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

Configuring Simulation Algorithms with ParamILS

Robert Engelke
Roland Ewald

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

ABSTRACT

Simulation algorithms often expose various numerical parameters, e.g., to control the size of auxiliary data structures or to configure certain heuristics. While this allows to fine-tune a simulator to a given model, it also makes simulator configuration more complex. For example, determining suitable default parameters from a multi-dimensional parameter space is challenging, as these parameters shall work well on a broad range of models. Instead of manually selecting parameter values, the configuration space of a simulation algorithm can also be searched automatically. We investigate how well PARAMILS (Hutter et al. 2009), an iterated local search algorithm for algorithm configuration, can be applied to simulation algorithms, and discuss its implementation in context of the open-source modeling and simulation framework JAMES II.

1 PARAM-ILS

PARAMILS searches for an algorithm's parameter configuration that results in optimal performance on a (randomly drawn) sequence of problem instances. It tries to find good parameter configurations by starting from an initial configuration, evaluating it, and then changing some of its parameters at once (sometimes all). The new configuration is evaluated and used as the starting point for a hillclimbing, to find a local optimum. The whole procedure is repeated until a termination condition (e.g., overall execution time) is met.

During hillclimbing, two configurations are compared by applying them to the sequence of problem instances (in our case, model setups) and recording their performance (in our case, execution time). Hutter et al. (2009) present two techniques for this comparison, called BasicILS and FocusedILS. BasicILS evaluates both configurations on the same fixed sequence of problem instances. FocusedILS, on the other hand, stops the comparison at some problem instance if one configuration performs worse than the other (on a sufficiently large sub-sequence of problem instances). Another abort mechanism for configurations which perform poorly is to cap the ressources for them, depending on the best configuration found so far.

While PARAMILS is not restricted to any specific kind of algorithm, it has been predominantly applied to SAT-solvers (Hutter et al. 2009). Additional issues may arise for simulation algorithms. For example, when evaluating stochastic simulation algorithms, the noise inherent in execution time measurements may be much larger, as it does not only stem from the underlying hardware and operating system, but also from the diversity of possible simulation trajectories and the corresponding computational loads to simulate them. Such effects may hamper the applicability of PARAMILS and thus need to be investigated carefully.

2 IMPLEMENTATION & INTEGRATION INTO JAMES II

Our implementation of PARAMILS is integrated into the modeling and simulation framework JAMES II (Himmelspach and Uhrmacher 2007), so it can be used with any JAMES II simulator. A major challenge

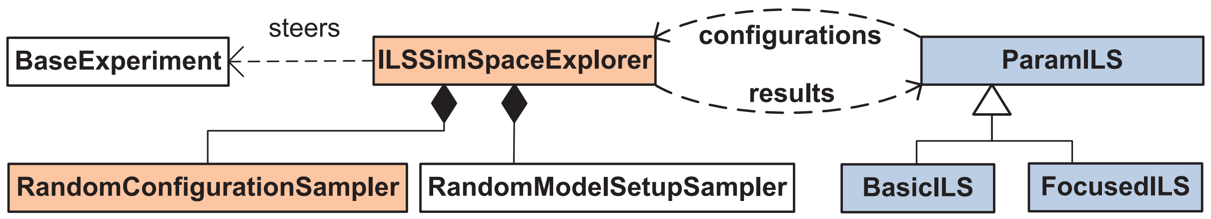

Figure 1: Apart from implementing the PARAMILS variants as such (blue types), a custom exploration component that supports random sampling of hierarchical simulator configurations and their numerical parameters (orange types) had to be added to the JAMES II experimentation layer. A PARAMILS instance runs in a separate thread and sends requests to the experimentation layer. A request triggers the execution of a specific simulator configuration on a specific model setup. Eventually, the PARAMILS instance receives the results of the run and continues the search.

was to support the random sampling of both simulator configurations and simulation models in a way that is transparent to PARAMILS. In particular, how to deal with the hierarchical parameterization of simulators and their sub-components (i.e., plug-ins) in JAMES II? Each simulator — and each plug-in it relies on, etc. — may have an arbitrary number of numerical parameters that shall be searched. This results in a *hierarchy* of parameters, from which we sample in proportion to the dimensionality of the parameter space for each sub-tree. For example, consider a simulator that may either rely on plug-in A, with one numerical parameter, or plug-in B, with three numerical parameters. In this case, our sampling mechanism is $\frac{2^3}{2^1} = 4$ times more likely to draw a configuration for plug-in B. This is done to ensure that the larger parameter space of plug-in B is explored sufficiently. Figure 1 shows how the PARAMILS variants are connected to the experimentation layer of JAMES II, mostly by relying on already established concepts regarding the exploration of a 'simulation space', i.e., a space of model setups and simulator configurations (see Ewald and Uhrmacher 2009).

A first evaluation of both BasicILS and FocusedILS was conducted for the discrete-event stochastic simulation algorithm τ-leaping (Cao et al. 2006), which has four numerical parameters that can be adjusted. Preliminary results indicate that the approach is feasible to find good parameter configurations. The approach should be particularly suitable to tune simulators with several sub-components that exhibit numerical parameters, e.g., parallel and distributed discrete-event simulators (Fujimoto 2000) that rely on partitioning, synchronization, and load balancing mechanisms. We plan to investigate this in future work.

ACKNOWLEDGMENTS

This research has been supported by the DFG (German Research Foundation), research project `EW 127/1-1` (*ALeSiA*).

REFERENCES

Cao, Y., D. T. Gillespie, and L. R. Petzold. 2006, Jan. "Efficient step size selection for the tau-leaping simulation method.". *The Journal of Chemical Physics* 124 (4).

Ewald, R., and A. M. Uhrmacher. 2009. "Automating the Runtime Performance Evaluation of Simulation Algorithms". In *Proceedings of the Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 1079–1091: IEEE Computer Society.

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*. Wiley.

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

Hutter, F., H. Hoos, K. Leyton-Brown, and T. Stützle. 2009. "ParamILS: An automatic algorithm configuration framework". *Journal of Artificial Intelligence Research* 36:267–306.