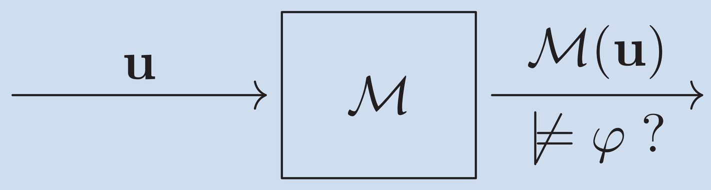
Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search

Zhenya Zhang[1,2], Gidon Ernst[1], Sean Sedwards[3], Paolo Arcaini[1], Ichiro Hasuo[1,2]
[1] *National Institute of Informatics, Tokyo, Japan*
[2] *The Graduate University for Advanced Studies, Hayama, Japan*
[3] *University of Waterloo, Waterloo, Canada*

Problem

- Falsification problem is defined as follows:
  - **Given:** a *model* $\mathcal{M}$ (that takes an input signal $\mathbf{u}$ and yields an output signal $\mathcal{M}(\mathbf{u})$), and a *specification* φ (a temporal formula)
  - **Answer:** *error input*, that is, an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ
- Challenges:
  - Black/Grey box model, e.g., model in Simulink, etc.
  - Continuous (infinite) input space



Preliminary

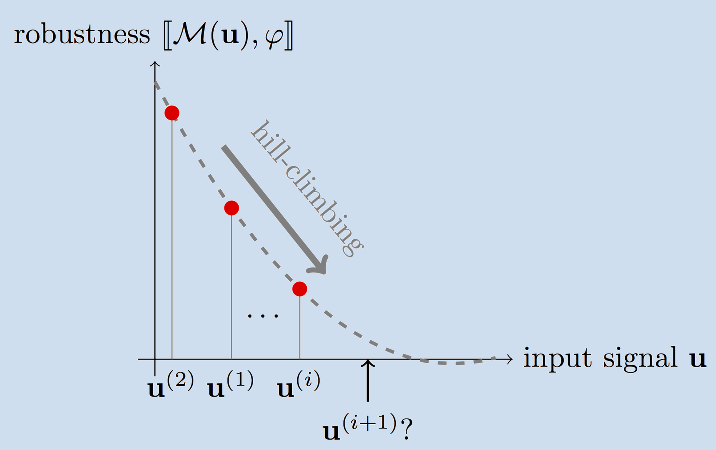
Robust semantics of temporal formulas

- Boolean satisfaction: $\mathbf{v} \models \varphi$ or $\mathbf{v} \not\models \varphi$
- Quantitative robustness: $[\![\mathbf{v}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$

Optimization-based technique:

- Goal: $\min [\![\mathbf{v}, \varphi]\!]$
- "Hill-Climbing" algorithms: CMA-ES, Nelder-Mead, Simulated Annealing, etc.

