PMC-VIS: An Interactive Visualization Tool for Probabilistic Model Checking

Max Korn[1(✉)] , Julián Méndez[2(✉)] , Sascha Klüppelholz[1(✉)] ,
Ricardo Langner[2(✉)] , Christel Baier[1(✉)] , and Raimund Dachselt[2(✉)]

[1] Institute of Theoretical Computer Science, TU Dresden, Dresden, Germany
{max.korn,sascha.klueppelholz,christel.baier}@tu-dresden.de
[2] Interactive Media Lab Dresden, TU Dresden, Dresden, Germany
{julian.mendez2,ricardo.langner,raimund.dachselt}@tu-dresden.de

Abstract. State-of-the-art Probabilistic Model Checking (PMC) offers multiple engines for the quantitative analysis of Markov Decision Processes (MDPs), including rewards modeling cost or utility values. Despite the huge amount of internally computed information, support for debugging and facilities that enhance the understandability of PMC models and results are very limited. As a first step to improve on that, we present the basic principles of PMC-VIS, a tool that supports the exploration of large MDPs together with the computed PMC results per MDP-state through interactive visualization. By combining visualization techniques, such as node-link diagrams and parallel coordinates, with quantitative analysis capabilities, PMC-VIS supports users in gaining insights into the probabilistic behavior of MDPs and PMC results and enables different ways to explore the behaviour of schedulers of multiple target properties. The usefulness of PMC-VIS is demonstrated through three different application scenarios.

1 Introduction

Probabilistic model checking (PMC) is a well-established technique used in the field of formal verification to analyze and assess the behavior of probabilistic systems. Sources of probabilistic behavior include randomized algorithms as well as stochastic assumptions about the external use of the system (i.e., the system environment) and error probabilities. PMC combines concepts from probability theory and model checking to provide quantitative insights into the reliability and performance of such systems on various types of stochastic models. It is applicable at all stages of the life cycle of a system (i.e., in the design phase, at runtime, and at inspection time) for the evaluation of system properties, such as reliability, safety, and various cost and performance metrics.

State-of-the-art probabilistic model checkers, such as PRISM [23], Storm [15], and MRMC [20], have successfully been applied in various application fields,

---

M. Korn and J. Méndez—Authors contributed equally.

including computer science, engineering, robotics, and biology [2,7]. Nevertheless, understanding, debugging, and using computed results for a given general goal often involves a laborious manual process that is only supported by hand-written and tailored software scripts in combination with general-purpose tools for handling large datasets. Existing model checkers provide rather limited support for *(re)configuration* tasks (i.e., calibrating systems before or at runtime to complete set objectives under possibly multiple criteria) that involve systematically exploring and understanding (**1**) complex system behavior and (**2**) metrics returned by PMC processes. While there exist model checkers for PMC that provide graphical user-interfaces to inspect the model and its results, like PRISM [23], they do not incorporate advanced visualization techniques.

From the application viewpoint, the missing visual tool support for using computed PMC results has already been recognized in some domains, such as DNA sequencing [7] and automated driving [13]. The tools proposed in these fields provide support for solving concrete problems in their respective domains, yet they do not transfer to more domain-independent general goals such as (re)configuration. To the best of our knowledge, no PMC tool fully harnesses interactive visualization to support the exploration and facilitate the understanding of large models and their (functional or non-functional) properties.

In response, we set to create PMC-VIS, a visual tool that explicitly supports understanding the above-mentioned system behavior (**1**) and metrics (**2**) while remaining independent from a concrete application field. PMC-VIS connects the PRISM model checker on the backend side with a visualization frontend. Our focus is on *Markov Decision Processes* (MDPs) as operational models, in which the initial states stand for design alternatives (*configurations*) and the nondeterministic choices stand for possible *reconfiguration* steps. Metrics in this setting are of a quantitative nature and include probabilities and expectations of random variables, standing for either costs or gained rewards. The backend of our tool PMC-VIS consists of a simple API shell around PRISM that allows for calls to the model checker at runtime and a database wrapper for efficient data-exchange to the frontend. The frontend consists of a web-based application that enables *exploration of large MDPs* including features for comparing metrics computed by PMC attached to MDP states, actions and schedulers, while additionally supporting *finding suitable configurations* in families of MDPs and *reconfiguration in adaptive systems* modeled as MDPs. The current version of PMC-VIS, usage scenarios, and performance experiments are available at imld.de/pmc-vis.

