Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

An Adaptive Simulator for ML-Rules

Tobias Helms
Stefan Rybacki
Roland Ewald
Adelinde M. Uhrmacher

Albert Einstein Str. 22
University of Rostock
18059 Rostock, GERMANY

ABSTRACT

Even the most carefully configured simulation algorithm may perform badly unless its configuration is adapted to the dynamics of the model. To overcome this problem, we apply methods from reinforcement learning to continuously re-configure an ML-Rules simulator at runtime. ML-Rules is a rule-based modeling language primarily targeted at multi-level microbiological systems. Our results show that, for models with sufficiently diverse dynamics, an adaptation of the simulator configuration may even outperform the best-performing non-adaptive configuration (which is typically unknown anyhow).

1 TOWARDS AN ADAPTIVE ML-RULES SIMULATOR

The simulation of models defined in ML-Rules (Maus et al. 2011) is quite challenging, due to several aspects of the language. Most importantly, the so-called rule schemas need to be matched against the current population of chemical species, in order to generate all possible chemical reactions. As each ML-Rules species may have arbitrarily many attributes and can also be nested, their management raises several issues. For example, the simulator may need to retrieve all species from the population that have a certain attribute value, in order to calculate the reaction rate. If such look-ups occur frequently, it is more efficient to employ a dedicated data structure for this task, such as a gridfile (Hinrichs 1985). On the other hand, initializing and maintaining an additional data structure also introduces overhead, which may even slow down the overall simulation (e.g., when look-ups occur only rarely). Similar trade-offs arise in several other contexts, e.g., for the usage of advanced hash functions (which may speed up equality checks for nested species), or the execution of reactions.

Whether or not any of these more sophisticated extensions to the original ML-Rules simulator (see Maus et al. 2011) is effective depends to some extent on the model's structure (which could be analyzed before execution), but also on the current *state* of the model, i.e., the size and structure of the current species population. The model state is only available at runtime and changes with each simulated reaction event. Hence, only a dynamic re-configuration mechanism can ensure that the simulator configuration remains suitable over the whole execution period. Additionally, approaches like reinforcement learning (Sutton and Barto 1998) may even work unsupervised and without prior performance analyses, which are often required for automatic algorithm selection (see Ewald 2011).

To investigate the effectiveness of an adaptive ML-Rules simulator, we extend the original ML-Rules simulator regarding the aspects described above. The ML-Rules simulator is implemented on top of the modeling and simulation framework JAMES II (Himmelspach and Uhrmacher 2007), so new auxiliary data structures (like a gridfile) have been defined as plug-ins.

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

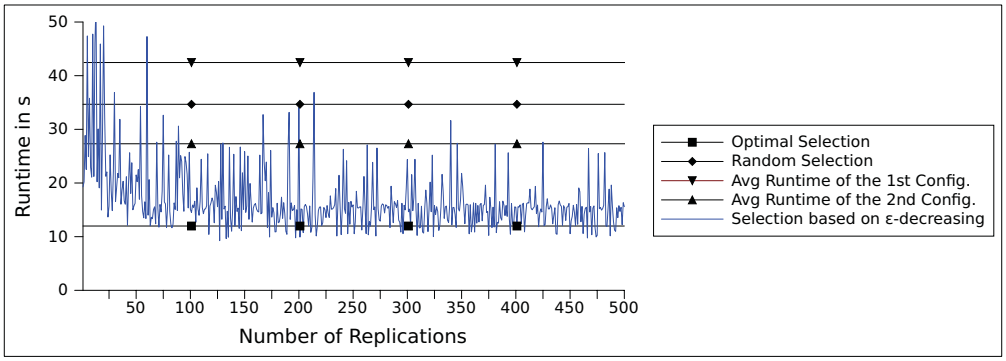

Figure 1: Sample results for the adaptive ML-Rules simulator.

2 RESULTS

As a proof of concept, we constructed a simple benchmark model that consists of a stochastic switch to activate one of two rules. The rules differ only slightly from each other, in that one rule makes use of some pattern matching while the other does not. This has an impact on the invocation pattern with which species are retrieved from the overall population (see sec. 1), which may be exploited by an adaptive simulator to improve overall performance. In order to show that the adaptive simulator works adequately, we configured the online adaptation mechanism to choose from one of two possible simulator setups, whereas one configuration is more suitable for the first rule of the benchmark model and the other configuration is more suitable for the second rule of the benchmark model. The adaptive simulator re-evaluates its performance (measured as the number of processed events) every 1000 steps. The selection mechanism relies on an ε-decreasing strategy (Vermorel and Mohri 2005). The resolution of the adaptive state space is relatively high and fixed, and finally the learning method is based on the Q-Learning algorithm (Watkins 1989) which in this case learns across replications of an experiment. Figure 1 shows that the runtimes of the replications converge with an increasing number of executed replications to the optimum runtime, i.e. the runtime based on optimal decisions. At the end the adaptive simulator reduced the runtime of an execution of a replication by about 50% compared to the permanent usage of the better of both configurations. These results show the potential of our approach.

ACKNOWLEDGMENTS

This research has been supported by the DFG (German Research Foundation), via research projects ALESIA (EW 127/1-1) and VASSIB (part of SPP 1335).

REFERENCES

Ewald, R. 2011. *Automatic Algorithm Selection for Complex Simulation Problems*. Vieweg + Teubner.

Himmelspach, J., and A. M. Uhrmacher. 2007. "Plug'n simulate". In *Proceedings of the 40th Annual Simulation Symposium*, 137–143: IEEE Computer Society.

Hinrichs, K. 1985, December. "Implementation of the grid file: Design concepts and experience". *BIT Numerical Mathematics* 25 (4): 569–592.

Maus, C., S. Rybacki, and A. M. Uhrmacher. 2011. "Rule-based multi-level modeling of cell biological systems". *BMC Systems Biology* 5:166.

Sutton, R. S., and A. G. Barto. 1998, May. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

Vermorel, J., and M. Mohri. 2005. "Multi-armed Bandit Algorithms and Empirical Evaluation". In *Machine Learning: ECML 2005*, Volume 3720 of *Lecture Notes in Computer Science*, 437–448.

Watkins, C. J. C. H. 1989. *Learning from delayed rewards*. Ph. D. thesis, University of Cambridge, England.