

COMBINING PROXELS AND DISCRETE PHASES

Claudia Isensee, Sanja Lazarova-Molnar, Graham Horton

Otto-von-Guericke University of Magdeburg
Department of Computer Science
Universitätsplatz 2, 39106 Magdeburg, Germany
claudia{sanja/graham}@sim-md.de

ABSTRACT

The analysis of discrete stochastic models such as generally distributed stochastic Petri nets can be done by using state space-based methods. They describe the behavior of a model by a Markov chain that can be solved mathematically. Formerly this approach often had to be discarded as unfeasible due to high memory and runtime costs.

The recently developed Proxel-based algorithm is a state space-based simulation method, and has already performed well in several application fields. Experiments suggest, that the selective use of discrete phase approximations could further improve the method, because they can often represent infinite support distribution functions with considerably fewer Markov chain states than proxels. By replacing certain on-the-fly proxel approximations by predetermined phase-type approximations, the total runtime and memory requirement of the simulation method could be drastically reduced for some test models. An efficient algorithm for the approximation of discrete phase-type distributions based on common optimization methods was recently introduced.

The formal inclusion of discrete phases into the proxel paradigm is another step toward a practically usable state space-based simulation method. Our hope is that such a combination of both approaches will lead to a competitive simulation algorithm.

1. INTRODUCTION

This paper will formally extend the proxel-based simulation algorithm by including discrete phase approximations. The current section will motivate the work and show previous developments. The next sections will describe the proxel-based algorithm and discrete phases. Then the changes in the structure of the probability element (proxel) and the algorithm will be described. Some experiments comparing the approaches, and a conclusion form the last part of the paper.

1.1. Motivation

Discrete stochastic models are usually simulated using discrete event simulation. When rare events are involved or a high accuracy for the results is required, this can become quite expensive due to the large number of replications necessary.

State space-based methods are deterministic and do not require replications; but even though they work well in theory, they are not used widely. This is partly due to the fact that the state space of a model can grow exponentially with the order of the representations of the non-Markovian distributions, a behavior called state-space explosion.

The recently developed proxel-based simulation algorithm tackles another drawback of state space-based methods. It does not require the solution of partial differential equations or the complete creation of the state space beforehand, but it generates the model states on-the-fly, and calculates their probabilities by using the instantaneous rate function (IRF). The method was successfully applied to problems involving rare events. Because of its deterministic behavior, the accuracy of the results is only dependent on the chosen time step and not on the number of replications.

The proxel-based method is based on the method of supplementary variables, and requires a separate variable for each concurrently enabled transition and each disabled transition with age memory policy. Because one variable value is needed for every time step of a distribution (see Section 2.1), the number of possible combinations rises steeply, as the size of the model increases.

Discrete phases (DPH) can often represent infinite support distribution functions with sufficient accuracy needing considerably less Markov chain states, than proxels can. But, finite support distribution functions such as Uniform, require a large number of phases to be approximated accurately with a small time step, which would result in expensive approximation runs. If all model activities were to be approximated by DPH, these would have to be calculated in advance, and the overall time required to approximate the distributions and run the simulation would be larger than with proxels. In addition, because DPH are calculated for one specific distribution, they cannot easily grasp for example model behavior that is dependent on dynamic values, such as number of failures of a particular part. This can however be done with proxels [1].

Therefore, by selectively replacing model activities distributions with discrete phase approximations when using the proxel-based algorithm, the size of the state space, and the overall runtime and memory requirement can decrease considerably, without diminishing the power of the proxel-based method. To be able to use DPH approximations instead of proxel approximations, the definition of the probability element proxel has to be extended, and the proxel-based simulation algorithm modified. To show that the combination of the two paradigms is possible, it is sufficient to include discrete phases into the original algorithm. All later extensions have the same general structure.

1.2. Previous Work

The proxel-based method was first introduced in [2]. The method has been successfully applied to fault-tree analysis [3], stochastic Petri nets (SPN) [4] and the analysis of project schedules [5]. Besides an extension for immediate transitions in SPN's [6], recently a new model description framework was developed [1] that

exploits the power of the proxel-based method beyond SPN's or fault trees.

