* score (line 13). Given a priority S^z, the *exploitation score* identifies the empirical reward $\texttt{Rew}(z, k-1)$, and it follows the formal definition in [12], that considers the robustness obtained in previous loops: the lower the observed robustness is, the higher the reward assigned to the arm is. The definition is as follows: $\texttt{Rew}(z, k-1) = (1 - \frac{\min_{l \in \{1,\dots,k-1\} \text{ s.t. } i_l = z} \mathsf{rb}_l}{\max_{l \in \{1,\dots,k-1\}} \mathsf{rb}_l})$.

The *exploration score* is a value negatively correlated to the number of attempts of the arm of priority S^z. At line 13, $N(z, k-1)$ identifies the number of attempts of priority S^z in the previous $k-1$ steps $S^{i_1} \dots S^{i_{k-1}}$. The scalar c is used to give more importance to either exploration or exploitation.

VI. Experimental Evaluation

In this section, we present the experiments we performed to evaluate the proposed approaches. In Section VI-A, we introduce two baseline approaches. Then, in Section VI-B, we introduce the experimental setup, and in Section VI-C, we illustrate the experiments and evaluate the results using some research questions.

A. Baseline Approaches

We compare the performance of our approaches with two baseline approaches proposed in [18]: 1) the *constraint embedding* method and 2) the *lexicographic method*. Both methods are based on penalties, i.e., when the input constraint is not satisfied, a penalty is added to the objective function. Therefore, the optimization algorithm considers not only the original objective function but also the penalty: it minimizes the fitness function with the aim of violating the specification φ while satisfying the input constraint ψ.

Constraint Embedding (CE) Method: This method embeds the constraint ψ into the specification φ, by trying to falsify the formula $\psi \rightarrow \varphi$. Namely, given an input signal $\mathbf{u}$ and the output signal $\mathcal{M}(\mathbf{u})$, the robustness $\rho(\langle \mathbf{u}, \mathcal{M}(\mathbf{u})\rangle) := [\![\langle \mathbf{u}, \mathcal{M}(\mathbf{u})\rangle, \psi \rightarrow \varphi]\!]$ is computed according to Definition 4, and the optimization algorithm is guided by ρ. Once ρ is negative, it implies that the constraint ψ is satisfied and the specification φ violated: so, a feasible falsification input has been found.

Lexicographic Method (LM): It uses a *global cost function GCF* that considers both the satisfaction of constraint ψ and the violation of the system specification φ. The method prioritizes the satisfaction of the constraint ψ: when ψ is unsatisfied, the (positive) degree $[\![\mathbf{u}, \neg\psi]\!]$ of dissatisfaction is added to *GCF* and it is guaranteed to dominate *GCF*. Instead, if the constraint is satisfied, *GCF* is only determined by the robustness $[\![\mathcal{M}(\mathbf{u}), \varphi]\!]$ related to the specification. The approach minimizes *GCF*, and, when this becomes negative for some $\mathbf{u}$, it means that $\mathbf{u} \models \psi$ and $\mathcal{M}(\mathbf{u}) \not\models \varphi$.

B. Experimental Setup

We experiment our approaches over the benchmarks used in the falsification community [27]. In order to make a comprehensive and reliable comparison, we select four Simulink models with 20 system specifications. The hardness of these specifications depends on their parameters; in our experiments, we vary the parameters to obtain problems of different difficulties. Each specification has been experimented with different input constraints taken from [18].

The constraints for the considered models usually predicate about the inputs over time. Therefore, we allow users to specify input constraints in STL, as done also in [18]; such constraints predicate over variables $u_1, \ldots, u_M$ (one for each input). However, the constraints we support in our approach are a combination of linear constraints (no temporal operators) defined over variables $\overrightarrow{u} = (u_{1,1}, \ldots, u_{1,M}, \ldots, u_{\mathsf{c},1}, \ldots, u_{\mathsf{c},M})$ (see Definitions 8 and 6), where, for each input u_i, there are c variables $u_{1,i}, \ldots, u_{\mathsf{c},i}$ (one for each control point). There is a straightforward translation between the two formats.

The specifications are reported in Table I. The input constraints are reported in Table I, where each constraint is in its two forms, namely, the format of Definition 8 and STL. Their IDs identify the corresponding Simulink models, described as follows.

Automatic Transmission (AT) [27]: It has two input signals, *th* (throttle) and *br* (brake), and produces output signals such as *speed*, *rpm*, *gear*, etc. The model is composed of six subsystems, one Stateflow chart, and 72 blocks in total. The *th* and *br* range over [0, 100] and [0, 325], respectively, each with five control points. We select specifications AT1, ..., AT14, regarding system's safety (see [12], [26], [27]). We consider five input constraints, covering both equalities and inequalities.

Abstract Fuel Control (AFC) [14]: It takes two input signals, *PA* (pedal angle) and *ES* (engine speed), and outputs a ratio μ reflecting the deviation of *air–fuel ratio* from its reference value. In our experiment, we set the range of $PA \in [8.8, 70]$ and $ES \in [900, 1100]$, each with five control points. The model is composed of 20 subsystems, and 271 blocks in total. Specifications AFC1 and AFC2 [12], [14] reason about the expected safety properties of the system. We specify two input constraints, one constraining the value of *ES* w.r.t. the value of *PA*, and another one constraining the value of *PA* over time.

