

Proxel-Based Simulation of Stochastic Petri Nets Containing Immediate Transitions

Sanja Lazarova-Molnar¹

*Institute for Simulation and Graphics
University of Magdeburg
Magdeburg, Germany*

Graham Horton²

*Institute for Simulation and Graphics
University of Magdeburg
Magdeburg, Germany*

Abstract

This paper discusses the analysis of stochastic Petri nets using the proxel-based simulation method. The paradigm of the proxel ("probability element") was recently introduced in order to provide a new algorithmic approach to analysing discrete-state stochastic models such as are represented by stochastic Petri nets (SPNs) or queuing systems. Proxel-based simulation is not related to either of the standard simulation approaches: it is in no way analogous to discrete-event simulation, and, although it is based on the model's underlying stochastic process and makes use of supplementary variables, it does not require the use of differential equations. Instead, the proxels trace the movement of probability from one state of the model to another using discretised time steps. Since stochastic Petri nets are a powerful and widespread tool for modelling stochastic processes, we are interested in finding out how the proxel-based method applies to them. The analysis of SPNs using the proxel-based method is the subject of this paper, with an emphasis on the treatment of immediate transitions.

1 Goals of the Paper

The goal of this paper is to extend the analysis of stochastic Petri nets using the recently introduced proxel-based method. The formal

¹ Email: sanja@isg.cs.uni-magdeburg.de

² Email: graham@isg.cs.uni-magdeburg.de

connections between the proxel-based method and stochastic Petri nets were established in [5], but the Petri nets described there excluded the use of immediate transitions. Many of the things explained in that paper, apply here as well, as this paper can be seen as a logical consequence of, and is based on the former one. Since the treatment of immediate transitions was left as an open problem, in this paper we will extend the definition and the application of the proxel-based simulation to SPNs containing immediate transitions. Because of the novelty of the proxel-based method, an extensive description of the method will be presented, as well as a description of its application to analysing stochastic Petri nets. A comprehensive example will be given in order to provide a complete view of the treatment of immediate transitions with the proxel-based method.

2 Introduction to SPNs and Proxels

2.1 Modelling with Petri Nets

Stochastic Petri nets are popular and powerful tool for modelling and analysing complex stochastic systems. They are a visual tool which provides a natural and intuitive representation of the user's conceptual model. A stochastic system is usually viewed as a function of time, which means that we are interested in computing its dynamic behaviour. In many cases it is a discrete-event system, meaning that the changes in the system occur at certain points in time as a consequence of the completion of certain activities in the system. Activities in SPNs are associated with transitions, which can either fire instantly, as soon as certain conditions become enabled (immediate transitions), or can be associated with probability distribution functions which determine the firing delays (timed transitions). Besides the transitions, an SPN is defined by a finite number of places, a finite number of arcs and an initial state (the initial marking of the Petri net). Note that our definition of SPNs is not restricted to exponential distributions.

Formally, the class of SPNs that will be treated in this paper can be described in the following way:

$$SPN = (P, T, A, G, m_0)$$

- $P = \{P_1, P_2, \dots, P_n\}$, the set of places, drawn as circles
- $T = \{T_1, T_2, \dots, T_m\}$, the set of transitions along with their distribution functions or probability values, drawn as bars
- $A = A^I \cup A^O \cup A^H$, the set of arcs, where A^O is the set of output arcs, A^I is the set of input arcs and A^H is the set of inhibitor arcs and each of the arcs has a multiplicity assigned to it,

- $G = \{g_1, g_2, \dots, g_r\}$, the set of guard functions which are associated with different transitions,
- m_0 - the initial marking of the Petri net.

