Parametric Falsification of Many Probabilistic Requirements under Flakiness

Matteo Camilli*, Raffaela Mirandola†
* Department of Electronics, Information and Bioengineering (DEIB)
Politecnico di Milano, Italy
Email: matteo.camilli@polimi.it
† Karlsruhe Institute of Technology (KIT), Germany
Email: raffaela.mirandola@kit.edu

***Abstract*—Falsification is a popular simulation-based testing method for Cyber-Physical Systems to find inputs that violate a formal requirement. It employs optimization algorithms to minimize a robustness metric that defines the satisfaction of a given property over an execution trace. Despite falsification representing an established approach, detecting violations considering many, possibly independent, requirements simultaneously, under flaky simulations is an open problem. We address this problem by proposing a novel approach that combines parametric model checking and many-objective optimization. We use parametric model checking to shift part of the complexity of the problem offline. We pre-compute numeric constraints for the satisfaction of all requirements on a parametric specification of the testing scenario. Flaky violations are then detected using many-objective optimization to explore the space of changing factors in the scenario and push the parameters out of all precomputed constraints. The results of our empirical evaluation using four open-source evaluation subjects with increasing complexity (number of requirements) show that our approach can falsify many requirements simultaneously, without hiding their individual contribution. The effectiveness, in terms of quantity and severity of violations, is significantly higher than random search as well as two selected state-of-the-art baseline approaches. Furthermore, the extra offline computation yields a negligible cost.**

***Index Terms*—falsification, many-requirements, flakiness**

I. Introduction

Falsification is an established testing method for Cyber-Physical Systems (CPSs) to detect violations of safety (or dependability) requirements usually expressed through a formal notation [1], [2]. These techniques are black-box and are guided by quantitative fitness functions that can estimate how far a candidate test is from violating a given requirement. Test cases are sampled from the search input space using, for instance, randomized or meta-heuristic search algorithms [3]. To compute fitness functions, the CPS under test is simulated in a closed loop with the surrounding environment for each candidate test case to push the search toward failures.

Falsification, sometimes called online testing (when requirements are not expressed formally), is an effective technique for complex CPSs (e.g., autonomous vehicles, robotic systems) in which other exhaustive verification methods are impractical. As a result, falsification has been recently adopted by many CPS testing approaches [4]–[6]. Identifying defects by systematically testing against rigorous requirements makes it highly valuable for safety-critical industry sectors like automotive [7] and aerospace [8], especially where stringent safety standards and certification needs exist.

However, existing falsification approaches exhibit limitations. According to Haq et al. [6], the majority of them do not consider there are often many (possibly independent) requirements $R_1, ..., R_n$ that must be considered together in practice. Testing each requirement R_i often needs expensive simulations, making it impractical to evaluate each R_i individually [9]. This is especially true in cases where the number of requirements grows faster than the complexity of test scenarios. For example, autonomous vehicles must adhere to numerous traffic laws, even in relatively simple driving scenarios [7]. Many existing approaches test the conjunction $\Phi = R_1 \wedge ... \wedge R_n$. In this case, Φ is violated if R_i does not hold for any i. Although this solution is computationally efficient, it leads to traceability issues. Another significant limitation is that existing techniques do not account for nondeterminism [10]. Nondeterminism in CPS simulators is a significant challenge in testing, as it can lead to varying outcomes across multiple simulations, resulting in *flakiness* [11] (i.e., inconsistent test results). Moreover, assuming full control over all possible factors in a simulation is unrealistic. Testers typically concentrate on a limited subset of factors considered most relevant. However, this selective focus can lead to undersampling, which may further amplify flakiness. In this setting, failures become stochastic events. Consequently, falsification shall quantify the likelihood of requirement violations alongside identifying potential occurrences.

To overcome the aforementioned limitations, we present Parametric Falsification (PF), a novel falsification approach that deals with many *probabilistic* requirements. PF combines two distinct techniques: parametric model checking [3] and many-objective optimization [12]. The two techniques are applied in sequential steps: (1) *offline analysis* of the testing scenario, and then (2) *online optimization* to automatically generate test cases that estimate the likelihood of violations under potential flakiness of test results.

We leverage parametric model checking to reduce problem complexity by pre-computing, for each probabilistic requirement, numerical intervals that constrain the parameters of a stochastic specification for the testing scenario. These pre-computed intervals shift the complexity of falsification from requirements to specification parameters. As a result, the cost of test case generation remains unaffected by the number of requirements. The pre-computed probability bounds allow us to recast test case generation as a many-objective optimization problem [12] that accounts for flaky simulations potentially leading to stochastic violations.

The results of our empirical evaluation show that our approach can falsify multiple probabilistic (or even deterministic) requirements simultaneously under flaky test results. The effectiveness is significantly higher than random search and two selected state-of-the-art baseline approaches tailored to multiple requirements: Focused Falsification [13] (FF) and Many-Objective Reinforcement Learning for Online Testing [6] (MORLOT). We also show that the extra cost required by the offline analysis is negligible.

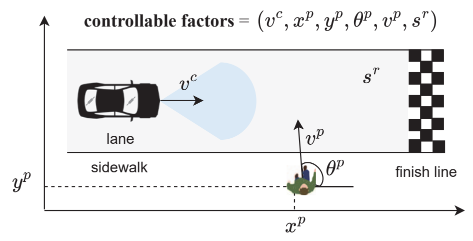
Our contributions can be summarized as follows:

- PF, a novel falsification approach for CPSs under many probabilistic requirements that combines parametric model checking and many-objective optimization;
- An empirical evaluation of PF considering four evaluation subjects (open-source CPS benchmark) in terms of the effectiveness of the online optimization step, and cost introduced by the offline analysis step;
- A publicly available replication package, including sources and instructions to replicate our experiments.

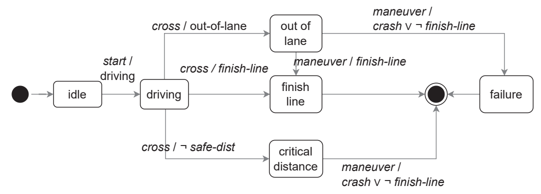
The paper is organized as follows. In Sec. II, we introduce our problem statement illustrated through a running example. In Sec. III, we introduce background notions. In Sec. IV, we describe in detail the two sequential steps (offline analysis and online optimization) of PF, while we discuss our empirical evaluation and threats to validity in Sec. V. In Sec. VI, we discuss related work. We conclude the paper in Sec. VII and provide details on the replication package in Sec. VIII.

II. Problem Statement

CPSs under test may have many requirements that must be considered simultaneously. Let us consider an Advanced Driver Assistance System [14] (ADAS) with an automated emergency braking component. The ADAS has constraints on the vehicle's speed limit, must stop at stop signs, and must adhere to other relevant traffic laws. To falsify such requirements, the CPS under test (e.g., the ADAS) is embedded into and interacts with its surrounding environment in a closed loop. Because of apparent risks and costs, falsification is performed through simulation rather than in-field testing. Simulation platforms allow the falsification process to take control of the relevant factors affecting the CPS, such as system and environmental factors. As shown in Fig. 1a, the vehicle drives from the left-hand side of an urban street toward the finish line, while a pedestrian starts crossing from the middle of the street. The tuple $(v^c, x^p, y^p, \theta^p, v^p, s^r)$ represents some controllable factors in the scene including vehicle speed, position, speed, orientation of the pedestrian, and road shape, respectively. Figure 1b shows a UML state diagram that represents this scenario. We use standard annotations attached to transitions to represent *events/measurable outcomes*. As an


controllable factors = $(v^c, x^p, y^p, \theta^p, v^p, s^r)$
v^c
s^r
lane
v^p
sidewalk
θ^p
finish line
y^p
x^p

(a) High-level schema of the test scenario.



(b) Specification of the test scenario

Fig. 1: ADAS example (adapted from Nejati et al. [14]).

example, from *driving* state, the pedestrian starts crossing. At this point, there are three possible target states according to the outcome measured from the simulation. For instance, the simulation enters the *critical distance* if the emergency braking component does not preserve the safe distance from the pedestrian. Relevant factors affecting this outcome include system configuration, such as the speed of the vehicle v^c as well as environmental factors, such as the initial position of the pedestrian x^p, y^p, and the shape of the road s^r (either straight, curved or ramped).

Simulated CPSs have inherent sources of nondeterminism from inevitable sensor and actuator inaccuracies to stochastic algorithms (e.g., vision and navigation), often leading to flaky tests [10], [11]. Further, testers often select a relatively small number of factors according to their domain knowledge (e.g., the tuple in Fig. 1a). This may further exacerbate flakiness. Indeed, the occurrence of a requirement violation is a stochastic phenomenon that can be triggered by the joint effect of controllable factors and dynamically changing factors that are not under the control of the tester. In this setting, falsification shall deal with probabilistic requirements to quantify the likelihood of a violation rather than simply spot its occurrence. A requirement for ADAS may state that "the probability of keeping a safe distance from the crossing pedestrian must be greater than 0.999." Another example is "the probability that the vehicle goes out of the lane is less than 0.01." Estimating the probability with high confidence is however expensive since it requires many simulations. If the requirements are not considered simultaneously, no practical budget may be sufficient to check each requirement exhaustively, as they must be falsified sequentially using a small portion of the budget.