Neural Network Controller (NN): It is a neural network controller for a magnet system from Mathworks. Specifications NN1 and NN2 [27] formalize the safety requirement about the position *Pos* of the magnet w.r.t. its reference value *Ref*. The input signal *Ref* ranges over $[1, 3]$ with four control points. The model is composed of 11 subsystems, including one neural network-based controller, and 104 blocks in total. We consider two input constraints: the first one requiring *Ref* to be nondecreasing, and the second one requiring *Ref* to be larger of 2.5 in at least one time point.

TABLE I
BENCHMARK SET ($\Delta_t(\mathbf{w}) = \mathbf{w}^t - \mathbf{w}$). (A) TEMPORAL SPECIFICATIONS φ. (B) INPUT CONSTRAINTS ψ

(a)

Spec. ID	Temporal specification in STL
AT1	$\square_{[0,30]}\ (speed < 120)$
AT2	$\square_{[0,30]}\ (gear = 3 \rightarrow speed \geq 20)$
AT3	$\square_{[0,30]}\ (gear = 4 \rightarrow speed \geq 35)$
AT4	$\neg(\square_{[10,30]}((50 < speed) \wedge (speed < 60)))$
AT5	$\neg(\square_{[10,30]}((53 < speed) \wedge (speed < 57)))$
AT6	$\square_{[0,29]}(speed < 100) \vee \square_{[29,30]}(speed > 75)$
AT7	$\square_{[0,29]}(speed < 100) \vee \square_{[29,30]}(speed > 70)$
AT8	$\square_{[0,30]}(rpm < 4770 \vee \square_{[0,1]}(rpm > 1000))$
AT9	$\square_{[0,30]}(rpm < 4770 \vee \square_{[0,1]}(rpm > 700))$
AT10	$\square_{[0,30]}(rpm < 3000) \rightarrow \square_{[0,20]}(speed < 65)$
AT11	$\square_{[0,10]}\ (speed < 50) \vee \diamond_{[0,30]}\ (rpm > 2700)$
AT12	$\square_{[0,10]}\ (speed < 50) \vee \diamond_{[0,30]}\ (rpm > 2520)$
AT13	$\square_{[0,26]}(\Delta_4(speed) > 40 \rightarrow \Delta_4(gear) > 0)$
AT14	$\square_{[0,27]}(\Delta_3(speed) > 30 \rightarrow \Delta_3(gear) > 0)$
AFC1	$\square_{[11,50]}(\mu < 0.23)$
AFC2	$\square_{[11,50]}(\diamond_{[0,10]}(\lvert\mu\rvert < 0.05))$
	$Req \equiv \square_{[0,16]}(\neg closeRef \rightarrow reachRef)$ $closeRef \equiv \lvert Pos - Ref\rvert \leq \alpha_1 + \alpha_2 \cdot \lvert Ref\rvert$ $reachRef \equiv \diamond_{[0,2]}(\square_{[0,1]}(closeRef))$
NN1	Req with $\alpha_1 = 0.003, \alpha_2 = 0.04$
NN2	Req with $\alpha_1 = 0.015, \alpha_2 = 0.03$
FFR1	$\neg\diamond_{[0,5]}(x, y \in [3.9, 4.1] \wedge \dot{x}, \dot{y} \in [-1, 1])$
FFR2	$\neg\diamond_{[0,5]}(x, y \in [3.95, 4.05] \wedge \dot{x}, \dot{y} \in [-0.5, 0.5])$

(b)

Constr. ID	ψ in format of Def. 8	ψ in STL
ψ^1_{AT}	$\bigwedge_{k=1}^{\mathsf{c}}(th_k = 0 \vee br_k = 0)$	$\square_{[0,30]}(th = 0 \vee br = 0)$
ψ^2_{AT}	$\bigwedge_{k=1}^{\mathsf{c}}(th_k \leq 20 \vee br_k \leq 50)$	$\square_{[0,30]}(th \leq 20 \vee br \leq 50)$
ψ^3_{AT}	$\bigwedge_{k=1}^{\mathsf{c}}(th_k \geq 3 \cdot br_k \vee br_k \geq 3 \cdot th_k)$	$\square_{[0,30]}(th \geq 3 \cdot br \vee br \geq 3 \cdot th)$
ψ^4_{AT}	$\bigwedge_{k=1}^{\mathsf{c}-1}(th_k \geq 70 \rightarrow th_{k+1} \leq 10)$	$\square_{[0,24]}(th \geq 70 \rightarrow th^6 \leq 10)$
ψ^5_{AT}	$\bigwedge_{k=2}^{\mathsf{c}}(th_k = 0 \vee br_k = 0) \wedge br_1 = 0$	$\square_{[6,30]}(th = 0 \vee br = 0) \wedge \square_{[0,6]}(br = 0)$
ψ^1_{AFC}	$\bigwedge_{k=1}^{\mathsf{c}}(PA_k \geq 50 \rightarrow ES_k \geq 1000)$	$\square_{[0,50]}(PA \geq 50 \rightarrow ES \geq 1000)$
ψ^2_{AFC}	$\bigwedge_{k=1}^{3}(PA_k \leq PA_{k+1})$	$\square_{[0,20]}(\Delta_{10}(PA) \geq 0)$
ψ^1_{NN}	$\bigwedge_{k=1}^{\frac{\mathsf{c}}{2}}\left(\bigvee_{j=1}^{2} Ref_k \geq 2.5\right)$	$\square_{[0,14]}\ (\diamond_{[0,6]}(Ref \geq 2.5))$
ψ^2_{NN}	$\bigwedge_{k=1}^{\mathsf{c}}(Ref_k \leq Ref_{k+1})$	$\square_{[0,15]}(\Delta_5(Ref) \geq 0)$
	$conf(u_i, u_j) \equiv \bigvee_{\sim\in\{\leq,\geq\}}(u_i \sim 0 \wedge u_j \sim 0)$	
ψ^1_{FFR}	$\bigwedge_{k=1}^{\mathsf{c}}(conf(u_{1_k}, u_{3_k}) \wedge conf(u_{2_k}, u_{4_k}))$	$\square_{[0,5]}\ (conf(u_1, u_3) \wedge conf(u_2, u_4))$

