MDPVIS: An Interactive Visualization for Testing Markov Decision Processes

Sean McGregor, Hailey Buckingham, Rachel Houtman,
Claire Montgomery, Ronald Metoyer, and Thomas G. Dietterich
Oregon State University
1148 Kelley Engineering Center
Corvallis, OR 97331

Abstract

Markov Decision Process (MDP) simulators and optimization algorithms integrate several systems and functions that are collectively subject to failures of specification, implementation, integration, and optimization. We present a domain agnostic visual analytic design and implementation for testing and debugging MDPs: MDPVIS.

A common approach for solving Markov Decision Processes is to implement a simulator of the stochastic dynamics of the MDP and a Monte Carlo optimization algorithm that invokes this simulator. The resulting software system is often realized by integrating several systems and functions that are collectively subject to failures (referred to as "bugs") of specification, implementation, integration, and optimization.

Since bugs are subject to the same stochastic processes as their underlying systems, detecting and characterizing bugs requires exploration with an "informed trial and error" (Sedlmair et al. 2014) process. This process involves writing an interactive client to manually execute transitions, followed by a visualization of state development as a policy rule is followed.

A domain agnostic visual analytic interface could facilitate testing and debugging, but during semi-structured interviews of MDP researchers we did not find anyone using a generic visualization tool for testing. We posit this is because researchers have heretofore not had access to a visualization that is easily connected to their MDP simulator and MDP optimizer.

In the following section we summarize the implementation and integration of MDPVIS presented in McGregor et al. (2015).

Implementation and Integration

MDPVIS' target users are researchers interested in steering the optimization itself, simulator developers who are interested in ensuring the policies optimized for the problem domain are well founded, or domain policy experts primarily interested in the outcomes produced by the optimized policy. In real-world settings these roles can be filled by a single person, or each role can be performed by a large team of developers and domain experts.

MDPVIS extends computational steering from the high performance scientific visualization community (Parker et al. 1996). Whereas computational steering traditionally refers to modifying a computer process during its execution (Mulder, van Wijk, and van Liere 1999), we treat optimization as an open-ended process whose parameters are repeatedly changed for testing and debugging.

Sedlmair et al. (2014) label techniques for understanding the relationship between input parameters and outputs as Parameter Space Analysis (PSA), "...the systematic variation of model input parameters, generating outputs for each combination of parameters, and investigating the relation between parameter settings and corresponding outputs." This is a suitable definition for the MDP debugging and testing processes. Testing for MDP bugs requires exploring Monte Carlo rollouts. These rollouts are the output of the system under test, but since the distribution of these rollouts is defined by applying a policy in many successive states, the rollouts are tightly coupled with the parameter space of the MDP's component functions. Similarly, establishing bug causality (debugging) requires varying the model parameters and examining the resulting rollouts.

The VL/HCC paper explores MDP testing questions in the following six broad tasks introduced by Sedlmair et al. (2014):

1. **Fitting:** Do the outputs match real-world data or expectations?
2. **Outliers:** Are low probability events occurring with unexpected frequency?
3. **Partition:** Do different system parameters produce the expected differences?
4. **Optimization:** Did the optimization algorithm find the local optimum and does the policy exploit a bug in the specification or implementation?
5. **Uncertainty:** How confident are we in the proposed results?
6. **Sensitivity:** Do small changes to the system result in big changes to the optimized policy?

To test these questions, MDPVIS (Figure 1) has four computational steering control sets and three visualization areas.

The controls give the reward, model, and policy parameters that are exposed by the MDP's software. These layers are memoized in an exploration history that records the parameters and rollouts computed by the MDP.

The first visualization area shows state distributions at time steps under the current policy. The second visualization area gives the distribution of a variable's development through time. The last visualization area gives details of individual states.

Each of these steering controls and visualizations are designed to integrate with MDP simulators and optimizers using the same read-eval-print loop (REPL) that is typically implemented in current development practices.