III. Background

This section covers background concepts used in the following sections. We briefly introduce Markov Decision Processes,

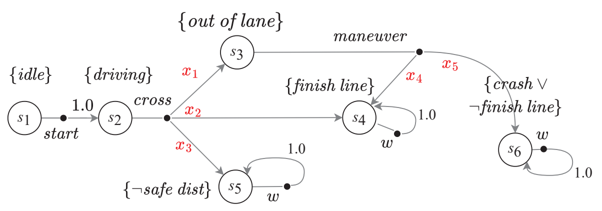


Fig. 2: MDP specification of the test scenario.

parametric model checking, and many-objective optimization.

A. Markov Decision Processes

A Markov Decision Processes [15] (MDP) is a Markov model that embeds stochastic behavior and nondeterministic inputs (or events). A MDP is defined as a tuple $\mathcal{M} = (S, s_0, A, \delta)$, where: S is a finite set of states ($s_0 \in S$ initial state); A is a finite set of actions; and $\delta : S \times A \rightarrow Dist(S)$ is a partial probabilistic transition function, where *Dist(S)* is the set of all discrete probability distributions over the countable set S. State transitions occur in two steps: a nondeterministic choice among the actions from state s: $A(s) = \{a \in A : \exists \delta(s, a)\}$; and then a stochastic choice of the successor state s', according to the probability distribution δ, such that $\delta(s, a)(s')$ represents the probability that a transition from s to s' occurs when a happens.

Figure 2 shows an MDP model that represents the test scenario illustrated in Fig. 1b. For example, from the *driving* state s_2, there is a single available action, *cross*, representing the pedestrian starting to cross. The three possible target states, s_3, s_4, and s_5, represent different measurable outcomes that may occur during the test scenario: the vehicle goes out of the lane, the vehicle successfully reaches the finish line, and the vehicle fails to maintain a safe distance, respectively. The likelihood of observing the target states is modeled through model *parameters* (e.g., x_1, x_2, and x_3) which define transition probabilities varying over regions of values.

B. Parametric model checking

Parametric model checking [16] is a verification technique in which formal models are parameterized, allowing the analysis to be performed on a family of models rather than a single instance. This technique identifies properties that hold universally across all instantiations or it discovers parameter values that lead to violations of critical properties.

Off-the-shelf model checkers, such as PRISM [16], support automated verification of Probabilistic Computation Tree Logic (PCTL) properties on parametric MDPs. PCTL is a branching-time logic that allows for probabilistic quantification of properties and is widely used as a specification language for probabilistic requirements in CPSs [17]. As an example, one may specify dependability requirements, such as "vehicle crash shall eventually occur in less than 5% of the cases." This requirement is formalized by the following PCTL property: $P_{<0.05}[\mathcal{F}\ crash]$, where $P_{<0.05}$ is the probabilistic quantification with upper-bound (0.05 probability), $\mathcal{F}$ is the "eventually" temporal operator, and *crash* is an atomic proposition that represents a measurable outcome. While a comprehensive knowledge of the syntax and semantics of PCTL is not required to understand our approach, readers interested in more detailed information can refer to Kwiatkowska et al. [16].

The result of parametric model checking of a PCTL property is given as a mapping from regions of parameters (hyper-rectangles) to truth values[1]. This result can be used to analyze how the parameters affect the satisfaction of the property.

C. Many-objective Optimization

Many-objective optimization [12] refers to solving multiple objectives simultaneously (typically more than 4 and up to 15) using meta-heuristic ES, such as NSGA-III [19] and MOEA/D [20]. These algorithms rely on multiple fitness functions (one per objective) that quantify the goodness of candidate solutions. The outcome is a set of solutions satisfying as many objectives as possible. In the context of software testing, many-objective optimization has been applied to cover multiple test targets (e.g., branches of a program). The idea is to recast such testing problems into optimization problems by defining candidate solutions (i.e., test cases) and fitness functions for all objectives. MOSA [21] is a well-known algorithm for test suite generation designed to tackle optimization problems with many structural coverage objectives. FITEST [22] extends and improves the efficiency of MOSA by decreasing the population size as the number of uncovered objectives decreases.

Alternative approaches dealing with many-objective optimization based on RL exist [6], [23]. MORLOT [6] is a recent approach to online testing of CPSs that embed neural networks and run in dynamic environments (e.g., autonomous vehicles). It leverages tabular-based RL [24] to generate sequences of environmental changes while keeping many reward tables (one per objective) to determine the changes so that they are more likely to achieve any of the uncovered objectives.

IV. Parametric Falsification

This section presents our novel approach Parametric Falsification (PF) that addresses the problem introduced in Sec. II.

A. Overview

Figure 3 shows an overview of the two main steps of PF: (1) offline analysis and (2) online optimization. The offline step employs parametric model checking to derive constraints on the simulated scenario specification. It transforms PCTL requirements into numerical intervals that bound probability values in a parametric MDP. These pre-computed bounds enable violation detection during the online step, reframing test case generation as an optimization problem that increases flakiness until requirements are violated. Here, test case generation costs depend on MDP parameters, remaining unaffected by the number of requirements.

[1]Parametric model checking ensures that any existing relationships between parameters are properly considered when calculating hyper-rectangles [18].

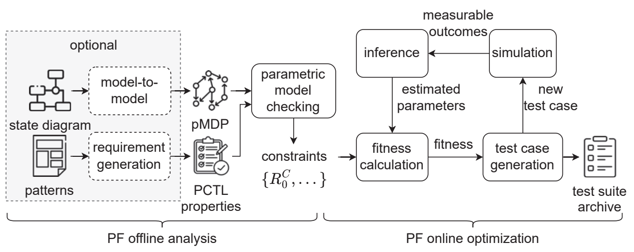


Fig. 3: Overview of Parametric Falsification.

B. Offline Analysis

Our approach adopts parametric MDPs to specify a test scenario, such as the model shown in Fig. 2. Optionally, the tester can describe the scenario using a UML state diagram, like the one illustrated in Fig. 1b, and then use a simple model-to-model transformation procedure that takes as input the diagram and produces the corresponding parametric MDP as follows. The procedure creates a one-to-one mapping between states in the UML diagram to states in the corresponding MDP. For instance, the initial *idle* state maps to the initial state s_1 in the MDP model. Each event in the state diagram maps to an action in the MDP model. For instance, the event *cross* received in the state *driving* maps to the *cross* action out of s_2. If a state-event pair has a single target state in the state diagram, the corresponding transition in the MDP is annotated with probability 1.0. If the pair has more than one target state, the transition is annotated with a parameter x_i that represents an unknown probability. As an example, the three transitions out of s_2 in Fig. 2 are annotated with three parameters x_1, x_2, x_3, each one representing the probability of the corresponding outcome given the event *cross*. Each state in the MDP is then associated with a set of atomic propositions. This set is the conjunction of the measurable outcomes attached to all incoming edges in the state diagram. As an example, let us consider s_5 and the corresponding state *critical distance* in the state diagram. This state represents a hazard occurring when the distance between the vehicle and the pedestrian is below a given threshold. This situation is detected whenever the vehicle is *driving* and the outcome $\neg$*safe-dist* measured by the simulator occurs. Therefore, $\neg$*safe-dist* becomes an atomic proposition that holds in the target state s_5.

As shown in Fig. 2, the specification includes absorbing (final) states that indicate the end of a test case execution. For systems designed to run indefinitely, a final state may represent a quiescent state where execution can be safely terminated.

Given the parametric MDP specification, requirements are expressed using PCTL properties. To simplify the definition of requirements, the tester may adopt standard specification patterns using structured natural language [25], [26]. Table I lists selected examples of patterns, requirements, and corresponding PCTL formulas for our running example[2]. Requirements are created from patterns by replacing each «condition» placeholder with a Boolean expression of atomic propositions, each «operator» with a comparison symbol ($>$, $<$, $\geq$, or $\leq$) and each «value» with a probability.

[2]Note that some requirements might express deterministic constraints. In such cases, PCTL formulas can impose probability ≥ 1 (or ≤ 0).

Given a set of PCTL requirements and the parametric MDP specification of the scenario, we use parametric model checking to calculate a mapping from sets of intervals (parameter valuations) to truth values (true/false) for each requirement. For each requirement R_j, we pre-compute the interval I_{x_i} for each parameter x_i. The set $R_j^c = \{I_{x_i}, \forall i\}$ represents the constraint for the satisfaction of R_j, expressed as acceptable values for all x_i. The set of constraints, one for each requirement, represents the input of the online optimization.

Note that requirements may express deterministic constraints like "a bad event shall never occur." In this case, constraints represent specific edge cases where each probabilistic interval I_{x_i} is either $[0, 0]$, or $[1, 1]$, respectively.

C. Online Optimization

Existing approaches to multi-requirement falsification either define a single objective to test the conjunction or split the budget according to given priority criteria [13]. In existing approaches covering independent objectives during falsification, the complexity of the problem depends on the number of requirements [27], [28]. These approaches do not explicitly consider the stochastic nature of requirement violations under flaky simulations. PF exploits offline analysis to shift the focus from requirements to parameters of the scenario specification. This means that the number of requirements can grow without affecting the complexity of the optimization step.