Free Floating Robot (FFR) [12]: The four input signals $u_1, u_2, u_3, u_4 \in [-10, 10]$ are boosters for a robot, and the goal is to steer it from $(x, y) = (0, 0)$ to $(4, 4)$ in a 2-D space. The model contains 32 blocks in total. We take four control points for each input signal. The imposed constraint is a real one [12]: u_1 and u_3 should be both positive or negative, and so should u_2 and u_4; otherwise, the boosters would conflict with each other.

Experimental Platform: In our experiments, we use Breach [4] (ver 1.2.13) with CMA-ES (the state of the art). The experiments are executed on an Amazon EC2 c4.2xlarge instance (2.9-GHz Intel Xeon E5-2666 v3, 15-GB RAM).

C. Evaluation

We performed a set of experiments using the two baseline approaches constraint embedding (CE) and lexicographic method (LM), and the three proposed approaches Fixed-Priority (Fix), All-Priorities (ALL), and MAB-Priority (MAB). Since Fixed-Priority requires to select a given priority, for each benchmark, we randomly selected two priorities S^1 and S^2: we name the two settings as $\texttt{Fix}_{S^1}$ and $\texttt{Fix}_{S^2}$.

In our context, an *experiment* consists of the execution of an approach A (CE, LM, $\texttt{Fix}_{S^1}$, $\texttt{Fix}_{S^2}$, ALL, or MAB) over a specification φ for 30 *trials*, with different random seeds; each single trial has been executed with a time budget K of 900 s. For each experiment, we define the *success rate* SR as the number of trials in which a falsifying input was found, and measure the average execution *time* of the successful trials. Note that time is correlated with the number of simulations, because simulation is much more computationally expensive than other steps, e.g., proportional transformation.

In the following, we compare two approaches $A_1, A_2 \in \{\texttt{CE}, \texttt{LM}, \texttt{Fix}_{S^1}, \texttt{Fix}_{S^2}, \texttt{ALL}, \texttt{MAB}\}$ by comparing SR using the nonparametric Wilcoxon signed-rank test with 5% level of significance [28]. The null hypothesis is that there is no statistical significant difference in applying A_1 or A_2 in terms of SR; if the null hypothesis is rejected, we check the alternative hypothesis that A_1 is better than A_2 (higher SR).

Experimental results are reported in Table II. The gray cells are local best performers: they have the best SR with minimum time.

Table III reports the results of the Wilcoxon signed-rank test between each pair of techniques in terms of SR.

We now analyze the results using three research questions.

RQ1 *Do the proposed approaches outperform the two baseline approaches?*

In this RQ, we want to assess whether we improve w.r.t. the state of the art. From Table II, we observe that sometimes the two baseline approaches CE and LM are not able to find any feasible falsifying input over the 30 trials: for example, AT7 for almost all the constraints, and AT8 and AT9 for ψ^4_{AT}. Our proposed approaches, instead, are almost always successful in at least one trial. Exceptions are $\texttt{Fix}_{S^1}$ with AT6, $\texttt{Fix}_{S^2}$ with AT7, and ALL with AT5, all under constraint ψ^1_{AT}.

Also when the two baseline approaches do find at least a falsifying input, our approaches in general perform better.

The statistical tests in Table III confirm the previous qualitative evaluation: all our approaches are statistically better than the two baseline approaches.

Note that both the baseline approaches and our proposed approaches modify the fitness landscape. However, in the baseline approaches, the fitness landscape is given by the *composition* of the degree of violation of the constraints and of the robustness; in this way, the falsification task performed by hill climbing is complicated. Moreover, note that the *scales* (i.e., orders of magnitude) of constraint violation and robustness may be very different, and this has been shown to affect the effectiveness of falsification algorithms [26]. In our approach, instead, hill climbing operates over a fitness landscape that, although deformed, it is only given by robustness.

We notice that in many cases the proposed approaches improve the baselines in time. This is reasonable: since the baselines make objective functions much more complex, they need more simulations (thus time) to find the falsifying input. Note that all infeasible samplings require simulation, so wasting time for falsification.

We want now to assess the effect of the proportional transformation on the execution time. For each experiment, we have computed the average simulation time (*time/#sim*) for all the techniques (not reported for the sake of space); note that, for the proposed approaches, such value also includes the time required by the transformation. We observed that there is no significant difference between the approaches, meaning that the computational cost of the transformation is negligible.

RQ2 *How do the three proposed approaches compare each other in terms of SR?*

We are here interested in assessing which is the best approach (among the three proposed ones) in terms of SR (in the given time budget). From Table II, we notice (as already observed in RQ1) that in very few cases the Fix approach may be not effective: this shows that, in some cases, choosing the *wrong* priority can affect the performance; a more detailed analysis will be given in RQ3. In one case, we observe that also the ALL method is not effective: although this is an exception considering all the other results of ALL, it is a signal that such an exhaustive approach may be ineffective, in particular when there are many priorities to consider.

