Fast Koopman Surrogate Falsification using Linear Relaxations and Weights

Stanley Bak[1][0000-0003-4947-9553], Abdelrahman Hekal[2][0009-0008-9685-0558], Niklas Kochdumper[3][0000-0001-6017-7623], Ethan Lew[4][0000-0002-6509-6846], Andrew Mata[1][0009-0002-7435-332X], and Amir Rahmati[1][0000-0001-7361-1898]

[1] Department of Computer Science, Stony Brook University, NY, USA
[2] School of Computing, Newcastle University, Newcastle Upon Tyne, UK
[3] Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
[4] Galois Inc., OR, USA

Abstract. Recent work demonstrated that using Koopman surrogate models to falsify black-box models against signal temporal logic specifications is highly effective. However, the bottleneck of this approach arises from the mixed-integer linear program optimization used to synthesize the falsifying trajectory. The complexity of mixed-integer linear programming can be prohibitive, increasing exponentially with the number of binary variables. In this work, we introduce a new weighted robustness encoding that eliminates the need for binary variables. We also propose a new weighting scheme for Koopman operator linearization that aims to compensate for inaccuracies in the learned model. We evaluate our approach using a set of benchmarks from the ARCH falsification competition. Our weighting methods significantly improve computational efficiency and reduce the number of simulations needed to find falsifying traces.

Keywords: Cyber-physical systems, signal temporal logic, falsification, Koopman operator linearization, linear programming relaxation

1 Introduction

Trusted autonomy of cyber-physical systems in safety-critical environments has become a rapidly growing area of research and development. Safety requirements for such systems are often specified using signal temporal logic (STL) formulae [26]. STL enables the rigorous expression of complex safety specification by combining temporal operators (e.g., *globally*, *eventually*, *until*), logical operators (e.g., *and*, *or*, *not*), and continuous-time predicates over system variables.

However, the verification of cyber-physical systems remains challenging due to the inherent complexity of system dynamics and the temporal nature of the specifications. Traditional techniques often involve exhaustive exploration of the system's state space or discrete event sequences, which can be computationally prohibitive for large-scale, continuous systems with black-box components.

Orthogonal to formal verification, which aims to prove a system's correctness, falsification offers a promising approach for addressing the challenges that arise when reasoning about the safety of such systems. Falsification aims to find a trajectory of a system that violates a safety property or leads to a bad state. Due to the complexity and often black-box nature of many real-world systems, the majority of works focus on numerical methods for falsification with STL safety requirements [1,2,30,37]. Recent work [6] has shown great promise in falsifying such systems by constructing a surrogate model using Koopman linearization and deploying it in a robustness-guided optimization. In particular, inspired by previous work on controller synthesis for STL requirements [34,35], the authors formulate the optimization problem using binary variables to encode the robustness of STL formulae and solve it using mixed-integer linear programming (MILP).

It is well known that MILP, in the general case, is NP-hard [15], which hinders its applicability, particularly as the problem size increases. To address this issue, we propose a new heuristic encoding termed *weighted robustness encoding*, which eliminates the need for binary variables, leading to a linear program. The proposed approach is iterative: we refine the optimization problem based on critical predicates and the corresponding time points from previous iterations. By assigning weights to predicates and time points, we prioritize satisfaction or violation of specific predicates at particular times, effectively guiding the optimization process. We show that our linear relaxation of the MILP formulation significantly reduces computational efforts, particularly for complex requirements. For instance, in the case of the $CC5$ benchmark from the ARCH falsification competition [11], and with a discretization time step of 2 seconds, our weighted encoding requires only 2.32 seconds, compared to 1705 seconds for the MILP encoding.

To further improve the falsification process, we introduce a novel weighting scheme for learning the Koopman linearized system. Our key observation is that we can achieve more effective modeling of system dynamics by increasing the weighting of critical trajectories, i.e., trajectories that are closer to falsifying the system. Furthermore, we extend our method to the weighting of states, where higher accuracy is desired in states more prominent in the STL requirement. The weighting scheme is highly effective, significantly reducing the number of simulations required to find a falsifying trace for all challenging ARCH benchmarks, where 10 or more simulations are required.

2 Related Work

Falsification techniques have recently seen great advancements, with several approaches developed to improve performance and scalability. Pioneer tools such as S-TaLiRo [3] and Breach [10] rely on a mixture of randomized simulations and optimization approaches to guide the search towards a falsifying trace. Subsequent tools with different strategies have since been developed. The tool falsify [40] adopts a grey-box method to learn the system's behavior and is implemented with a deep reinforcement learning algorithm. FORESEE [42] uses a Monte-Carlo tree search to specifically address the *scale problem* [14] within the domain of falsification. Falstar [12] gradually scales inputs based on a guided search that adapts to local complexity.

Recent approaches utilize model learning, such as surrogate probabilistic models with Bayesian optimization [9,27], mealy machines [39], and system identification techniques [28]. A promising recent work by Bak et al. [6] leverages Koopman linearization techniques to learn a surrogate model and synthesizes a falsifying trace using MILP optimization.

However, MILP complexity poses significant challenges for scalability. Researchers have since focused on mitigating this issue. Kurtz et al. [20] propose a new encoding that unintuitively increases the number of binary variables but results in a tighter convex relaxation. Conversely, they later suggest yet another encoding that reduces the number of binary variables [21], offering its own benefits. Several works aim to eliminate the need for binary variables altogether. Notable works include heuristic methods such as smooth approximations of robustness with gradient-based methods [16,23,32], convex-concave programming [38], differential dynamic programming [19], and control barrier functions [25]. Yet, these heuristic approaches overlook system-specific information. In contrast, Saha and Julius [36] propose an iterative approach that leverages such information to refine adaptive optimization. However, the approach has limitations, often incurring a large number of iterations or failing to converge to a desired solution. In this work, we address these limitations with our new iterative strategy.

3 Problem Formulation

In this work, we consider general black-box cyber-physical systems. We view a system $\mathcal{M}$ as a mapping from input signals $u(t) \in \mathcal{U}$ to output signals $y(t) \in \mathbb{R}^l$, where $\mathcal{U}$ is a compact set of input values (input space) at each point in time. A bounded-time simulation of a system for time horizon T is then denoted by:

$$y(t) = \mathcal{M}\big(u(t)\big). \tag{1}$$

Note that the set of input signals can contain time-varying control inputs, external inputs, and/or initial conditions. We represent a trajectory of the system by a tuple $\tau = (u(t), y(t))$.

Specifications are used to describe the desired system behavior and are represented as signal temporal logic formulae [26]:

Definition 1 (Signal Temporal Logic). *Given a set of atomic predicates $p \in \mathcal{A}$ which are defined as $p := f(y) > 0$, where $f: \mathbb{R}^l \to \mathbb{R}$ is a nonlinear function, the syntax for a signal temporal logic formula is*

$$\varphi := \text{True} \;\big|\; p \;\big|\; \neg\varphi \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \Box_{[a,b]}\varphi \;\big|\; \Diamond_{[a,b]}\varphi \;\big|\; \varphi_1 U_{[a,b]} \varphi_2,$$

where a and b with $b \geq a$ are non-negative scalars denoting time bounds. For a signal $y(t): \mathbb{R}_{\geq 0} \to \mathbb{R}^l$, the semantics of STL is defined as follows:

$$\begin{aligned}
y(t) \models p \quad &\Leftrightarrow \quad f(y(0)) > 0 \\
y(t) \models \neg\varphi \quad &\Leftrightarrow \quad \neg(y(t) \models \varphi) \\
y(t) \models \varphi_1 \wedge \varphi_2 \quad &\Leftrightarrow \quad (y(t) \models \varphi_1) \wedge (y(t) \models \varphi_2) \\
y(t) \models \varphi_1 U_{[a,b]} \varphi_2 \quad &\Leftrightarrow \quad \exists c \in [a,b] : y(t+c) \models \varphi_2 \\
&\qquad\quad \wedge\ \forall d \in [0,c) : y(t+d) \models \varphi_1.
\end{aligned}$$

Moreover, the semantics for the finally operator $\Diamond_{[a,b]}\varphi := \text{True}\; U_{[a,b]}\varphi$ *and the globally operator* $\Box_{[a,b]}\varphi := \neg\Diamond_{[a,b]}\neg\varphi$ *directly follows from the semantics of the until operator* $\varphi_1 U_{[a,b]}\varphi_2$.