2 Background and Related Work

Markov Decision Processes (MDP) are formal, stochastic models used to describe systems that exhibit both controllable and uncontrollable behaviour [3]. MDPs are typically represented as directed graphs with *states* as vertices and *actions* (also referred to as *transitions*) as edges. Edges can additionally express uncertainty, or in other words, the same action may lead to different states according to a probability distribution. We will use the term *PMC results* to refer to the

output of the PMC, including probabilities of temporal events and expectations of random variables. Apart from the sole purpose of evaluation by means of a quantitative analysis, the computed PMC results can be used to decide on different design alternatives, determine appropriate values for system parameters (i.e., parameter synthesis for configuration) and to synthesize suitable adaptation policies (i.e., strategy synthesis for reconfiguration in adaptive systems) by examining *schedulers*, the functions that resolved the non-determinism during computation.

Visual Tools for Model Checking. HMMEditor [7] and TraceVis [13] exemplify the usage of PMC results for specific application scenarios which do not transfer to domain-independent (re)configuration tasks. On the other hand, tools like Palette [19] and Theseus [11] visualize model checking outcomes, but not the model or its inherent behaviour. Another example of this is MDPVis [25], which provides insights for debugging through a succinct overview of MDPs of arbitrary size, yet it does not visualize individual decisions nor the model itself, making it quite abstract for understanding the model behavior. Some model checking toolboxes in adjacent fields, such as mCRL2 [5], CADP [10] and UPPAAL [4] already contain capabilities to create visual support for understanding their models by visualizing them in graph form. In the field of MDPs, this also has been partially addressed for learning scenarios [27] as well as for the understanding of model checking counterexamples [17], using multiple coordinated views. But all these tools were evaluated on relatively small graphs, making scaling to large MDPs remain an open challenge.

Degree-Of-Interest for Large Decision Graphs. Large graphs are typically handled using aggregation and clustering methods [12], (e.g., ZAME [8] and ASK-GraphView [1] and HybridVis [24]), or focus+context techniques [6,30]. However, noting that a complete graph overview may not be necessary for decision graphs (as the interest is typically on a few decisions at a time), we make use of Degree-Of-Interest (DOI) graphs [14] instead. This approach for large graph exploration shows only an initial set of nodes and delegates to the user the responsibility of revealing, on-demand, the neighbors of nodes of interest. In this way, the graph is revealed progressively. This is fitting for MDPs as we can expand by discreet time steps of the model. However, while visual clutter and subsequently, cognitive load, start low, they increase as the graph is expanded in this approach. To overcome this limitation, we take further inspiration from concepts for iterative graph exploration on sequential views [16,29].

Parallel Coordinates Plots for Multivariate Graphs and Decision-Making. The surveys on multivariate network visualization [21,26] present various approaches fitting to our challenge of joining per-state PMC results to MDPs. Most prominently, multiple coordinated view setups featuring *Parallel Coordinates Plots* (PCPs) [18] are used to effectively display and explore the attributes alongside or within the graph view. A PCP consists of a set of n parallel lines axes for each of the dimensions or attributes of n-dimensional data. The data points are then represented as polylines connecting the axes at the values of each data
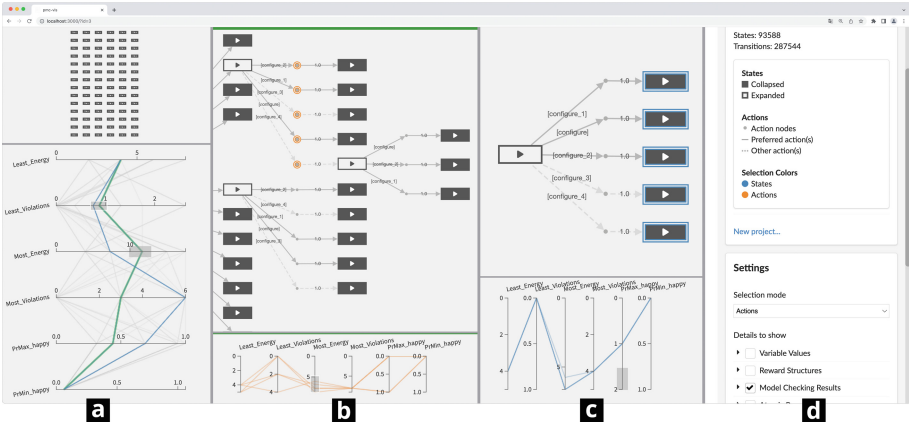
Fig. 1. Screenshot of PMC-VIS, illustrating three *panes* (a, b, c), each showing a different part of the MDP. The settings sidebar is visible on (d).

point's attributes. By organizing the data in such a way, it is possible to detect patterns and correlations between attribute values. PCPs typically incorporate interaction techniques such as axes re-ordering, brushing and highlighting. By brushing on an axis x, the user selects the set of data points where the value for x is within the range determined by the brushed area. Multiple brushes can be specified simultaneously on different axes, which is why PCPs are employable for multi-criteria decision-making (e.g., Parasol [28]). HybridVis [24] also exemplifies the usage of a PCP alongside a graph view. Using the before-mentioned abstraction approaches, this graph view scales, although visually cluttered, to a few thousand nodes. The PCP encodes the multi-variate nodes of the graph, which in turn allows coordinated filtering and highlighting in both views.