Optimization-based method

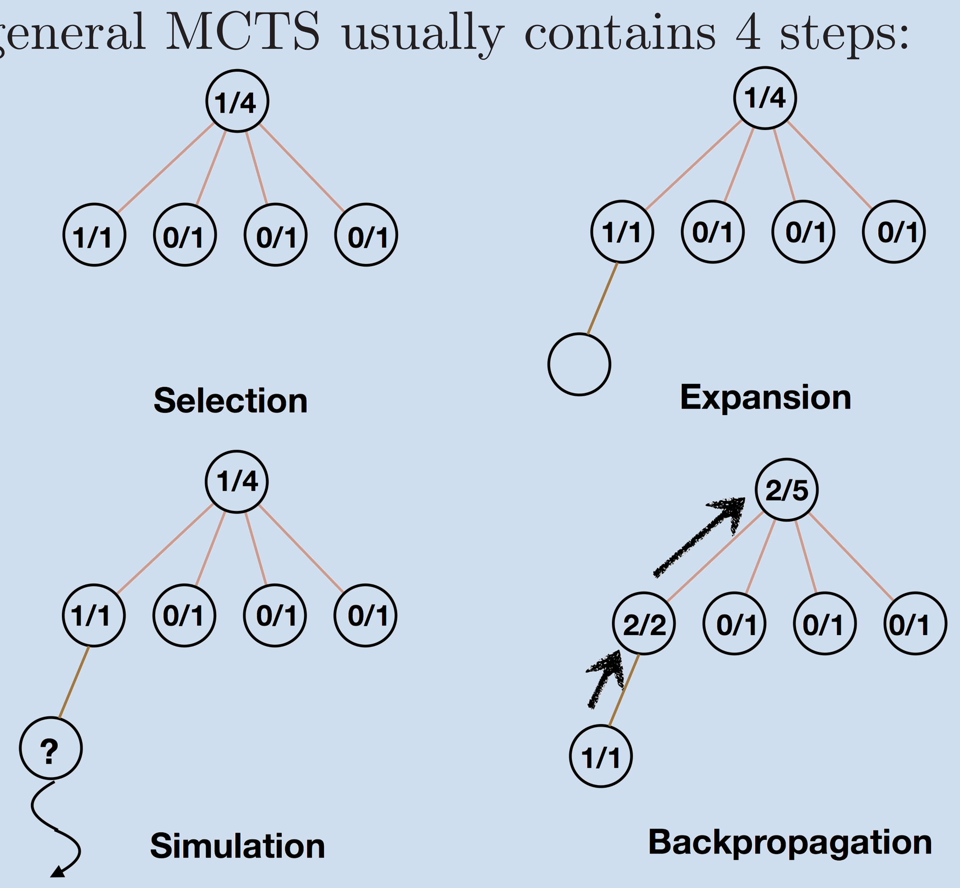


"Hill-Climbing" algorithm:

- In the i-th sampling one tries an input signal $\mathbf{u}^{(i)}$. The corresponding output signal $\mathbf{v}^{(i)} = \mathcal{M}(\mathbf{u}^{(i)})$ is shown below it.
- Optimization algorithm decides a new input signal $\mathbf{u}^{(i+1)} = (u_1^{(i+1)}, \dots, u_K^{(i+1)})$. and $\mathbf{u}^{(i+1)}$ makes the robustness smaller.

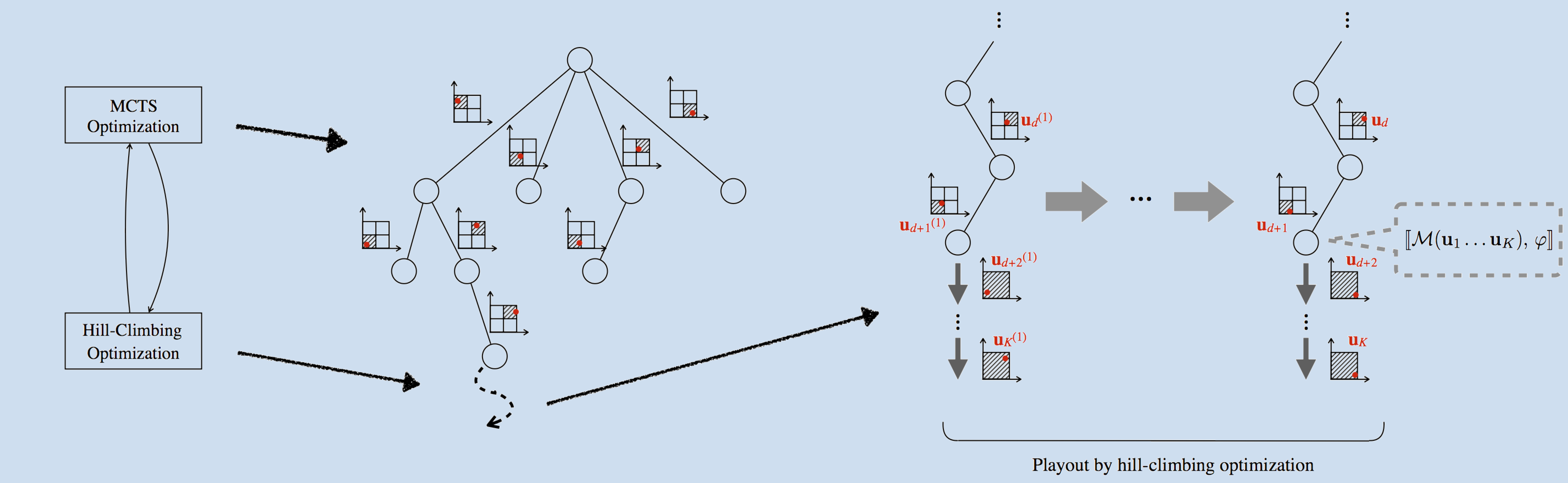
Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm that focuses more on those promising branches rather than the whole tree. A general MCTS usually contains 4 steps:



