Steady State Solution for Models with Geometric and Finite Support Activity Duration

András Horváth
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
horvath@di.unito.it

Abstract

This paper addresses steady state solution of discrete time stochastic models in which every activity duration is given either by a geometric or a finite support distribution. Finite support distributions can be described by discrete time phase type (DPH) distributions. The behaviour of the whole stochastic model is given by a discrete time Markov chain (DTMC). The DTMC is subject to the so-called state space explosion. We present a technique for obtaining the steady state solution that alleviates this problem. The technique is based on Gaussian elimination combined with an iterative technique.

1 Introduction

This paper deals with models in which duration of an activity (task,job) can be given either by a memoryless distribution or by a finite support distribution. In any state of the system a set of tasks is active with no limitation on the number of active finite support tasks. Completion of a task alters the global state of the model and may cause to interrupt active tasks or to start inactive ones. These models can arise either from description of discrete time systems or from approximation of continuous time, non-Markovian models.

Many solution techniques were presented for continuous time, non-Markovian models. The supplementary variable approach [13] was applied in [15, 25, 18] for the analysis of non-Markovian Petri nets with various preemption policies (see [1, 5] for the definition of preemption policies). All these works assume that a single supplementary variable is enough to describe the state of the stochastic model (there is only one active non-exponential activity at a time). For the description of models in which more than one non-exponential task is active, more than one supplementary variable is needed (see, for example, [16]). Unfortunately, no good solution technique is known for such models in general. Solution techniques for models with exponential and deterministic durations are implemented in DSPNexpress [20]. This tool put a limit on the number of concurrent deterministic activities is limited.

An approach to solve non-Markovian models is discrete time approximation. Finite support distributions can be approximated by DPH distributions maintaining the finite support property [3, 4]. Infinite support can be approximated by DPH distributions through fitting procedures [17]. Exponential distributions can be approximated by geometric distributions. The discrete time approximation gives rise to DTMC. The structure of the DTMC allows for very efficient storage of the transition matrix through Kronecker expressions [21, 22, 2]. Solution techniques for Markov chains described by Kronecker expressions are discussed in [11, 7, 10]. There remains however the problem of storing the vector containing the probabilities of the states. To overcome this problem, in [6, 12] approximate stationary measures are computed based on aggregation while [9] proposes a compact Kronecker representation for the vector which leads to an approximate solution.

In this paper an exact solution technique is proposed for steady state analysis for a special class of DTMCs. By successive application of Gaussian elimination the number of states of the DTMC can be decreased. Then the resulting reduced DTMC can be solved by any iterative method. Gaussian elimination can cause fill-in of the transition matrix of the reduced DTMC. The problem of finding the minimal fill-in is NP complete [26], and therefore, heuristics are used. The most widely used is the minimal degree algorithm [14]. In general, these heuristic algorithms are hard to implement in such a way that the entries of the reduced matrices are not stored directly but computed on the fly based on the parameters of the model. We propose an algorithm which, by exploiting the structure of the model, performs Gaussian elimination in such a way that prevents fill-in and allows for computing entries of the reduced matrix on the fly. This way the computations are carried out on vectors which have significantly less number of entries than the number of states of the DTMC.

The paper is organized as follows. Section 2 introduces the considered class of DTMCs. The proposed solution technique is presented in Section 3. A simple numerical example is provided in 4. Possible directions for future work is discussed in Section 5. The paper is concluded in Section 6.

2 Considered Model Class

In this section we describe the model class whose steady state computation will be addressed. We start by discussing the distribution of the duration of the activities. Then the composition of the activities into a complex model is described. The presented description can be easily obtained from high level description languages like Petri nets or process algebra. The section ends with presenting the DTMC that describes the stochastic behaviour of the model.