3 PMC-VIS: Visualising Probabilistic Model Checking

As an interdisciplinary team, bridging the visualization and formal verification communities, we designed and developed PMC-VIS, a tool that integrates visualization techniques and efficient retrieval methods. A screenshot of PMC-VIS can be seen in Fig. 1. PMC-VIS consists of two web servers, *Backend* and *Frontend*. Our architecture decouples the heavy probabilistic model checking computations done in the *Backend* from the visualization and interaction service provided by the *Frontend*. The Frontend can request data for states or subgraphs of interest from Backend, and uses this data to populate interactive visualizations.

Backend: The Backend server manages instances of the model checker PRISM [23], using its publicly available API[1]. As opposed to retrieving the data from resulting log files, we extract and store it in a database while it is generated during the stages of the PMC process. *Before model checking*, we extract structural information from the model input: existence of variables and their domains
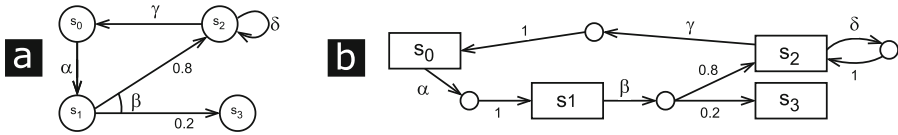
---

[1] github.com/prismmodelchecker/prism-api.

Fig. 2. Representation of MDPs. Version (a) is a more typical, compact version, with edges for actions and vertices for states. We propose version (b) for PMC-VIS, with a more stretched but explicit representation of actions versus outcomes.

(i.e., possible values they can take), labels for the states, parameters for experiments, and reward functions. *After model construction*, we create two tables, for (1) reachable states in the model, together with information about variables, labels and rewards, and (2) reachable actions, along with their labels and possible outcomes. *After model checking a property*, the computed PMC results are extracted for every reachable state of the MDP and stored in the above-mentioned tables, along with the scheduler actions taken or possibly taken.

Frontend: The Frontend of PMC-VIS is a web-based application that visualizes MDPs and their conjunct PMC results through sequential *panes*. Each pane has two sub-panes, for a *graph view* and an *attribute view*. Every graph view uses the Degree-Of-Interest (DOI) [14] approach to reveal the MDP on demand, and every attribute view uses configurable Parallel Coordinates Plots (PCPs) [18] to navigate through the PMC results related to selections within the graph view. Both panes and sub-panes are re-sizable, and the content within them adapts to the available space. For example, Fig. 1a shows a vertical PCP to better use the height of the sub-pane, whereas the PCPs in Fig. 1b and c are horizontal. PMC-VIS uses both traditional context menus and direct interactions (e.g., shift+click, double click) to support both novices and experienced users.

DOI MDPs: We designed a representation of MDPs, visible in Fig. 2b. For the *states*, we use rectangular nodes. For the edges, instead of overloading their meaning to indicate both *actions* and probabilistic *outcomes* (as shown in edge β of Fig. 2a), we explicitly separate these meanings by introducing handle nodes for the actions as label-less circles. This allows the user to more easily scan and parse the edges, since an outgoing edge from a state will always carry the action name (e.g., α, β, γ), and any outgoing edge from a handle node shows the probability of reaching the state that it points to. For example, when only glancing over Fig. 2a one could mistakenly get the impression that there are two outgoing actions from s_1 (as opposed to just β). This effect worsens when, (e.g., for each state) there are several outgoing actions, each with several probable outcomes. In reconfiguration tasks, where the user needs to decide between several actions, the handle nodes also simplify the selection of actions by area (e.g., using rectangle or lasso selections over the circle nodes instead of edges). The labels of the states are provided by the Backend as either text or icons. The scheduler choices are represented as solid edges, with the sub-optimal choices shown as dashed edges. Our proposed representation suits the implementation of an incremental DOI

approach well, since the data transfers are always small despite the introduced components. Expanding states reveals their immediate next actions and states. This expanding can be done within the same pane, or onto one, or several, adjacent panes.

PMC Results via PCPs: Alongside each DOI graph, a PCP is shown in each pane to explore the additional PMC results. The PCPs in our PMC-VIS tool support axes re-ordering and brushing for selections, hovering to preview selections, and other miscellaneous options for numerical, boolean and nominal data. The data that is shown on the PCP of a pane is linked to the selection of nodes on its respective graph view, meaning that states (blue) and actions (orange) can be loaded onto the PCP for filtering based on the shown PMC results, which in turn can refine the selection made on the graph view through e.g., axes brushing. The PCPs can be seen at the bottom of the panes visible in Fig. 1a–c. Additionally, details for each state on the DOI graph can be shown on-demand via tooltips.