To guide the search toward signals that violate the specification, we use the quantitative robustness for temporal logic [13], which specifies how robustly a signal satisfies an STL formula.

Definition 2 (Robustness). *The quantitative robustness semantics of STL is represented by a function* $\mathcal{Q}(\varphi, y(t))$ *that maps an STL formula* φ *and a signal* $y(t)$ *to a scalar value. This function is recursively defined as follows:*

$$\begin{aligned}
\mathcal{Q}(\text{True}, y(t)) &:= \infty \\
\mathcal{Q}(p, y(t)) &:= f(y(t)) \\
\mathcal{Q}(\neg\varphi, y(t)) &:= -\mathcal{Q}(\varphi, y(t)) \\
\mathcal{Q}(\varphi_1 \wedge \varphi_2, y(t)) &:= \min\big(\mathcal{Q}(\varphi_1, y(t)), \mathcal{Q}(\varphi_2, y(t))\big) \\
\mathcal{Q}(\varphi_1 \vee \varphi_2, y(t)) &:= \max\big(\mathcal{Q}(\varphi_1, y(t)), \mathcal{Q}(\varphi_2, y(t))\big) \\
\mathcal{Q}(\Box_{[a,b]}\varphi, y(t)) &:= \min_{c\in[t+a,t+b]} \mathcal{Q}(\varphi, y(c)) \\
\mathcal{Q}(\Diamond_{[a,b]}\varphi, y(t) &:= \max_{c\in[t+a,t+b]} \mathcal{Q}(\varphi, y(c)) \\
\mathcal{Q}(\varphi_1 U_{[a,b]}\varphi_2, y(t)) &:= \max_{c\in[t,t+b]} \min\Bigg(\mathcal{Q}(\varphi_2, y(c)), \\
&\qquad\qquad \min_{d\in[c-a,c]} \mathcal{Q}(\varphi_1, y(d))\Bigg).
\end{aligned}$$

The robustness measures the extent to which a signal $y(t)$ *satisfies an STL formula* φ*, where* $\mathcal{Q}(\varphi, y(t)) \geq 0$ *entails satisfaction* $y(t) \models \varphi$*, and larger values for* $\mathcal{Q}(\varphi, y(t))$ *indicate stronger satisfaction.*

Our overarching goal is to solve the falsification task, which is defined as follows:

Problem 1 (Falsification). Given a black-box model $\mathcal{M}$ as in (1), a system specification in the form of a signal temporal logic formula φ as in Def. 1, and a set of uncertain inputs $\mathcal{U}$, falsification attempts to find an input signal $u(t) \in \mathcal{U}$ such that the corresponding output $y(t) = \mathcal{M}(u(t))$ violates the specification, $y(t) \not\models \varphi$.

Recent work by Bak et al. [6] proposed a new approach for the falsification of black-box cyber-physical systems using surrogate Koopman models. The approach iteratively constructs a Koopman model $\mathcal{M}_K$ from system trajectories and uses this model to determine critical inputs that maximally violate the system specification by minimizing the robustness in Def. 2.

Adopting an abstracted view, the dynamics of the surrogate Koopman model can be represented by a linear discrete-time system

$$y_{k+1} = \mathcal{M}_K(y_k, u_k), \tag{2}$$

where $k \in \{1,...,n\}$ is the time index for a discretization with time step size Δt. We use n to denote the number of time indices of the system for a time horizon T.

The surrogate approach in [6] encodes the robustness of STL formulae over the Koopman surrogate models using a MILP formulation. This is made possible because the model dynamics are linear, and the `min()` and `max()` operators that define the robustness in Def. 2 can be encoded using binary variables and the Big-M notation. Consider predicates of linear nature, such that $p := f(y) \geq 0$ [5]. The encoding for predicates, and the `min()` and `max()` operators is defined as:

Predicate Encoding. For each atomic predicate p in the STL formula φ, and for each time index k, a continuous decision variable r_k^p represents the robustness of the predicate at time index k, such that:

$$r_k^p = f(y_k). \tag{3}$$

Minimum Operator Encoding. Consider a minimum operator $r^\varphi = \min(r^{\varphi_1}, r^{\varphi_2}, ..., r^{\varphi_m})$ where r^{φ_i} represents the robustness of subformulas. The following constraints can be used to encode the minimum operator, where $z_i \in \{0,1\}$ are auxiliary binary variables, and M is a large positive constant:

$$r^\varphi \leq r^{\varphi_i} \quad \forall i \in \{1,...,m\}, \tag{4a}$$
$$r^{\varphi_i} - M(1-z_i) \leq r^\varphi \leq r^{\varphi_i} + M(1-z_i) \quad \forall i \in \{1,...,m\}, \tag{4b}$$
$$\sum_{i=1}^{m} z_i = 1. \tag{4c}$$

Maximum Operator Encoding. Similarly, for $r^\varphi = \max(r^{\varphi_1}, r^{\varphi_2}, ..., r^{\varphi_m})$ where r^{φ_i} represents the robustness of subformulas, the constraints are as follows:

$$r^\varphi \geq r^{\varphi_i} \quad \forall i \in \{1,...,m\}, \tag{5a}$$
$$r^{\varphi_i} - M(1-z_i) \leq r^\varphi \leq r^{\varphi_i} + M(1-z_i) \quad \forall i \in \{1,...,m\}, \tag{5b}$$
$$\sum_{i=1}^{m} z_i = 1. \tag{5c}$$

We remark that in literature, MILP robustness encoding is most often used for controller synthesis, where the goal is to maximize the satisfaction of the STL formula. Falsification and controller synthesis are intrinsically connected, as falsifying an STL formula is equivalent to satisfying its negation. Additionally, reasoning over satisfaction is intuitively more straightforward to understand, as it directly aligns with the goal of meeting specified requirements. For this reason, we formulate the falsification problem in terms of maximizing the satisfaction of the negated specification:

[5] Alternatively, predicates can be any convex or even nonlinear function for which corresponding solvers can be leveraged.

Problem 2 (Robustness Optimization). Given a surrogate Koopman model $\mathcal{M}_K$, a (negated) system specification in the form of a signal temporal logic formula φ, and a set of uncertain inputs, maximize the satisfaction (i.e., robustness) of the system specification subject to system constraints, i.e.,

$$\begin{aligned} \max_{u(t)\in\mathcal{U}} \quad & \mathcal{Q}(\varphi, y(t)) \\ \text{s.t.} \quad & y_{k+1} = \mathcal{M}_K(y_k, u_k) \quad \forall k \in \{1,\dots,n\}. \end{aligned} \tag{6}$$

For the remainder of this paper, we therefore focus on the satisfaction of STL formulas rather than violation.