Activity Duration

Activities of the model are divided into two classes. The set $\mathcal{G} = \{g_1, g_2, \dots, g_{N_g}\}$ of cardinality N_g contains the activities with geometric duration. The probability that a geometric activity $g_i, 1 \le i \le N_g$ finishes in a step is denoted by p_i. The set $\mathcal{F} = \{f_1, f_2, \dots, f_{N_f}\}$ of cardinality N_f is the set of the activities with finite support distribution. The probability that a finite support task $f_i, 1 \le i \le N_f$ finishes in $1 \le n \le m_i$ steps is denoted by $q_{i,n}$ where m_i is the largest number of steps for finite support activity f_i to finish. Further, $\bar{q}_{i,n}$ stands for the probability that finite support activity f_i finishes after $1 \le n \le m_i$ steps given that it did not finish before, i.e., $\bar{q}_{i,n} = q_{i,n}/(1 - \sum_{j=1}^{n-1} q_{i,j})$. By assuming that finite support activities finish with probability 1, we have $\sum_{j=1}^{m_i} q_{i,j} = 1$ for all $1 \le i \le N_f$. When a finite support activity is active since x time units we say that its age is x. The age of an activity is larger or equal to 0 and must be smaller than its maximal finishing time.

Composed Model

The composed model has N global states[1]. The set of active tasks in state $i, 1 \le i \le N$ is denoted by $\mathcal{E}_i$. Further, $\mathcal{E}_i$ is divided into two subsets according to the type of the activities: $\mathcal{E}_i^g$ ($\mathcal{E}_i^f$) contains the set of active geometric (finite support) tasks in global state i. In order to define the effect of the completion of activities, we introduce the quantity

$$s_{i,\mathcal{C},j} = Pr\{\text{next global st. is } j \mid \text{current global st. is } i, \text{ tasks in } \mathcal{C} \text{ finish}\}$$

with $1 \le i, j \le N$ and $\mathcal{C}$ is a non-empty subset of $\mathcal{E}_i$. For a model to be correct we must have $\sum_{j=1}^{N} s_{i,\mathcal{C},j} = 1$ for all $1 \le i \le N$, $\mathcal{C} \in 2^{\mathcal{E}_i}$ and $\mathcal{C} \ne \emptyset$.

When a global state change $i \to j$ occurs the tasks that were active in the last state but are not active in the current one (task in $\mathcal{E}_i - \mathcal{E}_j$) get disactivated (preempted). The applied preemption policy is preemptive repeat different, i.e. the portion of the task already performed is lost. The tasks that are present in $\mathcal{E}_j - \mathcal{E}_i$ get activated.

[1]We use the term *global state* in order to distinguish between states of the composed model and states of the underlying DTMC.

DTMC Underlying the Composed Model

A state of the DTMC underlying the composed model is characterized by the global state and the age of the active finite support tasks. Since the geometric distribution is memoryless, it is not necessary to keep track of the age of geometric activities. Global state i is represented by $\prod_{j:f_j \in \mathcal{E}_i^f} m_j$ states of the DTMC. The number of states of the whole DTMC is $\sum_{i=1}^{N} \prod_{j:f_j \in \mathcal{E}_i^f} m_j$. Given a state S of the DTMC, S_g stands for its global state and $S_{f_i}, f_i \in \mathcal{E}_{S_g}^f$ denotes the age of finite support activity f_i.

The transition matrix $\boldsymbol{T}$ that describes the behaviour of the underlying DTMC can be built as follows. We start from $\boldsymbol{T} = \boldsymbol{0}$ and consider first the state transitions without finishing of activities

$$t_{S,S'} = \left(\prod_{i:g_i \in \mathcal{E}_{S_g}^g} (1 - p_i) \right) \times \left(\prod_{i:f_i \in \mathcal{E}_{S_g}^f} (1 - \bar{q}_{i,S_{f_i}+1}) \right) \quad (1)$$

with $S'_g = S_g$ and for all $i : f_i \in \mathcal{E}_{S_g}^f$ we have $S'_{f_i} = S_{f_i} + 1$. The expression in (1) gives the probability that none of the active tasks finish. When none of the tasks finish the process remains in the same global state and the age variables of the active finite support tasks is incremented by one. As expected (1) results in 0 if any of the active finite support tasks reaches its maximal finishing time. Next, in order to consider finishing of activities, the following operation is performed for every state S of the DTMC and every non-empty subset $\mathcal{C}$ of $\mathcal{E}_{S_g}$