Observing the statistical tests in Table III, we can draw more definitive conclusions. Selecting one particular priority (Fix) does not make any difference: $\texttt{Fix}_{S^1}$ and $\texttt{Fix}_{S^2}$ are statistically equivalent. Considering all the priorities (ALL) is sometimes better than considering only one (ALL is better than $\texttt{Fix}_{S^2}$), but sometimes it is equivalent (ALL is equivalent to $\texttt{Fix}_{S^1}$). This means that none of the two techniques, Fix and ALL, overcomes the other, because they highly depend on the considered specification and constraint. On the other hand, we observe that the MAB approach outperforms all the other proposed approaches: this shows that the MAB efficiently learns the priority that must be used because it provides the lower robustness.

RQ3 *How does the selection of priority S in the Fixed-Priority approach affect SR?*

As observed in RQ2, the priority chosen by the Fixed-Priority approach can influence the falsification ability. In this

TABLE II
EXPERIMENTAL RESULTS. (SR: THE NUMBER OF SUCCESSES OUT OF 30 TRIALS. #SIM: THE NUMBER OF SIMULATIONS. TIME IN SECONDS.) (a) AUTOMATIC TRANSMISSION (AT). (b) ABSTRACT FUEL CONTROL (AFC). (c) NEURAL NETWORK CONTROLLER (NN). (d) FREE FLOATING ROBOT (FFR)

(a)

		AT1			AT2			AT3			AT4			AT5			AT6			AT7		
		SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time
ψ^1_{AT}	CE	22	334	130.1	6	320	121.0	1	186	66.1	25	318	120.1	26	741	286.6	7	1094	481.6	0	-	-
	LM	9	477	194.2	0	-	-	0	-	-	15	291	116.0	9	860	348.5	1	399	154.1	0	-	-
	$\texttt{Fix}_{S^1}$	28	111	37.3	27	71	23.6	17	84	28.0	9	246	94.7	7	1014	418.9	0	-	-	2	315	127.0
	$\texttt{Fix}_{S^2}$	24	216	78.1	26	106	36.2	13	87	29.1	19	238	84.2	9	834	331.0	1	193	85.4	0	-	-
	ALL	30	382	124.0	30	145	47.0	29	269	88.2	28	464	161.4	0	-	-	29	988	343.5	19	1143	428.0
	MAB	30	744	251.3	30	57	28.1	30	127	49.5	30	473	164.6	16	2000	741.5	28	1603	604.6	14	1849	726.9
ψ^2_{AT}	CE	28	216	75.6	13	90	31.4	3	78	27.5	30	228	82.3	29	527	193.9	10	620	263.9	0	-	-
	LM	20	402	157.2	9	95	35.0	3	93	35.4	28	224	86.8	29	454	166.7	4	629	272.9	0	-	-
	$\texttt{Fix}_{S^1}$	29	93	32.3	18	109	38.6	21	97	33.9	20	176	66.5	16	917	390.9	7	563	232.0	8	463	200.1
	$\texttt{Fix}_{S^2}$	30	137	46.1	22	76	26.0	13	100	33.9	23	261	93.5	24	689	269.3	4	191	83.9	4	747	290.9
	ALL	30	410	132.1	30	359	117.8	28	308	101.3	24	755	273.0	0	-	-	27	885	306.3	21	1368	500.5
	MAB	30	480	160.5	30	119	47.0	30	228	82.0	30	497	172.1	14	2067	781.6	28	1578	601.9	27	1787	681.7
ψ^3_{AT}	CE	10	211	82.8	23	179	64.9	11	168	61.7	18	372	140.7	15	778	310.9	0	-	-	0	-	-
	LM	1	439	177.4	26	131	48.3	11	167	64.5	26	397	155.2	24	741	292.1	5	530	241.4	6	874	366.7
	$\texttt{Fix}_{S^1}$	30	110	37.2	20	79	27.3	16	78	26.8	29	219	78.6	23	852	337.8	16	489	193.6	13	748	303.6
	$\texttt{Fix}_{S^2}$	29	127	43.1	22	93	32.2	13	100	33.9	27	215	74.3	28	675	257.5	9	537	213.6	4	675	283.0
	ALL	30	540	177.0	27	390	132.2	25	706	240.9	30	431	149.2	2	1432	676.4	12	1309	486.6	3	1865	699.1
	MAB	30	736	249.3	30	114	45.3	30	246	88.0	30	347	122.3	16	2087	767.3	24	1710	634.2	15	1830	708.0
ψ^4_{AT}	CE	17	1105	460.7	23	89	31.4	14	70	24.2	30	114	40.2	29	448	162.8	2	1433	571.5	4	1170	487.7
	LM	14	1154	480.5	21	112	41.9	14	87	31.9	30	120	44.3	30	478	181.3	1	1055	469.6	1	1563	610.5
	$\texttt{Fix}_{S^1}$	14	543	208.0	18	92	30.8	10	127	43.1	28	108	36.0	26	504	180.8	5	228	89.3	4	411	165.9
	$\texttt{Fix}_{S^2}$	21	376	141.9	21	89	29.7	19	78	25.8	30	115	38.4	28	414	149.1	21	187	65.8	17	206	77.4
	ALL	30	795	273.7	21	263	92.3	16	181	63.6	30	191	64.2	27	1104	402.3	22	828	294.9	21	1437	401.2
	MAB	27	583	203.5	30	160	61.8	29	258	95.4	29	164	62.7	29	634	214.1	13	636	242.4	16	592	199.7
ψ^5_{AT}	CE	14	272	106.1	12	167	61.0	4	437	165.4	23	632	258.2	14	919	376.7	9	1093	490.2	0	-	-
	LM	12	706	276.6	0	-	-	0	-	-	28	538	217.1	15	947	402.3	0	-	-	0	-	-
	$\texttt{Fix}_{S^1}$	30	90	30.2	28	66	23.0	20	63	21.5	30	131	45.6	28	973	383.0	13	526	218.2	4	1186	463.6
	$\texttt{Fix}_{S^2}$	25	88	30.2	27	57	19.1	28	69	23.1	30	187	63.4	21	804	326.2	26	151	52.0	28	313	111.8
	ALL	30	435	140.6	23	189	80.3	30	321	106.2	26	318	110.5	6	1565	646.0	30	966	330.7	23	1225	441.1
	MAB	30	503	168.8	30	69	30.5	30	80	35.3	30	260	93.4	28	1351	497.0	26	745	271.8	21	1122	414.8