Phase-type distributions were first described in detail by Neuts in [7] and a new approach to the efficient approximation of discrete phase-type distributions has been introduced [8]. In contrast to former methods [9] it is based on common optimization methods, does not restrict the distribution function to be approximated and was developed and tested with a focus not only on the result quality but also on the performance of the algorithm.

2. PROXELS

2.1. The Proxel-based Simulation Algorithm

The proxel-based method for the analysis of discrete stochastic systems is based on the method of supplementary variables. A detailed description of the original proxel-based simulation algorithm can be found in [2]. The method creates and analyzes the state space of a model on-the-fly. Non-Markovian distribution functions are approximated using their instantaneous rate functions (IRF). In contrast to the usual supplementary variable-based methods, proxels do not require differential equations, but are intuitive and purely algorithmic. For some classes of models the method shows significant runtime and accuracy improvements over discrete event simulation [10].

The idea behind the method is, to start the model off in its initial state and then to track every possible path, which the system can take in conjunction with the probability, that this path will be taken. Only the chosen simulation step size influences the accuracy of the results, because this approach is deterministic.

2.2. The Proxel Approximation Scheme

Proxels as a state space-based algorithm are a way of solving a discrete-time Markov chain. Therefore, non-Markovian distribution functions need to be approximated. The approximation consists of one Markov chain state for every time step of the distribution and has the structure shown in Figure 1. Here Δ stands for the simulation step size used.

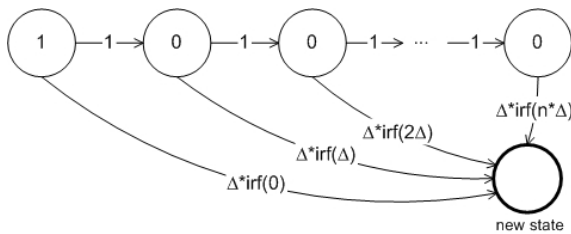


Figure 1: Proxel Approximation Scheme.

3. DISCRETE PHASE-TYPE DISTRIBUTIONS

Discrete phase-type distributions are another way of approximating a non-Markovian distribution function through a Markov chain. The time-to-absorption in such a chain tries to mimic the behavior of the original distribution function. An introduction to the topic can be found in [7] and some advantages of the discrete type over the continuous are mentioned in [8].

3.1. Discrete Phase Approximation Scheme

In contrast to a proxel approximation only a limited number of phases is used for the approximation of a distribution function. The resulting Markov chain has the structure shown in Figure 2. The structure has been adapted from the one presented in [9]. The p_i are the one step transition probabilities from one state to the next and the a_i represent the initial probabilities of the phase representation.

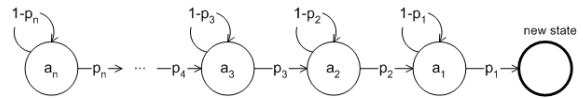


Figure 2: DPH Approximation Scheme.

3.2. Approximation Algorithm

The approximation algorithm described in [8] is based on common optimization methods. It tries to minimize the distance between the discretized original distribution function and the output of the DPH. In the experiments described in that paper, the algorithm performs particularly well for rather smooth infinite support distribution functions such as Weibull and Lognormal. These could often be approximated with very few phases in only a couple of seconds. Finite support distribution functions on the other hand, such as Uniform, needed considerably more phases for close fits, and the runtime of the algorithm increased accordingly.

This suggests, that proxels should be used for the approximation of finite support distribution functions, and functions with dynamic dependencies, because the phase approximations have to be calculated beforehand. Phases in the other hand should be used for the representation of rather smooth and infinite support distribution functions. More detailed recommendations are a subject of future research.

4. FORMALIZATION

4.1. The Probability Element

The proxels as the basic units of the proxel-based simulation algorithm completely define the state of the system at discrete points in time. A proxel contains the probability p that the simulation model has reached a specific state S at a given point in time via a specific path R . We assume for now, that the system to be simulated is modeled by a stochastic Petri net, as in [2].

When using supplementary variables, the discrete state of the model dS (the marking of the stochastic Petri net), has to be supplemented by a vector τ containing the ages of the active transitions, i.e. the number of time steps, that a transition has been enabled, but has not fired yet. These variables are also needed for transitions with memory policy Race Age.