$$t_{S,S'} \mathrel{+}= s_{S_g,\mathcal{C},S'_g} \left(\prod_{i:g_i \in \mathcal{E}^g_{S_g}, g_i \notin \mathcal{C}} (1-p_i) \right) \times \left(\prod_{i:f_i \in \mathcal{E}^f_{S_g}, f_i \notin \mathcal{C}} (1-\bar{q}_{i,S_{f_i}+1}) \right) \times \left(\prod_{i:g_i \in \mathcal{E}^g_{S_g}, g_i \in \mathcal{C}} p_i \right) \times \left(\prod_{i:f_i \in \mathcal{E}^f_{S_g}, f_i \in \mathcal{C}} \bar{q}_{i,S_{f_i}+1} \right) \quad (2)$$

where for all $i : f_i \in \mathcal{E}^f_{S'_g}$

$$S'_{f_i} = \begin{cases} S_{f_i}+1 & \text{if } f_i \in \mathcal{E}^f_{S_g}, f_i \in \mathcal{E}^f_{S'_g} \text{ and } f_i \notin \mathcal{C}, \\ 0 & \text{if } f_i \in \mathcal{E}^f_{S_g}, f_i \in \mathcal{E}^f_{S'_g} \text{ and } f_i \in \mathcal{C}, \\ 0 & \text{if } f_i \notin \mathcal{E}^f_{S_g} \text{ and } f_i \in \mathcal{E}^f_{S'_g}. \end{cases} \quad (3)$$

The expression in (2) gives the probability that the activities in $\mathcal{C}$ finish and the next global state is S'_g. The ages of the activities change according to (3). In (2) $+=$ is applied because from a state S different sets of finishing activities can lead to the same state. Note that for every state of the DTMC in which at least one age is grater than 0, the probability of leaving the state in one step is 1.

For models in which there are active finite support tasks in some global states the transition matrix is highly sparse and has a lot of structure. For what concerns the computation of the steady state results, it is not necessary to build the transition matrix because its entries can be computed on the fly. The main problem is the storage of the iteration vector which has as many entries as many states there are in the Markov chain.

3 Computation of the Steady State Solution

Basic Step

The basic step of the procedure consists in eliminating a state by Gaussian elimination[2]. We consider elimination only of those states which the DTMC leaves in one step with probability 1. Let us consider a DTMC with n states described by its transition matrix $\boldsymbol{T}$ for which $t_{n,n} = 0$ (without loss of generality we assume that the state to eliminate is state n). We construct a new matrix $\boldsymbol{T'}$ of size $(n-1) \times (n-1)$ whose entries are given as

$$t'_{i,j} = t_{i,j} + t_{i,n}\ t_{n,j} \quad (4)$$

and a new vector $\boldsymbol{u'}$ of size $n-1$ whose entries are

$$u'_i = u_i + t_{i,n}\ u_n \quad (5)$$

where $\boldsymbol{u}$ is a vector of size n with all entries equal to 1. For example, if

$$\boldsymbol{T} = \left| \begin{array}{ccc} 0.5 & 0.2 & 0.3 \\ 0.1 & 0.5 & 0.4 \\ 0.2 & 0.8 & 0.0 \end{array} \right|$$

then

$$\boldsymbol{T'} = \left| \begin{array}{cc} 0.56 & 0.44 \\ 0.18 & 0.82 \end{array} \right| \quad \text{and} \quad \boldsymbol{u'} = |\ 1.3 \ \ 1.4\ | \ .$$

Matrix $\boldsymbol{T'}$ is a proper transition matrix again. The vector $\boldsymbol{u}$ can be seen as a normalization vector for the DTMC described by $\boldsymbol{T'}$ and is used to recover the steady state solution of the original DTMC from the steady state solution of the reduced DTMC. For a transition matrix with more than one 0 entry in its diagonal, the procedure described by (4) and (5) can be repeated starting from $\boldsymbol{T'}$ and $\boldsymbol{u'}$.