Each transition is defined as $T_i = (F, type)$, where $type \in \{enabling, age, immediate\}$ is the type of memory policy if it is a timed transition or "*immediate*" if the corresponding transition is an immediate one. F is a cumulative distribution function if the corresponding transition is a timed one. Immediate transitions have a constant value instead of a distribution function assigned to them, which is used for computing the probability of firing of an immediate transition if more than one are enabled at once. The sets of arcs are defined such that

$$A^O = \{a^o_1, a^o_2, \dots, a^o_k\}, A^I = \{a^i_1, a^i_2, \dots, a^i_j\}, \text{ and } A^H = \{a^h_1, a^h_2, \dots, a^h_{ij}\},$$

where

$$A^H, A^O \subseteq P \times (T \cup I) \times \mathcal{N}, A^I \subseteq (T \cup I) \times P \times \mathcal{N}.$$

We denote by $M = \{m_0, m_1, m_2, \dots\}$ the set of all reachable markings of the Petri net. Each marking is a vector made up of the number of tokens in each place in the Petri net, $m_i = (\#P_1, \#P_2, \dots, \#P_n)$. We distinguish two different kinds of markings, vanishing (if any immediate transitions are enabled) and tangible (if no immediate transitions are enabled). The set of all reachable markings is the discrete state space of the Petri net. The changes from one marking to another are consequences of the firing of enabled transitions which move (destroy and create) tokens, creating the dynamics in the Petri net. This makes the firing of a transition analogous to an event in a discrete-event system. The markings of a Petri net, viewed as nodes, and the possibilities of movement from one to another, viewed as arcs, form the reachability graph of the Petri net.

The class of stochastic Petri nets that was treated until now excluded the possibility of having immediate transitions. Immediate transitions can also be treated with proxels, which is the main subject of this paper.

Another issue when analysing SPNs with proxels are the memory policies of the timed transitions. Every timed transition has a memory policy assigned to it, which determines the behaviour of the transition's enabling time when it becomes disabled. It can either have age or enabling memory policy, which specifies whether the time one timed transition was enabled until another fired, will be remembered (age memory policy), or reset (enabling memory policy).

The proxel-based method operates on the reachability graph of the Petri net; therefore the first step prior to the proxel-based analysis of the Petri net will be its construction. The assumption that we make is that the reachability graph is finite, i.e. the Petri net is bounded.

2.2 The Proxel-based Approach to Simulation

Proxels were recently introduced [2] as a new technique for analysing discrete-state stochastic models such as queuing systems or stochastic Petri nets. For this class of models, the analysis is usually carried out using Monte Carlo simulation. By contrast, proxel-based simulation is deterministic, and works with the state-space of the model, which is extended to include supplementary variables that represent transitions' age intensities. Normally, this approach entails constructing and solving partial differential equations [1], but proxels yield a purely algorithmic approach to the simulation, in which differential equations are avoided completely. The goal of the new approach is to develop an easily-understood, deterministic algorithm, which, for certain classes of models at least, may prove to be competitive with discrete-event simulation.

A proxel is a basic computational unit for the simulation algorithm, which represents the probability that at a given time, the simulation model has reached a specific state via a specific path. We use the term "Proxel" as an abbreviation of "probability element" by analogy to the well-known "pixel" (picture element) in Computer Graphics.

The idea behind the simulation algorithm is to discretise the continuous stochastic process of the SPN using a discrete time step dt . This yields a computational model consisting of a set of discrete states at each time point, each of which has a certain probability of occurring. The proxel simulation algorithm creates the discrete states on the fly and tracks probability as it is redistributed between these states as time progresses in discrete steps.

We illustrate the general idea using the same example from [5] which is based on a simple SPN, whose reachability graph is shown in Figure 1. This Petri net has three markings, m_0 , m_1 , and m_2 . The delays for each transition are described by probability distributions. We assume the SPN to be in marking m_0 at time $t = 0$.

