Application of Proxels to Queuing Simulation

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

Abstract

Queuing theory strives to provide analytical solutions to a number of queuing problems. Unfortunately, closed analytical expressions can not be derived for every type of queuing system. Discrete event simulation on the other hand can find solutions to any kind of queuing problem, but the results are stochastic and only specific to a single parameterization. Proxel simulation can bridge the gap between these two methods by providing accurate deterministic results of the performance measures of a wide range of queuing systems. Furthermore one can solve some stiff problems far more quickly than when using discrete event simulation. The paper presents the general approach and a tool implementing the Proxel-based simulation of queuing systems. Experiments validate the method and show the range of applicability.

1 Motivation

Queuing Theory is one of the oldest branches of simulation. Still today many communications problems are analyzed and solved using basic queuing theory mechanisms. The basic components of a queuing system are queues, servers and customers. The goal of the analysis of such a queuing system is finding analytical expressions for such performance measures as steady state queue length, throughput and utilization.
The more complex the models get, the more difficult it is to derive analytical expressions for the performance measures. For a number of queuing problems it is not even possible to get these analytical expressions. Most of these difficult problems involve general distributions for the arrival and service distributions or multiple server environments.
It would be desirable to have a tool that yields deterministic results for these models' performance, if no analytical expression is available. Discrete event simulation is an alternative to queuing analysis. Unfortunately, it has the drawback of being stochastic and only providing results for a specific parameterized problem, compared to the general solution of queuing theory. Therefore the results are not comparable and often not of use to queuing analysts.
Using Proxels, the simulation of queuing models becomes more attractive and useful to queuing analysts. Proxels still can only provide results for a specific queuing system, but the results are deterministic and can be obtained to an arbitrary accuracy. One can obtain quick and dirty results for new queuing problems, that have not been analyzed thoroughly yet, or increase the quality of results for problems that can not be solved analytically by common queuing theory.

*Fakultät für Informatik, Institut für Simulation and Graphik, Universitätsplatz 2, 39106 Magdeburg, Germany

Another advantage of Proxel simulation is that it computes the transient solution for performance measures, as well as the probabilities of the discrete system states at no extra computation cost. This enables the solution of stiff queuing problems, where only one or few system states are of interest, which occur rarely but have a great impact on system performance. The emptying or overflowing of a buffer for example should happen rarely, but can be disastrous for the system simulated.
The approach presented in this paper builds a bridge between the world of queuing theory and discrete stochastic model simulation. We think that a tool that can analyze single queuing models using Proxels can greatly help queuing analysts in their work. The tool performs the actual simulation, then some of the most common performance measures for queuing models are calculated, which can be plotted.
In the next section we summarize some of the basics of queuing theory and Proxel based simulation. The following section documents the implementation of the approach and the difficulties involved in it. The experiments section has two parts, a validation experiment and three benchmark experiments. The last section provides a summary and outlook.

2 State of the Art

2.1 Queuing Systems Analysis and Simulation

Queuing theory is the study of systems, that can be described by queues, servers and some kind of customers or jobs (both will be used synonymous). Kendall's notation describes a single queuing process as a 5-tuple $A/B/X/Y/Z$, where A is the interarrival distribution of the customers, B the service time distribution, X the number of parallel servers, Y the maximum number of customers allowed in the system and Z the queuing strategy. Only $A/B/X$ are required elements, Y is then per default ∞ and the queuing strategy Z is FIFO(FCFS). A thorough description of the possible symbols and their meaning in Kendalls Notation can be found in [GH98].
The most common measures that are obtained when analyzing a system with one single queue are the following: Server *utilization* ρ describes the fraction of time that the server is busy, or the mean fraction of active servers, in the case of multiple servers. *Throughput* λ describes the number of jobs, whose processing is completed in a single unit of time. *Queue length* Q is the number of jobs waiting in the queue at a given time. *Waiting time* W is the time that the jobs spend in the queue waiting to be served. The *Number of Jobs in the system* K at a given time as well as the probability of a given number of jobs in the system π_i are often needed. (see [BGdMT98] for more details)
Queuing theory now tries to find analytical expressions for these measures in relation to the given queuing model parameters. If no analytical expression can be found, numerical results, generating functions or Laplace transforms are obtained. But in some cases there is no way to find a general description for the desired measures. Then simulation is the only way to obtain results for the performance measures for a specific system specification.
Simulation however is not commonly used in queuing analysis, since its results are much less accurate and general. Simulation has two major drawbacks compared to closed analytical result. The simulation is performed for one specific system parameterization and the