Assuming that the steady state solution of the DTMC described by $\boldsymbol{T'}$ is $\boldsymbol{\pi'}$, we can recover the steady state solution for the DTMC described by $\boldsymbol{T}$ as

$$\begin{aligned} \pi_i &= \frac{\pi'_i}{\sum_{j=1}^{n-1} \pi'_j\ u'_j} \quad \text{for} \quad 1 \le i \le n-1, \\ \pi_n &= \sum_{i=1}^{n-1} (u'_i - 1)\ \pi_i\ . \end{aligned} \quad (6)$$

For the example given above $\boldsymbol{\pi'} = |0.29\ 0.71|$ from which one can recover the steady state solution of the original chain $\boldsymbol{\pi} = |0.21\ 0.52\ 0.27|$.

Successive Application of the Basic Step

By successive application of the procedure described in the previous section the size of the DTMC can be reduced. The iteration vector as well becomes smaller. The problem is that if an unsuitable set of states is eliminated the resulting reduced transition matrix fills up with non-zero entries loosing its structure.

The way in which states are eliminated has to fulfill two requirements. First, it must be possible to determine the entries of the final reduced transition matrix

[2]For detailed description of the following expressions see, for example, Section 2.2.1 of [24].

without performing the reduction state by state. Second, the computation of the steady state probabilities of the global states of the original model must be possible without the storage of the steady state probability vector corresponding to the "unreduced" DTMC.

In order to have reduction procedure that meets the above requirements, a state S is removed if and only if

- there is at least one active finite support task and none of the ages are 0, i.e., $\mathcal{E}^f_{S_g} \neq \emptyset$ and for all $i : f_i \in \mathcal{E}^f_{S_g}$ we have $S_{f_i} \neq 0$; and
- state S is reachable in one step only from states where the set of active finite support tasks is identical to the set of active finite support tasks in S, i.e., $t_{S',S} \neq 0$ implies $\mathcal{E}^f_{S'_g} = \mathcal{E}^f_{S_g}$.

A state fulfilling the above requirements is reachable in one step only from a limited number states. As a result, eliminating these states leads to a controllable reduction process. It is easy to see that if some states of global state i are to eliminate than the number of states eliminated from global state i is $\prod_{j:f_j \in \mathcal{E}_i}(m_j - 1)$.

Before proceeding with the description of the reduced transition matrix, we need to introduce the following notation. The set of global states from which some of the states are removed will be denoted by $\mathcal{R}$. Instead, $\bar{\mathcal{R}}$ stands for the set of global states from which none of the states are removed. For a set of finite support activities $\mathcal{A}$, the matrix $\boldsymbol{T}_\mathcal{A}$ of size $N \times N$ describes how the process moves among those global states in which the set of active finite support tasks is $\mathcal{A}$ assuming that none of the finite support tasks finish. Entries of $\boldsymbol{T}_\mathcal{A}$ are given as

$$t_{\mathcal{A},i,j} = \begin{cases} 0 & \text{if } \mathcal{E}^f_i \neq \mathcal{A} \text{ or } \mathcal{E}^f_j \neq \mathcal{A}, \\ \sum_{\mathcal{C}:\mathcal{C}\subseteq\mathcal{E}^g_i} s_{i,\mathcal{C},j} \left(\prod_{k:g_k\in\mathcal{C}} p_k\right) \times & \\ \qquad \left(\prod_{k:g_k\notin\mathcal{C}}(1-p_k)\right) & \text{otherwise.} \end{cases}$$

Entries of $\boldsymbol{T}^k_\mathcal{A}$ will be denoted by $t^{(k)}_{\mathcal{A},i,j}$. Let $\boldsymbol{R}$ stand for the transition matrix of the DTMC resulting from the elimination of all the states that fulfill the criteria presented above. In the following the computation of $\boldsymbol{R}$ is presented.

It is clear from (4) that during the elimination of a state S, if $t_{S',S} = 0$, then the entries in the row corresponding to S' do not change. As a consequence, the entries in those rows of $\boldsymbol{R}$ that correspond to the global states in $\bar{\mathcal{R}}$ can be determined by (1) and (2). For what concerns the vector $\boldsymbol{u}$ which is used to recover the steady state solution of the original DTMC, we have $u_S = 1$ for every S such that $S_g \in \bar{\mathcal{R}}$.