Our approach defines the optimization problem as follows. Let S be the CPS under test and $F = \{F_1, ..., F_K\}$ the set of controllable factors of the simulation (e.g., speed of the vehicle, and position of the pedestrian in our running example). Let a test case $t = (f_1, ..., f_K)$ be a tuple of values assigned to factors for a simulation of S. Given a set of probabilistic requirements $\{R_1, R_2, ...\}$, the degree of a violation for a requirement R_j produced by S under test case t depends on the actual value of the parameters compared to the numerical intervals prescribed by the constraint R_j^c. The actual value of the parameters can be measured by monitoring the simulation of S for t. In particular, a simulation runs the specified scenario multiple times to collect a sample of measurable outcomes out of each event. For example, the distance between the vehicle and the pedestrian can be used to check whether the state *critical distance* holds after the event *cross*. In the same way, the distance between the vehicle and the center of the lane can be used to check if the simulation yields the state *out of lane*. For each state-event, we count the occurrences of the corresponding target states observed in a simulation to estimate the actual value of the parameters. While simulation results are used to update the actual parameter values, the pre-computed constraints forming the oracle stay unchanged.

More formally, each state-event maps to a *categorical* distribution [29] whose concentration parameters $x_1, x_2, ...$ model

TABLE I: Selected examples of probabilistic requirements.

#	pattern name	pattern	requirement example	PCTL formula
R_0	probabilistic absence	the probability that «condition» does not hold globally must be «operator» than/to «value»	the probability that *critical distance* does not hold globally must be *greater* than 0.999	$P_{>0.999}[\mathcal{G}\ \neg critical\ distance]$
R_1	probabilistic existence	the probability that «condition» eventually holds must be «operator» than/to «value»	the probability that *out of lane* eventually holds must be *less* than 0.001	$P_{<0.001}[\mathcal{F}\ out\ of\ lane]$
R_2	probabilistic until	the probability that «condition» holds until «condition'» must be «operator» than/to «value»	the probability that $\neg out\ of\ lane$ holds until *finish line* must be *greater* than 0.995	$P_{>0.995}[\neg out\ of\ lane\ \mathcal{U}\ finish\ line]$

the probability of the corresponding target states. To estimate the concentration parameters we use statistical (Bayesian) inference [30] to update a *Dirichlet* distribution [31] (i.e., the natural conjugate prior of the categorical one). The inference process incorporates the knowledge gained from the samples (i.e., the occurrence of measurable outcomes) and estimates each parameter x_i through the expected value as follows: $\hat{x_i} = (c_i + \alpha_i)/(N + \sum_k \alpha_k)$, with α_i hyperparameter of the distribution[3], c_i number of occurrences of the corresponding target state, and N total number of samples. In addition to the expected value $\hat{x_i}$, we compute the corresponding Credible Interval (CI), denoted as $CI(x_i)$, by calculating the highest density region of the distribution for a given credibility level (e.g., 95%). CIs are used to adjust the parameters of an initially inaccurate MDP specification by progressively refining the posterior knowledge [31].

CIs are also used to determine the termination of a simulation for a test case t. When the final state of the specification is achieved, we start a fresh run from the initial state by restoring the initial configuration of the scenario (according to t) until one of the following alternative conditions holds:

- *no violation*: $\forall i, j\ CI(x_i) \subseteq I_{x_i}$ with $I_{x_i} \in R_j^c$
- *flaky violation*: $\exists i, j$: $CI(x_i) \cap I_{x_i} = \emptyset$ with $I_{x_i} \in R_j^c$

In essence, we observe a *flaky violation* for a given test case t, if the corresponding simulation S produces nondeterministic results, and the estimated probability of certain events fails to satisfy the constraints defined by requirements.

Let us consider the specification in Fig. 2. If the CI of x_1 (i.e., probability that *out-of-lane* state follows *driving*) is $[0.005, 0.01]$, there is a flaky violation for R_1 which, according to Table I, constrains x_1 in the range $[0, 0.001]$. Conversely, there is no violation if the simulation either deterministically avoids *out-of-lane* states or exhibits flakiness that remains within the specified boundaries.

At the end of the execution of a test case t, we measure the *severity of violation* for each x_i. The severity is a quantitative measure of the magnitude of flakiness. Let $I^*_{x_i}$ be the smallest interval for x_i considering all constraints R^c_j. We define the severity of violation using a *signed distance* as follows:

$$dist(x_i) = \begin{cases} \min(|\hat{x_i} - \inf I^*_{x_i}|, |\hat{x_i} - \sup I^*_{x_i}|) & \text{if } \hat{x_i} \in I^*_{x_i} \\ -\min(|\hat{x_i} - \inf I^*_{x_i}|, |\hat{x_i} - \sup I^*_{x_i}|) & \text{otherwise} \end{cases} \tag{1}$$

Essentially, $dist(x_i)$ is the smallest distance from violating a constraint if all the constraints are satisfied. In case there is a violation, $dist(x_i)$ is a negative value measuring the smallest distance from the farthest constraint. Let us consider $I^*_{x_1} = [0, 0.001]$. If $\hat{x_1}$ is 0.0007, there is no violation and the smallest distance (from the upper bound) is 0.0003. If $\hat{x_1}$ is 0.01, there is a violation as the estimate is higher than the upper bound. In this case, the distance is negative and is equal to -0.009.

Ideally, we would like to find those test cases that yield the smallest negative *dist* value for all x_i. These represent test cases where the likelihood of violating one or more requirements is higher and, therefore, they yield higher severity. For this reason, our optimization problem uses n *dist* functions as objectives $dist(x_1), ..., dist(x_n)$: one objective function for each specification parameter. The set of controllable factors F defines instead the search space.

PF addresses the optimization problem described above by using many-objective ES. Similar to existing work, PF uses the notion of archive to keep the set of test cases satisfying the objectives. Algorithm 1 presents the pseudo-code. It takes as input a set of specification parameters X, a set of constraints C (each one for each requirement), and returns an archive (i.e., a test suite) A that aims to maximally achieve individual objectives, that is, minimize $dist(x_1), ..., dist(x_n)$ according to constraints C. The algorithm begins with the initialization of A, and the population P (lines 1–2). The algorithm simulates the scenario according to each test case in P to estimate the value of the parameters (line 3). Given the estimates, the algorithm calculates fitness scores D by using the function *dist* for each parameter (line 4) and then updates the archive by including the test cases which obtain negative scores (line 5). Until the search runs out of budget, the algorithm repeats the following steps. It generates a set of new test cases P' from the previous population P using *crossover* and *mutation* operators (line 7), simulates the scenario according to P' and updates the archive base on the fitness score (line 8-10) as in the initialization, and then creates the next generation P from the current one and P' by using *selection* operators based on D (line 11). The algorithm ends by returning the archive A including the test cases violating at least one requirement.

V. Evaluation

This section describes the empirical evaluation of PF. We answer the following research questions:

RQ1: What is the effectiveness of PF compared to other existing approaches for multi-requirement falsification in terms of detected flaky violations?

[3]The inference starts from a non-informative prior ($\alpha_i = 0.5$ for all i).

Algorithm 1: Pseudo-code of PF

```
Input: Set of parameters $X = \{x_i\}$, Set of constraints $C = \{R_j^c\}$
Output: Archive (of test cases) $A$
1  Archive $A \leftarrow \emptyset$
2  Set of test cases $P \leftarrow initPopulation(|X|)$
3  Set of estimates $\hat{X} \leftarrow simulateScenario(P)$
4  Set of distance values $D \leftarrow calculateDist(\hat{X}, C)$
5  $A \leftarrow updateArchive(A, P, D)$
6  while budget is not finished do
7      $P' \leftarrow generateOffspring(P)$
8      $\hat{X} \leftarrow simulateScenario(P')$
9      $D \leftarrow calculateDist(\hat{X}, C)$
10     $A \leftarrow updateArchive(A, P, D)$
11     $P \leftarrow nextGeneration(P, P', D)$
12 end
```

TABLE II: Complexity of the selected study subjects.

subject	states	transitions	search space	parameters	requirements
ADAS1 [14]	6	9	6	5	3
ADAS2 [14]	6	9	6	5	6
RR [32]	13	22	9	5	6
UAV [33]	23	86	7	10	12

RQ2: What is the effectiveness of PF compared to other existing approaches for multi-requirement falsification in terms of severity of flaky violations?

RQ3: What is the overhead of the offline analysis of PF?

To answer RQ1 and RQ2, we compare the results obtained by using different falsification approaches using the same testing budget (total number of simulations). We answer RQ1 by comparing the number of violations for each individual requirement, while we answer RQ2 by comparing the severity of the violations in terms of distance between the actual value of parameters and constraints. To answer RQ3, we quantify the extra cost (in execution time) required by the offline analysis.

A. Design of the Evaluation

1) Evaluation subjects: We designed and carried out an experimental campaign to evaluate PF and the other baseline approaches using the same budget and the same set of evaluation subjects: 4 open-source CPS benchmarks. Table II lists the subjects and the studies describing them in more detail. We selected existing benchmarks having nontrivial complexity according to different factors, such as number of states and transitions in the scenario specification, number of variables in the search space, number of specification parameters, and number of requirements.