4 Weighted Robustness Encoding

In this section, we propose a new heuristic approach for solving Prob. 2, using weighted robustness encoding. The encoding relies on an iterative approach, which is incorporated within the falsification loop of the work in [6], yielding the overall framework summarized in Alg. 1. First, we generate a random trajectory (Line 1) that is used to initialize the training set (Line 2). Next, we learn a Koopman model (Line 5) and then use our new weighted robustness encoding to optimize for a critical trajectory that we suspect to satisfy the (negated) system specification (Line 6). Note that we use $\mathcal{Q}_w$ to denote weighted robustness. If the trajectory fails to satisfy (falsify) the system, we look to identify the critical predicates and corresponding critical time indices (Line 12). We define the critical predicates $p \in \mathcal{A}_{crit}$ as all predicates that incur negative robustness on the system and prevent its satisfaction (falsification). The corresponding critical time index k^p represents the time at which the system violates the critical predicate the most. We further identify a critical value v^p for each critical predicate as the minimal robustness value for the predicate or the robustness value at the critical time index. The critical value is then used to update the offset and refine our heuristic approach at the corresponding critical time index (Line 13). We iteratively modify each identified critical predicate in the STL formula in search of all critical predicates (Line 14–15). At the same time, the previously extracted critical trajectory is used to refine the learned Koopman model (Line 18), and the process is repeated.

In this work, we look to eliminate the need for binary variables in traditional MILP encoding. To this end, we introduce a weighted robustness encoding that replaces equations (3), (4) and (5). Our approach is inspired by the previous work of Saha and Julius [36], where constraints are iteratively added for the most critical predicate p to enforce satisfaction (violation) at the corresponding critical time index k^p:

$$f(y_{k^p}) \geq 0. \tag{7}$$

The approach can be easily modified to maximize satisfaction (violation) of each constraint by adding a decision variable $r^p_{k^p}$ to each constraint which is to be maximized, such that:

$$f(y_{k^p}) \geq r^p_{k^p}. \tag{8}$$

In the remainder of this section, we detail our approach and highlight how we overcome the limitations of Saha's approach [36].

Algorithm 1: Koopman Falsification with Weighted Robustness

```
Data: Black-box CPS model $\mathcal{M}$, (negated) STL formula
      $\varphi$, input space $\mathcal{U}$, time horizon $T$, maximum number of simulations $N_{\max}$
Result: Counterexample trajectory $\tau$
1  $\tau \leftarrow$ RunSimulation($\mathcal{M}$,RandomInput($\mathcal{U}$),$T$)
2  $\mathcal{T} \leftarrow \{\tau\}$                      // Initialize set of training data
3  $\mathcal{B} \leftarrow \{0\}^m, \quad \mathcal{W} \leftarrow \{1\}^{m\times n}$
4  for $i=1$ to $N_{max}$ do
5      $\mathcal{M}_K \leftarrow$ LearnKoopmanModel($\mathcal{T}$)
6      $u^* \leftarrow \underset{u(t)\in\mathcal{U}}{\text{argmax}}\ \mathcal{Q}_w(\varphi,y(t),\mathcal{B},\mathcal{W})$ s.t. $y_{k+1}=\mathcal{M}_K(y_k,u_k)$
7      $\tau \leftarrow$ RunSimulation($\mathcal{M}$,$u^*$,$T$)
8      if $\mathcal{Q}(\varphi,\tau)>0$ then
9          return $\tau$                    // Counterexample found
10     end
11     do
12         $p$, $k^p$,$v^p \leftarrow$ getCriticalPredTimes($\varphi$,$\tau$)
13         $b^p \leftarrow b^p+v^p, \quad w_k^p \leftarrow w_k^p+v^p$
14         $p^* \leftarrow p-v^p$
15         $\varphi^* \leftarrow \varphi[p^* \rightarrow p]$              // Update STL with modified predicate
16     while $\mathcal{Q}(\varphi^*,\tau)<0$
17     $\mathcal{B} \leftarrow \{b^1,b^2,...,b^m\}, \quad \mathcal{W} \leftarrow \{w_1^1,w_1^2,...,w_1^m,w_2^m,...,w_n^m\}$
18     $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau\}$                 // Update trajectory set
19 end
```

4.1 Weighted Predicate Encoding

We augment Eq. (3) by introducing offsets $\mathcal{B}=\{b^1,b^2,...,b^m\}$ for each predicate p along with weights $\mathcal{W}=\{w_1^1,w_1^2,...,w_1^m,w_2^m,...,w_n^m\}$ for each predicate and time index k, such that:

$$r_k^p = w_k^p\ (f(y_k^p)+b^p). \tag{9}$$

The offset b^p, first introduced by Bak et al. [6], linearly shifts predicate values to counteract inaccuracies in the learned Koopman model. For our work, it serves a dual purpose, where the offset adjusts for both errors in the Koopman model and the heuristic robustness encoding. The newly introduced weight w_k^p is used to favor the satisfaction of particular predicates at specific time indices. Prior to the first iteration of the main loop in Alg. 1, all offsets and weights are initialized to zero and one, respectively (Line 3). In the following iterations, we augment the offset values and weights accordingly (Line 13) for all critical predicates $p\in\mathcal{A}_{crit}$ and the corresponding robustness values v^p, and time indices k^p

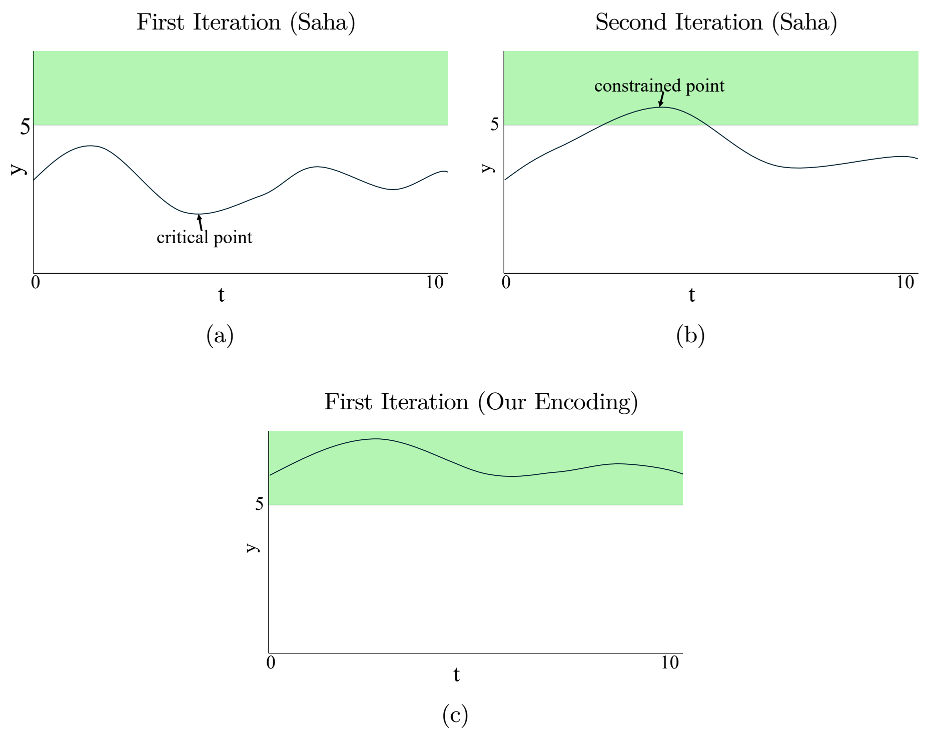

Fig. 1: Resultant trajectories (black line) and goal set (green area) for Example 1. The top left (a) shows a random initial trajectory, which is the resultant trajectory after the first iteration of the iterative MILP approach by Saha and Julius [36]. The marked critical point is the point closest to the target set. Top right (b) shows the trajectory after the second iteration of Saha's approach. Bottom (c) shows the trajectory after one iteration using our encoding, satisfying the STL formula in Example 1.