Now we turn our attention to the states corresponding to global states in $\mathcal{R}$. In particular, let us consider state S which is in global state $S_g \in \mathcal{R}$ and is present in the reduced DTMC. Entries of the row of $\boldsymbol{R}$ that corresponds to S and entry u_S of $\boldsymbol{u}$ can be determined by Algorithm 1. In order to determine $\boldsymbol{R}$, Algorithm 1 has to be executed for all S such that $S_g \in \mathcal{R}$.

Demonstration that the transition matrix of the DTMC resulting from the reduction process is equal to the matrix resulting from the execution of the algorithm is rather elementary but cumbersome. Here we confine ourselves to describe the quantities computed during the algorithm. All these quantities can be interpreted considering the behaviour of the original DTMC. Being in state S, the quantity S_{max} computed in step 3 gives the maximal number of time units that can elapse before finishing at least one of the active finite support tasks. The quantity P_1 computed in step 5 is the probability that assuming that the process is in state S none of the finite support tasks finish before k time units. In step 7, P_2 is computed by multiplying P_1 with the probability that the process is in global state l after k time units as a consequence of finishing of geometric activities whose finishing does not change the set of active finite support tasks. In step 10, P_2 is multiplied by the probability that the tasks in set $\mathcal{C}$ finish after k steps. In step 12, we identify the state S' that we reach if we start in state S, none of the finite support activities finish in k steps, the set of active finite support tasks remain the same for k steps, and after k steps the finishing of activities including at least one finite support task leads to global state m. Finally, in step 13 P_3 is multiplied by the probability that finishing of the tasks in $\mathcal{C}$ leads to global state m. The resulting quantity must be added to $r_{S,S'}$. Entries of the normalization vector $\boldsymbol{u}$ are updated in step 8.

Intuitively, state S of the reduced DTMC, for which $S_g \in \mathcal{R}$, incorporates the behaviour of all those states that can be reached from S without finishing or interrupting of finite support tasks. All these states are eliminated from the original DTMC.

Having obtained the steady state solution $\boldsymbol{\pi}$ for the DTMC described by $\boldsymbol{R}$, the vector $\boldsymbol{v}$ of length N containing the steady state probabilities of the global states of the original model can be computed by Algorithm 2. Algorithm 2 starts with normalizing according to vector $\boldsymbol{u}$ (step 1). Then, in steps 2-4 we compute how the probabilities of the states of the reduced DTMC contribute to the probabilities of the global states. Steps 6 and 8 are identical to steps 3 and 5 of Algorithm 1 and were described earlier. Step 10 considers the probabilities of the states that are eliminated from the original DTMC according to (6).

Algorithm 1 Computation of $r_{S,S'}$ for a given S ($S_g \in \mathcal{R}$) and for all S'

```
1: set $r_{S,S'} = 0$ for all $S'$
2: set $u_S = 0$
3: set
       $$S_{max} = \max_{i:f_i \in \mathcal{E}^f_{S_g}} \{m_i - S_{f_i} - 1\}$$
4: for $k = 0$ to $S_{max}$ do
5:   set
       $$P_1 = \prod_{i:f_i \in \mathcal{E}^f_{S_g}} \frac{\sum_{j=S_{f_i}+k+1}^{m_i} q_{i,j}}{\sum_{j=S_{f_i}+1}^{m_i} q_{i,j}}$$
6:   for all $l$ such that $1 \le l \le N$ and $\mathcal{E}^f_l = \mathcal{E}^f_{S_g}$ do
7:     set
         $$P_2 = P_1 \; t^{(k)}_{\mathcal{E}^f_{S_g},S_g,l}$$
8:     perform $u_S += P_2$
9:     for all $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{E}_l$ and $\mathcal{C}^g \neq \emptyset$ do
10:      set
           $$P_3 = P_2 \left( \prod_{i:g_i \in \mathcal{E}^g_l, g_i \notin \mathcal{C}} (1 - p_i) \right) \times \left( \prod_{i:g_i \in \mathcal{E}^g_l, g_i \in \mathcal{C}} p_i \right) \times \left( \prod_{i:f_i \in \mathcal{E}^f_l, f_i \notin \mathcal{C}} (1 - \bar{q}_{i,S_{f_i}+k+1}) \right) \times \left( \prod_{i:f_i \in \mathcal{E}^f_l, f_i \in \mathcal{C}} \bar{q}_{i,S_{f_i}+k+1} \right)$$
11:      for all $m$ such that $1 \le m \le N$ do
12:        set $S'$ the state such that $S'_g = m$ and for all $i : f_i \in \mathcal{E}^f_m$
             $$S'_{f_i} = \begin{cases} S_{f_i} + k \\ \quad \text{if } f_i \in \mathcal{E}^f_l, f_i \in \mathcal{E}^f_m \text{ and } f_i \notin \mathcal{C} \,, \\ 0 \\ \quad \text{if } f_i \in \mathcal{E}^f_l, f_i \in \mathcal{E}^f_m \text{ and } f_i \in \mathcal{C} \,, \\ 0 \\ \quad \text{if } f_i \notin \mathcal{E}^f_l \text{ and } f_i \in \mathcal{E}^f_m \,. \end{cases}$$
13:        perform $r_{S,S'} += P_3 \; s_{l,\mathcal{C},m}$
14:      end for
15:    end for
16:  end for
17: end for
```

