Quality assurance of cyber-physical systems (CPS) has been recognized as an important challenge, where many CPS are hybrid systems that combine the discrete dynamics of computers and the continuous dynamics of physical components. Unfortunately, analysis of hybrid systems poses unique challenges, such as the limited applicability of formal verification.

Due to these difficulties, falsification problem is pursued by a increasing number of researchers for quality assurance. The goal of falsification is to find a counterexample that violates the specification, so it is a testing issue rather than formal verification. The problem is formalized as that given a system model $M$ and a specification $\varphi$ answer whether there is an input $u$ such that the corresponding output signal $M(u)$ violates $\varphi$. Here, $M$ is usually a black or grey box model, meaning that it is infeasible to investigate into the model, $u$ is an input signal and $M(u)$ is its corresponding output signal. The property $\varphi$ is usually given in temporal logic.

One typical way of solving falsification problem is the optimization-based strategy, thanks to the quantitative robustness semantics of temporal logic such as STL [1]. Instead of the Boolean satisfaction relation $v \models \varphi$, robust semantics assigns a quantity $[v, \varphi] \in \mathbb{R} \cup \{\infty, -\infty\}$ that tells us, not only whether $\varphi$ is true or not, but also how robustly the formula is true or false. This allows one to employ hill-climbing optimization by setting the robustness score $[v, \varphi]$ as the objective function. Hill-climbing optimization is usually based on the previous observations to iteratively search for an input that minimizes the output. The goal here for falsification is just to optimize the value of output till it becomes negative.

Optimization-based falsification has been implemented in the state-of-the-art tool Breach [2]. One can specify the goal property and the type of input signal. Breach provides multiple optimization solvers, including CMA-ES [3], Nelder-Mead, and Simulated Annealing.

**TIME-STAGING HEURISTIC RULE**

We introduce a heuristic rule that enhances the falsification ability of the optimization-based strategy. We illustrate this method by a motivating example from an automotive system model [6]: The STL requirement $\phi \equiv \square_{[0, 30]}(speed < 120)$ means that the speed should remain below 120 in the time interval $[0, 30]$. To falsify it, the speed should exceed 120 at some moment during $[0, 30]$. We assume the input signal is piecewise constant.

The optimization-based approach is shown in Fig. 1. First, it tries the $i$-th input signal, and get an output signal after running the simulation. Note that the robustness score is the vertical distance between the peak of the output signal and the constant 120. Then, the solver optimizes it by trying the $(i + 1)$-th input signal. The robustness score of the resulting output signal is better.

Our observation is that here the corresponding output signal of $u_{i}^{(1)}$ and $u_{i}^{(2)}$ is better than that of $u_{i}^{(1+1)}$ and $u_{i}^{(1)}$; however, the corresponding output signal of $u_{i}^{(1+1)}$ and $u_{i}^{(1+1)}$ is better than that of $u_{i}^{(1)}$ and $u_{i}^{(1)}$. If we combine $u_{i}^{(1)}$, $u_{i}^{(2)}$, $u_{i}^{(1+1)}$, $u_{i}^{(1+1)}$ to be a new signal, the resulting output signal will be better than either of the $i$-th or $(i + 1)$-th output signal.
TABLE I: Experimental results.

| spec. | S1 | S2 | S3 easy | S3 hard | S4 easy | S4 mad | S4 hard | S init | S stable |
|-------|----|----|---------|---------|---------|--------|---------|--------|---------|
| algorithm | time #/20 | time #/20 | time #/20 | time #/20 | time #/20 | time #/20 | time #/20 | time #/20 | time #/20 |
| CMA-ES | 75s 0* | 79s 0* | 89s 1* | 35s 0* | 85s 0* | 66s 0* | 89s 0* | 33s 0* | 50s 0* |
| +TS | 52s 0* | 15s 0* | 16s 0* | 23s 0* | 9s 0* | 21s 10 | 23s 0* | 16s 0* | 14s 0* |
| +A-TS | 41s 18 | 15s 17 | 9s 16 | 21s 10 | 23s 0* | 23s 0* | 21s 10 | 23s 0* | 23s 0* |
| SA | 50s 0* | 43s 0* | 37s 9 | 55s 0* | 35s 6 | 36s 9 | 47s 5 | 30s 0 | 42s 1 |
| +TS | 37s 20 | 33s 16 | 11s 19 | 33s 8 | 21s 14 | 25s 13 | 51s 0 | 34s 0 | 42s 7 |
| +A-TS | 34s 20 | 18s 17 | 9s 18 | 26s 4 | 16s 18 | 21s 11 | 30s 2 | 34s 0 | 42s 5 |
| GNM | 6s 0* | 61s 0* | 56s 0* | 55s 0* | 43s 0* | 46s 0* | 53s 0* | 50s 0* | 66s 0* |
| +TS | 42s 20* | 15s 20* | 13s 20* | 25s 20* | 11s 20* | 45s 0* | 52s 0* | 30s 20* | 20s 20* |
| +A-TS | 20s 20* | 16s 20* | 10s 20* | 26s 20* | 13s 20* | 45s 0* | 43s 0* | 37s 0* | 19s 20* |