4.2 Weighted Minimum Operator Encoding

The encoding of the `min()` operator hinges on the insight that in Prob. 2, our objective is to maximize the robustness of the STL formula. For a `min()` operator, $r^{\varphi} = \min(r^{\varphi_1}, r^{\varphi_2}, \ldots, r^{\varphi_m})$, maximizing r^{φ} directly translates to maximization of each r^{φ_i}. Thus, we can remove the binary variables z_i and directly encode r^{φ} as

$$r^{\varphi} \leq r^{\varphi_i} \quad \forall i \in \{1, \ldots, m\}. \tag{10}$$

For a specific subset of STL formulae, this encoding is even exact:

Proposition 1. *If the quantitative robustness of an STL formula according to Def. 2 consists of only predicates and the* `min()` *operator (i.e., no* `max()` *operator), then our weighted robustness encoding is exact in the first iteration.*

Proof. Let $r^{\varphi} = \min(r^{\varphi_1}, r^{\varphi_2}, \ldots, r^{\varphi_m})$. Our objective is to maximize r^{φ}. According to the constraints in Eq. (10), the maximum value of r^{φ} is the minimum value among $r^{\varphi_1}, r^{\varphi_2}, ..., r^{\varphi_m}$. Therefore, if r^{φ} is maximized, it follows that each r^{φ_i} is also maximized and that r^{φ} corresponds to the minimum of these maximum values.

We note that the iterative addition of constraints in [36] will also eventually reach exact optimality. However, it can incur a large number of iterations. To better illustrate the differences, we use the following example:

Example 1. We consider a system with one output signal $y(t)$ and a system specification $\varphi = \square_{[0,10]} y > 5$ that we want to satisfy over a simulation time of $10s$.

Fig. 1 shows the resultant trajectories from our approach and Saha's iterative algorithm [36]. Fig. 1a shows a random initial trajectory. The corresponding critical point is the point that violates the specification $\varphi = \square_{[0,10]} y > 5$ the most, i.e., the minimum point. Saha's approach [36] begins with such a random trajectory and then adds a constraint at the corresponding critical point. The solver then synthesizes the trajectory in Fig. 1b. However, to satisfy the STL requirement in Example 1, the entire trajectory must be contained within the target region $y > 5$. It is thus evident that in the worst case, Saha's approach requires n iterations to find a satisfying trajectory, where n is the number of time indices. For a time step of $\Delta t = 0.1$, it would take 100 iterations to fully encode the robustness of the formula in Example 4. On the other hand, our encoding ensures that the robustness of the predicate $y > 5$ is maximized at each time index k and therefore only requires one iteration to find a satisfying trajectory as shown in Fig. 1c.

4.3 Weighted Maximum Operator Encoding

Unlike the minimum operator, we cannot directly encode the `max()` operator by removing the binary variables [6]. Instead, we replace the traditional MILP encoding of the `max()` operator in Eq. (5) with a new encoding as follows:

$$r^{\varphi} = \frac{1}{m} \sum_{i=1}^{m} r^{\varphi_i}. \tag{11}$$

The above encoding is akin to a multi-objective optimization problem, in which individual objectives are aggregated into a single objective through scalarization. The objective is to maximize a linear combination of all components within the `max()` operator. For the initial iteration in Alg. 1, each component r^{φ} is weighted equally. For later iterations, and as the weighting for predicates is augmented with critical values, the weights in the multi-objective optimization problem are adjusted accordingly. This allows the objective function to prioritize the satisfaction of certain components over others based on their criticality. For instance, consider the following example:

Example 2. We consider a system with one output signal $y(t)$ and a system specification $\varphi = \lozenge_{[0,10]} y > 5$ that we want to satisfy over a simulation time of $10s$.

[6] if $r^{\varphi} \geq r^{\varphi_i}$ and the objective is to maximize r^{φ}, then the problem is unbounded

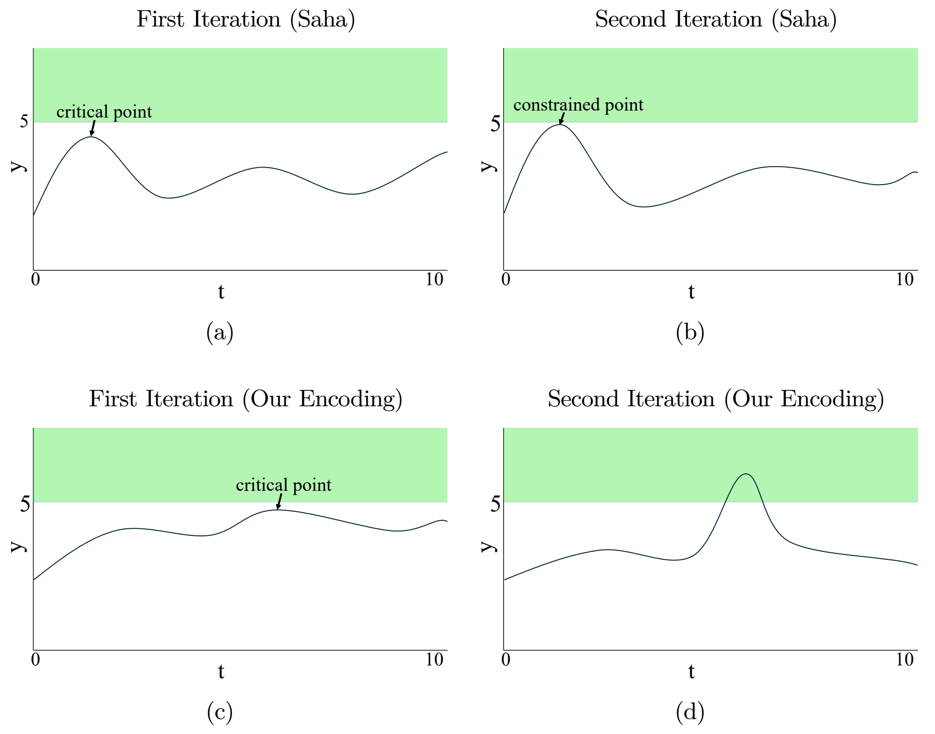

Fig. 2: Resultant trajectories (black line) and goal set (green) for Example 2. The top left figure (a) shows a random initial trajectory, which is the resultant trajectory after the first iteration of the iterative MILP approach by Saha and Julius [36]. The marked critical point is the point closest to the target set (max). Top right (b) shows the trajectory after the second iteration of Saha's approach. The bottom left (c) is the trajectory after one iteration using our encoding. The bottom right (d) shows the trajectory after the second iteration of our algorithm, satisfying the STL formula in Example 2.

From Def. 2, the robustness of the specification in Example 2 can be represented as:

$$r^{\varphi} = \min_{t \in [0,10]} r_t^{\varphi}, \tag{12}$$

where each r_t^{φ} represents the degree of satisfaction of the predicate $y > 5$ at time t. Following from the weighted encoding of predicates in Eq. (9) and from the maximum operator in Eq. (11), the robustness is encoded as

$$r^{\varphi} = \frac{1}{n} \sum_{k=1}^{n} w_k^p (y_k - 5 + b^p), \tag{13}$$

where n is the number of discrete time indices k in the time interval [0,10]. As all weights are initialized to 1 in the initial iteration of Alg. 1, the overall robustness r^{φ} is equally comprised of the degree of satisfaction of $y > 5$ at each time index k. Thus, the optimization Prob. 2 aim to maximize satisfaction at each time index k with equal reward. In subsequent iterations, the weighting of critical points is increased, and the objective function's priorities are shifted accordingly.

We illustrate resultant trajectories from Saha's approach [36] and our work in Fig. 2. First, we show a random initial trajectory in Fig. 2a, which corresponds to the trajectory after one iteration of the approach in [36]. The identified critical point is the maximum point. In the following iteration, a constraint is added at the critical point according to Eq. (8), and satisfaction at the critical point is maximized. However, the critical point for the random initial trajectory is not guaranteed to prove critical for the system overall. As the algorithm aim to maximize the initially identified point in subsequent iterations, the solver may never converge to a desired trajectory. Fig. 2b shows the next iteration of the algorithm, where the constrained point is maximized and once again corresponds to the critical point but never reaches the target state, i.e., never satisfies the STL requirement. Conversely, our approach in its first iteration aim to maximize satisfaction at all points with equal reward. The resultant trajectory is depicted in Fig. 2c. The weighting of the critical point is then modified as in Line 13 in Alg. 1. The iterative manipulation of weights based on critical points results in convergence to the desired behavior. Fig. 2d shows the final trajectory after the second iteration, where the STL formula in Example 2 is satisfied. In general, STL requirements with *weighted maximum encoding* are not guaranteed to converge to a solution. We show, however, that in practice, our encoding is highly successful, always finding a desired solution.