result of one run is just another random variable, so that replications have to be performed. In [HK06] an approach is presented for the transient analysis of a queuing system. The $GI/G/1$ system is initially defined in discrete time and the transition equations for the discrete time points are derived. The state space of the system is inherently limited by a stability condition and only the relevant state transitions at arrival and service instances are considered. The approach seems promising, and according to the authors can be extended to a number of queuing problems. The example in the paper is only a small one, considering the two discretization points for the arrival and service distributions. It is not clear, how this can be extended efficiently to more discretized continuous distributions with multiple discretization points. The transition equations as well as the state space of the system can get quite complex when the problem gets larger, since all possible combinations of the arrival and service of a customer have to be considered.
The approach presented in this paper is similar to the idea presented in [HK06]. The main difference here is the use of an already established and well researched simulation algorithm, which enables us to build a more general tool that already addresses some of the problems of state space explosion and complexity. With the tool presented in this paper, we strive to relieve at least one of the drawbacks of common discrete event simulation. The described approach helps obtain deterministic results of arbitrary accuracy for a given queuing system specification.

2.2 Proxel-based Simulation

Our recently developed Proxel-based simulation method ([LM05, Hor02]) does not require replications and produces deterministic results with an arbitrary accuracy for a discrete stochastic simulation model. It is a state space-based simulation method, which does not require the explicit generation of the models state space beforehand, but does that on-the-fly. The discrete state space of a model is expanded by the discretized age information of the activated transitions, and is thereby transformed into a discrete-time Markov chain (DTMC). One state of the DTMC (called Proxel = Probability element) contains the discrete model state, the relevant age information, the current simulation time and the probability of that combination. The state-space expansion enables us to determine the transition probabilities into the next possible Markov chain states solely based on the information contained in such a Proxel, using the so called instantaneous rate function.
The method is very flexible in its definition of system state. Virtually anything can be contained in a data structure defining the discrete system state. The only problem is the state space explosion resulting from the expansion of an already large discrete system state space. This can be met with a number of methods, but it confines the method to the analysis of small simulation models. Therefore we chose to limit this tool to the analysis of systems with a single queue.

3 Implementation

The Proxel algorithm can be applied to any kind of discrete event system, such as a queuing system. It starts from an initial system state and determines the possible next states and the

probabilities of the transitions into these states within the discretization time step.
The discrete state space of a queuing system is theoretically unlimited, if the specification does not contain a specified calling source size or system capacity. Stable queuing systems that can serve jobs faster than they arrive, have a steady state solution. Only these systems are considered further here. Besides the number of customers in the queue, the discrete state of the system is made up of the occupation state of the server(s), increasing the state space by a factor of 2^m (m being the number of servers in the system). The Proxel algorithm expands a discrete state by the ages of the non-Markovian distributions, which can lead to large DTMCs. A non-Markovian distribution in this paper is any stochastic distribution that does not have a constant firing rate over time, and one needs more than one Markov chain state to represent it accurately. Basically anything but the Exponential distribution (Geometric in the discrete case) is considered non-Markovian.
The challenge of the implementation is to define the system states in a way that can be stored, retrieved and searched efficiently. For queuing systems we decided to store the system state parameters directly as components of the Proxel data structure, so that a Proxel now contains the following elements