Settings: PMC-VIS has a sidebar on the right with several options to modify the shown attributes of the PMC results, as well as several graph layout options (e.g., force-directed, hierarchical) that can be applied to each *pane* individually. Understanding the part of the MDP shown in each pane may be easier using different graph layout options for particular cases, and this flexibility in configuration supports varying user preferences within each individual pane. The settings sidebar is visible in Fig. 1d.

Multi-pane Possibilities: Our multi-pane approach allows the users to expand as much as desired and to create structural "check-points" by expanding a selection of states from the MDP onto new panes. Doing so creates multiple work spaces within the same MDP, enhancing the scalability of the DOI approach by turning the model exploration into a task that can be distributed and completed asynchronously. Furthermore, a pane can be cloned onto a neighbor pane, for comparison and to save previous states. This approach also supports backtracking to previous states and work spaces, to re-evaluate decisions and explore branching paths of the MDP. Lastly, the content of any pane can be exported/imported, allowing users to completely off-load their progress from a browser tab and start directly from any state or state selection that they had reached before.

Implementation: The Backend uses *dropwizard*[2] to establish a RESTful web-server that wraps PRISM and manages an SQL-lite database. The Frontend of PMC-VIS uses the *Cytoscape.js*[3] library (v3.25.0 [9]) for the graph views, enhanced through several of its add-ons (e.g., for graph layouts), alongside the well-established *D3.js*[4] library (v7.8.4) to construct the PCPs. This selection of libraries and frameworks ensures efficient loading and rendering times.

---

[2] dropwizard.io.
[3] js.cytoscape.org.
[4] d3js.org.

4 Usage Scenarios and Performance

To illustrate the capabilities of PMC-VIS, we provide four example models and several performance measurements on the accompanying artifact [22] and website imld.de/pmc-vis. In the following, we exemplify the usage of PMC-VIS for three individual usage scenarios on the model of a server management system (SMS). This model, consisting of 93,588 states, describes how a number of *tasks* is distributed to a number of *servers*. The usage scenarios are further illustrated in the Appendix of this paper.

Scenario 1 (Configuration) deals with the model configuration, in which the user is interested in finding a good server setup. In order to do this, a selection over the initial 84 states must be made. Using the PCP, the user can refine a selection that satisfies some criteria of interest. For example, by brushing the axis for $PrMaxHappy$ near value 1, the user ensures that only states that maximize the probability of successfully completing all tasks are selected. After similarly brushing over the lowest values of maximum and minimum energy consumption ($MostEnergy$ and $LeastEnergy$), a selection of only 3 states is achieved and can be expanded on an adjacent pane. Other patterns can be seen in the PCP that may inform different selections, for which the user can simply go back to the original pane to change the selection.

Scenario 2 (Exploration) explains how through the exploration of the MDP in different panes, it is possible for the user to distinguish 3 *phases* the model goes through, which helps construct a visual understanding of the model behavior while using PMC-VIS: (1) generating tasks, (2) re-configuring, and (3) assigning work. These phases repeat until some termination criteria is reached (e.g., a fixed number of phases have been completed). The split between these different phases also highlights the value of using different panes to make sure that no partial work is lost, which in systems with a single view is often not possible.

Lastly, **Scenario 3 (Strategic Exploration)** describes the additional means by which the users of PMC-VIS can construct *strategies* while navigating the MDP, informed by the accompanying PMC results. These strategies are essentially a list of choices that must be made in order to find certain paths in the MDP, that fulfill the goals of the SMS. Users can discover strategies by comparing per-node tooltips, PCPs in different panes (for both states and actions within the MDP), and MDPs with different highlighted schedulers. Schedulers, and particularly, the comparison of multiple scheduler options, helps the user understand how non-determinism is solved for maximum and minimum PMC results. Thus, the flexible exploration and comparison facilities of PMC-VIS aid in making sense of schedulers over multiple properties without the need to trigger new model checking computations. Ultimately, the creation of a strategy may span many panes, and disconnected, asynchronous work, which is why PMC-VIS also provides a feature to export a collection of marked nodes, which can be loaded onto a pane to explore gaps until a complete strategy has been developed.

Performance Experiment: Our goal was to provide both fast build time and smooth interaction with the MDP regardless of its total size. Thus, we measured the build and response times of multiple models of similar structure but exponentially increasing size (measured in number of states of the MDP), all solved for the same properties. With respect to the **model computation times,** our experiment shows that the overhead introduced by the creation and initial insertion into the database, compared to only executing the PMC procedures, is mainly bound to the database writing speed, where we perform around 40–50 operations per millisecond. However, all operations scale similarly with respect to model size, meaning we can compute PMC results normally with PRISM. Likewise, with regards to the **model exploration**, the Frontend performance is influenced by the response times of the Backend, which we kept under a second for small requests (1 to 5 states) or within a few seconds for large (10 queried states) requests, even on a model with 10^7 states. Beyond this, on a laptop with Intel(R) Core(TM) i7-7500U CPU and 16 GB of RAM, and using a Chromium 114.0.5735.133 browser, a single pane continues to operate smoothly with over 500 nodes. However, we do not foresee users working with large requests often, nor with that much content on a single pane.

