Online Appendix to: Automatic Runtime Adaptation for Component-based Simulation Algorithms

TOBIAS HELMS, ROLAND EWALD, STEFAN RYBACKI,
and ADELINDE M. UHRMACHER, University of Rostock

A.1. Policy Experiments

To examine the scalability of our approach with respect to the number of actions, we repeat the benchmark ML-Rules experiment setup (100 replications with 40,000 events each, each experiment repeated 50 times) with 36 instead of two actions (see Section 5.2). This is a more challenging learning task because the Q matrix is larger (see Section 3). For preprocessing, we use a static state space generalization with configuration c_2 . Here, in contrast to Helms et al. [2013], the 36 actions correspond to 36 configurations of the ML-Rules SA, combined with only one adaptation condition (adapt every 1,000 events). In Helms et al. [2013], we used 12 SA configurations and three adaptation conditions. Our previous results show that, at least in this setting, considering multiple step count adaptation conditions only worsened the learning efficiency and did not improve the overall performance.

With 356 actions, the chosen policy determining the tradeoff between exploration and exploitation becomes more important, and, thus, we use five different policies for this experiment: ϵ -greedy, ϵ -decreasing, *Upper Confidence Bound 1 (UCB1)* [Auer et al. 2002], *Interval Estimation* [Kaelbling 1993], and *SoftMax* (e.g., Vermorel and Mohri [2005]). The ϵ -greedy strategy selects a random action with probability ϵ and the best action (based on the already gained knowledge) with probability $1 - \epsilon$. Thus, it is not a zero-regret strategy because it always selects random actions with the same probability. We use $\epsilon = 0.15$ for this policy for all experiments. In contrast, ϵ -decreasing reduces the probability to explore actions constantly (i.e., the probability to select a random action is $\min(1, \frac{\epsilon}{n})$, where n is the number of occurrences of the according state). Consequently, ϵ -decreasing is a zero-regret strategy. For this policy, we use $\epsilon = 5$ for all experiments. The deterministic *UCB1* policy initially chooses each action once. For each following decision, the action which maximizes

$$\hat{\mu}_i + \sqrt{\frac{2 \ln n}{n_i}}$$

is chosen, where $\hat{\mu}_i$ represents the current estimate of action i 's performance, n the number of occurrences of the according state, and n_i , the number of times action i has been selected in this state. Thus, the probability to take the best action increases over time as the second summand of the equation converges to zero. Consequently, this is a zero-regret strategy. Similar to *UCB1*, the *Interval Estimation* policy initially chooses each action once. Afterward, confidence intervals of the performance of each action are computed, controlled by a parameter $\alpha \in (0, 1)$. These confidence intervals are used to determine the next action. If α is fixed, this policy is not zero-regret. We set $\alpha = 0.05$ for all experiments. Finally, the *SoftMax* policy computes the probability p_i to choose

Fig. 1. The dynamic regret of the adaptive simulator setups on the ML-Rules synthetic benchmark model with five different action selection policies and 36 actions.

action i with

$$p_i = \frac{e^{\mu_i/\tau}}{\sum_{j=1}^{|A|} e^{\mu_j/\tau}},$$

where $\tau \in \mathbb{R}^+$ is a parameter (called temperature) chosen by the user. The higher τ is chosen, the more exploration takes place. This strategy is not zero-regret, but does not have to select each action once before using the gained knowledge. All in all, the parameter values we use are usual default parameter values of these policies. We use them because we do not want to complicate the configuration of the adaptive simulator here. Figure 1 shows the average dynamic regret of the adaptive simulator with the five policies. Even with 36 actions, except with *SoftMax*, the adaptive simulator outperforms the average performance of the 36 configurations, and it even outperforms the best nonadaptive SA after 100 replications with ϵ -greedy, ϵ -decreasing, and *Interval Estimation*. The policies that initially execute each SA configuration once (*UCB1* and *Interval Estimation*) perform worst (and similar) for the first 10 replications because they do not sufficiently use the gained knowledge at the beginning. Typically, several configurations of the SAs perform similarly compared to the best configuration due to parameters of the SAs with negligible influence. Thus, the simple policies ϵ -greedy and ϵ -decreasing perform best because it is not necessary to find the best configuration to reach almost-best performance, which fits the results presented in Ewald et al. [2009]. The other three complex policies choose bad actions several times to be sure that they are bad, but because the deviation of the execution times is small, these tries are not needed.