5 Weighted Koopman Operator Linearization

Koopman operator linearization [17,22] is a class of system identification methods that leverages the Koopman operator theory for learning nonlinear system dynamics. These methods produce competitive results for the analysis [29,7], control [31,18], and verification [4,5] of dynamical systems. Koopman operator linearization methods involve finding a nonlinear transformation from the original state space $y \in \mathbb{R}^l$ to *Koopman observables* by an observable function $g:\mathbb{R}^l \to \mathcal{H}$, where $\mathcal{H}$ is often a high dimensional vector space $\mathbb{R}^q, q > l$. This transformation enables learning a linear system

$$g(y_{k+1}) = Kg(y_k), \tag{14}$$

where $K \in \mathbb{R}^{q \times q}$ is the linear operator. Accordingly, Koopman operator linearization methods require learning the transformation g and operator K. To keep the math simple, we omit the external inputs u_k to the Koopman model in this section. However, these methods are easily extended to dynamical systems with input spaces by incorporating Koopman with inputs and control (KIC) [33]. KIC modifies Koopman's methods by augmenting the states with the inputs to accommodate controlled systems that are applicable to the systems we use in evaluation.

The falsification framework in Alg. 1 uses extended dynamic mode decomposition (eDMD) to learn Koopman surrogate models from a collection of trajectories.

Definition 3 (eDMD). *Given a sequence of data points $y_1, \ldots, y_n \in \mathbb{R}^l$, eDMD determines the system matrix $K \in \mathbb{R}^{q \times q}$ of the Koopman model in* (14) *by solving the* *following optimization problem:*

$$K = \underset{K \in \mathbb{R}^{q \times q}}{\operatorname{argmin}} \sum_{k=1}^{n-1} \| g(y_{k+1}) - K g(y_k) \|_2^2,$$

where $g : \mathbb{R}^l \to \mathbb{R}^q$ *is the observable function of the Koopman model* (14).

While eDMD traditionally fixes the observable function g, we select the best observables via hyperparameter tuning. The AutoKoopman framework [24] tunes hyperparameters for Koopman methods efficiently, automating the selection of optimal observables.

To improve the utility of the Koopman models for falsification, we employ weighting schemes for eDMD. Weighted regression problems are highly beneficial in contexts where some data points or parameter values are more significant than others due to their criticality or uncertainty. For system identification, a simple scheme is to weight each time index y_k in the training trajectories by a scalar $w_k \in \mathbb{R}_{\geq 0}$. Online DMD methods, for example, use this weighting to learn time evolving dynamics, weighting trajectory points non-zero over a time window [41,8].

Definition 4 (W-eDMD). *Weighted eDMD extends eDMD as given in Def. 4 by introducing scalar weights* $w_1, \ldots, w_n \in \mathbb{R}_+$ *for each time point*

$$K = \underset{K \in \mathbb{R}^{q \times q}}{\operatorname{argmin}} \sum_{k=1}^{n-1} w_k \| g(y_{k+1}) - K g(y_k) \|_2^2,$$

which allows some observations to contribute more weight to the objective.

Finally, a more sophisticated weighting scheme considers each state at every time point. Here, each of the l states of the state vector $y_k \in \mathbb{R}^l$ has a different weight, which is stored in the weight vector $w_k \in \mathbb{R}^l_{\geq 0}$. Since we chose the weight for each state to reflect its critically for the system specification, the objective function for surrogate model identification will prioritize the states that are most relevant for falsification.

We formulate a novel eDMD variant, called state weighted eDMD (SW-eDMD). The challenge we are facing with this approach is that in the objective function in Def. 3, we can only weight the observables g, but not the states y. We therefore have to transform the weights for the states to corresponding weights for the observables. To achieve this, we utilize the Jacobian matrix of the observable function, whose absolute value corresponds to the sensitivity of each observable to changes in the state variables. Observables that are more sensitive (i.e., have larger absolute derivatives) to highly weighted states are given more influence over the model learning objective.

Definition 5 (SW-eDMD). *State-weighted eDMD extends eDMD as given in Def. 4 by weighting both time indices and states using vector-values weights* $w_1, \ldots, w_n \in \mathbb{R}^l_{\geq 0}$

$$K = \underset{K \in \mathbb{R}^{q \times q}}{\operatorname{argmin}} \sum_{k=1}^{n-1} \left\| \operatorname{diag}(|J_y(y_k)| w_k)(g(y_{k+1}) - K g(y_k)) \right\|_2^2,$$

where $J_y = [\partial g/\partial y_{(1)}, \cdots, \partial g/\partial y_{(l)}]$ is the Jacobian matrix of g with respect to the state variables y. Here, notation $y_{(j)}$ denotes the j-th state variable of a state vector y, and the operator $\mathrm{diag}(v) \in \mathbb{R}^{q\times q}$ *with $v \in \mathbb{R}^q$ returns a square matrix with vector v on its diagonal.*

SW-eDMD, like eDMD, is solved by learning an operator K via convex optimization. To avoid overfitting in the presence of sparse or noisy data, eDMD often adds rank adaptation as a regularization method, retaining only the most significant modes. We implement rank adaptation by introducing a rank hyperparameter $1 \leq \sigma \leq q$ and learning the two rank-reduced matrices U and V instead of learning K directly,

$$K = UV, \quad U \in \mathbb{R}^{q\times\sigma},\, V \in \mathbb{R}^{\sigma\times q}. \tag{15}$$

The selection of a suitable rank hyperparameter σ can also be automated, for example by using cross-validation.

The main intuition behind the weighted approaches for Koopman linearization is that we want to bias learning towards critical subsets of data. In particular, for a set of learning trajectories, we aim to increase the weighting of trajectories that are closer to falsifying (satisfying) the system. The weighting of each trajectory is then directly proportional to its robustness. Such that for each trajectory τ with robustness $\mathcal{Q}(\varphi,\tau)$, we directly use the robustness as the weight for all time steps.

$$w_k = \mathcal{Q}(\varphi,\tau) \quad \forall k \in \{1,\ldots,n\}. \tag{16}$$

We note that while this setup allows for different weighting of time points, we choose to weight all time points equally. The rationale is that weighting time points differently could negatively impact the weighted robustness encoding approach described in Sec. 4, which already applies different weights to critical time points.

Another important observation is that for a given STL formula, state variables do not carry the same importance. In particular, it is expected that state variables more heavily featured in the STL formula have a larger influence on the overall robustness. Following the recursive encoding of robustness, we assign weighting priority with SW-eDMD in proportion to the number of predicates for which a state variable is featured. We start by setting the weight of each state to the total number of time indices n. We then recursively traverse through each operator in the STL formula, updating the weights. For this, we view the STL formula as a tree with temporal operators acting as the nodes, as visualized in Fig. 3. The weights grow multiplicatively based on the number of time indices n_φ associated with each temporal operator along a path. We define a *path* as a complete traversal from the root of the tree (STL formula) to a leaf (atomic predicate). For instance, consider the path to atomic predicate p_1 in Fig. 3, for a system with a time step of $\Delta t = 0.1$ seconds. At the root node, the temporal operator $\square_{[0,2]}$ has an associated number of time indices $n_\varphi = 20$. Similarly, for the next operator on the path $\lozenge_{[3,4]}$ spanning a time horizon of $1s$, we have $n_\varphi = 10$. The overall weight for each state i in predicate p_1 is then $w_{(i)} = 200n$. This process is repeated for all atomic predicates and associated states. The state weights returned are then used to bias learning of Koopman operators in the SW-eDMD framework.