ADAS1 represents a bigger version of our running example in Sec. II. The level of complexity in the scenario specification is still relatively low (6 states and 9 transitions). ADAS2 is the same as ADAS1 but has twice as many requirements (6 rather than 3).

RR represents a search and rescue robot that supports emergency circumstances (e.g., fire, hurricane) to carry out rescue tasks in a target location. The robot navigates in the search area and avoids obstacles using vision sensors and LiDAR. The complexity of the scenario specification is higher (13 states, 22 transitions), while we keep the same number of specification parameters and requirements as in ADAS2. The search space has 9 factors including system properties (e.g., energy level, robot speed) and environment properties (e.g., lighting conditions, presence of smoke).

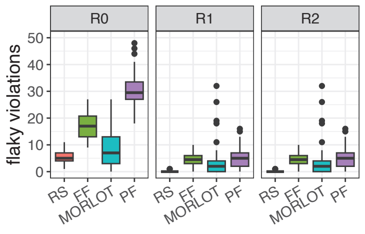
UAV represents an autonomous team of unmanned aerial vehicles carrying out a surveying mission in a hostile environment. The objective of the team in this scenario is to reach mission targets on the ground (through downward-looking sensors) and, at the same time, avoid threats (via forward-looking sensors). The complexity of the scenario specification and the number of parameters is higher (23 states, 86 transitions, 10 parameters). The number of requirements is doubled compared to RR (12 requirements), while search space has 7 factors including system properties (e.g., flying speed, team formation) and environment properties (e.g., weather conditions, number of threats).

We test all subjects leveraging an existing platform for simulating the stochastic behavior of CPSs [32]. The simulation platform allows us to control the effect of changing factors in the search space through a combination of custom linear, non-linear, and non-convex functions. We also controlled the intervals constraining the specification parameters to avoid trivial cases (i.e., requirements always satisfied or violated). The details of the simulation platform, functions, and constraints can be found in the supporting material (see Sec. VIII).

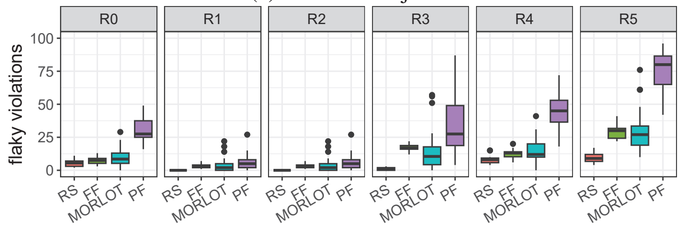
2) Methods under comparison: For comparison with PF, we use two existing approaches for multi-requirement falsification: Focused falsification [13] (FF) and Many-Objective Reinforcement Learning for Online Testing [6] (MORLOT). FF splits the budget among requirements and runs smaller falsification sub-problems focusing on one requirement at a time but computing the fitness also for the others to start the search from good candidate test cases. Each sub-problem controls only the subset of factors for which the requirement is sensitive. Subsets of factors are defined automatically through a preliminary global sensitivity analysis (using 30% of the budget). MORLOT [6] uses tabular-based RL to generate sequences of environmental changes by keeping a reward table for each objective (i.e., specification parameter in our case). Reward tables are used to determine the changes that are more likely to achieve any of the uncovered objectives. To make MORLOT comparable with PF, we keep controllable factors constant within each simulation, allowing us to collect samples and estimate the probability of violations under flaky results. In contrast, RL explores and exploits environmental changes across different simulations to cover all objectives.

We also use Random Search (RS) as a baseline. RS generates assignments to the factors in the search space using uniform random sampling. It also maintains an archive for all the assignments that lead to requirements violations. We use this baseline to get insights into the complexity of the falsification problem and quantify the relative effectiveness of other approaches: PF, FF, and MORLOT.

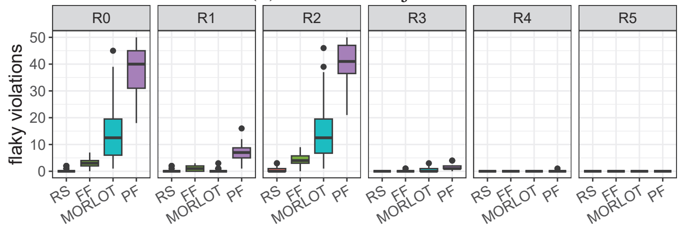
Note that one might also consider surrogate-assisted methods [5], [14]. We do not exclude the exploration of these approaches (to reduce the cost) which remain, for now, outside the scope of our work.



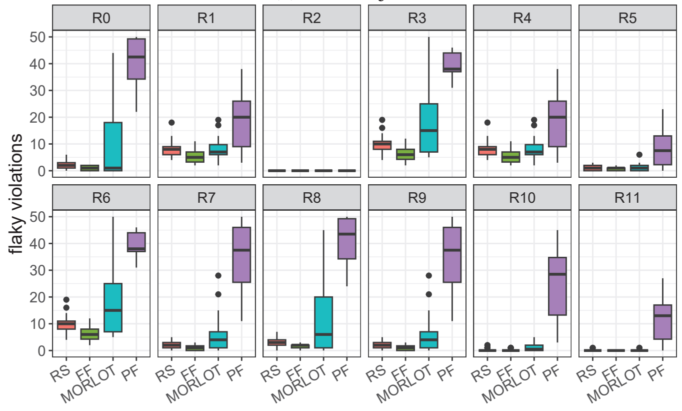
(a) ADAS1 subject



(b) ADAS2 subject

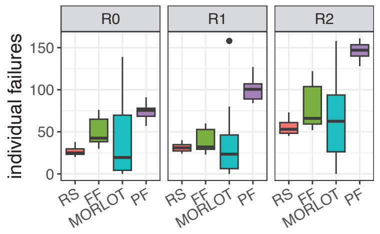


(c) RR subject

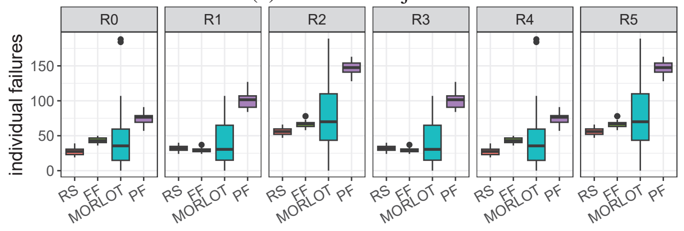


(d) UAV subject

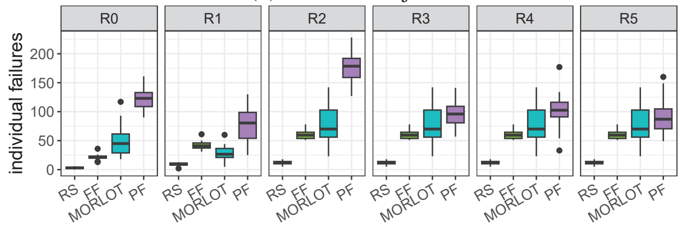
Fig. 4: Number of flaky violations (the higher, the better).



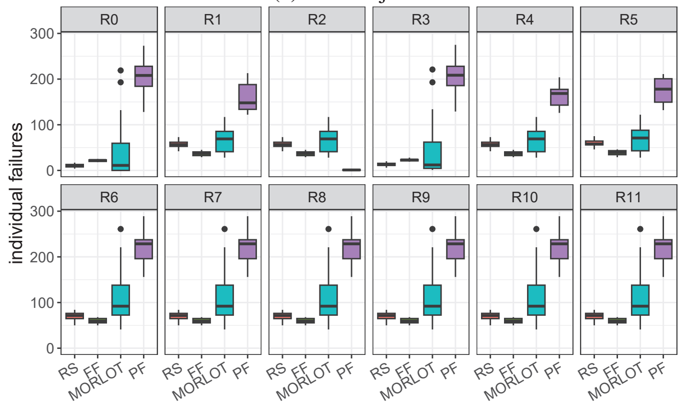
(a) ADAS1 subject



(b) ADAS2 subject



(c) RR subject



(d) UAV subject

Fig. 5: Number of individual failures (the higher, the better).

3) Statistical tests: To reduce the risk of obtaining results by chance, we account for randomness in all the approaches by repeating the falsification process 30 times with the same budget, that is, 900 total simulations. Based on our preliminary evaluations, we found that the severity of the flakiness (i.e., our fitness) reaches a plateau within this budget. According to the guideline introduced by Arcuri et al. [34], we apply the non-parametric Mann–Whitney U tests [35] to evaluate the statistical significance of the results. We also measure Vargha and Delaney's $\hat{A}_{AB}$ [36] to compute the effect size of the difference between A and B. We adopt the following standard classification: effect size $\hat{A}_{AB}$ $(= 1 - \hat{A}_{BA})$ is small, medium, and large when its value is greater than or equal to 0.56, 0.64, and 0.71, respectively.

4) Evaluation testbed: All experiments have been executed on a desktop machine running UBUNTU 22.04.1 LTS, equipped with Intel Xeon Silver 4114 CPU at 2.20GHz (16 cores) and 32GB RAM.