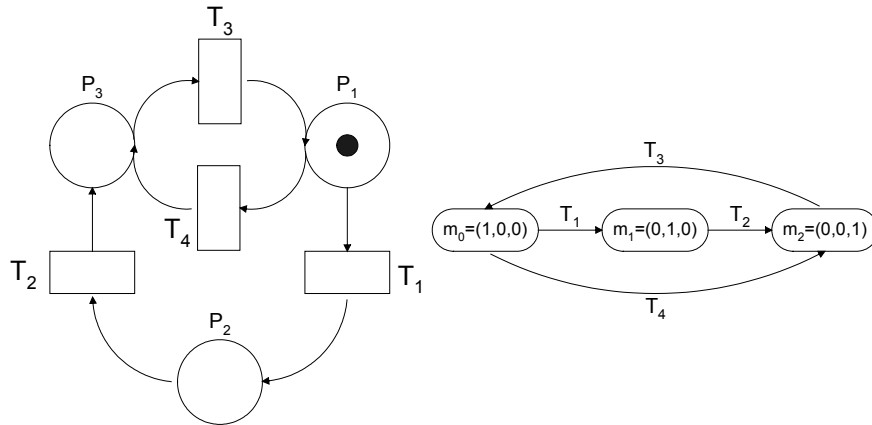


Figure 1: Example model

We now choose a discrete time step dt . This user-defined parameter must be chosen carefully, since it affects the accuracy of the simulation; specifically, it must be chosen so that the probability of the model making two or more state changes during any time interval $(t, t+dt)$ is significantly smaller than the probability of one state change only. The proxel-based method makes the approximating assumption that at most one state change occurs during a time interval of length dt .

Figure 2 shows the states reached by the model at times 0, dt , and $2*dt$. The state is composed of two parts - the marking of the SPN and the length of time that a transition has been enabled without firing. The latter is known as the age intensity; it is needed to compute the probabilities for each state change, as will be explained in the next section. For this simple model, only one age intensity variable is needed. For more general models, several age intensity variables will be necessary, as will be seen in the example in section 3.

At time $t = 0$, the model is in state $(m_0, 0)$, where the second component is the time that the enabled transitions have been enabled. At time $t = dt$, the model can have done any of three things - transition to marking m_1 or m_2 , or remain in marking m_0 . The probability for each of these occurrences can be computed from the distribution functions describing the firing delays of the transitions. States at time $t = 2*dt$ and later are generated correspondingly. Proxel-based simulation is the repeated computation of the new set of states and their probabilities as simulation time progresses.

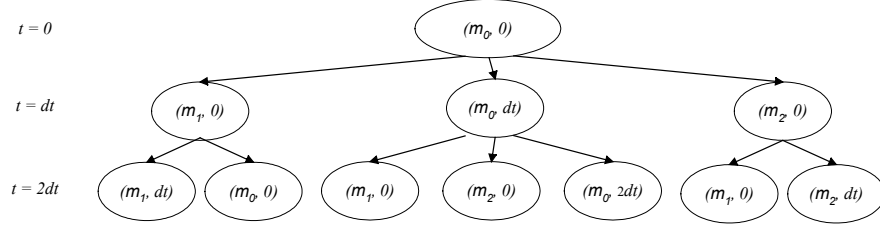


Figure 2: The first few states reached by the model

The set of reachable markings of the Petri net corresponds to the set of discrete states of the system. The state changes are associated with transitions. The state space is created based on the reachability graph, which is the origin for the Proxel-based analysis of the stochastic Petri net. The analysis begins with the probability equal to 1 of the Petri net being in the initial marking at time zero. Based on the transitions enabled in the initial marking and their distribution functions, this probability of one is distributed among the states at subsequent time steps, as will be explained in detail in the next section.

3 Proxel-based Simulation of SPNs

3.1 Definitions

Proxels are elements that completely define the state of the system at discrete points in time. We assume that the system is modelled as a stochastic Petri net, having the restrictions that we specified in the second section. Let PN be a stochastic Petri net, defined as follows:

$$PN = (P, T, A, m_0), \text{ with}$$

$$P = \{P_1, P_2, \dots, P_n\}, T = \{T_1, T_2, \dots, T_m\}.$$

A proxel Px is defined as the following:

$$Px = (S, t, R, Pr).$$

$S = (m, \vec{\tau})$ is the state of the system, where m is the marking and $\vec{\tau}$ is the *age intensity vector*, which contains the elapsed enabling time of a set of timed transitions. t is the global simulation time and Pr denotes the probability that the model is in state S at time t , given that it has been reached through the sequence of states that represent tangible markings, $R = (S_1, S_2, \dots, S_s)$. The proxels that represent tangible markings will be referred to as *tangible proxels*. The null sequence is denoted by $\emptyset = ()$. The age intensity vector $\vec{\tau}$ is needed for a complete

definition of the state of the Petri net, because the firing rate of each timed transition is dependent on how long it has been enabled. The *instantaneous rate function* IRF, denoted by $\mu(\tau)$, is used for this purpose. It takes as a parameter τ , the enabling time of the timed transition that caused the change from the previous to the current marking, and is computed from the distribution function that is associated with the transition. Henceforth, the term "state" will refer to the vector $S = (m, \vec{\tau})$.