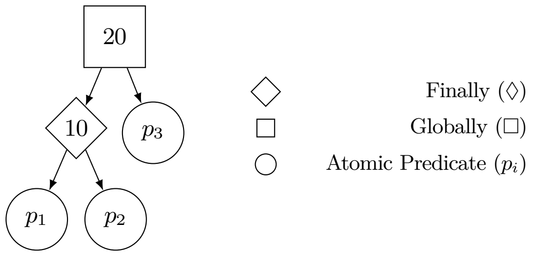

Fig. 3: Tree structure for STL formula $\varphi = \Box_{[0,2]}(\Diamond_{[3,4]}(p_1 \wedge p_2) \ \vee \ p_3)$ for a system with a time step of $\Delta t = 0.1$ seconds. The values in temporal operator nodes represent the number of time indices for the operator n_φ.

6 Evaluation

In this section, we present the empirical evaluation of our proposed approach. We incorporate *weighted robustness encoding* and *weighted Koopman operator linearization* in the open-source tool FReaK- Falsification using Reachability and Koopman [7]. We use the default settings for the tool and the default parameters in [6] unless otherwise specified. We evaluate our approaches on the diverse set of benchmarks from the 2022 ARCH competition [11], where we average the results over 10 attempts. As evaluation metrics, we record the *falsification rate* FR, which specifies how many of the 10 runs were successful, and the *average and median number of simulations*, $\overline{S}$ and $\widetilde{S}$, of the real system that is needed to falsify the specification. We also report the percentage of overall computation time spent running simulations of the real system, $\text{R} = \frac{\text{Simulation Time}}{\text{Total Time}}(\%)$. A higher R-value directly correlates to less computation time spent in search of critical trajectories. We use $N_{\text{max}} = 1000$ as the limit on the number of iterations of Algorithm 1. All presented computations were done on a 3.4GHz AMD Ryzen 9 5950X 16-core processor with 64GB memory.

We divide our evaluation section into two main parts. First, in Sec. 6.1, we evaluate the *weighted robustness encoding* and compare its efficiency against the traditional MILP encoding in [6] and Saha's approach in [36]. In Sec. 6.2, we discuss the results obtained using *weighted Koopman operator linearization*.

We remark that in previous work by Bak et al. [6], the comparison with state-of-the-art falsification tools shows that the Koopman falsification method with traditional MILP encoding is highly effective. The approach outperforms all other participating tools on 16 out of 19 benchmarks from the ARCH falsification competition. For brevity, we omit the comparison with other tools from this paper. Instead, we focus on comparing our novel weighted approaches with traditional Koopman falsification. We also omit benchmarks for which no falsifying trace was found.

[7] https://github.com/Abdu-Hekal/FReaK

Table 1: Comparison of the performance of different robustness encodings on the benchmarks from the ARCH competition (instance 1), where the results are averaged over 10 executions. The evaluation metrics are the mean $\overline{S}$ and median $\widetilde{S}$ number of simulations for successful runs, the simulation time percentage R, as well as the falsification rate FR.

Bench.	**MILP**				**Saha [36]**				**W-Rob**			
	FR	$\overline{S}$	$\widetilde{S}$	R	FR	$\overline{S}$	$\widetilde{S}$	R	FR	$\overline{S}$	$\widetilde{S}$	R
AT1	10	7.9	4.0	43.5	10	10.7	10.5	46.7	10	5.1	2.5	46.9
AT2	10	2.1	2.0	61.0	3	572	579	32.6	10	2.0	2.0	62.4
AT51	10	5.7	4.5	22.4	10	3.9	4.0	28.1	10	22.7	8.0	25.9
AT52	10	1.3	1.0	37.3	10	1.8	1.0	41.2	10	1.2	1.0	56.8
AT53	10	1.7	1.0	37.9	10	1.5	1.0	48.9	10	1.6	1.0	43.1
AT54	10	1.9	1.0	33.0	10	2.1	1.0	39.1	10	2.4	1.0	33.5
AT6a	10	6.8	5.5	7.8	8	14.6	11.5	45.1	10	5.1	4.5	46.4
AT6b	10	11.0	6.0	40.5	10	20.9	19.0	43.5	10	6.1	5.0	44.1
AT6c	10	9.9	6.0	39.1	10	58.6	32.0	40.3	10	5.4	5.0	44.3
AT6abc	10	12.5	7.5	27.0	8	127	80.0	19.4	10	6.1	4.5	32.6
NN	10	2.0	2.0	51.5	10	4.4	4.5	51.8	10	2.0	2.0	59.5
NN$_{\beta=0.04}$	10	30.9	33.0	39.1	10	68.5	71.0	40.6	10	61.4	43.0	37.8
CC1	10	3.6	3.0	74.2	10	6.8	7.0	73.5	10	5.1	5.0	71.1
CC2	10	3.0	3.0	67.3	10	6.6	4.5	73.4	10	3.1	3.0	72.9
CC3	10	5.7	5.5	57.0	10	11.9	10.5	65.7	10	6.2	4.0	63.5
CC4	10	188.5	140.0	57.0	10	150	112	63.1	10	158	150	59.5
CC5	10	48.2	10.5	49.9	10	18.8	16.5	57.7	10	37.4	38.5	55.1
CCx	10	110.5	62.0	50.2	10	61.7	49.5	49.1	10	166	147	42.6
SC	10	41.6	29.5	6.7	0	-	-	-	10	123	104	6.7

6.1 Weighted Robustness Encoding

Here, we evaluate the efficiency of the weighted robustness encoding in Sec. 4. We compare our weighted encoding (W-Rob) against the traditional encoding (MILP) in Table 1. It is expected that traditional encoding incurs a lower number of simulations, as it exactly solves Prob. 2. However, our weighted encoding offers a unique advantage in this context. The learned Koopman model $\mathcal{M}_K$ is not exact, so that the robustness for $\mathcal{M}_K$ does not directly correspond to that of the real system. Herein lies the benefit of our encoding, where the weighting of predicates and the corresponding critical points are modified based on the behavior of the real system. Essentially, weighting adjusts not only for inaccuracies in the heuristic encoding, but also for inaccuracies in model learning. In fact, we observe in Table 1 that W-Rob, in comparison with MILP, records a lower number of average simulations per falsifying trace for 10 of the 19 benchmarks.

The main advantage the weighted encoding is expected to offer over traditional MILP encoding is reduced computational complexity. As seen in Table 1, W-Rob

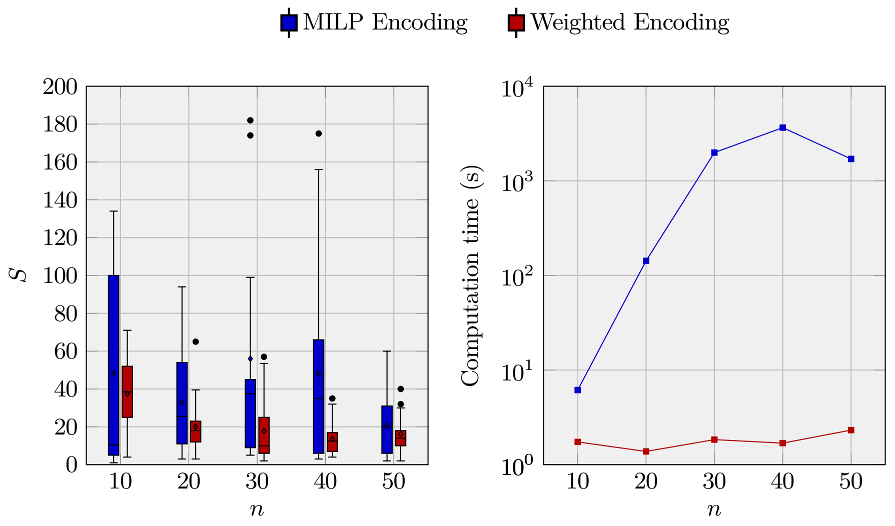

Fig. 4: Number of simulations S (left) and computation time in seconds for the optimizer (right) for the CC5 benchmarks from the ARCH competition over 10 executions, where we compare traditional MILP encoding and our new weighted encoding for an increasing number of time points n. The black dots in the left figure represent the outliers from all executions. The y-axis for the figure on the right is displayed on a logarithmic scale.