$$P_x = (q, \vec{s}, \tau_q, \vec{\tau_s}, p)$$

The elements meaning the following: q the number of jobs in the queue, $\vec{s}$ the occupation status of the servers, τ_q the age of the arrival process, $\vec{\tau_s}$ the ages of the different service processes and p the probability of the combination. If specified, the remaining number of jobs in the calling source is also encoded. The simulation time is not explicitly included, because only the Proxels of one time step are held in the data structure.
For the current implementation, the jobs in the queue do not have attributes, therefore queuing strategies cannot be modeled yet, but are a subject of future research. The program was structured modularly, so that an extension to attributed jobs is easily possible. With the given Proxel structure, the Proxel algorithm works as follows: The initial Proxel is created and for every simulation time step the following loop is performed.

```
01 FOR every Proxel px
02   p_arr = P(arrival);
03   FOR EACH occupied server s_i
04     p_serv_i = P(service i is finished)
05   ENDFOR
06   p_stay = 1-(sum(p_serv_i)+p_arr)
07   normalize_probabilities();

08   IF(p_arr > 0) create_arrival_proxel();
09   FOR EACH occupied server s_i
10     IF(p_serv_i > 0) create_service_finished_proxel();
11   ENDFOR
12   IF(p_stay > 0) create_stay_proxel();
13 ENDFOR
```

The first part (`02-07`) calculates the probabilities of possible state changes. The probability of an arrival (`p_arr`) can be deduced from the age of the arrival process (for how long has there not been an arrival) and the instantaneous rate function of the arrival distribution itself. The probability of any of the servers being finished can be calculated analogously (`p_serv_i`). If the sum of the probabilities of the state changes is smaller than 1, there is a probability of staying in this discrete system state (`p_stay`), which has to be considered. The second part of the loop (`08-12`) actually generates the Proxels representing the next system states for the next simulation time step. This step is encapsulated in functions that modify the state variables according to the event. The functions add customers to the queue and take them out of service and also check the conditional event of a server being empty and customers waiting in the queue. Therefore this does not have to be considered as a separate immediate state change.
After the simulation of the model, performance measures have to be calculated to extract the relevant information from the simulation result.

3.1 Calculating Performance Measures

During the whole Proxel simulation the probabilities for the discrete system states are logged in a two-dimensional array that contains the probabilities of each system state at each point in time. Due to the coding of the system state into an integer the array is only sparsely populated, a better data structure (hash or tree) for storing these results is again subject of future research. As the state definition will grow much more differentiated when adding job attributes, a better storing of results is essential when adding them.
The `create_service_finished_proxel` function also logs the finished services in a separate array for each time step by adding the probability of the service finished event to the throughput for this time step $throughput[k]$. This works analog to an impulse reward ([WHHE05]). The other performance measures are comparable to rate rewards and can therefore be calculated as a post processing step as described in the following. Δt denotes the simulation time step, m the total number of servers in the system, k the current time step, $tmax$ the maximum simulation time and $kmax = tmax \div \Delta t$ the maximum number of simulation time steps.
Transient server utilization $\rho[k]$ and average utilization $\bar{\rho}$:

$$\rho[k] = \sum_{states\ s} busy\_servers(s)/m * P(s)[k] \quad (1)$$

$$\bar{\rho} = \sum_{k} \rho[k]/timesteps \quad (2)$$

Transient throughput of the system $\lambda[k]$, throughput average $\bar{\lambda}$ and total λ:

$$\lambda[k] = throughput[k]/\Delta t \quad (3)$$

$$\lambda = \sum_{k} throughput[k] \quad (4)$$

$$\bar{\lambda} \quad = \quad \lambda / tmax \tag{5}$$

Transient queue length $Q[k]$ and average queue length $\bar{Q}$:

$$Q[k] \quad = \quad \sum_{states\ s} queued\_jobs(s) * P(s)[k] \tag{6}$$

$$\bar{Q} \quad = \quad \sum_{k} Q[k] / kmax \tag{7}$$

Average job waiting time $\bar{W}$:

$$\bar{W} \quad = \quad \sum_{k} Q[k] * \Delta t / \lambda \tag{8}$$

Transient number of jobs in the system $K[k]$ and average number of jobs in the system $\bar{K}$:

$$K[k] \quad = \quad \sum_{states\ s} (busy\_servers(s) + queued\_jobs(s)) * P(s)[k] \tag{9}$$

$$\bar{K} \quad = \quad \sum_{k} K[k] / kmax \tag{10}$$

Transient probability for a defined number of jobs π_i:

$$\pi_i[k] \quad = \quad P(s | job\_number(s) = i)[k] \tag{11}$$

Assuming the simulation has converged to steady state, the values of the transient measures in the last simulation time step $kmax$ are the steady state measures: e.g. steady state utilization $\rho = \rho[kmax]$. The average measures are interesting for queuing systems with a limited calling source, since the steady state of these systems is empty.
In order to be able to easily parameterize and analyze a queuing system, a user interface was designed, which is described in the next section.

3.2 Interface

The graphical user interface (see Figure 1) enables the user to specify the queuing system, execute the simulation, plot the transient results, and read the scalar performance measures. The upper part of the GUI allows input of the arrival and service process as a known general probability distribution. The process definition through a distribution file or a phase-type distribution can easily be added as choices. The user then has to specify the number of servers in the system. Optionally a system capacity and a calling source size can also be specified. The simulation parameters time step Δt and maximum simulation time $tmax$ can also be modified. When the system is fully specified, the user can start the simulation.

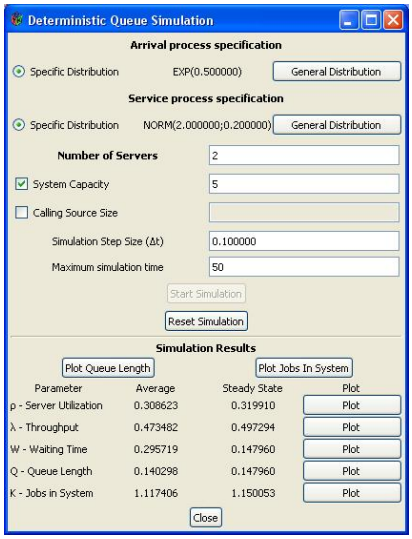

Figure 1: User interface for queue simulation using Proxels

The lower part of the GUI shows the simulation output. The values of five scalar performance measures are represented directly after the simulation is finished as average and final values, (steady state, if the simulation has converged). The plot buttons invoke graphs, that represent the transient values of these measures. The two upper buttons display plots of the transient probabilities of the queue lengths q_i and the job numbers π_i.
The next section shows some experiments performed using the approach and tool described in this section.

4 Experiments

This section contains the results of four different experiments. The first experiment is done using a trivial queuing system and intends to validate the developed tool against the analytical results obtained from literature.
The other experiments take queuing models that do not yield analytical expressions for the performance measures, and demonstrate the abilities of the tool to find solutions for these kinds of problems.

4.1 Validation Experiment M/M/1

The basic queuing system M/M/1 has a Markovian arrival and service process and a single server. The parameters of the example considered here are the arrival rate $\mu_1 = 1$ and service rate $\mu_2 = 2$. The analytical results for the steady state performance measures

calculated through the formulas taken from literature ([BGdMT98]) are listed below.

$$\begin{aligned}
\rho &:= \mu_1/\mu_2 = \frac{1}{2} \\
\lambda &:= m * \rho * \mu_1 = \frac{1}{2} \\
\bar{W} &:= \frac{\rho/\mu_1}{1-\rho} = 1 \\
\bar{Q} &:= \frac{\rho^2}{1-\rho} = \frac{1}{2} \\
\bar{K} &:= \frac{\rho}{1-\rho} = 1 \\
\pi_0 &:= 1-\rho = \frac{1}{2} \\
\pi_i &:= (1-\rho)\rho^i = \frac{1}{2^{i+1}}
\end{aligned}$$