When including discrete phases, we need to extend the definition of a state even further. We introduce a new vector ϕ , which contains one element for each enabled (or race age) phase-type transition. The value i of the vector element indicates, in which phase of the phase-type distribution this particular proxel is situated, where $i = n$ means the starting phase and $i = 1$ the last phase before entering the next discrete state, analogous to Figure

2. The state of the system S is thus completely described by the discrete state of the model dS , the vector τ and the vector ϕ .

$$P = (S, R, p) = ((dS, \tau, \phi), R, p) \quad (1)$$

The new proxel P now has the structure shown in (1). Where S is the state that is composed of the discrete state of the model dS , the age intensity vector τ and the phase index vector ϕ , R the route that this state was reached by and p the probability of that combination. From now on in this paper the expression proxel always refers to this new kind of probability element, if not stated otherwise.

4.2. Modified Proxel Algorithm

In this section we will describe the modified algorithm, that comes with the extended probability element. As a reference we take the original proxel algorithm specified in [2], and only modify the necessary parts. If phase-type transitions can be conceptually included into the structure of the basic algorithm described in that paper, the inclusion into the more advanced versions will be also possible.

The algorithm is started with an initial proxel, representing the initial discrete state of the model with the probability one. It then determines all of the possible successor states, computes the probability with which the model will transition into those within a discrete time step Δ , and creates new proxels representing these transitions. This process is repeated with all created proxels recursively, until the specified maximum simulation time is reached. As a new feature, the system may also transit into the next phase of a phase-type transition without changing the discrete state of the model.

Successor states are determined by the enabled transitions. The transition probabilities are taken from the phase-type distributions specification, or computed via the instantaneous rate function, depending on the current transition. The proxels generated over time can be interpreted as a tree, where the initial proxel forms the root and each layer contains the proxels generated at one discrete point in simulation time.

There are two major modifications of the algorithm necessary, in order to be able to treat discrete phase-type distributions. Firstly, a preprocessing step before adding a proxel to the tree is necessary if the proxel generated takes the model into a new discrete state. It is checked whether in the new state any formerly not enabled phase-type transitions are active, and then the proxel is split up into n phases, where the probability is divided according to the initial probability vector of the phase-type distribution. If more than one DPH becomes enabled, this process is repeated recursively. Secondly, the transition from one phase to the next within a DPH is not the firing of a transition, and does not change the discrete state of the model, so only the according ϕ vector element and the τ vector are modified.

4.2.1. Implementation

A proxel $P = (m, \tau, \phi, t, p)$ is made up of 5 elements: the discrete state m (marking of the Petri net), the age intensity vector of the enabled or race age transitions τ , the phase index vector containing the information about the current phase ϕ , the simulation time t and the probability of that combination p . The route is omitted in this implementation, because by definition it does not influence

the future of the system and is partially implicitly included in the configuration of the vectors τ and ϕ .

The generated proxels are temporarily stored in a queue Q , and processed one by one. $P.x$ denotes the element x of proxel P . The variable $\pi_m(t)$ is used to store the total probability to be in state (marking) m at time t . $tmax$ is the maximum simulation time, Δ is the discrete simulation step size, and k denotes the number of discrete time steps. \emptyset denotes the empty set, T a transition in the Petri net and assignments are denoted by a \leftarrow symbol.

A Transition T also has attributes: $T.ph$ is TRUE, if the transition is of phase-type. $T.id$ is the index of the ϕ vector (τ vector), that contains the phase (age intensity) information for the transition. $T.a[i]$ and $T.p[i]$ denote the elements of the initial probability and transition probability vectors of the phase-type transition T .

The initial proxel is not necessarily uniquely defined. The initial activation of a phase-type transition might result in several initial proxels representing one discrete state. However, the first proxel to be preprocessed is $P_0 = (m_0, \vec{0}, \vec{0}, 0, 1)$. The algorithm terminates, when the queue that stores the proxels is empty.