records a higher R-value than MILP encoding for 16 of the 19 benchmarks. Although there can be substantial differences in R values, such as in the $AT6a$ benchmark (7.8 and 46.4), the R values across most benchmarks are comparable for each encoding. This can be attributed to the limited number of time indices n for robustness encoding used in the previous Koopman falsification work [6] to ensure MILP feasibility. In general, it is expected that a larger number of time indices will result in more accurate results, as it will allow for a more fine-grained analysis of the system behavior. This holds particularly true for complex requirements. Consider requirement CC5:$=\Box_{[0,72]}\Diamond_{[0,8]}(\neg(\Box_{[0,5]}y_{(2)}-y_{(1)}\geq 9)\wedge(\Box_{[5,20]}y_{(5)}-y_{(4)}\geq 9))$ from the ARCH competition. The time horizon for the problem is $T=100s$, and the default discretization time step in Table 1 is $\Delta t=10s$, resulting in $n=10$. In Fig. 4, we evaluate the number of simulations S (left) and computation time for the optimizer (right) for an increasing number of time indices. As expected, the number of simulations required to falsify the system generally reduces for a finer discretization. It is also worth noting that our weighted method needs fewer simulations than traditional MILP across all discretization levels. Further, the reduced computation time for our weighted encoding is evident, particularly noticeable with finer discretization levels, where traditional MILP encoding exhibits exponential growth in computation time. In fact, for $n=50$ time indices where the discretization time step is $\Delta t=2s$, our weighted encoding requires 2.32 seconds compared to 1705 seconds for MILP encoding.

Table 2: Comparison of the performance of different weighted Koopman approaches on the benchmarks from the ARCH competition (instance 1), where the results are averaged over 10 executions. The evaluation metrics are the mean $\overline{S}$ and median $\widetilde{S}$ number of simulations for successful runs, the simulation time percentage R, as well as the falsification rate FR.

Bench.	**eDMD**				**W-eDMD**				**SW-eDMD**			
	FR	$\overline{S}$	$\widetilde{S}$	R	FR	$\overline{S}$	$\widetilde{S}$	R	FR	$\overline{S}$	$\widetilde{S}$	R
AT1	10	5.1	2.5	46.9	10	4.6	3.0	47.0	10	5.1	4.0	41.8
AT2	10	2.0	2.0	62.4	10	2.0	2.0	63.6	10	2.0	2.0	59.1
AT51	10	22.7	8.0	25.9	10	27.2	20.5	25.8	10	22.7	8.0	23.8
AT52	10	1.2	1.0	56.8	10	1.7	1.0	44.6	10	1.2	1.0	56.0
AT53	10	1.6	1.0	43.1	10	1.4	1.0	49.7	10	1.6	1.0	38.7
AT54	10	2.4	1.0	33.5	10	1.9	1.0	40.0	10	2.4	1.0	31.3
AT6a	10	5.1	4.5	46.4	10	6.0	5.0	45.7	10	5.1	4.5	42.5
AT6b	10	6.1	5.0	44.1	10	8.2	5.5	43.0	10	7.0	5.0	40.0
AT6c	10	5.4	5.0	44.3	10	6.2	3.0	45.1	10	6.7	5.0	39.7
AT6abc	10	6.1	4.5	32.6	10	3.2	3.0	38.2	10	6.1	4.5	30.4
NN	10	2.0	2.0	59.5	10	2.4	2.0	55.8	10	2.0	2.0	60.1
$\text{NN}_{\beta=0.04}$	10	61.4	43.0	37.8	10	91.7	49.0	41.5	10	40.1	42.5	39.8
CC1	10	5.1	5.0	71.1	10	6.1	5.0	71.4	10	7.5	7.0	68.1
CC2	10	3.1	3.0	72.9	10	2.7	3.0	73.9	10	2.7	3.0	70.9
CC3	10	6.2	4.0	63.5	10	8.0	6.5	60.0	10	4.4	4.0	61.4
CC4	10	158	150	59.5	10	93.6	77.5	60.0	10	90.2	72.5	55.2
CC5	10	37.4	38.5	55.1	10	22.4	15.5	54.2	10	29.2	22.0	50.1
CCx	10	166	147	42.6	10	198	168	41.6	10	88.8	82.5	39.7
SC	10	123	104	6.7	10	150	140	6.9	10	66.3	32.5	5.8

Another natural candidate for our comparison is Saha's approach [36] as discussed throughout the paper. The approach incrementally introduces constraints, resulting in low computation times (i.e., large R values), as seen in Table 1. Saha's approach performs well on some benchmarks, particularly the CC benchmarks. This is once again due to the small number of time points ($n=10$) used for the CC benchmarks. With fewer time points, fewer constraints are needed, and the iterative approach can quickly converge to a falsifying trace. However, for benchmarks that require a larger number of time indices, Saha's approach typically requires a larger number of simulations, such as for AT6c ($n=30$), where it requires, on average, 58.6 simulations compared to our 5.4. Additionally, it can encounter failures on particular runs, as in AT2 ($n=30$), where it fails to find a falsifying instance for 7 of the 10 runs and incurs an average of 572 simulations in the 3 instances where it succeeds. For the SC ($n=150$) benchmark, it fails to produce a falsifying trace for any run. The results are consistent with observations discussed in Sec. 4.

6.2 Weighted Koopman Operator Linearization

In Table 2, we evaluate the weighted approach for Koopman linearization introduced in Sec. 5. We compare baseline eDMD with W-eDMD, where trajectories are weighted, and SW-eDMD, where both trajectories and states are assigned individual weights. While W-eDMD incurs a lower number of simulations on some benchmarks compared to eDMD, e.g., CC4, CC5, the performance is not consistent across all benchmarks. W-eDMD incurs a larger number of simulations on average for 11 of the 19 benchmarks. One reason for this behavior is overfitting, where, for particular benchmarks, the robustness of trajectories are vastly different, leading to a heavily skewed weighting scheme. Another important aspect is that the hyperparameters in the previous Koopman falsification work [6] were tuned for eDMD, and we retain these parameters for direct comparison purposes. Conversely, and despite these factors, SW-eDMD demonstrates significant improvement over eDMD, particularly for challenging benchmarks. For example, SW-eDMD needs 40.1 simulations to find a falsifying trace for NN_β, whereas eDMD requires 61.4 simulations. Similarly, for the SC benchmark, SW-eDMD requires 66.3 simulations compared to 123 for eDMD. In fact, for all benchmarks that require more than 10 simulations to find a falsifying trace, SW-eDMD outperforms eDMD. The R values are similar across all three approaches, with a slight increase in computation time (decrease in R) for SW-eDMD, which can be attributed to its use of the third-party optimizer, Clarabel [8].

7 Conclusion

Falsification based on MILP and Koopman surrogate models has proven highly effective [6]. However, MILP performance significantly worsens with increasing size and complexity. In this paper, we introduced a new heuristic encoding that reduces the problem to a linear formulation. We rely on an iterative weighted approach that guides the optimization based on real system behavior. We evaluate our approach on the set of benchmarks from the ARCH falsification competition and show its efficacy, particularly for more complex problems. On a similar note, we proposed a new weighting method for learning the Koopman surrogate models. We developed a novel eDMD variant termed State Weighted Extended Dynamic Mode Decomposition (SW-eDMD) to bias learning toward states and trajectories most relevant to property violation. We show that the benefit of SW-eDMD is particularly evident for complex requirements, where it incurs a significantly fewer number of simulations to converge to a solution.