The Proxel simulation is performed using a time step of $\Delta t = 0.1$ and to a maximum simulation time of 100, which ensures the convergence of the simulation. This resulted in 54 Proxels per time step, corresponding to a DTMC with 54 states. The simulation needed under 10 seconds to complete on a laptop with a $2.66GHz\ \ Pentium4$ processor with and $512MB$ RAM. The steady state results of the scalar performance measures are the following $\rho = \frac{1}{2}; \lambda = \frac{1}{2}; \bar{W} = 1; \bar{Q} = \frac{1}{2}; \bar{K} = 1$. The transient and steady state results for the probabilities of different numbers of jobs in system can be seen in Figure 2.
The results obtained from the simulation are as close to the analytical ones as can be expected of numerical results. This shows the validity of the tool and the described formulas for the performance measures. There is no a difference due to the discretization of the continuous distributions, which strengthens the confidence in the implemented method further.

4.2 Benchmark Experiment 1 $G/G/1$

In this experiment a queuing system is simulated, that does not have an analytical solution in queuing theory. As specific example the parameterization $N(2; 0.4)/N(1.5; 0.2)/1$ is used. A single server system with normally distributed interarrival and service times. The Proxel simulation is performed using a time step of $\Delta t = 0.1$ and to a maximum simulation time of 50. The computation time needed was less than one second, and the resulting DTMC contained 1260 states.
The steady state results of the scalar performance measures are as follows $\rho = 0.75; \lambda = 0.5; \bar{W} = 0.02; \bar{Q} = 0.02; \bar{K} = 0.77;$
Figure 3 left shows the transient probabilities of different numbers of jobs in system over the course of the simulation time. Since the the mean service time is smaller than the mean interarrival time, the system is stable and the probabilities of the system containing 0 or 1 jobs are largest. Due to the non-Markovian service and arrival distribution, the result is not smooth, but the Proxel method can capture the expected values in every simulation time

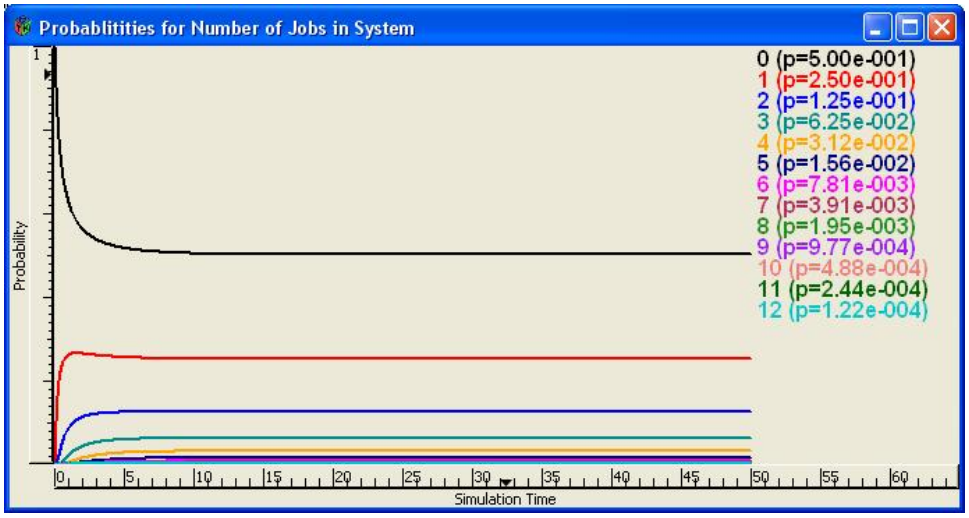

Figure 2: Validation experiment: Transient Probabilities for number of jobs in system

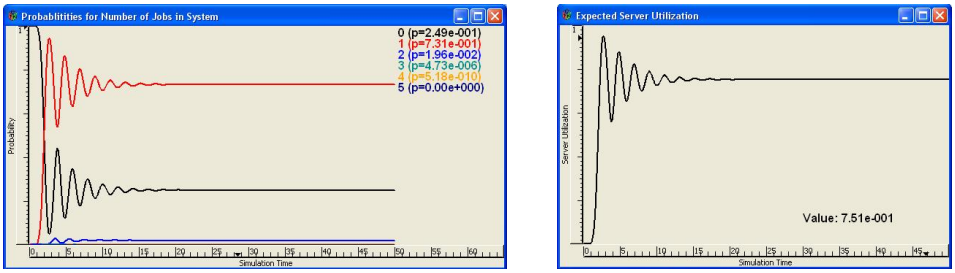

Figure 3: Benchmark experiment 1: Transient Probabilities for number of jobs in system (left) and server utilization (right)

step. The server utilization (Figure 3 right) exhibits a similar transient phase before it goes into steady state at $\rho = 1.5/2 = 0.75$.