Algorithm 2 Recovering steady state probabilities of global states

```
1: set $\boldsymbol{v} = \boldsymbol{0}$ and $\boldsymbol{\pi} = \boldsymbol{\pi}/(\sum_S \pi_S \; u_S)$
2: for all state $S$ of the reduced DTMC do
3:   perform $v_{S_g} += \pi_S$
4: end for
5: for all state $S$ of the reduced DTMC such that $S_g \in \mathcal{R}$ do
6:   set
       $$S_{max} = \max_{i:f_i \in \mathcal{E}^f_{S_g}} \{m_i - S_{f_i} - 1\}$$
7:   for $k = 1$ to $S_{max}$ do
8:     set
         $$P_1 = \prod_{i:f_i \in \mathcal{E}^f_{S_g}} \frac{\sum_{j=S_{f_i}+k+1}^{m_i} q_{i,j}}{\sum_{j=S_{f_i}+1}^{m_i} q_{i,j}}$$
9:     for all $l$ such that $1 \le l \le N$ and $\mathcal{E}^f_l = \mathcal{E}^f_{S_g}$ do
10:      perform
           $$v_l += \pi_S \; P_1 \; t^{(k)}_{\mathcal{E}^f_{S_g},S_g,l}$$
11:    end for
12:  end for
13: end for
```

Implementation

Efficient implementation of the algorithm is not straightforward and its discussion is out of the scope of the paper. We provide only a few notes here. As suggested by Algorithm 1, several steps of the computation are repeated many times. The ordering of the states has a heavy effect on the execution time. Moreover, several quantities can be stored, once computed, and reused after on resulting in faster computations. The right ordering and the quantities that are worth to store depend on the structure of the model.

For testing the algorithm we have implemented four variants for the steady state solution. First, we either use the complete (original) DTMC or the reduced (compressed) one. Second, we either build and store the transition matrix in sparse form or compute its entries on the fly. When the transition matrix is built in advance, the computations are carried out by the LAS-Pack sparse matrix package [23]. For all the four cases the pure Gauss-Seidell method as presented in [24] is applied. More sophisticated iterative steady state solution methods could also be implemented (see [8] for a broad overview) but we believe that the conclusions drawn hereinafter are widely valid.

4 Numerical Example

As an example, we consider a discrete time M/D/S/K queue which is not of major interest from modeling point of view but, by changing its parameters, it allows us to test the proposed procedure in several different situations. For this model those global states are reduced in which all the servers are active. The utilization is set to 0.8 for all the test cases.