For each timed transition $T_i = (F_i, type_i)$ with distribution function F_i and type $type_i \neq "immediate"$, the IRF $\mu_i(\tau)$ is calculated according to:

$$\mu_i(\tau) = \frac{F_i'(\tau)}{1 - F_i(\tau)}$$

The instantaneous rate function is used for calculating the probability that the timed transition will fire within the time interval $(\tau, \tau + dt)$, if it has been active for time τ . We approximate this probability by $\mu_i(\tau) * dt$. This is the fact that makes it possible to compute the probabilities for each state as time progresses.

Immediate transitions are associated with vanishing markings, therefore the proxels that represent vanishing markings are referred to as *vanishing proxels*. This also applies to the states that represent vanishing markings, they are referred to as *vanishing states*. The model spends zero time and therefore their simulation time component is not advanced. The probability values of the successor proxels of a vanishing proxel are computed as a multiplication of the probability value of the vanishing proxel and the probability that the corresponding immediate transition would fire. The latter is computed using the following formula:

$$P(T_i \text{ fires}) = \frac{p_i}{\sum_{T_j \text{ is enabled}} p_j}, \text{ where } T_{i/j} = (p_{i/j}, \textit{immediate}).$$

This means that the probability with which one immediate transition fires depends on the "probability" values of the other enabled immediate transitions (the word probability is in quotations because it needs not to be a probability value, i.e. between 0 and 1).

From the above said it is obvious that immediate transitions do not represent an additional burden for the method and do not require a special treatment. Instead, they are treated in the same way as timed transitions, without advancing the simulation time component. We find this to be an advantage for the proxel-based approach.

The proxel-based method distributes the initial probability of 1 for being in the initial state among all of the subsequent states of the model. The proxels are the computational units that store and keep track of the flow of probability from one state to another. This means that the proxel-based simulation repeatedly generates from each proxel the set of successive proxels, until the end of the simulation time has been reached. In this manner, a tree structure of proxels is created, which will be referred to as the "proxel tree". The proxel tree is actually the state space of the model in terms of proxels.

During the process of generating the proxel tree, at any discrete time step, the same state may be generated many times, each time via a different sequence of predecessors, i.e. a different route R_r . In order to obtain the total probability for that state and to optimise the storage of the proxels, the probabilities of all of that state's instances are summed up and a new proxel that represents the state is generated. This proxel is stored as a representative of the corresponding state and its route parameter is set to the union of all the routes that lead to that state at the current discrete time step. This can be formally represented in the following way:

$$Px = (S, t, \bigcup_{i:R_i \xrightarrow{t/dt} S} R_i, P(\text{model in } S \text{ at time } t))$$

$$P(\text{model in state } S \text{ at time } t) = \sum_{i:R_i \xrightarrow{t/dt} S} P(\text{model in } S \text{ at time } t | S \text{ reached via } R_i)$$

$$\xrightarrow{n} = \text{"leads to in } n \text{ steps"}$$

Then, at each discrete time step, this procedure repeats. In many cases, after a certain number of discrete time steps, a steady state is reached, which means that the sums of the probabilities of the tangible proxels that represent the same discrete state of the system converge to a stationary value. The formal representation of the probabilities of the different discrete states at different time steps as well as the calculation of the proxels' probabilities in the next discrete time step based on the previous one is the following:

$$P(\text{model in marking } m \text{ at time } t) = \sum_{\tau: S_k=(m,\tau)} P(\text{model in state } S_k \text{ at time } t)$$

$$P(\text{model in } S_k \text{ at time } t+dt) = \sum_{i,j:R_j \xrightarrow{t/dt} S_i} P(\text{model in } S_i \text{ at } t+dt) * \mu(\tau_{ij}) * dt * p_{jk}$$

where τ_{ij} is the age intensity of the timed transition that caused the state change from S_i to S_j and S_i is an intermediate vanishing state, in which case p_{lk} is the probability with which the relevant immediate transition that is enabled in the vanishing marking fires. If there is not an intermediate vanishing marking, then $S_i = S_k$ and $p_{lk} = 1$. $\mu(\tau_{ij})$ and p_{lk} are zero if the corresponding state changes are not possible.