The oscillation in the two graphs is due to the aging of the arrival (service) process and the resulting time-dependent arrival (service) rate. With the given arrival distribution ($N(2; 0.4)$) the first customer cannot arrive until about one time unit has expired, and only after another time unit this first arriving customers can possibly finish service. This causes the initially large oscillation of the transient state probabilities and performance measures. The oscillation fades out when as simulation time the state space reaches its final extent, and more and more states contribute to the average value. This behavior is similar to the transient period at the beginning of a discrete event simulation run.

4.3 Benchmark Experiment 2 $G/M/c/K$

In this experiment a queuing system is simulated, that does not have an analytical solution in queuing theory. The specific example is the following $N(1.2; 0.1)/Exp(0.5)/2/6$. The arrival process has normally distributed intervals, the service is Markovian and the system has 2 servers, as well as a capacity of 6 jobs. The Proxel simulation is performed using a time step of $\Delta t = 0.1$ and to a maximum simulation time of 50. The computation time needed, was less than one second, and the resulting DTMC contained 167 states.
The steady state results of the scalar performance measures are as follows $\rho = 0.68; \lambda = 0.80; \bar{W} = 0.81; \bar{Q} = 0.81; \bar{K} = 2.43;$

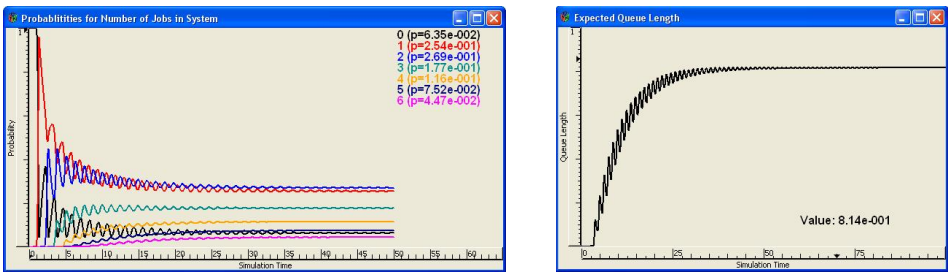

Figure 4: Benchmark experiment 2: Transient Probabilities for number of jobs in system (left) and queue length (right)

Figure 4 left shows the transient probabilities of different numbers of jobs in system over the course of the simulation time. As specified, the maximum number of jobs in the system is 6. Due to the non-Markovian arrival distribution, the result is not smooth, but the Proxel method can capture the expected values in every time step. The queue length (Figure 4 right) exhibits a similar transient phase before it goes into steady state at 2.43. The oscillation, representing the transient phase of the simulation has the same reasons as in the previous section.

4.4 Benchmark Experiment 3 $M/G/c/K$

In this experiment a queuing system is simulated, that could represent a small call center. The parameterization is the following $Exp(1.25)/N(1; 0.2)/2/17$. The callers arrive through a Poisson process with an arrival rate of $1.25/minute$, the service is normally distributed with a mean of 1 minute and the system has 2 servers, as well as a queue capacity of 15 jobs (system capacity 17). The interesting state for the manager is the improbable case that the queue is full, which means, that an incoming call could be rejected, and a potential customer is lost.
The Proxel simulation of the system with a discretization time step of $\Delta t = 0.1$ and a maximum simulation time of 50 took 5.9 seconds to compute, and the resulting DTMC contained 5563 states. The steady state results of the scalar performance measures are as