		AT8			AT9			AT10			AT11			AT12			AT13			AT14		
		SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time	SR	#sim	time
ψ^1_{AT}	CE	5	865	371.2	4	1631	655.1	18	541	215.8	7	679	291.2	9	559	229.3	6	325	118.5	16	281	101.5
	LM	3	201	77.5	9	439	170.3	3	747	310.1	2	1688	705.2	2	1519	617.6	9	236	88.2	8	289	109.2
	$\texttt{Fix}_{S^1}$	22	208	74.7	20	226	87.1	29	172	61.7	26	265	96.6	19	423	161.3	25	157	55.8	30	70	23.1
	$\texttt{Fix}_{S^2}$	22	198	70.6	17	342	132.2	28	332	121.2	19	245	86.7	14	316	120.9	22	246	88.2	27	74	25.3
	ALL	27	553	184.6	22	658	232.4	30	454	150.3	28	629	224.5	14	1176	483.2	28	995	351.8	30	281	95.0
	MAB	30	493	177.4	30	997	355.0	30	271	96.5	30	368	126.9	26	1536	570.3	30	1085	398.6	30	207	75.3
ψ^2_{AT}	CE	13	715	289.9	9	909	359.6	27	513	203.4	25	577	229.4	20	878	352.2	10	234	84.2	17	115	39.7
	LM	8	208	86.8	13	255	96.1	25	566	231.0	22	601	250.0	23	575	227.5	9	215	76.3	16	165	60.5
	$\texttt{Fix}_{S^1}$	25	87	30.7	19	111	40.6	29	116	40.3	26	159	57.3	22	380	146.7	27	142	50.4	30	72	25.0
	$\texttt{Fix}_{S^2}$	24	100	33.7	22	112	38.6	27	162	56.1	23	192	69.7	19	350	132.8	26	130	45.4	30	87	29.9
	ALL	29	470	157.6	24	564	197.5	30	444	146.7	28	666	237.0	5	1097	420.9	27	704	244.0	30	400	133.5
	MAB	30	511	180.8	30	704	248.1	30	220	80.5	30	376	133.0	30	1608	598.3	30	942	340.3	30	228	84.0
ψ^3_{AT}	CE	1	507	216.9	0	-	-	22	470	184.2	19	719	293.5	1	574	215.1	5	365	133.9	12	226	81.3
	LM	1	124	56.1	4	658	275.7	23	451	182.4	27	760	295.9	1	1883	831.8	11	412	153.0	14	238	85.1
	$\texttt{Fix}_{S^1}$	20	117	41.1	22	140	49.8	29	212	75.2	27	522	196.4	11	538	213.5	19	178	62.5	26	91	31.2
	$\texttt{Fix}_{S^2}$	19	138	48.2	14	150	52.9	28	275	98.1	26	491	182.6	6	394	161.0	18	177	63.7	28	98	34.0
	ALL	24	502	174.8	15	672	253.4	30	623	209.8	18	873	331.0	1	565	261.2	21	778	285.5	28	377	125.8
	MAB	30	866	315.7	30	851	299.0	30	394	139.9	29	851	296.4	8	1991	742.8	30	762	273.9	30	255	92.0
ψ^4_{AT}	CE	0	-	-	0	-	-	30	106	36.7	29	216	77.0	28	717	274.3	2	361	127.9	5	223	78.5
	LM	0	-	-	0	-	-	30	132	48.5	28	148	76.7	26	758	297.2	7	482	179.9	3	150	53.7
	$\texttt{Fix}_{S^1}$	14	155	52.8	15	156	53.8	30	97	31.5	30	198	67.4	25	814	311.8	8	97	32.3	21	73	24.3
	$\texttt{Fix}_{S^2}$	20	187	66.2	12	130	46.9	30	126	41.7	29	172	57.8	29	553	205.7	11	195	67.8	17	66	22.1
	ALL	13	379	146.1	10	572	172.5	30	180	61.0	30	288	97.9	19	1180	382.3	10	759	183.0	24	462	72.5
	MAB	22	441	164.7	24	514	181.5	30	174	67.4	30	255	94.7	28	862	305.2	18	352	131.1	26	155	56.3
ψ^5_{AT}	CE	11	1209	514.2	7	1050	427.0	26	423	165.8	14	530	222.9	14	617	243.3	5	401	149.0	5	343	124.8
	LM	5	413	173.1	4	405	164.6	20	472	199.8	16	587	253.7	4	873	376.0	6	454	169.6	8	317	116.1
	$\texttt{Fix}_{S^1}$	23	89	30.5	18	131	45.4	30	116	39.9	30	100	34.0	30	251	88.8	7	119	41.7	16	95	32.8
	$\texttt{Fix}_{S^2}$	15	91	30.5	13	130	44.4	30	177	61.1	18	248	92.4	13	400	154.4	13	86	30.1	22	93	32.2
	ALL	21	495	169.3	14	455	168.6	30	346	114.5	26	826	300.2	5	953	388.0	25	694	237.2	29	511	171.7
	MAB	30	571	206.9	30	687	242.1	30	228	82.8	30	291	104.4	28	1045	376.7	28	605	217.8	30	302	108.2