One important aspect when constructing the proxel tree is dealing with the different memory policies of the timed transitions in the Petri net. Age policy transitions "remember" their activation times when they become disabled owing to another transition firing. When re-enabled, the clock that measures the time until the transition fires resumes from where it left off. For this reason, every age policy transition may require an additional component in the age intensity vector, which adds to the complexity of the simulation. If M is the set of reachable markings, then the dimension D of the age intensity vector is bounded by:

$$D = \max_{m \in M} \{|\text{enabled enabling policy transitions in } m| + |\text{age policy timed transitions}|\}$$

The first components of the age intensity vector are the age intensities of all timed transitions that have enabling policy and have been enabled at different points in time. The second component contains the age intensities of the age memory transitions. The values of the variables corresponding to enabled transitions increase with time while the timed transitions they represent remain enabled. Each of them is set to zero when the corresponding timed transition fires. The age intensities of the currently disabled age policy transitions remain constant until the transitions are once again enabled. We believe there is a way to make an optimal mapping, so that at every point in time only the relevant age policy transitions are represented in the age intensity vector, and at the same time it has the lowest possible dimension. This is still a subject of study; for now we will operate with the previously described mapping tactic.

3.2 Example

In this section an example will be presented of how the proxel-based method functions when the Petri net contains both types of transitions (timed and immediate). For this purpose, the Petri net model shown in Figure 3 will be used. This SPN represents a limited queuing system with two servers in which the likelihood that the customer chooses each of them when both are free is probabilistic. For reasons of simplicity the maximal number of customers in the queue was chosen to be two.

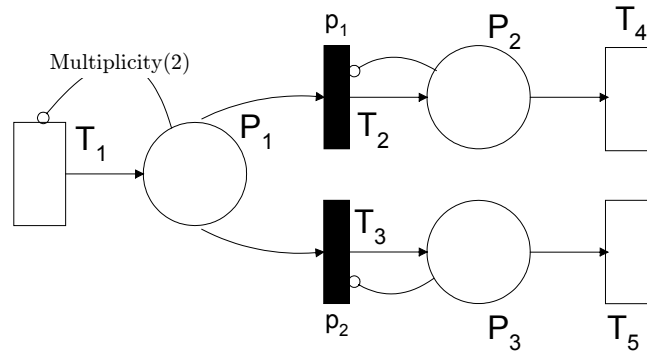


Figure 3: SPN model of a queuing system with two servers and one queue

The initial marking is $m_0=(0, 0, 0)$. There are three transitions, defined as follows:

- $T_i = (F_i, \textit{enabling}), i = 1,4,5$
- $T_i = (p_i, \textit{immediate}), i = 2,3$

Two of the transitions are immediate, and three are timed ones. Because of the possibility that all three of the timed transitions are enabled at the same time with different elapsed times, the dimension of the age intensity vector is three. The reachability graph of the Petri net is shown in Figure 4. Vanishing markings are shaded, and the arcs representing immediate transitions represented by thicker lines. In Figure 5 the tree represents all the possible developments of the Petri net within two discrete time steps, in terms of markings.