follows $\rho = 0.475; \lambda = 1.25; \bar{W} = 0.394; \bar{Q} = 0.394; \bar{K} = 1.66;$. The probability of the system being full and consequently the queue being full was determined at $\pi_{17} = 2.16e{-}7$.
A discrete event simulation experiment was done using the simulation tool Simplex3 [Sch00]. 1000 independent simulation runs were performed, starting with an empty system and running to a simulation time of $1,000,000$ minutes. The stiffness of the model made fewer or shorter simulation runs impossible, since then the queue never filled up. The simulation tracked the total time that the queue was filled. The fraction of total simulation time that the queue was filled is equivalent to the total probability of the queue being full.
The experiment took 15 minutes to complete. The queue being full happened in total only 7 times in total for about 1.54 minutes during these simulation runs. The confidence interval for the probability $(-8.52e-10; 3.948e-9)$ was computed using an $\alpha = 0.01$. The number of 1000 replications is not nearly enough for the rareness of the event, consequently the confidence interval is of no practical use since it includes an negative probability and underestimates the probability of the queue filling up dramatically. Furthermore is the computation time much larger than that of the Proxel simulation.
The filling up of the queue is a rare event, which forces one to perform many replications of the discrete event simulation. The Proxel simulation does not suffer from that problem. This experiment shows that using Proxels one can solve simulation problems, that do not have an analytical solution, and that cannot be feasibly solved using discrete event simulation.
The validation experiment shows that the developed algorithm and performance measure formulas are correct, and makes further experiments useful. The benchmark experiments show the applicability to other queuing problems and the kind of results that can be obtained.

5 Summary and Outlook

This paper describes how Proxels can be applied to the simulation and analysis of queuing problems. This approach can be seen as an alternative to discrete event simulation of queuing models, since it provides deterministic transient and steady state results for the performance measures. It can analyze problems that one can not solve analytically and therefore provides a help to queuing analysts. Some stiff problems can be solved far more quickly than when using discrete event simulation.
We first described the methods implementation and some problems involved in it. A validation experiment compared the results with the analytical results from queuing theory. The benchmark experiments showed the results that can be obtained for not analytically solvable problem, and that the method presented is faster that discrete event simulation on some problems.
Still, not all possible elements of queuing problems have been implemented. The current implementation does not support attributed customers, and therefore most queuing disciplines are not included yet. But the implementation is modular, and can be extended to include that. Attributed jobs, different queuing disciplines and more advanced arrival processes, such as MMPP are a subject of future work. However, adding attributes to the jobs will significantly increase the state space of the resulting model. Therefore this approach

will always be limited to very few attributes per job, or attributes with small value sets. The performance of the method will also largely depend on the implemented storage and retrieval strategies, because these are the bottlenecks of the Proxel based method. Another feature that is interesting when looking at the transient behavior of some problems is the initial state of the model. This will be the next step in improving the current tool.
When the tool is expanded to arrival processes that are specified by a discretized statistical distribution, one might use the output process of one queuing system as the input process for a next queuing system, and thereby enable the analysis of queuing networks. This however is only speculation and another possible subject of future research.
All in all the current tool presents a good start toward a general simulator for single queuing models.

References

[BGdMT98] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, New York, 1998.

[GH98] Donald Gross and Carl M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 3rd edition, 1998.

[HK06] Gerhard Hasslinger and Sebastian Kempken. Transient analysis of a single server system in a compact state space. In *Proceedings of 13th International Conference on Analytical and Stochastic Modelling Techniques and Applications*. European Council for Modelling and Simulation, May 2006.

[Hor02] Graham 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.

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

[Sch00] Bernd Schmidt. *Einführung in die Simulation mit Simplex3*. SCS European Publishing House, Erlangen, 2000.

[WHHE05] F. Wickborn, G. Horton, S. Heller, and F. Engelhard. A general-purpose proxel simulator for an industrial software tool. In *Proceedings of 18th Symposium Simulationstechnik (ASIM 2005), Erlangen, Germany*. SCS European Publishing House, September 2005.