We built MDPVIS as a data-driven web application. The web stack emphasizes standard data interchange formats that are easily linked to MDP simulators and optimization algorithms. We identified four HTTP requests (initialize, rollouts, optimize, and state) that are answered by the MDP simulator or optimizer. In each case the current values of the steering controls are sent to a web server acting as a bridge between the HTTP request and the syntax expected by the REPL.

1. ***/initialize*** – Ask for the steering parameters that should be displayed to the user. The parameters are a list of tuples, each containing the name, description, minimum value, maximum value, and current value of a parameter. These parameters are then rendered as HTML input elements for the user to modify. Following initialization, MDPVIS automatically requests rollouts for the initial parameters.
2. ***/rollouts?QUERY*** – Get rollouts for the parameters that are currently defined in the user interface. The server returns an array of arrays containing the state variables that should be rendered for each time step.
3. ***/optimize?QUERY*** – Get a newly optimized policy. This sends all the parameters defined in the user interface and the optimization algorithm returns a newly optimized policy.
4. ***/state?IDENTIFIER*** – Get the full state details and images. This is required for high dimensional problems in which the entire state cannot be returned to the client for every state in a rollout

The most domain-specific element of any MDP visualization is the representation of a specific state. In Figure 1 individual states are given as two dimensional images of landscape fuel levels. This is a visualization that our forestry colleagues typically generate for natural resource domains. The fourth HTTP request can optionally return images to accommodate these landscapes and arbitrary domain images. These landscapes can be rendered without any changes to the MDPVIS code base.

A live version of the visualization is available at *MDPvis.github.io* for a wildfire suppression policy domain (Houtman et al. 2013). The visualization has been tested on Google Chrome and Firefox and is responsive to a variety of screen resolutions.

In the VL/HCC paper we presented a use-case study to provide anecdotal evidence of the utility of MDPVIS on the wildfire problem. The case study involved user sessions with our forestry economics collaborators who have formulated an MDP optimization problem to study fire suppression policies. When applying MDPVIS we found numerous simulator and the optimization bugs.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1331932.

References

Houtman, R. M.; Montgomery, C. A.; Gagnon, A. R.; Calkin, D. E.; Dietterich, T. G.; McGregor, S.; and Crowley, M. 2013. Allowing a Wildfire to Burn: Estimating the Effect on Future Fire Suppression Costs. *International Journal of Wildland Fire* 22(7):871–882.

McGregor, S.; Buckingham, H.; Dietterich, T. G.; Houtman, R.; Montgomery, C.; and Metoyer, R. 2015. Facilitating Testing and Debugging of Markov Decision Processes with Interactive Visualization. In *IEEE Symposium on Visual Languages and Human-Centric Computing*.

Mulder, J. D.; van Wijk, J. J.; and van Liere, R. 1999. A survey of computational steering environments. *Future Generation Computer Systems* 15(1):119–129.

Parker, S. G.; Beazley, D.; Johnson, C. R.; City, S. L.; and Mechanisms, I.-b. C. 1996. Computational Steering Software Systems and Strategies.

Sedlmair, M.; Heinzl, C.; Bruckner, S.; Piringer, H.; and Möller, T. 2014. Visual parameter space analysis: A conceptual framework. *To Appear in IEEE TVCG (Proc. InfoVis)* 20(12).