Results with single server are reported in Figure 1-5. Service time is either 10, 100 or 1000 and the buffer size is varied between 2 and 200. Figure 1 shows the number of states of the complete and of the reduced DTMC. For this model, the number of states in the reduced model does not depend on the service time. Note that the number of states reflects the memory requirement of the computations. The number of states in the complete model is approximately $D \times K$ while it is about K for the reduced one. Consequently the vectors to store the steady state probabilities are D times smaller. Figure 2 reports the number of non-zero entries in the matrices. Thanks to the choice of the set of eliminated states, the number of non-zero entries remains low in the reduced model. The number of iterations to reach the desired precision (computation is stopped when the error is below 1e-08) is depicted in Figure 3. The number of iterations is somewhat lower for the reduced models because the reduction makes the DTMC more "compact" in a probabilistic sense as well. The time to achieve the results is shown in Figure 4 for the case when the transition matrices are stored in the memory and in Figure 5 when the entries of the transition matrices are computed on the fly. When the matrices are stored, one iteration takes less time with a reduced matrices than with the corresponding complete one and less iterations are necessary. However, since the reduced matrix must be built and the steady state probabilities have to be recovered (Algorithm 2) the proposed method is slower for large values of K. When the computations are performed without storing the transition matrices, the proposed reduction method is either slower or faster depending on the service time. As illustrated by Figure 4 and 5, the computations without storing the reduced matrix are slower, however, the slow down is not drastic.

Figures 6 and 7 show the structure of the transition matrices before and after reduction with $D = 10$ and $K = 100$. (The figures were produced with MatView [19]). The original transition matrix has 1001 states while the reduced has 101. In the reduced matrix we have a single state for having $x, 1 \leq x \leq 100$ jobs in the buffer.