1. Selection: select one "promising" child according to UCB1 algorithm.
2. Expansion: create a new child.
3. Simulation: randomly select a sequence of children until the game is decided, and record the result.
4. Backpropagation: update the winning and visiting times of the nodes in the path.

Algorithm overview



Basic algorithm:

- Time-staging. Divide the time bound into K intervals to find a sequence $\mathbf{u}_1, \dots, \mathbf{u}_K$.
- Node expansion. Use a *partitioning* of the input space $I_1 \times \cdots \times I_M$ as the children set A.
- Child selection. Define that $reward = 1 - \frac{R(wa)}{\max_{w' \in \mathcal{T}} R(w')}$, and apply UCB1 algorithm $\arg\max_{a \in A} \left(reward + c\sqrt{\frac{2 \ln N(w)}{N(wa)}} \right)$ to select the best child.
- Simulation. Apply optimization solvers in the selected sub-region sequence with time budget.
- Backpropagation. Update the reward of parent if the newly computed reward is better.
- Postprocessing. Apply optimization algorithm again if no solution is returned.

Variation:

- Progressive widening. Unlike the basic algorithm, the children of higher layer are possible to be expanded earlier than the children on the second layer.

Experimental evaluation

		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

S1 $\square_{[0,30]}$ $(speed < 120)$
S2 $\square_{[0,30]}$ $(gear = 3 \rightarrow speed \geq 20)$
S3 $\lozenge_{[10,30]}$ $(speed \leq 53 \vee speed \geq 57)$
S4 $\square_{[0,29]}$ $(speed < 100) \vee \square_{[29,30]}(speed > 65)$
S5 $\square_{[0,30]}(rpm < 4770 \vee \square_{[0,1]}(rpm > 600))$
Sbasic $\square_{[11,30]}(\neg(|AF - AF_{\mathrm{ref}}| > 0.05 * 14.7))$
Sstable $\neg(\lozenge_{[6,26]}\square_{[0,4]}(AF - AF_{\mathrm{ref}} > 0.01 * 14.7))$
Strap $\neg\lozenge_{[0,5]}(x, y \in [3.9, 4.1] \wedge \dot{x}, \dot{y} \in [-1, 1])$

Model:

- AT: Automatic Transmission
- AFC: Abstract Fuel Control
- FFR: Free Floating Robot

Algorithm:

- Breach: optimization
- Basic: MCTS + Hill-Climbing
- P.W.: MCTS + Hill-Climbing + progressive widening

Conclusion & Future work

In this work we have presented a two-layered optimization framework for hybrid system falsification. It combines Monte Carlo tree search—a widely used stochastic search method that effectively balances exploration and exploitation—and hill-climbing optimization—a local search method whose use in hybrid system falsification is established in the community. Our experiments demonstrate its promising performance.

Two directions for future work:

- Compute a quantitative coverage metric from the result of our MCTS algorithm.
- Explore variations of robust semantics to mitigate discrete propositions.

Acknowledgements

This work is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), Japan Science and Technology Agency.