The modified proxel-based simulation algorithm is the following.

```

1:  $Q \leftarrow \emptyset$ 
2: pre_addproxel( $m_0, \vec{0}, \vec{0}, 0, 1$ )
3: WHILE  $Q \neq \emptyset$ 
4:    $P \leftarrow$  getproxel()
5:    $\pi_{P.m}(P.t) \leftarrow \pi_{P.m}(P.t) + P.p$ 
6:   IF ( $P.t < tmax$ )
7:     addproxel( $P.m, update(P.\tau, P.\phi, P.m, \emptyset), P.t + 1,$ 
        $P.p * (1 - \Delta * \sum_{-T.ph} h_T(\tau) - \sum_{T.ph} T.p[P.\phi_{T.id}])$ )
8:      $\forall T$ : IF(enabled( $P.m, T$ ))
9:       IF( $T.ph$  AND  $P.\phi_{T.id} > 1$ )
10:        addproxel( $P.m, update(P.\tau, P.\phi, P.m, T),$ 
           $P.t + 1, P.p * T.p[P.\phi_{T.id}]$ )
11:      ELSEIF( $T.ph$  AND  $P.\phi_{T.id} = 1$ )
12:        pre_addproxel(succ( $P.m, T$ ),
          update( $P.\tau, P.\phi, P.m, T$ ),
           $P.t + 1, P.p * T.p[1]$ )
13:      ELSE
14:        pre_addproxel(succ( $P.m, T$ ),
          update( $P.\tau, P.\phi, P.m, T$ ),
           $P.t + 1, P.p * \Delta * h_T(\tau)$ )

```

The following functions are used in the algorithm:

- succ(m, T) returns the marking reached from marking m when firing transition T .
- enabled(m, T) returns TRUE if transition T is enabled in marking m .
- pre_addproxel(P) if phase transitions have become enabled, splits up proxel P into phases and inserts them unto queue Q .
- addproxel(P) inserts proxel P into queue Q .
- getproxel() deletes a proxel from Q and returns its value.

update(τ, ϕ, m, T) updates the enabling time vector τ and phase index vector ϕ when transition T fires, or is advanced, in marking m .

memory(T) returns memory policy of transition T (i.e. ENABLE or AGE).

4.2.2. Explanation

First we will comment the algorithm line by line:

- Line 1: The proxel queue Q is initialized to the empty set.
- Line 2: The initial proxel P_0 representing the initial state of the model is inserted into the queue, possibly splitting it into phases for activated phase-type transitions.
- Line 3: Loop until the proxel queue is empty.
- Line 4: Get the next proxel P from the queue.
- Line 5: Add the probability of the current proxel $P.p$ to the solution.
- Line 6: Continue only if maximum simulation time $tmax$ has not yet been reached.
- Line 7: Add a new proxel, representing the case, that the SPN remains in the marking $P.m$, and also in the current phase of any phase-type transitions active.
- Line 8: Consider all transitions T that can fire in the marking of the current proxel $P.m$.
- Line 9: If the transition is of phase-type, and not in the last phase yet,
- Line 10: add a new proxel, representing the case, that the SPN remains in the marking $P.m$, and the phase of the current phase transition is advanced by one.
- Line 11: If the current transition is of phase-type, and in the last phase,
- Line 12: add a new proxel to the queue that represents the next marking of the Petri net after the firing of T , possibly splitting it into phases for activated phase-type transitions.
- Line 13: Otherwise (If the current transition is not of phase-type),
- Line 14: add a new proxel to the queue that represents the next marking of the Petri net after the firing of T , possibly splitting it into phases for activated phase-type transitions.

The probability to remain in a discrete state of the model (marking of the SPN) during the time period t to $t + \Delta$ is equal to one minus the sum of the probabilities to go to another state during this time period.

4.2.3. Memory policies

The function update() modifies the j -th element of the vector τ or ϕ of the transition T , according to the behavior and type of the j -th transition as follows:

```
update( $\tau, \phi, m, T$ )
FOR  $j = 1$  TO  $n_\tau(n_\phi)$  DO
  IF ( $j = T.id$ )
```

```

 $\tau_j \leftarrow 0$    ( $\phi_j \leftarrow \phi_j - 1$ )
ELSEIF ( $T.ph$  AND  $\phi_{T.id} > 1$  AND enabled( $m, T_j$ ))
 $\tau_j \leftarrow \tau_j + 1$    ( $\phi_j \leftarrow \phi_j$ )
ELSEIF (enabled( $m, T_j$ )) AND enabled(succ( $m, T$ ),  $T_j$ ))
 $\tau_j \leftarrow \tau_j + 1$    ( $\phi_j \leftarrow \phi_j$ )
ELSEIF (memory( $T_j$ ) = AGE)
 $\tau_j \leftarrow \tau_j$    ( $\phi_j \leftarrow \phi_j$ )
ELSE
 $\tau_j \leftarrow 0$    ( $\phi_j \leftarrow 0$ )
```

The function update() considers all $j = 1 \dots n_\tau$ supplementary variables and all $j = 1 \dots n_\phi$ phase variables, which are active in this marking, and modifies their values according to certain rules.