5   Conclusion

We contributed PMC-VIS to support exploration of MDPs and PMC configuration and reconfiguration tasks. Our solution incorporates DOI graphs and PCPs on a multi-pane approach that, making use of efficient model checking and retrieval methods, supports users in understanding model behaviour and conjunct PMC results. We discussed how PMC-VIS can be used to answer various formal verification questions, especially in regard to the (re)configuration tasks. We foresee that further formal model and verification methods would benefit from extensions of our approach. Thus, we look forward to further generalizing PMC-VIS in various directions towards an IDE for automata-based operational models, model checking of functional and nonfunctional properties and functionalities for various synthesis questions, including further features for what-if analysis.

Data-Availability Statement. PMC-VIS, the used models, scenarios and performance experiment are open source and available on our supplementary web page at imld.de/pmc-vis and in the accompanying artifact at Zenodo [22]. Further figures of PMC-VIS can be found in the Appendix of this paper.

Appendix

The following illustrations are appended to the paper for the interested reader to have a deeper look at PMC-VIS and our results. (See Figs. 3, 4, 5, 6, 7 and 8).
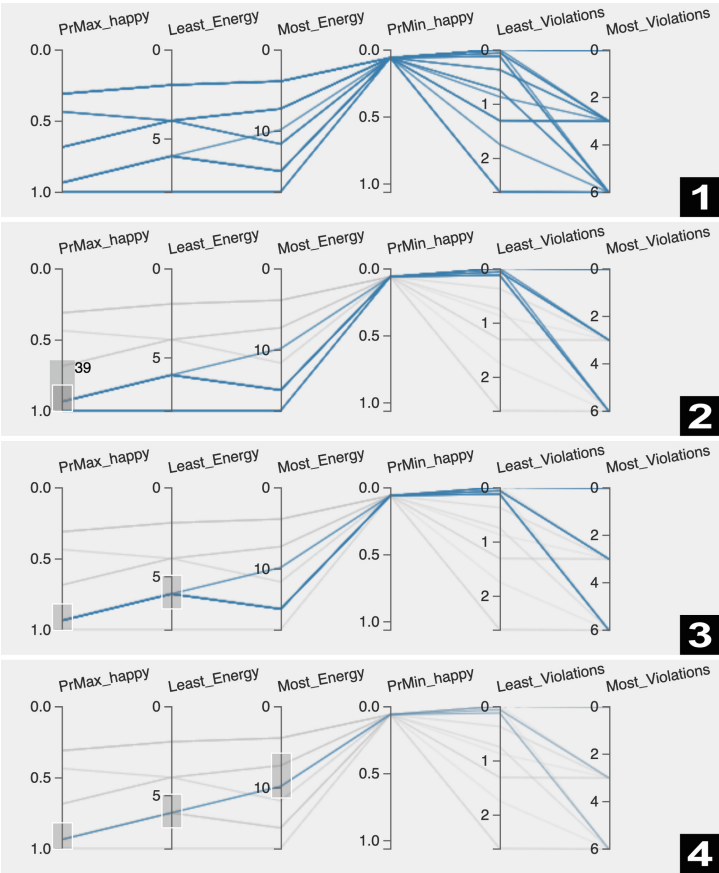

