Proxels Applied to Sensitivity Analysis and Optimization of Discrete Stochastic Models

Claudia Isensee, Graham Horton *
Otto-von-Guericke Universität Magdeburg
claudia@sim-md.de, graham@sim-md.de

Abstract

Simulation-based optimization or parameter tuning of discrete stochastic models becomes necessary, when no analytic expression for the goal function is available. Sensitivity analysis on the other hand is used to determine the required degree of detail that is needed when building a model. Both of these tasks are similar in the sense that one needs to repeatedly simulate a model with only slight changes in the tested parameter sets. Usually the steady state results using these slightly different parameter sets are also close to each other in some sense.

This paper shows that state space-based simulation can reuse old simulation results to arrive at a steady state solution faster than when starting from an initial model state, if the successive model configurations are in some sense close to each other. Performance gains can be generated for sensitivity analysis and optimization compared to restarting the simulation every time anew.

The paper will present the idea and an algorithm for performing successive runs of the Proxel simulation using the information from the previous ones. Experiments test the applicability to sensitivity analysis and optimization, and investigate the possible saving dependent on the size of the perturbation.

1 Introduction

1.1 Motivation

When building discrete stochastic simulation models, one is often faced with the question, whether a real system variable (such as the service time in a bank) has been measured and estimated well enough. This ”well enough” depends on the so called sensitivity of the model to changes in this variable. Sensitivity analysis in general is used to determine the required degree of detail that is needed when building a simulation model. The local gradient of an output measure is a good indicator of its sensitivity to parameter changes. The gradient is also needed when trying to optimize the system parameters using a hill-climbing optimization approach.

When no analytic expression for the gradient is available, it has to be determined experimentally. Using discrete-event simulation (DES) one has to perform replications for every parameter set in order to get an estimate of the gradient of the output measure, which can

*Fakultät für Informatik, Institut für Simulation and Graphik, 39106 Magdeburg, Germany

get expensive depending on the stiffness of the model. To determine the gradient of a goal function or output measure, one needs to simulate the model with only slight changes in the tested parameter sets. If the models behavior is not chaotic, the output values of these experiments should also be similar. So the final model states of the successive experiments should be closer to each other than to the initial model state. If that is the case, it would be advantageous, if one could somehow use the information contained in one solution to help determine the next one, and thereby save computation cost.

1.2 State of the art

Standard discrete-event simulation reruns all the replications if one wants to test a different parameter set, regardless of the size of the difference. Analytical methods on the other hand require a deep knowledge of the model to be simulated since they try to establish a mathematically formulated relationship between the model parameters and the simulation output. They are usually specialized for a certain class of models (queuing networks). If more general models grow above a trivial size, the mathematical calculations become very complex and setting up and solving the equations becomes impractical.([LK00]) Perturbation analysis observes one single path of the development of the stochastic system. It perturbs the path by some rule and determines a gradient of the desired output measure. This gradient is correct for that path, but is in itself a random variable and can only be an estimate of the actual gradient of the whole stochastic system. ([HC91]) Each of the examined existing approaches to sensitivity analysis and optimization has at least one serious drawback. They either require a possibly large number of replications, are not generally applicable, or result only in a sample of the desired output quantity.

1.3 Proxel-based Simulation

Our recently developed Proxel-based simulation method ([LM05, Hor02]) is generally applicable, does not require replications and produces deterministic results with an arbitrary accuracy. It is a state space-based simulation method, which expands a models discrete state space by the age information of the activated transitions, and thereby produces and solves a discrete-time Markov chain (DTMC). Due to this, the method has the ability to use the result from a previous simulation as valid starting point for a new simulation run. In this paper we further explain, why this is the case, and how this property can be exploited for sensitivity analysis and optimization.

2 Adapting the Proxel Algorithm

2.1 The Original Proxel-based Simulation Algorithm

The Proxel-based simulation method solves a discrete stochastic model by turning it into a DTMC and solving that. The non-Markovian transitions are expanded by their age information and the state transition probabilities can be calculated using the instantaneous rate function. The expanded model state space is built up on-the-fly and one of its elements

is called a Proxel (probability element), which contains the discrete model state, the age information and the probability of that combination at the current simulation time step.
The result at each step of the simulation is the probability vector of the whole reachable expanded model state space. By manipulating the discretization time step one can obtain a solution of chosen accuracy without having to do replications. This solution contains all the information needed to calculate performance measures like average queue length, activation time and throughput of a transition, or utilization of a server.

