The Journal of Open Source Software

POMDPStressTesting.jl: Adaptive Stress Testing for Black-Box Systems

Robert J. Moss[1]

1 Stanford University

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

POMDPStressTesting.jl is a package that uses reinforcement learning and stochastic optimization to find likely failures in black-box systems through a technique called adaptive stress testing (Lee, Mengshoel, & Kochenderfer, 2019). Adaptive stress testing (AST) has been used to find failures in safety-critical systems such as aircraft collision avoidance systems (Lee, Kochenderfer, Mengshoel, Brat, & Owen, 2015), flight management systems (Moss et al., 2020), and autonomous vehicles (Koren, Alsaif, Lee, & Kochenderfer, 2018). The POMDPStressTesting.jl package is written in Julia (Bezanson, Edelman, Karpinski, & Shah, 2017) and is part of the wider POMDPs.jl ecosystem (Egorov et al., 2017), which provides access to simulation tools, policies, visualizations, and—most importantly—solvers. We provide different solver variants including online planning algorithms such as Monte Carlo tree search (Coulom, 2006) and deep reinforcement learning algorithms such as trust region policy optimization (TRPO) (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015) and proximal policy optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). Stochastic optimization solvers such as the cross-entropy method (Rubinstein, 1999) are also available and random search is provided as a baseline. Additional solvers can easily be added by adhering to the POMDPs.jl interface.

The AST formulation treats the falsification problem (i.e. finding failures) as a Markov decision process with a reward function that uses a measure of distance to a failure event to guide the search towards failure. The reward function also uses the state transition probabilities to guide towards *likely* failures. Reinforcement learning aims to maximize the discounted sum of expected rewards, therefore maximizing the sum of log-likelihoods is equivalent to maximizing the likelihood of a trajectory. A gray-box simulation environment steps the simulation and outputs the state transition probabilities, and the black-box system under test is evaluated in the simulator and outputs an event indication and the real-valued distance metric (i.e. how close we are to failure). To apply AST to a general black-box system, a user has to implement the following Julia interface:

```julia
# GrayBox simulator and environment
abstract type GrayBox.Simulation end
function GrayBox.environment(sim::Simulation)::GrayBox.Environment end
function GrayBox.transition!(sim::Simulation)::Real end

# BlackBox.interface(input::InputType)::OutputType
function BlackBox.initialize!(sim::Simulation)::Nothing end
function BlackBox.evaluate!(sim::Simulation)::Tuple{Real, Real, Bool} end
function BlackBox.distance(sim::Simulation)::Real end
function BlackBox.isevent(sim::Simulation)::Bool end
function BlackBox.isterminal(sim::Simulation)::Bool end
```

The simulator stores simulation-specific parameters and the environment stores a collection of probability distributions that define the state transitions (e.g., Gaussian noise models, uniform control inputs, etc.). There are two types of AST action modes: random seed actions or directly sampled actions. The seed-action approach is useful when the user does not have direct access to the environmental distributions or when the environment is complex. When using directly sampled actions, the Transition and Evaluate functions can take in an environment sample selected by the solvers and apply it directly as input to the black-box system, allowing for finer control over the search. The interface is designed for straightforward extensions to other autonomous system applications. Explicitly separating the simulation environment from the system under test allows for wider validation of complex black-box systems.

Our package builds off work originally done in the AdaptiveStressTesting.jl package (Lee et al., 2019), but POMDPStressTesting.jl adheres to the interface defined by POMDPs.jl and provides different action modes and solver types. Related falsification tools (i.e. tools that do not include most-likely failure analysis) are S-TaLiRo (Annapureddy, Liu, Fainekos, & Sankaranarayanan, 2011), Breach (Donzé, 2010), and FalStar (Zhang, Ernst, Sedwards, Arcaini, & Hasuo, 2018). These packages use a combination of optimization, path planning, and reinforcement learning techniques to solve the falsification problem. The tool most closely related to POMDPStressTesting.jl is the AST Toolbox in Python (Koren et al., 2018), which wraps around the gym reinforcement learning environment (Brockman et al., 2016). The author has contributed to the AST Toolbox and found the need to create a similar package in pure Julia for better performance and to interface with the POMDPs.jl ecosystem.

Validating autonomous systems is a crucial requirement before their deployment into real-world environments. Using automated tools to search for likely failures allow engineers to address and resolve problems during development. Because many autonomous systems are in environments with rare failure events, it is especially important to incorporate likelihood of failure within the search to help inform the potential problem mitigation.

Research and Industrial Usage

POMDPStressTesting.jl has been used to find likely failures in aircraft trajectory prediction systems (Moss et al., 2020), which are flight-critical subsystems used to aid in-flight automation. A developmental commercial flight management system was stress tested so the system engineers could mitigate potential issues before system deployment (Moss et al., 2020). In addition to traditional requirements-based testing for avionics certification (RTCA, 2011), this work is being used to find potential problems during development. There is also ongoing research on the use of POMDPStressTesting.jl for assessing the risk of autonomous vehicles and determining failure scenarios of autonomous lunar rovers.

Acknowledgments

We acknowledge Ritchie Lee for his guidance and original work on adaptive stress testing and the AdaptiveStressTesting.jl package and Mark Koren, Xiaobai Ma, and Anthony Corso for their work on the AST Toolbox Python package and the CrossEntropyMethod.jl package. We also acknowledge Shreyas Kowshik for his initial implementation of the TRPO and PPO algorithms in Julia. We want to thank the Stanford Intelligent Systems Laboratory for their development of the POMDPs.jl ecosystem and the MCTS.jl package; particular thanks to Zachary Sunberg. We also want to thank Mykel J. Kochenderfer for his support and research input and for advancing the Julia community.

References

Annapureddy, Y., Liu, C., Fainekos, G., & Sankaranarayanan, S. (2011). S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (pp. 254–257). Springer. doi:10.1007/978-3-642-19835-9_21

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. doi:10.1137/141000671

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. *arXiv:1606.01540*.

Coulom, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *International Conference on Computers and Games* (pp. 72–83). Springer. doi:10.1007/978-3-540-75538-8_7

Donzé, A. (2010). Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification* (pp. 167–170). Springer. doi:10.1007/978-3-642-14295-6_17

Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K., & Kochenderfer, M. J. (2017). POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty. *Journal of Machine Learning Research*, *18*(26), 1–5.

Koren, M., Alsaif, S., Lee, R., & Kochenderfer, M. J. (2018). Adaptive Stress Testing for Autonomous Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)* (pp. 1–7). doi:10.1109/IVS.2018.8500400

Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., & Owen, M. P. (2015). Adaptive stress testing of airborne collision avoidance systems. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*. doi:10.1109/DASC.2015.7311450

Lee, R., Mengshoel, O. J., & Kochenderfer, M. J. (2019). Adaptive Stress Testing of Safety-Critical Systems. In *Safe, Autonomous and Intelligent Vehicles* (pp. 77–95). Springer. doi:10.1007/978-3-319-97301-2_5

Moss, R. J., Lee, R., Visser, N., Hochwarth, J., Lopez, J. G., & Kochenderfer, M. J. (2020). Adaptive Stress Testing of Trajectory Predictions in Flight Management Systems. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*.

RTCA. (2011, December). Software Considerations in Airborne Systems and Equipment Certification. DO-178C.

Rubinstein, R. (1999). The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, *1*(2), 127–190. doi:10.1023/A:1010091220143

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust Region Policy Optimization. In *International Conference on Machine Learning* (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv:1707.06347*.

Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., & Hasuo, I. (2018). 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. doi:10.1109/TCAD.2018.2858463