Hybrid System Falsification using Monte Carlo Tree Search (Abstract)

Zhenya Zhang[1], Gidon Ernst[2], Sean Sedwards[3], Paolo Arcaini[1], Ichiro Hasuo[1]
[1]National Institute of Informatics [2]LMU Munich [3]University of Waterloo
{zhangzy, arcaini, hasuo}@nii.ac.jp gidon.ernst@sosy.ifi.lmu.de sean.sedwards@uwaterloo.ca

Quality assurance of Cyber-Physical Systems (CPS) is drawing more and more attention from both academia and industry, not only because it is safety-critical, but it is also a challenging problem. Formal verification techniques based on system state exploration are infeasible to CPS due to its hybrid nature in which both discrete and continuous dynamics exist, and thus the state space is infinite. In contrast, testing techniques, namely *falsification*, which aims at finding a counterexample to refute a system specification, are well adapted for hybrid systems. Falsification problem can be illustrated as Fig. 1. $\mathcal{M}$ is the system model, $\mathbf{u}$ is an input signal for $\mathcal{M}$, and φ is a pre-defined system specification expressed in Signal Temporal Logic (STL) [1]. The goal of the problem is to find an input signal $\mathbf{u}$ such that the corresponding output signal $\mathcal{M}(\mathbf{u})$ violates φ.

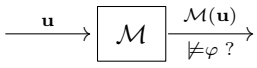

Fig. 1: Falsification

A typical way of solving falsification problems is based on the so-called *hill-climbing optimization* technique which aims at finding an input that minimizes the objective function. The objective function here is called *robustness*, ranging over real numbers, as defined in the robust semantics [2] of STL. The intuition of robustness for an output signal $\mathbf{w}$ related to φ reflects how robustly $\mathbf{w}$ satisfies φ: a larger positive value means that $\mathbf{w}$ is less vulnerable to violating φ, while a negative value means that $\mathbf{w}$ already violates φ. There have been several mature tools based on this idea, such as Breach [3], S-TaLiRo [4], in which hill-climbing optimization algorithms, like CMA-ES, Simulated Annealing, etc., are employed for falsification.

Monte Carlo Tree Search and Falsification

Hill-climbing optimization based falsification techniques, however, may easily fall into the "local optimum" trap and thus fail to falsify the specification. That is because most hill-climbing algorithms focus on exploitation of local optimum too much, ignoring the existence of global optimum elsewhere. A technique that helps to jump out from the trap is needed.

Z. Zhang, P. Arcaini, and I. Hasuo are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

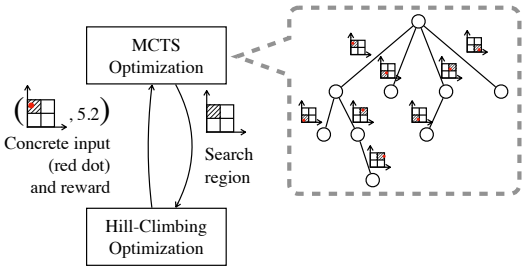

Fig. 2: A two-layered framework: MCTS for global guidance, and hill climbing for local search

Monte Carlo Tree Search (MCTS) [5] is an advanced artificial intelligence technique that balances exploration and exploitation in search-based testing problems. It has achieved a great success in computer-Go games where a globally optimal strategy is needed.

MCTS follows a 4-step strategy:

- Selection: start from the root R of a tree and select successive child nodes until a leaf is reached.
- Expansion: create a node N to expand the search.
- Playout: randomly select child from node N to complete the game and record a binary result.
- Backpropagation: update the number of wins and visits of each node on the path from N to R based on the playout result.

MCTS iteratively applies the 4 steps to update the winning rate (called as *reward*) of one path. In order not to bias local exploitation too much, MCTS employs *UCB1 (Upper Confidence Bound)*, where the number of visits to a given branch is also considered, to balance the exploration and exploitation during the search.

In this work, we borrow the idea of MCTS and build a two-layered framework as Fig. 2 shows. On the high level, MCTS globally guides the search by figuring out a promising direction; then, on the low level, hill-climbing optimization is employed in the local search to dig out a concrete solution in the direction given by MCTS.

The technical details follow [6].We divide the time bound of a signal into n intervals. Then we discretize the search space on each interval by partitioning it into m sub-spaces. A tree $\mathcal{T}$, of depth n and degree m, is built accordingly. Starting from the root R, we follow