2.2 Application to Gradient Estimation

Since the solution produced by the Proxel algorithm is the probability vector of a DTMC, our Proxel-based simulation method has the ability to reuse the information contained in one steady-state solution of a simulation model for the next experiment, with the same or a similar model. If the simulation converges, then the Proxel solution contains the probabilities of all reachable model states in steady state. To obtain a solution with a slightly different model parameter setting, one can start this new simulation using the results from the previous one as initial probability vector of the DTMC. Due to only small parameter changes, the new solution is most likely closer to the previous solution, than to an arbitrary initial system state. Therefore we expect to save computation cost by using the previous solution as a starting point in contrast to restarting the simulation every time. The closer the perturbed parameter set is to the original one of the previous run, the fewer iterations should be needed to find a new steady state solution. The resulting saving in computation time will most likely depend on the size of the perturbation in the model parameters.
The described approach can be used to test the sensitivity of the model to small parameter changes. One can also repeat these successive runs iteratively and thereby determine the gradient of the output measures in relation to input parameters over a longer parameter range. The gradient of the output measure or goal function can then be determined by extracting the scalar estimates from the two or more solutions and determining the gradient, or the direction of steepest ascent in the case of optimization.

2.3 Implementation Details and User Interface Description

No major changes had to be made to the core of the Proxel-based simulation algorithm, the challenges were rather in the implementation of the user interface and the correct optimization strategy. Furthermore a termination criterion had to be specified, where before the simulation was always performed up to a fixed maximum time. The developed user interface will only be described here briefly, due to space limitations. It enables the user to load a certain models description with discrete system states and the transitions between them. The user can then choose between plain simulation, sensitivity analysis or optimization.
The plain simulation interface compared to previous ones offers the possibility to simulate to a certain point in simulation time, plot the simulation results, and then continue from there, if the probabilities have not converged yet.
The sensitivity analysis dialog enables the user to specify a range and step size for any of the distributions parameters of the models transitions. Only one parameter can be varied at

one time, but this is only restricted by the current implementation and no general limitation. The results of the runs in terms of the steady state probabilities can be plotted.
The optimization dialog enables the user to specify a goal function, that has to be minimized or maximized. The goal function is a sum of products of one or multiple state probabilities and scalars. This way, most interesting goal functions can be described such as average queue length, server utilization, machine downtime etc. Only one distribution parameter can be varied in the current implementation (s.a.). As in the sensitivity dialog, the user can also specify a range for the parameter, but is not required to do so.
With this interface we were able to conduct some experiments to examine the behavior of the method, which will be described in the next section.

3 Experiment

3.1 Sensitivity Analysis Experiment

This section will apply the above described approach to a simple sensitivity analysis problem. The model describes an inventory system with exponentially distributed demands of about 1 per day ($TDemand \sim Exp(0.9 \ldots 1.1)$). The inventory can at most hold 10 items with a daily storage cost of 10 $Euro$ each. When there are only four items left in storage, a refill order is issued, which takes a random amount of days to arrive ($TRefill \sim Weibull(2, 4)$). The miss penalty for a demand that cannot be met is 1000 $Euro$. The objective is to examine the sensitivity of the overall cost of the inventory per day, with respect to the demand rate, in order to see if the demand rate has to be determined more precisely.

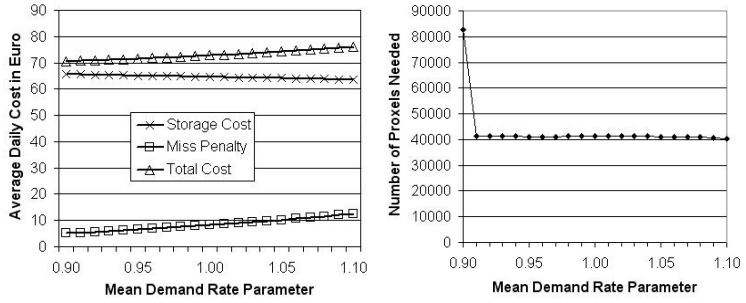

Figure 1: Inventory Cost for different Demand Rates (left) Number of Proxels Needed per Run with Demand Rate (right)

The sensitivity analysis was performed using a simulation time step of $\Delta t = 0.1$. The maximum allowed error before termination of one run was set to $\epsilon = 1e-12$. The range of the demand rate parameter was set to be between 0.9 and $1.1 demandperday$, with a step of 0.01. Figure 1 (left) shows the steady state results of the expected costs for different demand rates (calculated with successive simulation runs): storage cost, average miss penalty and

total cost, which is the sum of the miss penalty and storage cost. The average total daily cost varies between 70 $Euro$ and 76 $Euro$ in the specified interval, which is quite a big range, so the demand rate should be determined more precisely. The computation cost of each run with different demand rate, measured in the number of Proxels needed to find the steady state solution, can be seen in Figure 1 (right). This measure was chosen because it is independent of the current processor load and the time needed to process one Proxel is nearly constant. The computation cost of the initial run is about twice as big as the cost of the subsequent ones. The algorithm finds quite fast the general region of the solution but it needs a large number of simulation steps to actually converge to the final steady state. All but the first run have very similar computation cost, because the parameter difference is the same ($+0.01$) in every step. The user saves almost half of the runtime compared to starting each run from scratch, not to mention the setup time if the runs were started separately.