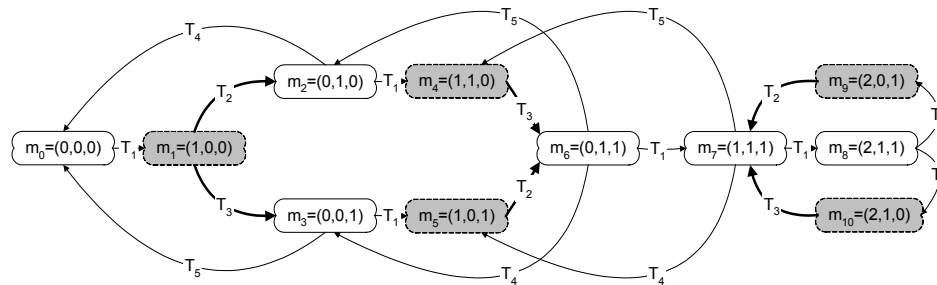


Figure 4: Reachability graph of the Petri net in Figure 3

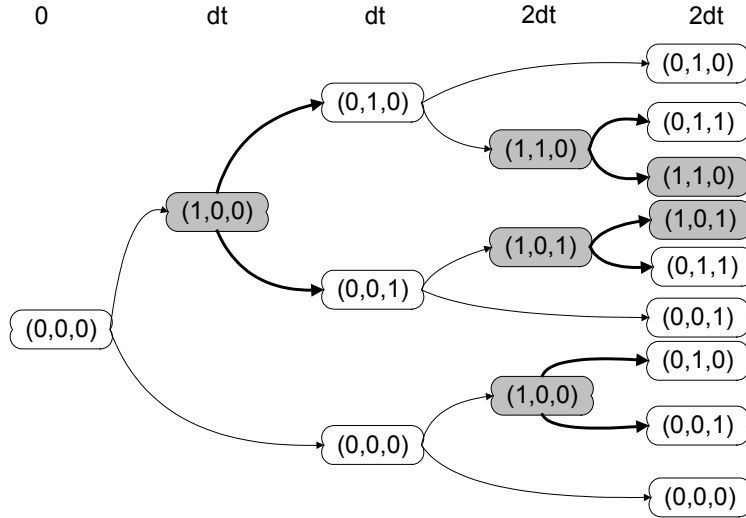


Figure 5: The first three levels of the markings tree based on the SPN in Figure 3

The age intensity vector complies with the following mapping (T_1, T_4, T_5) , i.e. the three timed transitions that have enabling memory policy are consequently mapped to the three components of the age intensity vector.

The beginning part of the proxel-tree for this model is presented in Figure 6. The proxel analysis operates on the complete reachability graph, without advancing the time in the case of vanishing markings. This means that whenever the model is in a vanishing marking, the probabilities of the subsequent proxels are instantly calculated by multiplying the probability of the vanishing proxel with the probabilities associated with the relevant immediate transitions. The simulation time during this operation does not advance, as shown in Figure 6.

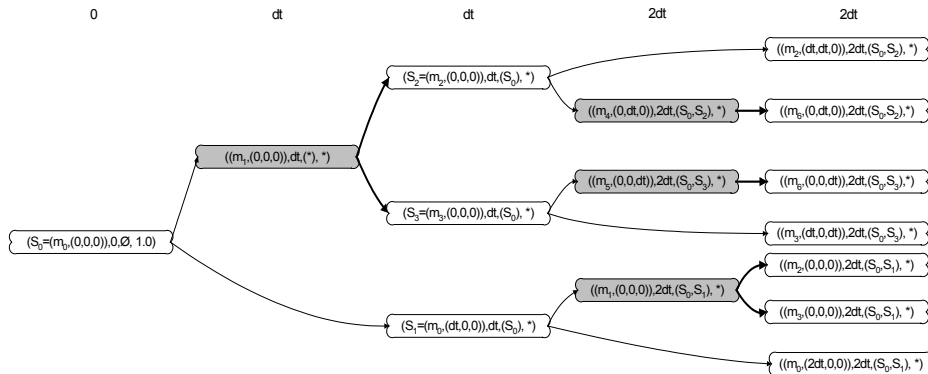


Figure 6: The first three levels of the proxel tree based on the SPN in Figure 3

The values of the probabilities are not stated explicitly in Figure 6, but are replaced by asterisks. In Figure 7, a result from a proxel-based simulation of this Petri net is presented. The value that was used for dt is 0.01, and the simulation was run up to time $t = 5$, i.e. there were 500 discrete time steps. The computation took 467.6 seconds on a 1.2 GHz Pentium III notebook. The distribution functions and the probability values associated with the transitions that were used in the simulation are the following:

- $F_1 \sim \text{Uniform}(0.1, 0.3)$, $F_4 \sim \text{Uniform}(0.4, 0.7)$, $F_5 \sim \text{Deterministic}(0.7)$
- $p_1 = 0.3$, $p_2 = 0.7$