Fig. 3. Related to Scenario 1 described in Sect. 4. Axes brushing on the parallel coordinates plot. The numbers (1) through (4) represent the order of the brushing steps to reach a selection of 3 states out of the initial 84.
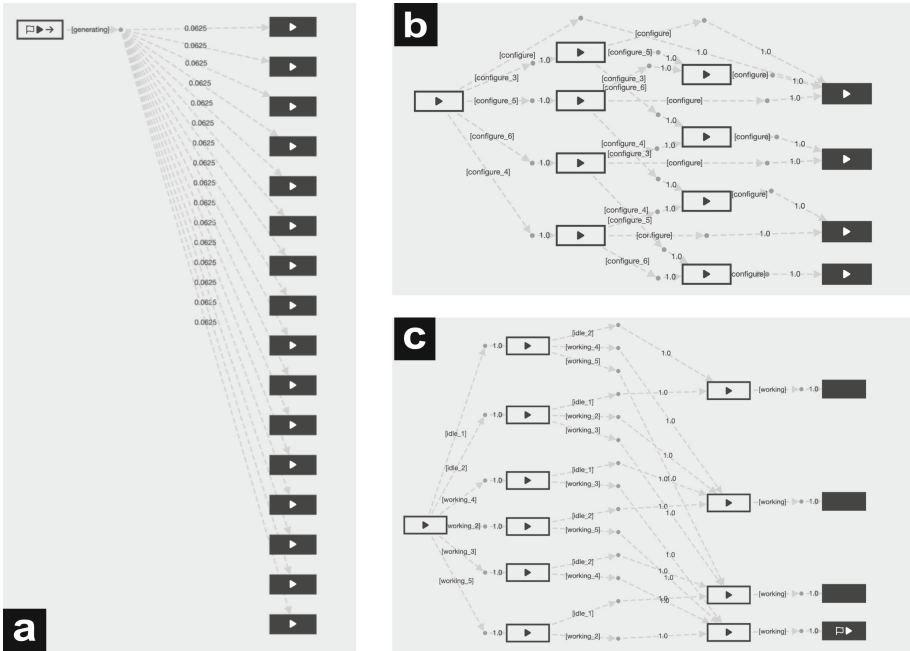
Fig. 4. Related to Scenario 2 described in Sect. 4. Three phases of our model shown as patterns in the graph views: (a) Generating tasks, (b) re-configuring servers, (c) assigning work to each server.
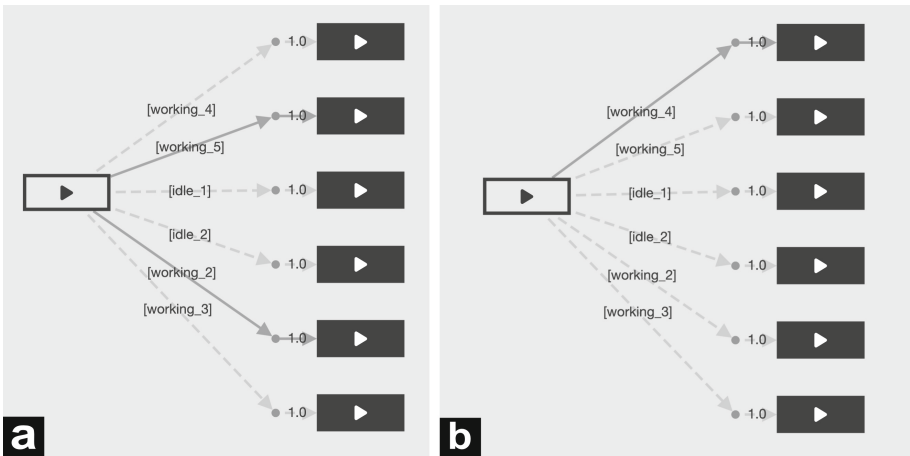
Fig. 5. Related to Scenario 3 described in Sect. 4. Different edge highlights depending on the selected scheduler. (a): $PrMax\_Happy$ was selected. (b): $Least\_Energy$ was selected.
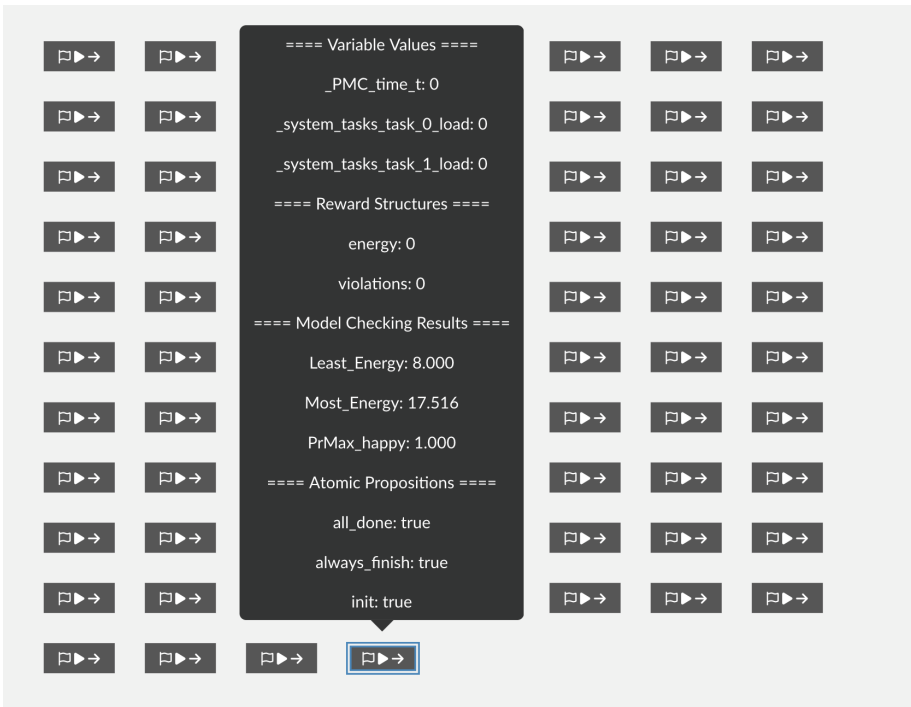
Fig. 6. Related to Scenario 3 described in Sect. 4. Shift+click on a state (with blue border) shows a tooltip, detailing all properties selected in the settings sidebar. (Color figure online)
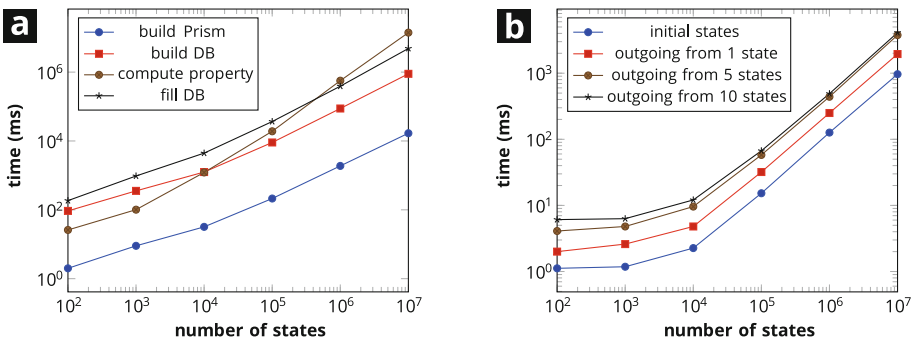