The rules for non phase-type transitions are the following: If the current transition is the one that fires, the age variable is reset to zero. If a phase-type transition only advances one phase, and does not change the marking of the SPN, or the transition is enabled in the current and the next state, the age variable is advanced by one. If the j -th transition is not enabled, the discrete state of the model changes, and the transition has memory policy AGE, then the value of the age variable is kept. Otherwise the age variable is reset to zero.

The rules for phase-type transitions are the following: If the current transition is the one that fires, the phase variable is decremented by one. If another phase-type transition only advances one phase, and does not change the marking of the SPN, or if the transition is enabled in the current and the next marking, the phase variable is kept. If the j -th transition is not enabled, the discrete state of the model changes, and the transition has memory policy AGE, then keep the value of the phase variable. Otherwise reset the phase variable to zero.

Supplementary variables, as well as phase variables can be mapped to different transitions in different markings, for example to save memory. This mapping must be computed and known beforehand.

4.3. Discussion

This section shows how the concept of discrete phase-type distributions can be included into the basic proxel algorithm. Obviously, the techniques for reducing the complexity of the algorithm and the memory requirement for the proxel storage that are mentioned in [2] also have to be applied to make this combination feasible. However, DPH and proxels can be combined without changing the basic structure of the algorithm. By leaving out phase-type transitions when specifying the model, the algorithm behaves exactly like an ordinary proxel algorithm. The necessary modifications can be adapted to the more advanced versions described in [6] or [1].

5. EXPERIMENTS

This section will show the differences between proxels and discrete phases regarding runtime and accuracy by comparing different experiments done with one simple stochastic Petri net. The tested Petri net is shown in Figure 3. It contains both finite and infinite support distribution functions, as well as one transition with age memory policy, and should therefore show the benefits of the combination of the two paradigms.

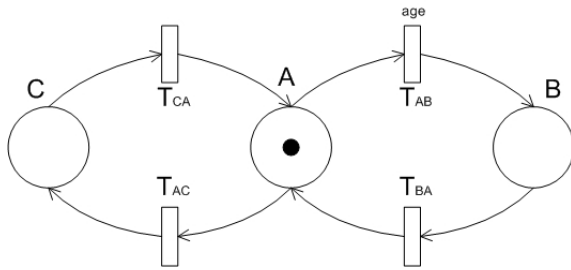


Figure 3: Test Petri net.

The transitions have the following distributions associated to them:

- $T_{AB} \sim Weibull(5, 5)$
- $T_{BA} \sim Normal(1, 0.2)$
- $T_{AC} \sim Uniform(1, 2)$
- $T_{CA} \sim Deterministic(1.2)$

The Petri net was simulated using a time step of $\Delta = 0.05$ and up to time 50. Three different kinds of experiments were conducted. In the first experiment *Prox*, which was also used as a reference, the Petri net was simulated using only proxels. In the experiment *Phas1 (Phas2, Phas3)* all four distributions were approximated through discrete phase-type distributions, using 4 (8,16) phases. In the last group of experiments *PnP1 (PnP2, PnP3)* the Weibull and Normal distributions were approximated using 4 (8,16) phases, and the Deterministic and Uniform ones were left as proxel approximations. The experiments showed differences in complexity, but also in the result quality, which will be explained in the next two sections.

5.1. Differences in Runtime and Memory

In Table 1 the runtime and memory complexity results of the experiments are summarized. The column 'Proxels' refers to the total number of proxels generated during the simulation, which is a measure for the complexity of the algorithm. The 'Total Time' is the sum of the actual runtime of the simulation algorithm (t-Sim) and the time needed to obtain discrete phase-type approximations for the different distributions (t-Approx).