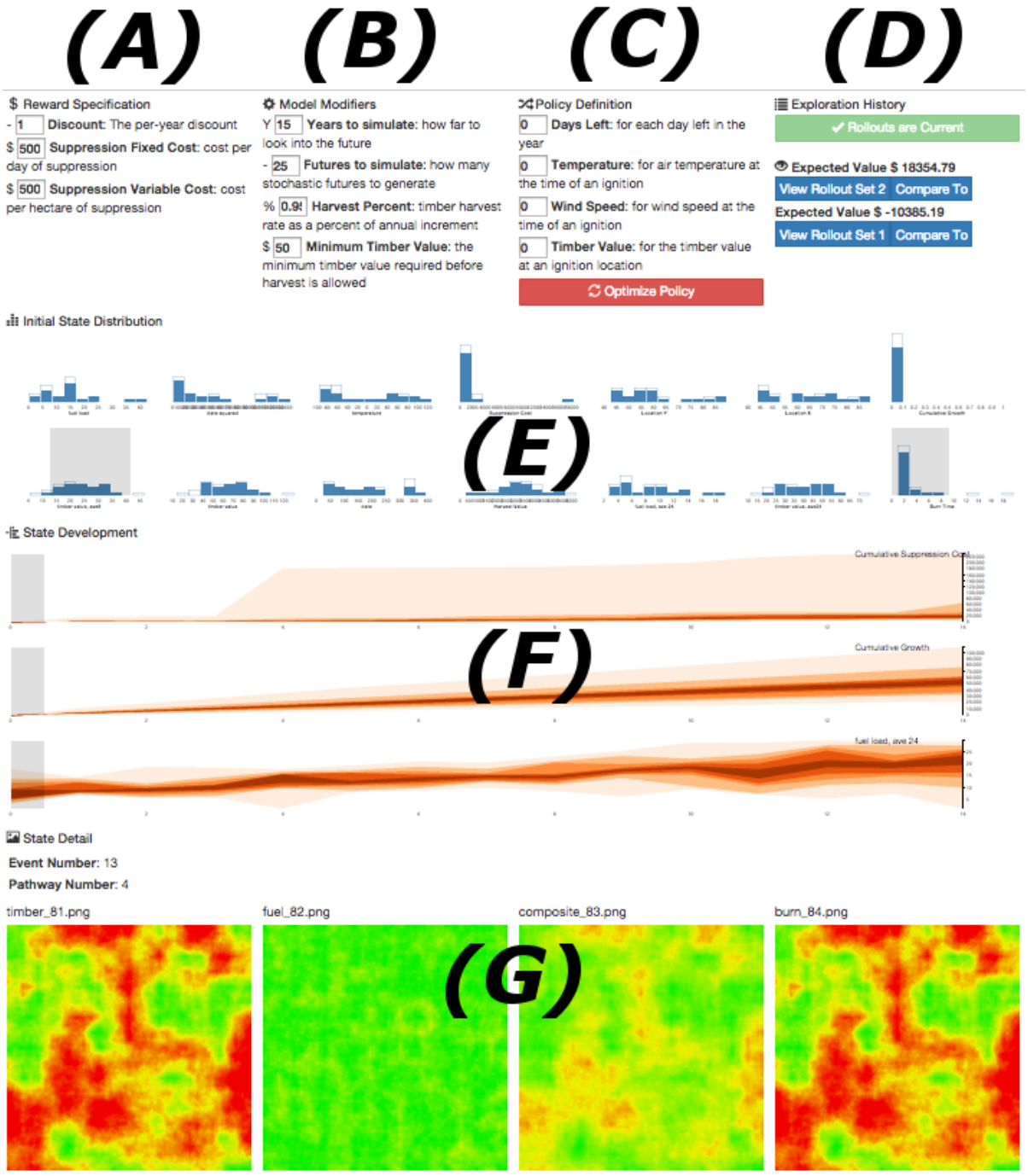

Figure 1: A high level overview of the Markov Decision Process visualization prototype: MDPVIS. The top row has the three computational steering controls for (A) the reward specification, (B) the model modifiers, and (C) the policy definition. A fourth panel gives the history of Monte Carlo rollout sets generated under the parameters of panels (A) through (C). Changes to the parameters enable the optimization button found under the policy definition and the Monte Carlo rollouts button found under the Exploration History section. The visualization has two buttons in the History panel for each set of Monte Carlo rollouts, one for visualizing the associated Monte Carlo rollouts and another for comparing two sets of rollouts. Below the control panels are visualization areas for (E) histograms of the initial state distribution, (F) fan charts for the distribution of variables over time, and (G) a series of individual states rendered by the simulator as images. For a readable version of the visualization we invite users to load the visualization in their browser by visiting *MDPvis.github.io*.