B. Evaluation Results

1) Number of flaky violations (RQ1): To answer RQ1, we executed PF and the selected baseline approaches for all evaluation subjects using the same budget (total simulations). We compare the approaches in terms of number of flaky violations found per each probabilistic requirement. There is no specific target for the number of violations; however, a higher count is preferable. A larger number of violations usually indicates a faster rate of discovery, resulting in increased

TABLE III: Statistical comparison results for RQ1.

subject	comparison		R_0		R_1		R_2		R_3	
	A	B	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$
ADAS1	FF	RS	5.08e-09	**0.99**	1.76e-08	**0.96**	1.76e-08	**0.96**	-	-
	MORLOT	RS	5.01e-02	0.65	2.30e-04	**0.76**	2.30e-04	0.76	-	-
	PF	RS	2.83e-09	**1.00**	1.65e-07	**0.92**	1.64e-07	**0.92**	-	-
	FF	MORLOT	5.25e-05	**0.80**	2.59e-02	0.66	2.59e-02	0.66	-	-
	PF	FF	6.97e-10	**0.96**	7.72e-01	0.52	7.32e-01	0.53	-	-
	PF	MORLOT	1.36e-09	**0.96**	6.92e-02	0.64	6.92e-02	0.64	-	-
ADAS2	FF	RS	1.94e-03	**0.76**	6.64e-10	**1.00**	6.64e-10	**1.00**	2.42e-09	**1.00**
	MORLOT	RS	1.99e-03	**0.76**	3.46e-05	**0.80**	3.44e-05	**0.80**	1.23e-06	**0.90**
	PF	RS	2.87e-09	**1.00**	1.90e-08	**0.95**	1.89e-08	**0.95**	2.69e-09	**1.00**
	FF	MORLOT	1.93e-01	0.40	3.16e-01	0.58	3.46e-01	0.57	6.37e-03	0.70
	PF	FF	2.76e-11	**1.00**	5.05e-02	0.64	5.05e-02	0.64	1.03e-04	**0.79**
	PF	MORLOT	6.26e-10	**0.97**	3.21e-02	0.66	3.38e-02	0.66	4.31e-05	**0.81**
RR	FF	RS	4.87e-07	**0.90**	4.59e-04	**0.77**	8.77e-08	**0.94**	2.14e-02	0.58
	MORLOT	RS	1.69e-11	**0.99**	8.34e-01	0.51	4.66e-11	**0.99**	3.05e-04	0.68
	PF	RS	7.74e-12	**1.00**	1.20e-11	**0.99**	1.24e-11	**1.00**	1.41e-09	**0.90**
	MORLOT	FF	7.93e-08	**0.90**	7.13e-05	0.23	5.80e-07	**0.88**	5.71e-02	0.61
	PF	FF	2.72e-11	**1.00**	7.68e-10	**0.96**	2.75e-11	**1.00**	4.25e-07	**0.85**
	PF	MORLOT	4.74e-07	**0.88**	1.33e-11	**0.99**	1.60e-08	**0.99**	1.16e-03	**0.73**
UAV	FF	RS	6.75e-03	0.30	3.38e-04	0.23	NA	NA	1.55e-05	0.18
	MORLOT	RS	4.60e-01	0.56	9.94e-01	0.49	NA	NA	2.75e-02	0.67
	PF	RS	2.73e-11	**1.00**	2.38e-05	**0.82**	NA	NA	2.88e-11	**1.00**
	MORLOT	FF	1.13e-01	0.62	1.02e-03	**0.75**	NA	NA	1.42e-06	**0.86**
	PF	FF	2.18e-11	**1.00**	7.07e-08	**0.90**	NA	NA	2.87e-11	**1.00**
	PF	MORLOT	2.21e-10	**0.98**	3.91e-05	**0.81**	NA	NA	5.18-10	**0.97**
subject	**comparison**		R_4		R_5		R_6		R_7	
	A	B	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$
ADAS2	FF	RS	1.51e-04	**0.82**	2.70e-09	**1.00**	-	-	-	-
	MORLOT	RS	1.21e-03	**0.77**	3.17e-08	**0.96**	-	-	-	-
	PF	RS	2.92e-09	**1.00**	2.95e-09	**1.00**	-	-	-	-
	FF	MORLOT	0.53	0.45	0.36	0.57	-	-	-	-
	PF	FF	3.19e-11	**0.99**	2.83e-11	**1.00**	-	-	-	-
	PF	MORLOT	3.76e-10	**0.97**	3.44e-10	**0.97**	-	-	-	-
RR	FF	RS	NA	NA	NA	NA	-	-	-	-
	MORLOT	RS	NA	NA	NA	NA	-	-	-	-
	PF	RS	1.60e-01	0.53	NA	NA	-	-	-	-
	FF	MORLOT	NA	NA	NA	NA	-	-	-	-
	PF	FF	1.60e-01	0.53	NA	NA	-	-	-	-
	PF	MORLOT	1.60e-01	0.53	NA	NA	-	-	-	-
UAV	FF	RS	3.38e-04	0.23	1.48e-01	0.40	1.55e-05	0.17	2.99e-03	0.28
	MORLOT	RS	9.94e-01	0.50	5.05e-01	0.55	2.75e-02	0.67	3.18e-03	0.66
	PF	RS	2.38e-05	**0.82**	1.82e-07	**0.89**	2.87e-11	**1.00**	2.63e-11	**1.00**
	FF	MORLOT	1.02e-03	**0.75**	3.29e-02	0.65	1.42e-06	**0.86**	1.79e-04	**0.78**
	PF	FF	7.07e-08	**0.90**	1.20e-08	**0.92**	2.87e-11	**1.00**	2.13e-11	**1.00**
	PF	MORLOT	3.91e-05	**0.81**	7.72e-07	**0.87**	5.18e-10	**0.97**	1.27e-10	**0.98**
subject	**comparison**		R_8		R_9		R_{10}		R_{11}	
	A	B	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$
UAV	FF	RS	3.71e-04	0.24	2.99e-03	0.28	2.86e-01	0.45	1.60e-01	0.47
	MORLOT	RS	1.85e-01	0.60	3.17e-02	0.60	1.67e-02	0.66	1.00	0.50
	PF	RS	2.74e-11	**1.00**	2.63e-11	**1.00**	7.70e-12	**1.00**	5.06e-11	**0.96**
	FF	MORLOT	8.53e-03	0.70	1.79e-04	**0.78**	1.03e-03	0.70	1.60e-01	0.53
	PF	FF	2.26e-11	**1.00**	2.13e-11	**1.00**	4.05e-12	**1.00**	1.63e-11	**0.97**
	PF	MORLOT	2.85e-10	**0.97**	1.27e-10	**0.98**	4.07e-11	**0.99**	5.06e-11	**0.96**

efficiency of the approach. We also repeat the experiments using a non-probabilistic interpretation of this latter metric, that is, we count the number of failures under the assumption that the requirements are classical (i.e., deterministic) rather than probabilistic. This analysis determines whether the performance differences between PF and the baseline methods can be attributed to the probabilistic interpretation of requirements. Additionally, it evaluates whether the approach remains effective in scenarios where the requirements are strictly deterministic.

To use MORLOT, we need to discretize the search space (in terms of states and actions) and define rewards specific to each evaluation subject as the method relies on a tabular-based RL, as anticipated in Sec. III. Based on preliminary experiments we defined a discretization that yields requirement violations and, at the same time, limits the state space explosion. Note that this may lead to a sub-optimal discretization, however, a systematic optimization would require more budget (in simulations) compared to the other approaches and, as a consequence, an unfair comparison. Further details about the discretization for each evaluation subject can be found in the supporting material (See Sec. VIII).

Concerning rewards, we define one reward function for each specification parameter of the evaluation subjects. Since we want to cause violations, we need higher reward values for more critical situations. Thus, for all parameters, we measure the distance between the actual value and the corresponding closest boundary (the smaller the distance, the more critical). According to the guideline introduced by Haq et al. [6], we adopt the following reward function to capture this behavior: $reward(x_i)$ is equal to $1/dist(x_i)$, if $dist(x_i) > 0$, equal to 10^6, otherwise. The constant 10^6 represents the maximum reward for any violation. Note that each value $dist(x_i)$ is between 0 and 1, so every parameter contributes equally to the reward function.

Besides the definition of states, actions, and rewards, we tuned RL parameters following the guideline reported by Mnih et al. [37]. To make the method more exploratory at first but gradually more exploitative, we dynamically decrease the probability ϵ of choosing a random action from 1.0 to 0.1 in the first 20% of the budget and then we keep it constant at 0.1. We set the learning rate $\alpha = 0.01$ and the discount factor $\gamma = 0.9$, as suggested by Haq et al. [6].

To use our approach PF, we need to select a specific optimization algorithm. In our experiments, we use NSGA-III [19], one of the popular and widely used algorithms for many-objective optimization problems. We tuned the parameters (mutation and crossover rates) following the guidelines introduced by the original studies [19]. We set the population size equal to the number of reference directions (i.e., twice the number of objectives in our case). Note that one may consider other algorithms, such as MOEA/D [20]. However, studying the effectiveness of PF by varying the specific evolutionary algorithm is out of the scope of this work.