Algorithm 1 Time-Staged Strategy

Require: falsification solver Falsify, system model $M$, STL formula $\varphi$, $T \in \mathbb{R}_{>0}$ and $K \in \mathbb{N}$ stages

1. $u \leftarrow ()$ 
   \[ \triangleright \text{start with the empty signal} () \]
2. for $j \in \{1, \ldots, K\}$ do
3. \quad $u' \leftarrow \text{Falsify} (M_u, \partial M(u)\rho_{\varphi}, T)$
4. \quad $u \leftarrow u \cdot u'$ 
5. end for
6. return $u$

Based on this observation, we design our time-staged algorithm as shown in Fig. 2 and Algorithm 1. At the first stage, we obtain input signals through random sampling and optimization, and select the one that results in the best output signal. At the following stages, we repeat the same strategy until one output signal that violates the property occurs. An in-depth discussion of the algorithm and comparison to related work can be found in the preprint [4]. A recent survey of the field is [5].

EXPERIMENTS

Table I shows our experimental results for two Simulink models, Automatic Transmission (AT) [6] and Abstract Fuel Control (AFC) [7]. We tested the performance of 9 algorithms shown in Col. 1, where CMA-ES, GNM, and SA are optimization algorithms implemented in Breach, the algorithms with “+TS” are the applications of our time-staging strategy to the optimization algorithms, and those with “+A-TS” are improved versions with the number of samples at each stage adaptive. Row 2 shows the properties we tested. For AT, they are S1: $\Box_{[0,30]} (v < 120)$, S2: $\Box_{[0,30]} (g = 3 \rightarrow v \geq 30)$, S3: $\Diamond_{[0,30]} (v \geq v_{\min} \vee v \geq v_{\max})$, where: $v_{\min} = 50$, $v_{\max} = 60$ (easy); $v_{\min} = 53$, $v_{\max} = 57$ (hard). S4: $\Box_{[0,10]} (v < v \vee \Diamond_{[0,30]} (\omega > \omega_{\max})$, where: $v_{\min} = 80$, $\omega_{\max} = 4500$ (easy); $v_{\min} = 50$, $\omega_{\max} = 2700$ (mid); $v_{\min} = 50$, $\omega_{\max} = 2520$ (hard). For AFC, they are $\neg (\Diamond_{[1,t_2]} \Box_{[0,t']} (AF - AF_{\text{ref}} > \delta > 14.7))$, where: $t_1 = 0$, $t_2 = 6$, $t' = 3$, $\delta = 0.07$ (init); $t_1 = 6$, $t_2 = 26$, $t' = 4$, $\delta = 0.01$ (stable). Starred numbers 0* or 20* indicate that GNM in Breach is deterministic so all trials yield the same result. In the table, we show the average runtime and success rate out of 20 trials.

DISCUSSION

From Table I, we can see that for some properties such as S3 easy, S3 hard and S stable, our new algorithms improve the efficiency. That is partially because that our algorithm reduces the search space. Let $U$ be the space for one input segment, and let $K$ be the total number of stages. The whole search space for unstaged optimization approach is $|U|^K$, while for our algorithm it is $K \cdot |U|$. However, this reduction comes with the risk of missing some falsifying input signals.

Also for some properties, our algorithm improves the falsification rate, notably for those properties that Breach cannot falsify at all such as S3 hard and S init. We can think about which kind of properties that our strategy suits for. If the counterexample input signal can be decomposed into several segments, which means that the global optimum is composed of local optimum, then our algorithm will solve the problem much faster.

However in the end, there are some properties such as CMA-ES on S2, SA on S4 hard where our strategy works not as well as Breach.

Acknowledgement. The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, and Grants-in-Aid No. 15KT0012, JSPS.

REFERENCES

[1] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In FORMATS 2004.
[2] A. Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In CAV 2010.
[3] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In CEC 2005.
[4] G. Ernst and I. Hasuo and Z. Zhang, and S. Sedwards. Time-Staging Enhancement of Hybrid System Falsification. arXiv:1803.03866, 2018.
[5] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Niković, and S. Sankaranarayanan. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In Lectures on Runtime Verification, 2018.
[6] B. Hoxha, H. Abbas, and G. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In ARCH 2014.
[7] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. R. Butts. Powertrain control verification benchmark. In HSCC 2014.