(b)

		AFC1			AFC2		
		SR	#sim	time	SR	#sim	time
ψ^1_{AFC}	CE	3	1181	771.4	10	438	282.7
	LM	7	901	591.2	10	452	295.4
	$\texttt{Fix}_{S^1}$	4	604	401.3	14	399	257.0
	$\texttt{Fix}_{S^2}$	10	787	516.6	11	215	140.5
	ALL	0	-	-	23	598	386.5
	MAB	0	-	-	4	975	604.0
ψ^2_{AFC}	CE	0	-	-	8	520	343.3
	LM	5	655	427.9	13	642	418.4
	$\texttt{Fix}_{S^1}$	7	492	316.8	14	543	358.2
	$\texttt{Fix}_{S^2}$	3	1028	702.4	11	587	380.3
	ALL	6	840	546.7	9	339	239.6
	MAB	5	550	355.8	13	637	412.1

(c)

		NN1			NN2		
		SR	#sim	time	SR	#sim	time
ψ^1_{NN}	CE	23	591	271.6	18	723	291.2
	LM	28	407	172.0	20	617	202.7
	$\texttt{Fix}_{S^1}$	30	303	121.7	26	461	192.3
	$\texttt{Fix}_{S^2}$	26	301	118.2	24	583	239.7
	ALL	28	549	211.4	20	556	226.9
	MAB	30	501	197.4	30	737	283.8
ψ^2_{NN}	CE	25	507	217.0	16	826	361.0
	LM	20	478	285.1	12	790	354.9
	$\texttt{Fix}_{S^1}$	26	379	152.6	22	459	192.5
	$\texttt{Fix}_{S^2}$	19	403	167.4	16	463	195.0
	ALL	27	543	218.9	18	716	333.1
	MAB	27	542	204.2	24	763	297.5

(d)

		FFR1			FFR2		
		SR	#sim	time	SR	#sim	time
ψ^1_{FFR}	CE	1	1492	477.4	1	1427	548.7
	LM	3	1456	346.1	0	-	-
	$\texttt{Fix}_{S^1}$	30	338	65.1	20	1467	398.6
	$\texttt{Fix}_{S^2}$	30	370	67.9	18	1325	370.1
	ALL	4	1751	460.7	0	-	-
	MAB	30	1034	226.5	3	920	316.9

RQ, we want to assess such influence. We selected eight specifications among those used previously and, for each of these, we also selected one constraint. For each specification, we run the Fixed-Priority approach using all the nonequivalent priorities in set *Prior* (see Section V-B). For each fixed priority, the falsification is run 30 times. Fig. 4 shows how the success rate SR changes by varying priorities.

We observe that, for some specifications (AT1, AFC1, NN2, and FFR2), the variability is low, meaning that the influence of the selected priority is small. Some of these specifications

TABLE III
WILCOXON SIGNED-RANK TEST BETWEEN TWO CONSIDERED APPROACHES A_1 AND A_2

A_1 \ A_2	CE	LM	$\texttt{Fix}_{S^1}$	$\texttt{Fix}_{S^2}$	ALL	MAB
CE	-	≡	✗	✗	✗	✗
LM	≡	-	✗	✗	✗	✗
$\texttt{Fix}_{S^1}$	✓	✓	-	≡	≡	✗
$\texttt{Fix}_{S^2}$	✓	✓	≡	-	✗	✗
ALL	✓	✓	≡	✓	-	✗
MAB	✓	✓	✓	✓	✓	-

Legend: (*nh* = null hypothesis, *ah* = alternative hypothesis)
✓: *nh* is rejected; A_1 is better than A_2 by *ah*.
≡: *nh* is not rejected.
✗: *nh* is rejected; A_2 is better than A_1 by *ah*.

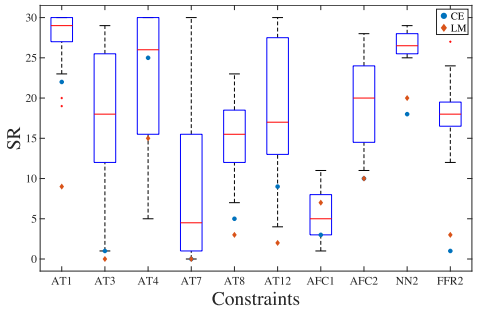

Fig. 4. Influence of selected priority S in the Fixed-Priority. (ψ^1_{AT} is used for AT specs, ψ^1_{AFC} for AFC specs, ψ^1_{NN} for the NN spec, and ψ^1_{FFR} for the FFR spec.)

are simple, such as AT1 and NN2, as we note that SR is high. AFC1 and FFR2 are not simple but they still exhibit small variance. For example, in FFR2, this is because the four different inputs (i.e., the four robot boosters) play similar roles in the system, and thus the priority over them does not matter.