Results. Figure 4 shows the distribution of violations for RS, FF, MORLOT, and PF over 30 repeats for each evaluation subject and each requirement. Table III shows the results of the statistical tests between different approaches. Columns A and B indicate the two approaches being compared. Columns p-value and $\hat{A}_{AB}$ indicate the statistical significance and effect size, respectively, when comparing A and B in terms of violations. Statistical significance (p-value < 0.05) is highlighted using gray cells, while boldface denotes a large effect size ($\hat{A}_{AB} > 0.71$). We use "NA" when both A and B cannot find any violation for a given requirement, while "-" indicates the requirement is missing.

The difference between PF and RS is statistically significant for all requirements and evaluation subjects while it is significant in 80% of the requirements when PF is compared to FF and MORLOT. In all these cases, $\hat{A}_{AB}$ is always

TABLE IV: Frequency of violations.

subject	method	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}
ADAS1	RS	**1.00**	0.06	0.06	-	-	-	-	-	-	-	-	-
	FF	**1.00**	**0.93**	**0.93**	-	-	-	-	-	-	-	-	-
	MORLOT	0.93	0.60	0.60	-	-	-	-	-	-	-	-	-
	PF	**1.00**	0.90	0.90	-	-	-	-	-	-	-	-	-
ADAS2	RS	**1.00**	0.00	0.00	0.66	**1.00**	**1.00**	-	-	-	-	-	-
	FF	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	-	-	-	-	-	-
	MORLOT	0.96	0.66	0.63	0.90	0.96	**1.00**	-	-	-	-	-	-
	PF	**1.00**	0.90	0.90	**1.00**	**1.00**	**1.00**	-	-	-	-	-	-
RR	RS	0.23	0.16	0.33	0.00	0.00	0.00	-	-	-	-	-	-
	FF	0.93	0.70	0.96	0.16	0.00	0.00	-	-	-	-	-	-
	MORLOT	**1.00**	0.20	**1.00**	0.36	0.00	0.00	-	-	-	-	-	-
	PF	**1.00**	**1.00**	**1.00**	**0.80**	**0.06**	0.00	-	-	-	-	-	-
UAV	RS	0.80	**1.00**	0.00	**1.00**	**1.00**	0.63	**1.00**	0.83	0.93	0.83	0.23	0.06
	FF	0.63	**1.00**	0.00	**1.00**	**1.00**	0.53	**1.00**	0.56	0.80	0.56	0.13	0.00
	MORLOT	0.56	**1.00**	0.00	**1.00**	**1.00**	0.70	**1.00**	0.83	0.90	0.83	0.50	0.06
	PF	**1.00**	**1.00**	0.00	**1.00**	**1.00**	**0.93**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.93**

greater than 0.71 when A is PF, meaning that PF has a large effect size when compared to the others in terms of detected violations. Note that when A is either PF or MORLOT and B is FF, the value $\hat{A}_{AB}$ increases when increasing the number of requirements. This means that the effectiveness of FF decreases when increasing the complexity while keeping the budget constant. In the UAV subject (highest complexity), FF is even worse than RS for all requirements. Both MORLOT and PF do not exhibit this weakness.

To further investigate the detection ability of the approaches we report in Table IV the detection frequency for all requirements and all evaluation subjects. Frequency 1.0 for R_i indicates that the corresponding method detects the violation of R_i in all 30 repeats. For each requirement, we emphasize the highest frequency using boldface.

For higher complexities (RR and UAV subjects), PF yields the highest frequency for all requirements. Particularly significant is R_4 for the subject RR. In this case, no methods but PF can detect violations. Another significant case is R_{11} for the subject UAV. PF yields detection frequency 0.93, while the other methods achieve 0.06 at most.

Figure 5 shows the distribution of failures for each evaluation subject using a non-probabilistic (deterministic) interpretation of the requirements. For example, according to Table I, requirement R_1 for ADAS1 states that "the probability that the vehicle goes out of the lane is less than 0.01." The corresponding deterministic interpretation is "the vehicle shall not go out of the lane," that is, "the probability that the vehicle goes out of the lane is zero."

Under this setting, failures could occur for deterministic requirements, even though their corresponding probabilistic interpretations in Fig. 4 do not yield violations (e.g., R_5 in RR, R_2 in UAV). Notice that this outcome is consistent as the degree of violation in these cases remains within the allowable bounds defined by the probabilistic constraints. It is also worth noting that a failure in a deterministic requirement can trigger cascading failures in other related requirements. In this case, the failure count is aligned across interrelated requirements, resulting in equivalent distributions (e.g., R_6–R_{11} for UAV).

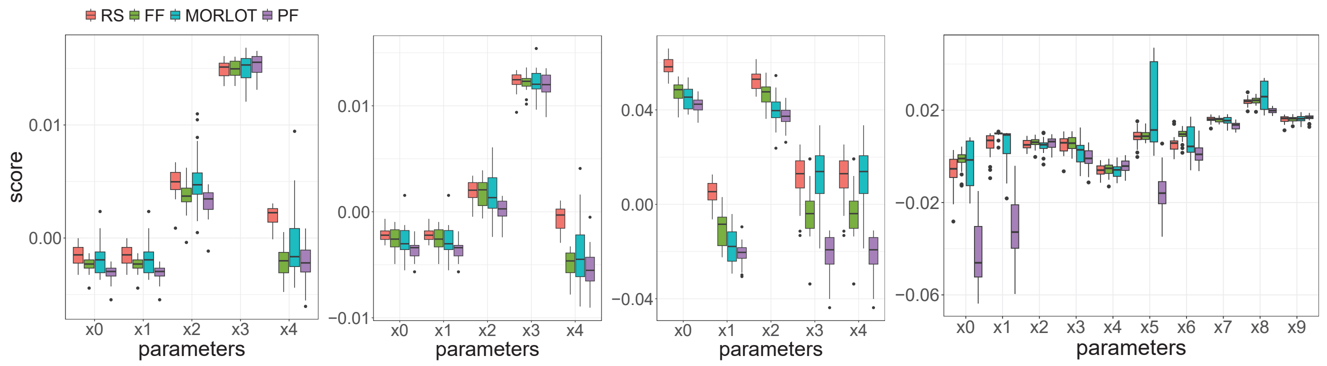
Overall, similar trends can be observed even under such deterministic constraints. As the complexity of the subject increases, PF demonstrates greater effectiveness compared to baseline methods. This is because PF steers the search toward regions with a higher likelihood of failure. Consequently, the total number of failures identified by PF is generally higher. MORLOT generally outperforms FF, except in cases involving low-complexity subjects (e.g., ADAS1), where FF performs similarly to RS.

RQ1 summary. PF generally overcomes the selected baselines especially when increasing the complexity of the specification and the number of requirements. PF is significantly more effective (with a large effect size), in terms of number of violations, than RS, FF, and MORLOT. Similar trends can be observed even when requirements express deterministic constraints.

2) Severity of flaky violations (RQ2): To answer RQ2, we use the same setup as in RQ1 but we measure the degree of violation for all parameters and subjects using the fitness score defined in Sec. IV. Recall that the score depends on the actual parameter values. Such values are probabilities estimated through multiple simulations for each test case. Probabilities are not known beforehand and depend on the simulator and the controllable factors of the tested scenario. Falsification minimizes the scores, potentially identifying negative values that represent flaky violations. In this setting, a lower score indicates greater severity.

Results. Figure 6 shows the lowest scores for all parameters of all evaluation subjects generated by different falsification methods over 30 repeats. Table V shows the results of the statistical tests. We use the same notation as RQ1: statistical significance is highlighted using gray cells, while boldface denotes a large effect size.

Overall, we can see that, compared to other methods, PF yields in general higher severity (lower scores) in all subjects, especially for relevant parameters (i.e., parameters whose actual value can exceed the corresponding boundaries). According to Table V, PF clearly outperforms MORLOT. The difference between MORLOT and RS is significant (significance level $\alpha = 0.05$) with a large effect size only in 30% of the cases. When A is PF and B is RS, we obtain statistical significance with a large effect size in 80% of the cases. Particularly significant is the comparison between PF and the baseline FF when the complexity of the subject increases. Considering the two subjects having lower complexity (ADAS1, ADAS2), the difference is significant in 60% of the cases, while PF yields a large effect size in 50% of the cases. Considering the two subjects having higher complexity (RR, UAV), the difference is significant in 90% of the cases, while PF yields a large effect size in 80% of the cases. This means that, compared to other methods, the effectiveness of PF, in terms of severity of violations, increases when the complexity of the subject increases.



(a) ADAS1 subject (b) ADAS2 subject (c) RR subject (d) UAV subject

Fig. 6: Effectiveness in terms of severity of flaky violations (the lower, the better).