Fig. 7. Related to the experiment described in Sect. 4. (a): Times for single model computation. (b): Average response time (gathered over 10,000 responses). These experiments were carried out on a server with Intel(R) Xeon(R) L5630 CPUs with in total 8 physical cores and 189 GB of DDR3 RAM.
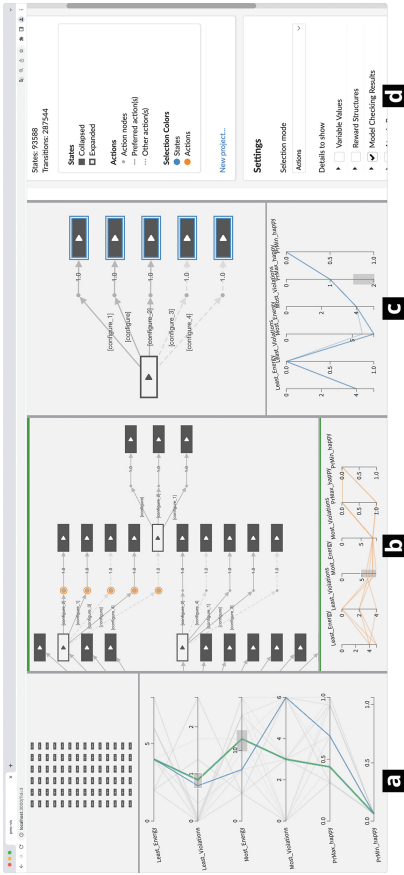
Fig. 8. Screenshot of PMC-VIS, showing 3 *panes* (a, b, c). Pane (a) shows the initial states of the MDP (as a grid of unconnected nodes above, and as polylines in the PCP below). Pane (b) shows a set of selected actions (in orange). Pane (c) shows a selection over the possible outcomes of a chosen action (states, in blue). (d) shows the settings sidebar, currently linked to pane (b) as indicated by the green border on top of this pane. (Color figure online)

References

1. Abello, J., van Ham, F., Krishnan, N.: ASK-GraphView: a large scale graph visualization system. IEEE TVCG **12**(5), 669–676 (2006). https://doi.org/10.1109/TVCG.2006.120

2. Baier, C., Hermanns, H., Katoen, J.-P.: The 10,000 facets of MDP model checking. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 420–451. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91908-9_21

3. Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press, Cambridge (2008)

4. Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., Yi, W.: UPPAAL — a tool suite for automatic verification of real-time systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) HS 1995. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0020949

5. Bunte, O., et al.: The mCRL2 toolset for analysing concurrent systems. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11428, pp. 21–39. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17465-1_2 ISBN 9783030174651

6. Card, S.K., Shneiderman, B., MacKinlay, J.D.: Readings in Information Visualization-Using Vision to Think. Series in Interactive Technologies. Morgan Kaufmann Publishers (1999). ISBN 1-55860-533-9

7. Dai, J., Cheng, J.: HMMEditor: a visual editing tool for profile hidden Markov model. BMC Genom. **9**(1), S8 (2008). https://doi.org/10.1186/1471-2164-9-S1-S8. ISSN 1471–2164

8. Elmqvist, N., et al.: ZAME: interactive large-scale graph visualization. In: 2008 IEEE PacificVis, pp. 215–222 (2008). https://doi.org/10.1109/PACIFICVIS.2008.4475479