On the other hand, for some specifications (e.g., AT3, AT4, AT7, and AT12), the variability is very high, going from SR of (almost) 0 to (almost) 30. This means that, in these cases, some priorities perform much better than others, because they tend to map points toward parts of the input space having low (possibly negative) robustness. In these specifications, the MAB-Priority approach is effective (see RQ2 and Table II): in AT3, AT4, and AT12 with ψ^1_{AT}, MAB-Priority approaches achieve the highest success rate; in AT7 with ψ^1_{AT}, it performs better than most of the other approaches.

We also mark the performance of `CE` and `LM` methods in Fig. 4. We note that, in almost all the cases, `CE` and `LM` do not perform as well as the median performance of the Fixed-Priority method; and in many cases, they perform even worse than the worst cases of the Fixed-Priority method. This strengthens our conclusion in RQ1 that the proposed approach performs generally better than the baseline approaches.

VII. RELATED WORK

Stochastic optimization-based falsification has drawn a lot of attention recently [1], [2], [4]–[12], [29], and has become one of the most effective approaches to CPS quality assurance. Most of research focuses on improving search techniques, especially handling the "exploration and exploitation" tradeoff. A comparison of falsification tools is given in [27].

However, the constrained falsification problem has not been investigated much. Indeed, solvers used in falsification frameworks (e.g., CMA-ES and `simulannealbnd`) in general do not support constraints. The first work considering constraints in falsification was [30], where timed automata were used to formalize the constraints and generate meaningful samples. However, that framework cannot be integrated into the state-of-the-art hill-climbing optimization-based falsification framework. Two more recent approaches [18] use penalties in the objective function to handle constraints; they are the *constraint embedding method* and *lexicographic method* that we used as baseline approaches in our experiments (see Section VI-A).

Constrained falsification is an instance of constrained optimization, which is a classical problem in optimization. Since the model in falsification is black box, white box methods used in constrained optimization cannot be used [31].

Instead, penalty-based approaches can also handle black box systems; they add *penalties* to the objective function for solutions not respecting the constraints. They are classified in different categories. Among these, *death penalty* [15] simply rejects infeasible solutions, while *adaptive penalty* uses in the objective function a factor proportional to the degree of constraints violation. Instances of adaptive penalties are those used for falsification in [18].

Our search space transformation idea is similar to a *decoding*-based method [19] for handling input constraints in evolutionary algorithms. Compared to our work, that technique has several weaknesses: first, the performance relies on defining an *origin* in the input space, which is feasible in simple problem settings, but not in our case; second, their method is not applicable to equality constraints.

Another mapping technique is employed in the *projected gradient descent* (PGD) method [31], where infeasible samples are projected to the closest point in the feasible area. In PGD, infeasible points are mapped to a restricted set of points, while feasible points stay the same. This leads to significant deformation of fitness landscapes, which makes the behavior of hill climbing in the search space very different from that in the input space. In contrast, our approach maps all points of the search space to the feasible area in a *proportionally spread way*, and exhibits smaller deformation of fitness landscapes.

VIII. CONCLUSION AND FUTURE WORK

In this article, we proposed a technique for handling constraints in hill-climbing-based hybrid system falsification. The approach defines a transformation that maps points from the unconstrained space to the constrained one; then it performs the search over the unconstrained space, guided by the robustness of the mapped points in the constrained space. The transformation requires to fix an order among the dimensions of the search space. We considered three ways to select the order: 1) Fixed-Priority; 2) All-Priorities; and 3) MAB-Priority. Experiments showed that the last one (based on MAB) outperforms the others in most cases.

As future work, we will consider a larger class of constraints (other than logical combinations of linear constraints), and of different complexity to better assess the scalability. Moreover, we also plan to consider other space mappings.

REFERENCES

[1] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, Sep. 2009.