3.2 Optimization Experiment

In this section we will examine the algorithms behavior when optimizing a small maintenance model with respect to machine availability. The model consists of a machine that is running most of the time and is maintained at regular intervals ($TMaintInt \sim N(170, 5)$), which also takes a random amount of time ($TMaint \sim N(1, 0.2)$). The machine can also fail before the maintenance starts ($TFail \sim N(200, 50)$), which then takes a random amount of time to repair ($TRepair \sim N(4, 0.5)$). The time to failure is reset after every maintenance. The objective is to maximize the steady state machine availability, since the machine only generates profit, when it is running. The variable parameter is the mean maintenance interval $TMaintInt$. The time between successive maintenances should be optimal, not causing too much downtime, but also preventing frequent failures.

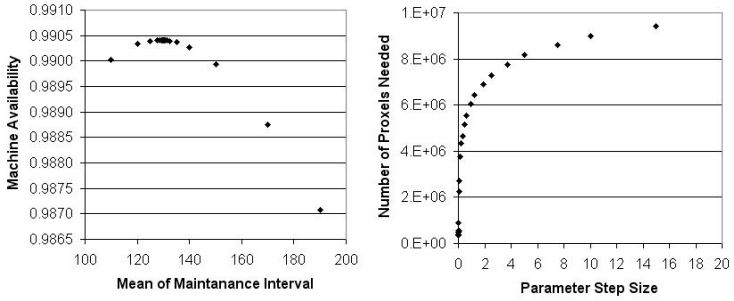

Figure 2: Optimization of Machine Availability Over Mean Maintenance Interval (left) Number of Proxels Needed over Parameter Step Size (right)

The optimization was conducted with a simulation time step of $\Delta t = 0.5$. The maximum allowed error before termination of one run was set to $\epsilon = 1e - 14$. The range of the mean maintenance interval parameter was set to be between 100 and 200 $hours$, with an initial step of 20. Figure 2 (left) plots the machine availability against the mean mainte-

nance interval that generated them. One can clearly see the optimum of the curve, which was determined at $mean = 130$. The computation cost of the successive runs (measured in the number of Proxels needed until convergence) with respect to the step sizes of the mean maintenance interval between the runs can be seen in Figure 2 (left). With decreasing parameter step size, the runtime until convergence to steady state is reduced exponentially. The user saves Proxels in almost every run, compared to the $8,277,630$ that were needed for the initial run at $mean = 170$. Assuming the same runtime as for the initial run for all replications, the user saves about half of the computation cost using the described method.

4 Summary and Outlook

This paper introduced an approach that uses a state space-based simulation method for sensitivity analysis and optimization of discrete stochastic models. To save computation time, successive runs are performed starting each one with the steady state result of the previous one, rather than with a arbitrary initial system state. Since the parameter sets of successive runs in optimization and sensitivity analysis are close, so should also be the system steady states. Using discrete event simulation, one would not be able to reuse old results in new simulation experiments, having to rerun replications for every parameter set. The described approach can be used to deterministically analyze and tune small stiff simulation models that would be very expensive when using discrete event simulation.
In the future, the current implementation should be extended to enable the user to vary more than one distribution parameter at a time, which is not a general limitation. More difficult, but also desirable would be the possibility to vary other system parameters, such as the reorder threshold of the inventory system, but since these parameters might influence the discrete state space of the model, this is more difficult and an area of further research. System optimization might be sped up by starting with a looser termination criterion in the first runs, allowing faster convergence, and increasing the accuracy, as the runs results converge around the optimum.

References

[HC91] Y.C. Ho and X.R. Cao. *Perturbation Analysis of Discrete Event Dynamic Systems*. Kluwer Academic Publishers, 1991.

[Hor02] G. Horton. A new paradigm for the numerical simulation of stochastic petri nets with general firing times. In *Proceedings of the European Simulation Symposium 2002*. SCS European Publishing House, 2002.

[LK00] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 2000.

[LM05] S. Lazarova-Molnar. *The Proxel-Based Method: Formalisation, Analysis and Applications*. PhD thesis, Otto-von-Guericke-University Magdeburg, November 2005.