9. Franz, M., et al.: Cytoscape.js 2023 update: a graph theory library for visualization and analysis. Bioinformatics **39**(1) (2023). https://doi.org/10.1093/bioinformatics/btad031. ISSN 1367–4811

10. Garavel, H., et al.: CADP 2011: a toolbox for the construction and analysis of distributed processes. STTT **15**(2), 89–107 (2013). https://doi.org/10.1007/s10009-012-0244-z. ISSN 1433–2787

11. Goldsby, H., Cheng, B.H.C., Konrad, S., Kamdoum, S.: A visualization framework for the modeling and formal analysis of high assurance systems. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MODELS 2006. LNCS, vol. 4199, pp. 707–721. Springer, Heidelberg (2006). https://doi.org/10.1007/11880240_49

12. Görke, R., Hartmann, T., Wagner, D.: Dynamic graph clustering using minimum-cut trees. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 339–350. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03367-4_30 ISBN 978-3-642-03367-4

13. Gros, T.P., Groß, D., Gumhold, S., Hoffmann, J., Klauck, M., Steinmetz, M.: TraceVis: towards visualization for deep statistical model checking. In: Margaria, T., Steffen, B. (eds.) ISoLA 2020. LNCS, vol. 12479, pp. 27–46. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83723-5_3

14. van Ham, F., Perer, A.: Search, show context, expand on demand: supporting large graph exploration with degree-of-interest. IEEE TVCG **15**(6), 953–960 (2009). https://doi.org/10.1109/TVCG.2009.108

15. Hensel, C., et al.: The probabilistic model checker storm (2020). arXiv: 2002.07080 [cs.SE]

16. Horak, T., Dachselt, R.: Hierarchical graphs on mobile devices: a lane-based approach. In: CHI MobileVis Workshop (2018)

17. Horak, T., et al.: Visual analysis of hyperproperties for understanding model checking results. IEEE TVCG **28**(1), 357–367 (2022). https://doi.org/10.1109/TVCG.2021.3114866. ISSN 1941–0506

18. Johansson, J., Forsell, C.: Evaluation of parallel coordinates: overview, categorization and guidelines for future research. IEEE TVCG **22**(1), 579–588 (2016). https://doi.org/10.1109/TVCG.2015.2466992

19. Kamhi, G., Fix, L., Binyamini, Z.: Symbolic model checking visualization. In: Gopalakrishnan, G., Windley, P. (eds.) FMCAD 1998. LNCS, vol. 1522, pp. 290–302. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49519-3_19 ISBN 9783540495192

20. Katoen, J.-P., et al.: The ins and outs of the probabilistic model checker MRMC. IPerform. Eval. **68**(2), 90–104 (2011). https://doi.org/10.1016/j.peva.2010.04.001. ISSN 0166–5316

21. Kerren, A., Purchase, H.C., Ward, M.O. (eds.): Multivariate Network Visualization. LNCS, vol. 8380. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06793-3

22. Korn, M., et al.: Interactive Visualization Meets Probabilistic Model Checking Artifact (2023). https://doi.org/10.5281/zenodo.8172531

23. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47

24. Liu, Y., et al.: HybridVis: an adaptive hybrid-scale visualization of multivariate graphs. JVLC **41**, 100–110 (2017). https://doi.org/10.1016/j.jvlc.2017.03.008. ISSN 1045–926X

25. McGregor, S., et al.: Facilitating testing and debugging of Markov Decision Processes with interactive visualization. In: IEEE VL/HCC 2015, pp. 53–61 (2015). https://doi.org/10.1109/VLHCC.2015.7357198

26. Nobre, C., et al.: The state of the art in visualizing multivariate networks. CGF **38**(3), 807–832 (2019). https://doi.org/10.1111/cgf.13728

27. Pfannkuch, M., Budgett, S.: Markov processes: exploring the use of dynamic visualizations to enhance student understanding. JSE **24**(2), 63–73 (2016). https://doi.org/10.1080/10691898.2016.1207404

28. Raseman, W.J., Jacobson, J., Kasprzyk, J.R.: Parasol: an open source, interactive parallel coordinates library for multi-objective decision making. EMS **116**, 153–163 (2019). https://doi.org/10.1016/j.envsoft.2019.03.005

29. Tan, Y.-Q., et al.: VecRoad: point-based iterative graph exploration for road graphs extraction. In: 2020 IEEE/CVF CVPR, pp. 8907–8915 (2020). https://doi.org/10.1109/CVPR42600.2020.00893

30. Wang, Y., et al.: Structure-aware fisheye views for efficient large graph exploration. IEEE TVCG **25**(1), 566–575 (2019). https://doi.org/10.1109/TVCG.2018.2864911