	Proxels	Total Time	t-Sim	t-Approx
<i>Prox</i>	7,453,387	56.421	56.421	–
<i>Phas1</i>	35,854	0.514	0.172	0.342
<i>Phas2</i>	134,635	1.156	0.640	0.516
<i>Phas3</i>	515,008	33.953	4.328	29.625
<i>PnP1</i>	1,188,317	8.639	8.468	0.171
<i>PnP2</i>	1,829,102	13.984	13.703	0.281
<i>PnP3</i>	2,768,184	51.109	22.390	28.719

Table 1: Runtime and memory complexity experiment results when comparing proxels, phases and their combination

The table shows, that when using phase approximations with just a few phases, the simulation time itself is greatly reduced to about 1/100 compared to only proxel approximations, because the

total number of proxels generated shrinks when the order of the representation of the non-Markovian distributions is reduced from 100 or more to just 4. When the number of phases is doubled in the *Phas* experiments, the number of proxels generated quadruples, because in two of the three discrete states of the model, two age variables have to be remembered, which results in a two dimensional matrix-like state space. In the *PnP* experiments, the doubling of the phase number results in not even twice the number of proxels generated. The simulation time seems to exhibit a similar behavior, at least for the jump from 4 to 8 phases.

When using phase approximations of order 8 or more, the approximation time takes up a good portion of the total time, because the approximation of larger phase-type distributions naturally takes longer to calculate. An exception to that rule is the approximation by a chain, that has as many phases, as the discretized distributions has time steps. The fitting algorithm initializes the solution correctly, and the actual computation time is reduced to zero. This trivial case was however not considered, since using the same number of phases, as proxels would be needed, does not reduce the state space, and thus would not reduce the simulation time considerably.

5.2. Differences in Results

Besides the difference in complexity, there is also a difference in result quality, when comparing the experiments. This is due to the approximation error that a discrete phase-type distribution has in comparison to a proxel approximation, because the representation is only a simplified version of the real distribution. The largest error of almost one was produced, when trying to approximate the Deterministic distribution function using only 4 phases, the smallest one with the normal distribution and 16 phases, $1.16E - 05$. The error was smaller for larger phase numbers and the more smooth distribution functions.

The results of the *Prox* experiment were taken as a reference, because these represent the most accurate results, one can achieve with the same time step and maximum simulation time. The steady state solution for the three states (A, B and C) was accurately reproduced by almost all 6 other experiments. Only the two experiments using only 4 phases for the approximations slightly differed in their results, but only in the third digit.

The transient solution of the model however, differentiated quite well between the different experiments. Taking the characteristic behavior of the model shown in Figure 4 as a reference, it is obvious, that using only four phases for all Petri net transitions does not reproduce the behavior, but using four phases only for the two infinite support distribution functions results in a much more accurate picture (see Figure 5). When using 8-phase approximations for all transitions, the results are still much less accurate, then when using 8 phases only for the two selected distributions (Figure 6). Using 16 phases for the approximations further enhances the quality of the transient solution, where the combination of proxels and phases is again better than just phases (Figure 7).

5.3. Evaluation

In the last two sections the two criteria result quality and performance of the algorithm were evaluated separately. Combining both criteria leads to a more meaningful distinction between the tested options. The original proxel simulation is the most accurate, especially regarding the transient solution, but does also need

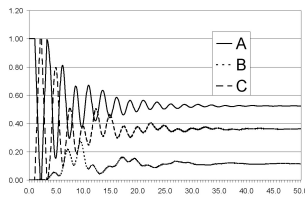


Figure 4: Prox - Transient solution using only proxels.

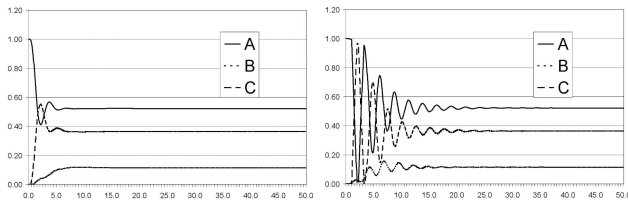


Figure 5: Phas1 and PnP1 - Transient solutions using 4 phases.

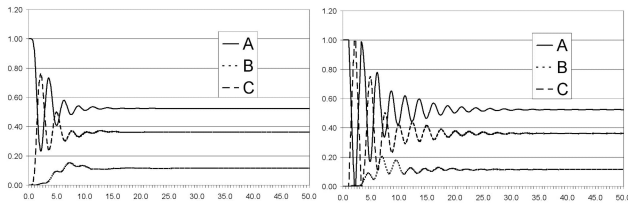


Figure 6: Phas2 and PnP2 - Transient solutions using 8 phases.