Acknowledgements. This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156, the ANR-JST research project CyPhAI under award numbers ANR-20-JSTM-0001 and JPMJCR2012, and the National Science Foundation under Award No. 2237229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or the United States Navy.

[8] https://clarabel.org

References

1. Abbas, H., Fainekos, G.: Convergence proofs for simulated annealing falsification of safety properties. In: Proc. of the Annual Allerton Conference on Communication, Control, and Computing. pp. 1594–1601 (2012)
2. Annapureddy, Y.S.R., Fainekos, G.: Ant colonies for temporal logic falsification of hybrid systems. In: Proc. of the Annual Conference of the IEEE Industrial Electronics Society. pp. 91–96 (2010)
3. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In: Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–257 (2011)
4. Bak, S., et al.: Reachability of black-box nonlinear systems after Koopman operator linearization. In: Proc. of the International Conference on Analysis and Design of Hybrid Systems. pp. 253–258 (2021)
5. Bak, S., et al.: Reachability of Koopman linearized systems using random Fourier feature observables and polynomial zonotope refinement. In: Proc. of the International Conference on Computer Aided Verification. pp. 490–510 (2022)
6. Bak, S., et al.: Falsification using reachability of surrogate Koopman models. In: Proc. of the International Conference on Hybrid Systems: Computation and Control (2024), Article No. 27
7. Bevanda, P., Sosnowski, S., Hirche, S.: Koopman operator dynamical models: Learning, analysis and control. Annual Reviews in Control **52**, 197–212 (2021)
8. Cheng, Z., Trépanier, M., Sun, L.: Real-time forecasting of metro origin-destination matrices with high-order weighted dynamic mode decomposition. Transportation Science **56**(4), 904–918 (2022)
9. Deshmukh, J., et al.: Testing cyber-physical systems through Bayesian optimization. ACM Transactions on Embedded Computing Systems **16**(5s) (2017), Article No. 170
10. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Proc. of the International Conference on Computer Aided Verification. pp. 167–170 (2010)
11. Ernst, G., et al.: ARCH-COMP 2022 category report: Falsification with ubounded resources. In: Proc. of the International Workshop on Applied Verification of Continuous and Hybrid Systems. pp. 204–221 (2022)
12. Ernst, G., Sedwards, S., Zhang, Z., Hasuo, I.: Fast falsification of hybrid systems using probabilistically adaptive input. In: Proc. of the International Conference on Quantitative Evaluation of Systems. pp. 165–181 (2019)
13. Fainekos, G., Pappas, G.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science **410**(42), 4262–4291 (2009)
14. Ferrère, T., et al.: Interface-aware signal temporal logic. In: Proc. of the International Conference on Hybrid Systems: Computation and Control. pp. 57–66 (2019)
15. Garey, M.R., Johnson, D.S.: A guide to the theory of NP-completeness. Computers and Intractability pp. 37–79 (1990)
16. Gilpin, Y., Kurtz, V., Lin, H.: A smooth robustness measure of signal temporal logic for symbolic control. IEEE Control Systems Letters **5**(1), 241–246 (2020)
17. Koopman, B.O.: Hamiltonian systems and transformation in Hilbert space. Proceedings of the National Academy of Sciences of the United States of America **17**(5), 315–318 (1931)
18. Korda, M., Mezić, I.: Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. Automatica **93**, 149–160 (2018)

19. Kurtz, V., Lin, H.: Trajectory optimization for high-dimensional nonlinear systems under STL specifications. IEEE Control Systems Letters **5**(4), 1429–1434 (2020)
20. Kurtz, V., Lin, H.: A more scalable mixed-integer encoding for metric temporal logic. IEEE Control Systems Letters **6**, 1718–1723 (2021)
21. Kurtz, V., Lin, H.: Mixed-integer programming for signal temporal logic with fewer binary variables. IEEE Control Systems Letters **6**, 2635–2640 (2022)
22. Kutz, J.N., Brunton, S.L., Brunton, B.W., Proctor, J.L.: Dynamic mode decomposition: Data-driven modeling of complex systems. SIAM (2016)
23. Leung, K., Aréchiga, N., Pavone, M.: Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. The International Journal of Robotics Research **42**(6), 356–370 (2023)
24. Lew, E., et al.: AutoKoopman: A toolbox for automated system identification via Koopman operator linearization. In: Proc. of the International Symposium on Automated Technology for Verification and Analysis. pp. 237–250 (2023)
25. Lindemann, L., Dimarogonas, D.V.: Control barrier functions for signal temporal logic tasks. IEEE Control Systems Letters **3**(1), 96–101 (2018)
26. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Proc. of the International Conference on Formal Modelling and Analysis of Timed Systems. pp. 152–166 (2004)
27. Mathesen, L., Pedrielli, G., Fainekos, G.: Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements. In: Proc. of the International Conference on Automation Science and Engineering. pp. 732–737 (2021)
28. Menghi, C., Nejati, S., Briand, L., Parache, Y.I.: Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In: Proc. of the International Conference on Software Engineering. pp. 372–384 (2020)
29. Mezić, I.: Analysis of fluid flows via spectral properties of the Koopman operator. Annual Review of Fluid Mechanics **45**, 357–378 (2013)
30. Nghiem, T., et al.: Monte-Carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Proc. of the International Conference on Hybrid Systems: Computation and Control. pp. 211–220 (2010)
31. Otto, S.E., Rowley, C.W.: Koopman operators for estimation and control of dynamical systems. Annual Review of Control, Robotics, and Autonomous Systems **4**, 59–87 (2021)
32. Pant, Y.V., Abbas, H., Mangharam, R.: Smooth operator: Control using the smooth robustness of temporal logic. In: Prof. of the International Conference on Control Technology and Applications. pp. 1235–1240 (2017)
33. Proctor, J.L., Brunton, S.L., Kutz, J.N.: Generalizing Koopman theory to allow for inputs and control. SIAM Journal on Applied Dynamical Systems **17**(1), 909–930 (2018)
34. Raman, V., et al.: Model predictive control with signal temporal logic specifications. In: Prof. of the International Conference on Decision and Control. pp. 81–87 (2014)
35. Raman, V., et al.: Reactive synthesis from signal temporal logic specifications. In: Proc. of the International Conference on Hybrid Systems: Computation and Control. pp. 239–248 (2015)
36. Saha, S., Julius, A.A.: An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications. In: Proc. of the American Control Conference. pp. 1105–1110 (2016)
37. Sankaranarayanan, S., Fainekos, G.: Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Proc. of the International Conference on Hybrid Systems: Computation and Control. pp. 125–134 (2012)
38. Takayama, Y., Hashimoto, K., Ohtsuka, T.: Signal temporal logic meets convex-concave programming: A structure-exploiting SQP algorithm for STL specifications. In: Proc. of the International Conference on Decision and Control. pp. 6855–6862 (2023)

39. Waga, M.: Falsification of cyber-physical systems with robustness-guided black-box checking. In: Proc. of the International Conference on Hybrid Systems: Computation and Control (2020), Article No. 11
40. Yamagata, Y., et al.: Falsification of cyber-physical systems using deep reinforcement learning. IEEE Transactions on Software Engineering **47**(12), 2823–2840 (2020)
41. Zhang, H., Rowley, C.W., Deem, E.A., Cattafesta, L.N.: Online dynamic mode decomposition for time-varying systems. SIAM Journal on Applied Dynamical Systems **18**(3), 1586–1609 (2019)
42. Zhang, Z., et al.: Effective hybrid system falsification using Monte Carlo tree search guided by QB-robustness. In: Prof. of the International Conference on Computer Aided Verification. pp. 595–618 (2021)