TABLE I: Experimental results

		AT model										AFC model				FFR model	
		S1		S2		S3		S4		S5		Sbasic		Sstable		Strap	
	Algorithm	succ.	time	succ.	time	succ.	time	succ.	time	succ.	time	succ.	time	succ.	time	succ.	time
	Random	10/10	108.9	10/10	289.1	1/10	301.1	0/10	-	0/10	-	6/10	278.7	10/10	242.6	4/10	409.3
CMA-ES	Breach	10/10	21.9	6/10	30.3	**10/10**	**193.9**	4/10	208.8	3/10	75.5	**10/10**	**111.7**	3/10	256.3	**10/10**	**119.8**
	Basic	10/10	15.8	10/10	108.5	10/10	697.1	7/10	786.8	9/10	384.4	10/10	182.0	7/10	336.9	10/10	338.0
	P.W.	**10/10**	**10.8**	**10/10**	**65.7**	10/10	728.6	**7/10**	**767.8**	**10/10**	**648.1**	10/10	177.1	**8/10**	**272.9**	10/10	473.9
GNM	Breach	**10/10**	**5.4**	**10/10**	**151.4**	0/10	-	0/10	-	0/10	-	**10/10**	**171.4**	0/10	-	0/10	-
	Basic	10/10	12.4	10/10	162.3	**10/10**	**185.6**	7/10	261.9	7/10	163.7	10/10	227.1	2/10	378.5	**10/10**	**162.2**
	P.W.	10/10	60.8	9/10	110.7	8/10	211.2	**8/10**	**313.0**	**10/10**	**178.7**	10/10	252.0	**6/10**	**153.2**	6/10	197.4
SA	Breach	**10/10**	**160.1**	0/10	-	3/10	383.7	0/10	-	3/10	80.4	0/10	-	6/10	307.0	3/10	92.8
	Basic	10/10	264.8	9/10	236.1	**8/10**	**385.6**	**8/10**	**505.3**	7/10	341.2	5/10	391.3	**8/10**	**273.8**	**10/10**	**273.2**
	P.W.	10/10	208.7	**10/10**	**377.6**	8/10	666.0	7/10	795.4	**10/10**	**624.2**	**8/10**	**665.7**	6/10	293.7	10/10	390.9

the 4 steps to conduct MCTS:

- Selection: initially, we just select R; otherwise, we select a child according to UCB1.
- Expansion: create a new node N for one sub-space that has not been expanded.
- Playout: collect the set of sub-spaces on the path from R to N, and incorporate the whole spaces in the remaining levels until reaching a leaf; run hill-climbing optimization on the sequence of spaces with a short timeout, and obtain robustness Rob_N.
- Backpropagation: compute reward of N, and update the reward and number of visits from N to R.

The reward $\mathcal{R}_N$ of a node N in $\mathcal{T}$ is defined as $\left(1 - \frac{Rob_N}{\max_{w \in \mathcal{T}} Rob_w}\right)$, where w is any node of $\mathcal{T}$. Furthermore, UCB1 algorithm returns one child of a node N according to $\underset{w \in N.Children}{\arg\max} \left(\mathcal{R}_w + c\sqrt{\frac{2 \ln V(N)}{V(w)}}\right)$, where $V(w)$ is the number of visits to a child node w of N, and c is a scalar for balancing exploitation to the "best" child and exploration to other children.

The algorithm illustrated until now is a basic two-layered framework, as shown in Fig. 2. As on one level the partitions to explore could be too many, in a variation of the approach we use progressive widening [7] to skip some of them for not spending too much on exploration.

Experimental evaluation and Discussion

We implemented our algorithms, namely *Basic* for the basic two-layered framework and *P.W.* for the improvement with progressive widening, and evaluated them on 3 Simulink benchmark models: Automatic Transmission (AT) [8], Abstract Fuel Control (AFC) [9] and Free Floating Robot (FFR) [10]. We compare the performance with random sampling and the state-of-the-art tool Breach [3] with 3 back-end hill-climbing algorithms, CMA-ES, GNM and SA. The specifications under test are S1-S5 for AT, Sbasic and Sstable for AFC, and Strap for FFR. A detailed introduction to all the benchmarks, specifications and experimental settings is in [6]. Table I presents the results in terms of success rate (out of 10 trials), and average time for successful runs. Based on the criterion that success rate is more important, local best performers are in boldface and global best performers w.r.t. each specification are colored green.

In Table I, if we label each specification as "hard" or "easy" according to the performance of random sampling, we observe that for "hard" specifications like S4 and S5, our approach performs better in terms of falsification rate, i.e., it enhances the ability to find counterexamples compared to the existing tools; for "easy" properties like S1 and Sbasic, it performs comparably with Breach. For many trials, our algorithm is more time-consuming: this is expected, as we spend more time on exploration. In this way, we improve the falsification rate and so the time overhead is still acceptable.

References

[1] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[2] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of FORMATS'10*, pages 92–106, Berlin, Heidelberg, 2010. Springer-Verlag.

[3] A. Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In CAV 2010.

[4] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Proceedings of TACAS'11/ETAPS'11*, pages 254–257, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of ECML'06*, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[6] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2894–2905, Nov 2018.

[7] R. Coulom. Computing "Elo ratings" of move patterns in the game of Go. *ICGA Journal*, 30(4):198–208, 2007.

[8] B. Hoxha, H. Abbas, and G. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *ARCH14-15*, vol. 34 of *EPiC Series in Computing*, pages 25–30. EasyChair, 2015.

[9] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain control verification benchmark. In *Proc. of HSCC '14*, pages 253–262, NY, USA, 2014. ACM.

[10] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu. Testing cyber-physical systems through bayesian optimization. *ACM Trans. Embed. Comput. Syst.*, 16(5s):170:1–170:18, 2017.