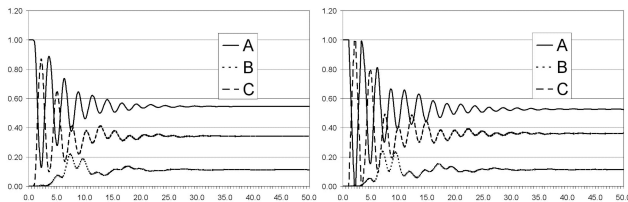


Figure 7: Phas3 and PnP3 - Transient solutions using 16 phases.

the longest to compute with almost a minute. The fastest solution is achieved when using only phase-type approximations of order 4, but when comparing the transient solution to the reference, the result quality is not sufficient. As always, the optimal combination is a trade off between performance and accuracy, and depends highly on requirements of the application itself. The best solution in the steady state case would be to use a small number of phases for all distribution approximations, because not the shape of the distributions is of concern that much, but more the mean. A good transient solution within 14 seconds was achieved when only substituting the two infinite support distribution functions by 8 phases.

6. CONCLUSION

6.1. Summary

This paper formalized the inclusion of discrete phases in the proxel-based simulation algorithm. The work was motivated by the different properties of the two ways to approximate a non-

Markovian distribution function. Discrete phases were successfully integrated into the original proxel-based simulation algorithm, and can therefore also be included in its extensions. An algorithm using this changed formalization of the proxel and the simulation algorithm was implemented, and tested. The experiments conducted suggest that combining the two paradigms will lead to an efficient simulation algorithm for certain models. Depending on the intended use of the result, proxels, phases or a combination of both is a better choice, and the developed algorithm is flexible enough to use all three possibilities.

6.2. Outlook

Ongoing research is concerned with the rules that govern the choice of using phases or proxels. Besides the obvious fact, that phases should not be used for dynamic distributions, almost every other factor is dependent on the intended use of the solution. General guidelines need to be developed, on when to use which approximation method. This will eventually lead to an effective proxel-based simulation algorithm, that combines proxels and discrete phases to exploit the advantages of both paradigms.

7. REFERENCES

- [1] Lazarova-Molnar, S., Horton, G., "A Description Framework for the Proxel-Based Simulation of a General Class of Stochastic Models" Submitted to 38th Annual Simulation Symposium, San Diego, CA, USA, 2005.
- [2] Horton, G., "A New Paradigm for the Numerical Simulation of Stochastic Petri Nets with General Firing Times," European Simulation Multiconference, Darmstadt, June 2002. SCS European Publishing House, 2002.
- [3] Lazarova-Molnar, S., Horton, G., "Proxel-Based Simulation for Fault Tree Analysis," 17. Symposium Simulationstechnik (ASIM 2003), SCS European Publishing House 2003.
- [4] Lazarova-Molnar, S., Horton, G., "Proxel-Based Simulation of Stochastic Petri Nets," Simulation und Visualisierung 2004. SCS European Publishing House 2004.
- [5] Isensee, C., Horton, G., "Proxel-Based Simulation of Project Schedules," European Simulation multiconference 2004. SCS European Publishing House 2004.
- [6] Lazarova-Molnar, S., Horton, G., "Proxel-Based Simulation of Stochastic Petri Nets Containing Immediate Transitions," On-Site Proceedings of the Satellite Workshop of ICALP 2003 in Eindhoven, Netherlands. 2003.
- [7] Neuts, M. F., "Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach," The John Hopkins University Press, Baltimore, 1981.
- [8] Isensee C., Horton G., "Approximation of Discrete Phase-Type Distributions," Submitted to 38th Annual Simulation Symposium, San Diego, CA, USA, 2005.
- [9] Bobbio, A., Horvth, A., Scarpa, M., Telek, M., "Acyclic Discrete Phase-type Distributions: Properties and a Parameter estimation Algorithm," Performance Evaluation, 54(1):1-32, 2003.
- [10] Lazarova-Molnar, S., Horton, G., "Proxel-Based Simulation of a Warranty Model," European Simulation multiconference 2004. SCS European Publishing House 2004.