TABLE V: Statistical comparison results for RQ2.

subject	comparison		x_0		x_1		x_2		x_3		x_4	
	A	B	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$
ADAS1	FF	RS	4.98e-03	**0.74**	4.98e-03	**0.74**	9.63e-04	**0.77**	9.02e-01	0.51	1.09e-08	**0.98**
	MORLOT	RS	1.44e-01	0.63	1.44e-01	0.63	7.52e-01	0.53	4.66e-01	0.44	1.84e-05	**0.85**
	PF	RS	7.37e-06	**0.88**	7.37e-06	**0.88**	5.61e-06	**0.86**	1.27e-01	0.37	1.39e-08	**0.98**
	FF	MORLOT	4.36e-01	0.55	4.36e-01	0.55	3.90e-03	**0.71**	4.16e-01	0.56	1.02e-01	0.62
	PF	FF	5.94e-05	**0.80**	5.94e-05	**0.80**	8.77e-02	0.63	5.04-02	0.35	9.35e-01	0.50
	PF	MORLOT	7.70e-04	**0.75**	7.70e-04	**0.75**	3.50e-05	**0.80**	4.07e-01	0.44	9.92e-02	0.62
ADAS2	FF	RS	1.11e-01	0.63	1.11e-01	0.63	8.8e-01	0.51	2.84e-01	0.59	2.95e-09	**1.00**
	MORLOT	RS	5.41e-02	0.66	5.41e-02	0.66	3.36e-01	0.58	5.13e-01	0.55	1.72e-05	**0.86**
	PF	RS	4.69e-07	**0.92**	4.69e-07	**0.92**	5.71e-07	**0.92**	2.71e-01	0.59	8.27e-12	**0.98**
	FF	MORLOT	6.03e-01	0.46	6.03e-01	0.46	6.41e-01	0.46	7.11e-01	0.47	3.04e-01	0.57
	PF	FF	2.64e-04	**0.77**	2.64e-04	**0.77**	1.46e-05	**0.82**	4.11e-01	0.56	1.90e-01	0.59
	PF	MORLOT	3.88e-03	**0.71**	3.88e-03	**0.71**	1.51e-03	**0.73**	6.52e-01	0.53	8.23e-02	0.63
RR	FF	RS	6.85e-09	**0.99**	1.16e-07	**0.94**	9.72e-04	**0.77**	2.19e-04	**0.81**	2.19e-04	**0.81**
	MORLOT	RS	1.69e-13	**0.99**	1.69e-13	**0.99**	1.20e-09	**0.99**	1.00	0.50	1.00	0.50
	PF	RS	3.00e-09	**1.00**	3.00e-09	**1.00**	4.24e-14	**1.00**	7.33e-09	**0.98**	7.33e-09	**0.98**
	MORLOT	FF	4.75e-02	0.65	2.24e-04	**0.78**	5.94e-05	**0.80**	1.24e-05	0.17	1.24e-05	0.17
	PF	FF	9.47e-06	**0.83**	1.27e-06	**0.86**	1.09e-08	**0.93**	3.32e-08	**0.91**	3.32e-08	**0.92**
	PF	MORLOT	9.88e-03	0.70	9.77e-02	0.63	2.60e-02	0.67	2.60e-10	**0.97**	2.60e-10	**0.97**
UAV	FF	RS	3.53e-03	0.28	5.75e-08	**0.91**	1.22e-01	0.38	4.96e-01	0.44	7.08e-01	0.47
	MORLOT	RS	3.35e-01	0.42	3.73e-01	0.43	7.39e-01	0.52	3.26e-02	0.66	6.75e-01	0.53
	PF	RS	2.35e-15	**0.98**	2.02e-16	**0.99**	3.87e-01	0.43	3.32e-06	**0.85**	2.02e-02	0.32
	MORLOT	FF	8.65e-01	0.48	3.94e-05	**0.80**	3.99e-02	0.65	0.33e-03	**0.72**	3.81e-01	0.56
	PF	FF	1.69e-17	**1.00**	3.01e-11	**1.00**	8.42e-01	0.51	1.15e-07	**0.89**	4.62e-02	0.35
	PF	MORLOT	7.61e-16	**0.99**	1.64e-15	**0.99**	1.87e-01	0.40	4.45e-02	0.65	8.00e-03	0.30

subject	comparison		x_5		x_6		x_7		x_8		x_9	
	A	B	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$	p-val	$\hat{A}_{AB}$
UAV	FF	RS	7.33e-01	0.47	1.08e-06	0.15	4.87e-01	0.55	4.20e-01	0.43	6.15e-01	0.53
	MORLOT	RS	1.17e-01	0.38	9.24e-01	0.49	1.92e-01	0.59	4.77e-01	0.44	6.89e-01	0.46
	PF	RS	3.21e-16	**0.99**	2.06e-05	**0.81**	3.64e-08	**0.91**	3.15e-10	**0.97**	1.07e-01	0.38
	MORLOT	FF	2.86e-01	0.41	1.97e-01	0.59	6.46e-01	0.53	5.51e-01	0.45	4.50e-01	0.44
	PF	FF	1.69e-17	**1.00**	1.98e-10	**0.92**	3.56e-11	**0.94**	3.52e-14	**0.97**	2.65e-02	0.33
	PF	MORLOT	5.07e-16	**0.99**	2.46e-03	**0.72**	1.58e-05	**0.82**	8.00e-06	**0.82**	1.37e-01	0.38

RQ2 summary. PF generally yields higher severity of the violations compared to RS, FF, and MORLOT. As the complexity of the specification increases, PF proves to be significantly more effective than FF, with a large effect size, in terms of the severity of violations. This effectiveness is observed in many more cases compared to MORLOT.

3) Cost overhead (RQ3): To answer RQ3, we executed the offline analysis step for all evaluation subjects to measure the cost (in execution time) of computing the boundaries for the specification parameters. The offline analysis is an extra step required by PF, which is generally not necessary in other falsification methods. Thus, our baseline is zero extra time. Concerning the online step of PF, the cost is comparable to that of the selected baseline methods as, in this case, the most expensive part lies in repeated simulations. As we use the same testing budget (i.e., the same number of simulations) across all methods, execution times are similar and within the same order of magnitude.

To run the offline analysis, we need to select a parametric model checker. In our experiments, we adopt PRISM [16], a popular probabilistic model checker that supports parametric MDP specifications and requirements expressed using PCTL. We use PRISM to verify PCTL requirements defined manually based on the original studies introducing our evaluation subjects. The MDP specification of each subject is mechanically derived from a textual description of the target scenario using a simple domain-specific language. Further details about the specification and requirements for all subjects can be found in the supporting material (See Sec. VIII).

Results. Figure 7 shows the distribution of the execution (wall-clock) time of the offline analysis step repeated 30 times, for all requirements and all evaluation subjects. The black triangle in the center of each box represents the average value.

We can observe that the median is close to zero for all subjects, while the average increases up to ~ 1 second in RR and ~ 2 seconds in UAV. It is apparent that the cost in these instances is minimal when compared to the optimization step. According to our experience, the time required by the optimization step is orders of magnitude higher for all subjects since it requires many simulations (~ 2.5 hours). Thus, the extra cost introduced by the offline analysis step is negligible. Furthermore, the outcome of the analysis does not change as long as the specification and the requirements are steady. In this case, boundaries can be reused across multiple runs.

RQ3 summary. The offline analysis takes a few seconds on average for each requirement. The extra cost introduced by the offline analysis step is negligible when compared to the optimization step. This means that offline analysis does not represent a bottleneck.

C. Threats to Validity

Using one simulation platform is a potential threat to external validity. To mitigate this issue, we considered more than one evaluation subject having increasing complexity in terms of specification, parameters, and number of requirements.

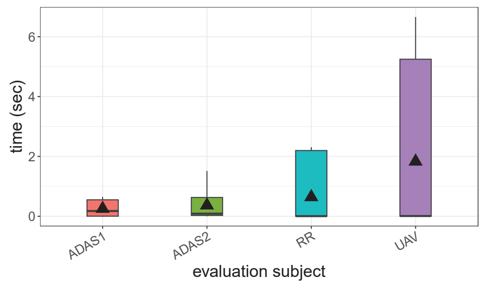


Fig. 7: Execution time of the offline analysis.

All the subjects are derived from the existing literature and they are open source, thus, they can be reused by other researchers for comparison. For each evaluation subject, we control the effect of changing factors (using non-linear, non-convex functions) and the intervals constraining the parameters of the specification to avoid trivial violations. Note that such a fine-grained manipulation increases the internal validity of our results compared to observations without manipulation.

Further studies with study subjects with more than 12 requirements would increase the generalizability of our results. However, this number aligns with existing falsification methods that handle multiple requirements simultaneously, such as MORLOT [6].

As anticipated above, the discretization (states and actions) used in MORLOT could potentially affect our results. Therefore, we do not exclude the possibility that MORLOT could overcome PF in case of optimal discretization. However, given the equal budget for all methods under comparison in our experimental setting, finding an optimal discretization within a few “trial and error” simulations is unlikely due to a huge space of possible discretization steps. Understanding the cost-effectiveness by varying the degree of discretization requires further investigation.