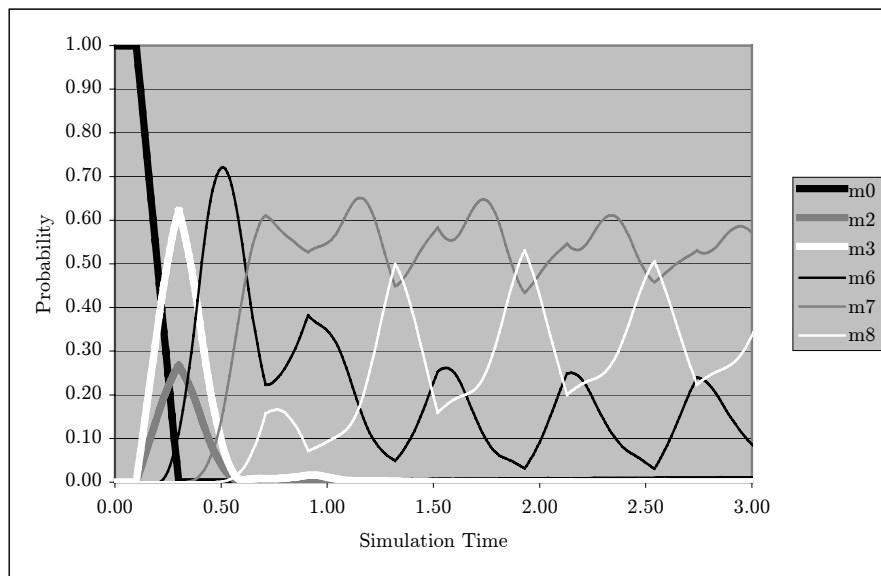


Figure 7: Solution (to time 3.0) of the example SPN, obtained by proxel-based simulation

In Figure 8, a beginning part of the output of the implementation for this example is shown. There, the order in which the proxels are processed can be seen, as well as the exact probability values, given the specific parameters for the distribution functions and the probability values. *AddProxel* shows the proxels that are generated and stored into the proxel tree, *Processing* shows a proxel that is currently being processed i.e. taken from the proxel tree and its successor proxels are being generated. *Vanishing* shows the vanishing proxels that are being processed i.e. their successor proxels being stored. The route parameter is omitted and replaced by the word "Route", for reasons of simplicity. Only the proxels with non-zero probability values are actually stored. That explains why the size of the tree in the first three steps is one.

```

AddProxel ((m0, ( 0dt,  0dt,  0dt)),  0dt, Route, 1.000000)

STEP 1

Processing ((m0, ( 0dt,  0dt,  0dt)),  1dt, Route, 1.000000)

Vanishing ((m1, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel ((m2, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel ((m3, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel ((m0, ( 1dt,  0dt,  0dt)),  1dt, Route, 1.000000)

STEP 2
Size of tree 1

Processing ((m0, ( 1dt,  0dt,  0dt)),  2dt, Route, 1.000000)

Vanishing ((m1, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel ((m2, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel ((m3, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel ((m0, ( 2dt,  0dt,  0dt)),  2dt, Route, 1.000000)

STEP 3
Size of tree 1

Processing ((m0, ( 2dt,  0dt,  0dt)),  3dt, Route, 1.000000)

Vanishing ((m1, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.250000)
AddProxel ((m2, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.075000)
AddProxel ((m3, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.175000)
AddProxel ((m0, ( 3dt,  0dt,  0dt)),  3dt, Route, 0.750000)

```

Figure 8: Output of the implementation for the example described in this section

4 Discussion of the proxel-based method

The Proxel algorithm is a state space-based approach to the simulation of the Petri net, whereby the states include both the marking and enabling time information of the transitions. The complexity of the algorithm thus depends directly on the size of discrete state space that is generated.

In principle, this state space is infinite, since for most enabling time distributions, arbitrarily long times are possible. However, in practice, the probability that a transition continues to be enabled decreases with time, so after a point, the proxels representing these states can be ignored. In our implementation, for example, proxels with a probability value less than $1E-12$ are not generated.