[2] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. 8th Int. Conf. Formal Model. Anal. Timed Syst.*, vol. 6246, 2010, pp. 92–106.
[3] 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 Syst. Mag.*, vol. 36, no. 6, pp. 45–64, Dec. 2016.
[4] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Proc. 22nd Int. Conf. Comput.-Aided Verif. (CAV)*, vol. 6174, 2010, pp. 167–170.
[5] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems*. Heidelberg, Germany: Springer, 2011, pp. 254–257.
[6] A. Adimoolam, T. Dang, A. Donzé, J. Kapinski, and X. Jin, "Classification and coverage-based falsification for embedded control systems," in *Computer Aided Verification*. Cham, Switzerland: Springer, 2017, pp. 483–503.
[7] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, "Stochastic local search for falsification of hybrid systems," in *Automated Technology for Verification and Analysis*. Cham, Switzerland: Springer, 2015, pp. 500–517.
[8] J. V. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu, "Testing cyber-physical systems through bayesian optimization," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, pp. 1–18, 2017.
[9] G. Ernst, S. Sedwards, Z. Zhang, and I. Hasuo, "Fast falsification of hybrid systems using probabilistically adaptive input," in *Quantitative Evaluation of Systems*. Cham, Switzerland: Springer, 2019, pp. 165–181.
[10] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods*. Cham, Swirzerland: Springer, 2015, pp. 127–142.
[11] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski, "Multiple shooting, CEGAR-based falsification for hybrid systems," in *Proc. Int. Conf. Embedded Soft. (EMSOFT)*, New Delhi, India, Oct. 2014, pp. 1–10.
[12] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo, "Two-layered falsification of hybrid systems guided by Monte–Carlo tree search," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2894–2905, Nov. 2018.
[13] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. 40th Int. Conf. Softw. Eng.*, Gothenburg, Sweden, 2018, pp. 1016–1026.
[14] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain control verification benchmark," in *Proc. 17th Int. Conf. Hybrid Syst. Comput. Control*, 2014, pp. 253–262.
[15] T. Bäck, F. Hoffmeister, and H. Schwefel, "A survey of evolution strategies," in *Proc. 4th Int. Conf. Genet. Algorithms*, San Diego, CA, USA, Jul. 1991, pp. 2–9.
[16] O. Kramer, "A review of constraint-handling techniques for evolution strategies," *Appl. Comput. Intell. Soft Comput.*, vol. 2010, pp. 1–11. Apr. 2010.
[17] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. London, U.K.: Springer, 2012.
[18] Z. Zhang, P. Arcaini, and I. Hasuo, "Constraining counterexamples in hybrid system falsification: Penalty-based approaches," in *Proc. 12th Int. Symp. NASA Formal Methods (NFM)*, vol. 12229. 2020, pp. 401–419.
[19] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, vol. 7, no. 1, pp. 19–44, Mar. 1999.
[20] T. Akazaki and I. Hasuo, "Time robustness in MTL and expressivity in hybrid system falsification," in *Computer Aided Verification*. Cham, Switzerland: Springer, 2015, pp. 356–374.
[21] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," in *NASA Formal Methods*. Cham, Switzerland: Springer, 2017, pp. 357–372.
[22] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proc. 15th ACM Int. Conf. Hybrid Syst. Comput. Control*, 2012, pp. 125–134.
[23] G. Fainekos, B. Hoxha, and S. Sankaranarayanan, "Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with S-TaLiRo," in *Proc. Int. Conf. Runtime Verif.*, 2019, pp. 27–47.
[24] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Edinburgh, U.K., 2005, pp. 1769–1776.
[25] H. Abbas and G. Fainekos, "Convergence proofs for simulated annealing falsification of safety properties," in *Proc. IEEE 50th Annu. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, 2012, pp. 1594–1601.
[26] Z. Zhang, I. Hasuo, and P. Arcaini, "Multi-armed bandits for Boolean connectives in hybrid system falsification," in *Proc. Int. Conf. Comput.-Aided Verif.*, 2019, pp. 401–420.
[27] G. Ernst *et al.*, "ARCH-COMP 2019 category report: Falsification," in *Proc. 6th Int. Workshop Appl. Verif. Continuous Hybrid Syst.*, 2019, pp. 129–140.
[28] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Heidelberg, Germany: Springer, 2012.
[29] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte–Carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proc. 13th ACM Int. Conf. Hybrid Syst. Comput. Control*, 2010, pp. 211–220.
[30] B. Barbot, N. Basset, and T. Dang, "Generation of signals under temporal constraints for CPS testing," in *NASA Formal Methods*. Cham, Switzerland: Springer, 2019, pp. 54–70.
[31] E. K. Chong and S. H. Zak, *An Introduction to Optimization*. New York, NY, USA: Wiley, 2004.

Zhenya Zhang received the bachelor's degree in software engineering from Northwestern Polytechnical University, Xi'an, China, in 2014, and the master's degree in computer science from the Institute of Software Chinese Academy of Sciences, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Graduate University for Advanced Studies (SOKENDAI), Hayama, Japan.

He is a Research Assistant with the ERATO MMSD Project, National Institute of Informatics, Tokyo, Japan, and a JSPS Research Fellow. His research topics include search-based testing, stochastic optimization, and their application to CPSs.

Paolo Arcaini received the Ph.D. degree in computer science from the University of Milan, Milan, Italy, in 2013.

He is a Project Associate Professor with the National Institute of Informatics (NII), Tokyo, Japan. Before joining NII, he held an Assistant Professor position with Charles University, Prague, Czechia. His main research interests are related to search-based testing, fault-based testing, model-based testing, software product lines, and automatic repair.

Ichiro Hasuo received the Ph.D. degree (*cum laude*) in computer science from Radboud University Nijmegen, Nijmegen, The Netherlands, in 2008, where he was affiliated with the Security of Systems Group and supervised by Prof. B. Jacobs.

He is an Associate Professor with the National Institute of Informatics (NII), Tokyo, Japan. He is at the same time the Research Director of the JST ERATO "Metamathematics for Systems Design" Project, and the Director of the Global Research Center for Systems Design and Mathematics, NII. Before joining NII, he was an Assistant Professor with the Research Institute for Mathematical Sciences (RIMS), Kyoto University, Kyoto, Japan, and a Lecturer and an Associate Professor with the Department of Computer Science, University of Tokyo, Tokyo. His research interests include mathematical (logical, algebraic, and categorical) structures in software science (especially formal methods); abstraction and generalization of deductive and automata-theoretic verification techniques; integration of formal methods and testing; and their application to cyber-physical systems and systems with statistical machine learning components.

Dr. Hasuo is a recipient of the Hiroshi Fujiwara Encouragement Prize for Mathematical Sciences in 2012 (supported by the Mathematical Society of Japan), the Best Paper Awards at CONCUR 2014 (jointly with Natsuki Urabe) and at ICECCS 2018 (jointly with Etienne Andre and Masaki Waga); the JST PRESTO Researchership; and the JST ERATO Project (three were granted in 2016 out of proposals from all scientific fields). He is an Area Editor of the *International Journal of New Generation Computing* and a Steering Committee Member for the conference/workshop series CALCO and GaLoP. He has served in more than 50 program committees of international conferences and workshops.