VI. Related Work

Many existing falsification approaches do not consider independent requirements altogether. Indeed, a common approach is to consider them one by one making the problem intractable when there are many requirements [6]. Several falsification techniques consider requirements expressed using a formal notation, typically Metric Temporal Logic [38] (MTL) or Signal Temporal Logic [39] (STL). In these approaches, a robustness metric quantifies how much an output signal satisfies a requirement. The problem is then formulated as robustness minimization [40]–[44]. Two well-known tools for falsification of individual MTL requirements are S-TaLiRo [41] and Breach [42] which rely on global optimization algorithms, such as simulated annealing, genetic algorithms, and stochastic local search [40]. Other approaches adopting multi-objective optimization [45], [46], Deep RL [47], have been proposed.

When considering the falsification of conjunctive requirements, different scales of signals can mask each other’s contribution to the robustness metric. This is known as the scaling problem [43]. Multi-requirement falsification techniques dealing with multiple conjunctive requirements have been recently introduced. Mathesen et al. [48] propose to use Bayesian stochastic optimization to model separately the contributions of each component of the conjunctive requirement. This alleviates the aforementioned masking problem. Eddeland et al. [13] proposes a focused multi-requirement falsification approach (FF), where the original multi-requirement problem is decomposed into multiple smaller falsification sub-problems. The selection of the requirement to focus on is done using a sensitivity analysis that looks for the inputs that are unlikely to affect the robustness value. This information is used to reduce the search space, thus, the dimension of the sub-problems.

MORLOT [6] deals with some of the research challenges we introduced in Sec. I. The authors present an online testing approach tailored to CPS that aims at falsifying multiple independent (informal) requirements. MORLOT incrementally generates sequences of environmental changes combining tabular-based RL and many-objective search.

Surrogate-assisted optimization has been explored in online testing of complex CPS to reduce costs [5], [14]. In this context, models mimic the simulator but are much less expensive to run. None of these studies have applied these techniques in the problem domain of falsification under flaky simulations.

Compared to existing methods, our unique contribution is a falsification approach that simultaneously addresses multiple (independent) probabilistic requirements while accounting for the potential flakiness of test results.

VII. Conclusion

In this paper, we present PF, a novel approach for CPS falsification that combines parametric model checking and many-objective optimization to falsify multiple (independent) probabilistic requirements under flaky simulations. Our empirical evaluation using four evaluation subjects shows that PF is often significantly more effective in terms of number and severity of violations than random search, FF, and MORLOT (with large effect size). This occurs especially when the complexity of the testing scenario grows. We also show that the extra cost introduced by the offline analysis step is negligible.

We plan to extend PF by leveraging surrogate models to reduce the cost of online optimization while preserving its effectiveness. We also plan to extend our study in order to compare the cost-effectiveness trade-off of RL approaches under optimal discretization of the state-action space.

VIII. Data Availability

The replication package of our experiments, including the implementation of PF, alternative approaches, and the instructions to set up and run the experiments including the simulation of the selected CPS benchmarks, are publicly available on Zenodo [49].

Acknowledgments

The work has been partially founded by the topic Engineering Secure Systems of the Helmholtz Association (HGF), by the pilot program Core Informatics at KIT (KiKIT) of HGF.

References

[1] C. Menghi, S. Nejati, L. Briand, and Y. I. Parache, "Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 372–384. [Online]. Available: https://doi.org/10.1145/3377811.3380370

[2] Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, "Falsifai: Falsification of ai-enabled hybrid control systems guided by time-aware coverage criteria," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1842–1859, 2023.

[3] C. Daws, "Symbolic and parametric model checking of discrete-time markov chains," in *Proceedings of the First International Conference on Theoretical Aspects of Computing*, ser. ICTAC'04. Berlin, Heidelberg: Springer-Verlag, 2004, p. 280–294. [Online]. Available: https://doi.org/10.1007/978-3-540-31862-0_21

[4] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 1016–1026.

[5] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 811–822. [Online]. Available: https://doi.org/10.1145/3510003.3510188

[6] F. Ul Haq, D. Shin, and L. C. Briand, "Many-objective reinforcement learning for online testing of dnn-enabled systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 1814–1826.

[7] Y. Sun, C. M. Poskitt, X. Zhang, and J. Sun, "Redriver: Runtime enforcement for autonomous vehicles," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3639151

[8] M. Hejase, A. Katis, and A. Mavridou, *Design, formalization, and verification of decision making for intelligent systems*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2024-2409

[9] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 432–442.

[10] A. Afzal, D. S. Katz, C. Le Goues, and C. S. Timperley, "Simulation for robotics test automation: Developer perspectives," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2021, pp. 263–274.

[11] M. H. Amini, S. Naseri, and S. Nejati, "Evaluating the impact of flaky simulators on testing autonomous driving systems," *Empirical Softw. Engg.*, vol. 29, no. 2, feb 2024. [Online]. Available: https://doi.org/10.1007/s10664-023-10433-5

[12] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2419–2426.

[13] J. Lidén Eddeland, A. Donzé, and K. Åkesson, "Multi-requirement testing using focused falsification," in *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3501710.3519521

[14] S. Nejati, L. Sorokin, D. Safin, F. Formica, M. M. Mahboob, and C. Menghi, "Reflections on surrogate-assisted search-based testing: A taxonomy and two replication studies based on industrial adas and simulink models," *Information and Software Technology*, vol. 163, p. 107286, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584923001404

[15] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[16] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*, G. Gopalakrishnan and S. Qadeer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 585–591.

[17] A. Filieri, G. Tamburrelli, and C. Ghezzi, "Supporting self-adaptation via quantitative verification and sensitivity analysis at run time," *IEEE Transactions on Software Engineering*, vol. 42, no. 1, pp. 75–99, 2016.

[18] E. M. Hahn, T. Han, and L. Zhang, "Synthesis for PCTL in parametric markov decision processes," in *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, ser. Lecture Notes in Computer Science, M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, Eds., vol. 6617. Springer, 2011, pp. 146–161. [Online]. Available: https://doi.org/10.1007/978-3-642-20398-5_12

[19] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[20] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[21] A. Panichella, F. M. Kifetew, and P. Tonella, "Reformulating branch coverage as a many-objective optimization problem," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1–10.

[22] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 143–154.

[23] R. Yang, X. Sun, and K. Narasimhan, *A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[25] M. Autili, L. Grunske, M. Lumpe, P. Pelliccione, and A. Tang, "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar," *IEEE Transactions on Software Engineering*, vol. 41, no. 7, pp. 620–638, 2015.

[26] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2208–2224, oct 2021.

[27] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 143–154.

[28] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 811–822.

[29] P. Congdon, *Bayesian models for categorical data*. John Wiley & Sons, 2005.

[30] A. Filieri, L. Grunske, and A. Leva, "Lightweight adaptive filtering for efficient learning and updating of probabilistic models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 200–211.

[31] C. P. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, 2nd ed. Springer, May 2007.

[32] M. Camilli, R. Mirandola, and P. Scandurra, "Enforcing resilience in cyber-physical systems via equilibrium verification at runtime," *ACM Trans. Auton. Adapt. Syst.*, vol. 18, no. 3, sep 2023. [Online]. Available: https://doi.org/10.1145/3584364

[33] G. Moreno, C. Kinneer, A. Pandey, and D. Garlan, "Dartsim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2019, pp. 181–187.

[34] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1–10. [Online]. Available: https://doi.org/10.1145/1985793.1985795

[35] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[36] A. Vargha and H. D. Delaney, “A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong,” *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000. [Online]. Available: http://www.jstor.org/stable/1165329

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[38] R. Koymans, “Specifying real-time properties with metric temporal logic,” *Real-Time Syst.*, vol. 2, no. 4, p. 255–299, oct 1990. [Online]. Available: https://doi.org/10.1007/BF01995674

[39] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[40] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, “Stochastic local search for falsification of hybrid systems,” in *Automated Technology for Verification and Analysis: 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings 13*. Springer, 2015, pp. 500–517.

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

[42] A. Donzé, “Breach, a toolbox for verification and parameter synthesis of hybrid systems,” in *Computer Aided Verification: 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*. Springer, 2010, pp. 167–170.

[43] Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, “On the effectiveness of signal rescaling in hybrid system falsification,” in *NASA Formal Methods Symposium*. Springer, 2021, pp. 392–399.

[44] w. s. ARCH, “Applied verification for continuous and hybrid systems,” 2014-2022. [Online]. Available: https://cps-vo.org/group/ARCH/archive

[45] Z. Ramezani, J. L. Eddeland, K. Claessen, M. Fabian, and K. Åkesson, “Multiple objective functions for falsification of cyber-physical systems,” *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 417–422, 2020.

[46] B. Barbot, N. Basset, T. Dang, A. Donzé, J. Kapinski, and T. Yamaguchi, “Falsification of cyber-physical systems with constrained signal spaces,” in *NASA Formal Methods: 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11–15, 2020, Proceedings 12*. Springer, 2020, pp. 420–439.

[47] Y. Yamagata, S. Liu, T. Akazaki, Y. Duan, and J. Hao, “Falsification of cyber-physical systems using deep reinforcement learning,” *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2823–2840, 2020.

[48] L. Mathesen, G. Pedrielli, and G. Fainekos, “Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 732–737.

[49] A. Author(s), “Parametric Falsification of Many Probabilistic Requirements under Flakiness – Replication package,” https://doi.org/10.5281/zenodo.13172358, 2024, [Online; accessed August-2024].