Enabling time distributions with finite support, such as the uniform and deterministic distributions are particularly advantageous, since they only generate non-zero firing probabilities over a short range of age intensities [4]. This means that the number of proxels generated per time step for each such transition is only one, and thus the contribution to the complexity is only linear in time. By contrast, distributions whose instantaneous rate functions are non-negligible over a wide or infinite range lead to an increase in the number proxels at each time step. Of course, the exponential distribution, being memoryless, is unproblematic, since it requires no age variable at all.

In general, the overall number of states to be processed grows exponentially with the number of non-exponential transitions that are concurrently enabled or are disabled and of age type and storing partially expired enabling times. This is true for any supplementary variable-based approach. The number of proxels needed at any time step grows towards this number as simulation time progresses. Despite appearances, the complexity of the method is not exponential in simulation time; at any time step there are many proxels with the same state, which are reached via different routes with different probabilities. These can of course be lumped together in a single proxel.

Adding immediate transitions to the SPN models that can be simulated did not substantially change the algorithm or its implementation. The new vanishing proxels are essentially treated in the same way as the tangible ones: the only substantial difference is that simulation time is not advanced for newly-created vanishing proxels. In our current implementation, vanishing proxels are not stored at all, but are processed "on-the-fly". We consider this ability to treat tangible and vanishing states identically to be a feature of our method; it contrasts, for example, with solution methods for GSPNs, where vanishing states require special treatment [1].

The proxel-based method can be interpreted as computing a certain discrete-time Markov chain, as described in [5]. This interpretation gives an additional insight into the nature of proxel-based simulation.

The proxel-based method performs well for models with rare events [2][4], such as are to be found in reliability and performability models. Such models contain events whose probability of occurring is very low. This can be a problem for Monte Carlo simulations, since a very large number of replications might be needed before the simulator "discovers" these low-probability events. By contrast, in the proxel simulation, events with large and small probabilities are treated identically - the difference in probability is reflected in the size of the numbers computed, but does not affect the course of the algorithm per se.

One additional advantage of the method over discrete-event simulation is that the discrete solution values computed are a first-order approximation to the true continuous solution. This allows results obtained using two different time steps dt to be extrapolated to a substantially more accurate value at the (fictitious) time step $dt = 0$, as described in [5].

5 Summary and Outlook

In this paper, the proxel-based analysis of SPNs was extended to include immediate transitions. It was shown that the treatment of immediate transitions is very straightforward and required no additional techniques. The existing definitions were extended in an

appropriate manner. In order to illustrate the method, a simple example was presented.

Like all supplementary variable-based methods, the state space can become very large. Thus much of our attention is devoted to developing techniques for reducing the size of the problem. One of these is to use an adaptable size of the time step [3], which can substantially improve the method's performance, especially for stiff models.

Other ideas include application to higher-level models such as coloured stochastic Petri nets. These should easily permit proxel-based simulation, since all the necessary state information can be encoded into the state vector and used to compute the probabilities for successor states.

References

- [1] German, R., "Performance Analysis of Communication Systems. Modelling with Non-Markovian Stochastic Petri Nets, " John Wiley & Sons, Ltd, 2000.
- [2] Horton, Graham, *A new paradigm for the numerical simulation of stochastic Petri nets with general firing times*, Proceedings of the European Simulation Symposium 2002, Dresden. Society for Computer Simulation, 2002.
- [3] Horton, Graham, and Sanja Lazarova-Molnar, *A Reduced-Stiffness Method for the Transient Solution of Discrete-Time Markov Chains*, Submitted to Numerical Solution of Markov Chains, University of Illinois at Urbana-Champaign, 2003.
- [4] Lazarova-Molnar, Sanja, and Graham Horton, *An Experimental Study of the Behaviour of the Proxel-Based Simulation Algorithm*, Simulation und Visualisierung 2003. SCS Verlag 2003.
- [5] Lazarova-Molnar, Sanja, and Graham Horton, *Proxel-Based Simulation of Stochastic Petri Nets*. Submitted to the 10th International Workshop on Petri Nets and Performance Models, University of Illinois at Urbana-Champaign, 2003.