A.2. Evaluation on Realistic Models

The benchmark model evaluated in Section 5.2 exhibits only two distinct simulation phases, each with a fixed amount of simulation events. Realistic models, in contrast, exhibit an unknown (potentially large) number of (hard to distinguish) phases, between which they may switch rapidly. Consequently, we conducted experiments with three realistic, complex ML-Rules models: a Cell Cycle model [Maus et al. 2011] for 100,000 simulation events, an Endocytosis model [Helms et al. 2012] for 100,000 simulation events, and a Wnt/ β -catenin pathway model [Mazemondet et al. 2012] for 500,000 simulation events. The adaptive simulator uses the same 36 actions as in the experiments described in Appendix A.1. Here, based on the results with the benchmark model, the adaptive simulator uses a static state space generalization with the configuration $c_4 = (5, 5, 0.1, 0.1)$ and the ϵ -decreasing ($\epsilon = 5$) policy. To compare the results of the adaptive simulator, we also executed experiments with an adaptive task runner, which adaptively selects SAs for whole simulation runs [Ewald et al. 2009]. For the adaptive

Fig. 2. Boxplots of the average execution times to compute one replication for all 36 SA configurations (A), the adaptive task runner executing 10 replications (B), the adaptive task runner executing 100 replications (C), the adaptive simulator executing 10 replications (D), and the adaptive simulator executing 100 replications (E). The horizontal dotted lines represent the execution time per replication for the corresponding model, averaged over all 36 SA configurations.

Table I. Average Execution Times for One Replication of the Corresponding Model, for the 36 SA Configurations, The Adaptive Simulator, and the Adaptive Task Runner

	Cell Cycle	Endocytosis	Wnt Pathway
36 SA configurations	24.5 s ($\sigma = 4.4$ s)	60.9 s ($\sigma = 13$ s)	44 s ($\sigma = 12$ s)
Adapt. Sim. executing ten replications	22.2 s ($\sigma = 0.9$ s)	62.1 s ($\sigma = 3.4$ s)	32.3 s ($\sigma = 1.2$ s)
Adapt. Sim. executing 100 replications	21.3 s ($\sigma = 0.7$ s)	55.9 s ($\sigma = 2.2$ s)	31 s ($\sigma = 1.15$ s)
Adapt. task runner executing ten replications	26.9 s ($\sigma = 1.8$ s)	63.3 s ($\sigma = 5.8$ s)	43.7 s ($\sigma = 4.5$ s)
Adapt. task runner executing 100 replications	21.8 s ($\sigma = 0.8$ s)	56.9 s ($\sigma = 1.5$ s)	33 s ($\sigma = 0.9$ s)

task runner, we also use ϵ -decreasing ($\epsilon = 5$). As before, each setup simulates 100 replications, and the overall experiment is repeated 50 times. Figure 2 shows the execution times to compute one replication for the three models; Table I lists their averages.

For all models, the adaptive simulator outperforms the average performance of the 36 SA configurations when executing 100 replications. For the Cell Cycle model and the Wnt Pathway model, the adaptive simulator even outperforms the average performance after ten replications. Furthermore, the results of the adaptive simulator are more robust compared to the results of the 36 configurations. The performance is more reliable (i.e., its standard deviation is smaller), and the worst-case performance of the adaptive simulator is much better than that of the 36 SA configurations. Thus, although the adaptive simulator does not perform better than the (typically unknown) best nonadaptive SA for one model, it is still beneficial to use it. Compared to the adaptive task runner, the adaptive simulator performs similarly when executing 100 replications but much better when executing 10 replications. The adaptive simulator performs better because it can use its learned knowledge at once, and it receives more data to learn from (the adaptive task runner only receives the overall runtime of each replication). The adaptive task runner, on the contrary, is focused on exploration for

Fig. 3. The species amount of the β -catenin protein within the Wnt/ β -catenin pathway model based on the simulation results of 100 replications is shown, once computed with a nonadaptive ML-Rules simulator (left) and once computed with the adaptive simulator (right). Both results are almost equal.

the first 10 replications and thus merely reaches average-case performance. For the Cell Cycle model, it performs even much worse than the average performance of the 36 SA configurations.

The performance differences of the adaptive simulator across the models are caused by the static state space generalization, which is too fine-grained for the Endocytosis model but suitable for the other two models. Approximately 120 states are visited on average while simulating the Endocytosis model, so that the adaptive simulator simply needs more runs to achieve better results. Based on the execution times of the 36 SA configurations, we can also determine the execution times of an approximately optimal adaptive simulator for the three models (see Section 5.1). For the Cell Cycle model as well as the Wnt Pathway model, the approximately optimal adaptive simulator performs only slightly better than the best nonadaptive SA. For the Endocytosis model, the best nonadaptive SA needs 45.6 s on average per simulation run, whereas 40 s on average could be reached with optimal adaptation decisions. The adaptive simulator either does not reach this execution time due to a still too coarse-grained state space or the state space with c_4 is sufficient, but simply more simulation runs are needed to learn better policies. Another reason could be that more features are needed for the states (i.e., more information about the model during the simulation is needed). All in all, the performance differences of the adaptive simulator across the three models suggest that a default configuration for static state space generalization is unlikely to yield good results in all situations. A dynamic generalization is required instead.

In addition to these performance analyses, Figure 3 shows simulation results of β -catenin proteins within the Wnt/ β -catenin pathway model based on 100 replications, once computed by a nonadaptive ML-Rules simulator and once computed by the adaptive simulator. The simulation results are almost equal and also, statistically, the simulation results of both simulators do not differ significantly. We checked this by executing the Wilcoxon signed-rank test for each observation point with a significance level of 0.01: For all tests, the hypothesis that the results do not differ significantly holds.

REFERENCES

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 2–3 (May 2002), 235–256. DOI: <http://dx.doi.org/10.1023/A:1013689704352>

- Roland Ewald, Stefan Leye, and Adeline M. Uhrmacher. 2009. An efficient and adaptive mechanism for parallel simulation replication. In *Proceedings of the 23rd Workshop on Principles of Advanced and Distributed Simulation (PADS'09)*. 104–113. DOI: <http://dx.doi.org/10.1109/PADS.2009.11>
- Tobias Helms, Roland Ewald, Stefan Rybacki, and Adeline M. Uhrmacher. 2013. A generic adaptive simulation algorithm for component-based simulation systems. In *Proceedings of the 27th Workshop on Principles of Advanced and Distributed Simulation (PADS'13)*. 11–22. DOI: <http://dx.doi.org/10.1145/2486092.2486095>
- Tobias Helms, Jan Himmelspace, Carsten Maus, Olivier Röwer, Johannes Schützel, and Adeline M. Uhrmacher. 2012. Toward a language for the flexible observation of simulations. In *Proceedings of the Winter Simulation Conference (WSC'12)*. 418:1–418:12.
- Leslie P. Kaelbling. 1993. *Learning in Embedded Systems*. MIT Press.
- Carsten Maus, Stefan Rybacki, and Adeline M. Uhrmacher. 2011. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology* 5, 166.
- Orianne Mazemondet, Mathias John, Stefan Leye, Arndt Rolfs, and Adeline M. Uhrmacher. 2012. Elucidating the sources of β -catenin dynamics in human neural progenitor cells. *PLoS ONE* 7, 8, e42792. DOI: <http://dx.doi.org/10.1371/journal.pone.0042792>
- Joannès Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*. 437–448.