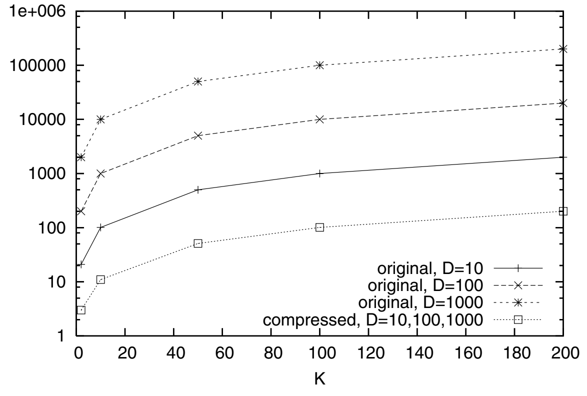

Figure 1. Number of states in DTMC with single server

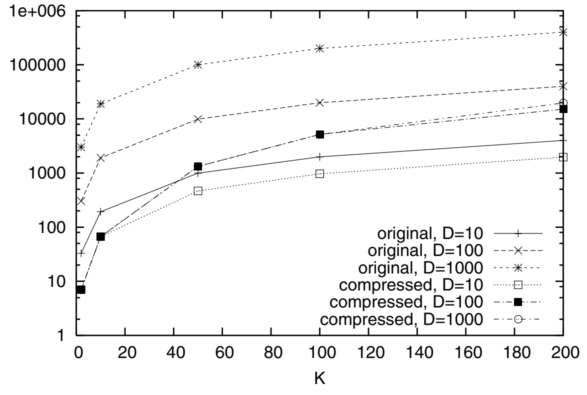

Figure 2. Number of non-zero entries in the DTMC with single server

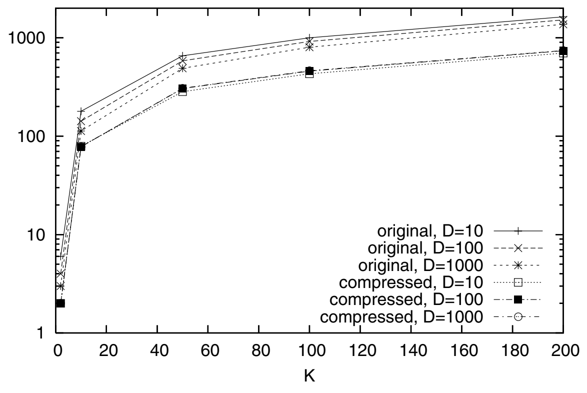

Figure 3. Number of iterations to achieve the results with single serve

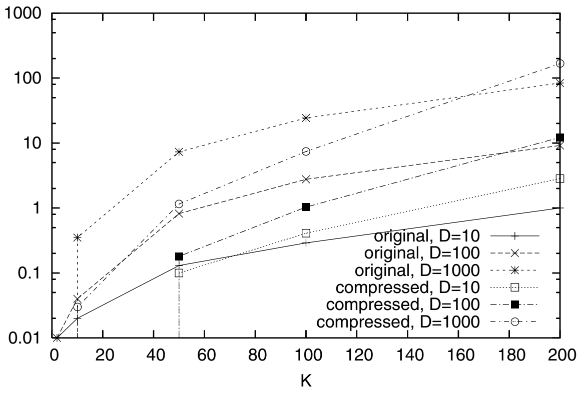

Figure 4. Time to achieve the results with DTMC stored in memory and single server

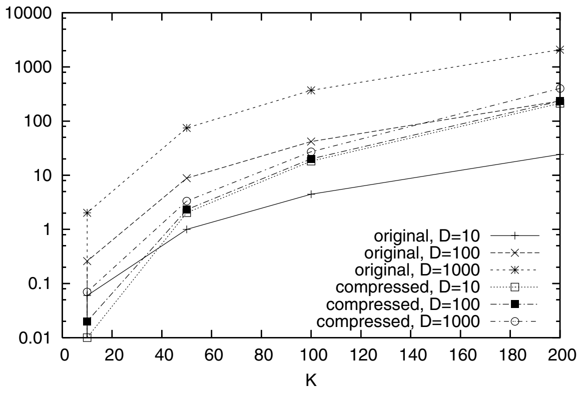

Figure 5. Time to achieve the results without storing the DTMC and with single server

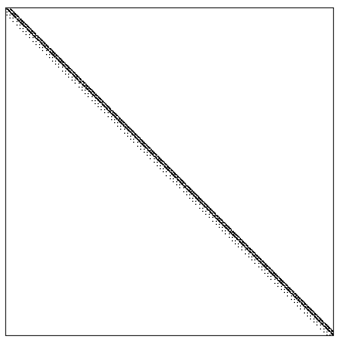

Figure 6. Structure of the complete DTMC with service time 10 and $K = 100$

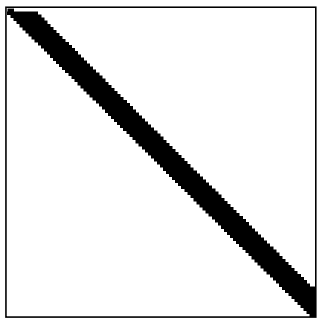

Figure 7. Structure of the reduced DTMC with service time 10 and $K = 100$

Figure 8-12 presents results with 2 servers. Service times of the servers (10,10), (10,100), (10,1000) or (100,100), buffer size is between 2 and 100. Figure 8 present the number of states in the DTMCs. For the cases (10,1000) and (100,100) the number of states are almost equal. The reduction is more significant in case of service times (100,100) than in case of service times (10,1000). This is due to the fact that we eliminate the states in which none of the servers are in their first phase. There are more states of this kind with service times (100,100). For this example the number of non-zero entries in the reduced matrices can be higher than in the original ones. As it can be seen in Figure 10, the number of iterations needed to achieve accuracy is lower for the reduced matrices. Case (10,1000) has particularly slow convergence even for small values of K. Runtimes are reported in Figures 11 and 12. As in the single server case, computations without storing the reduced matrix are slower but the slow down is not drastic.

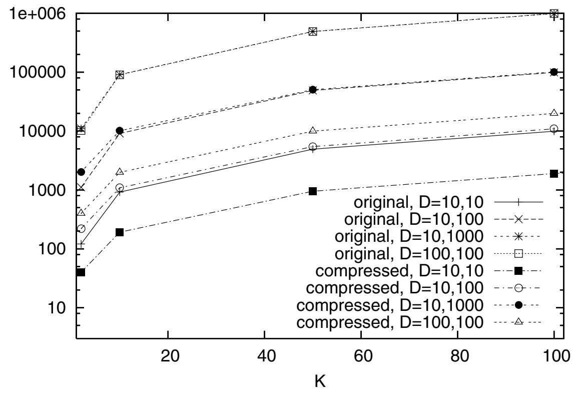

Figure 8. Number of states in DTMC with two servers

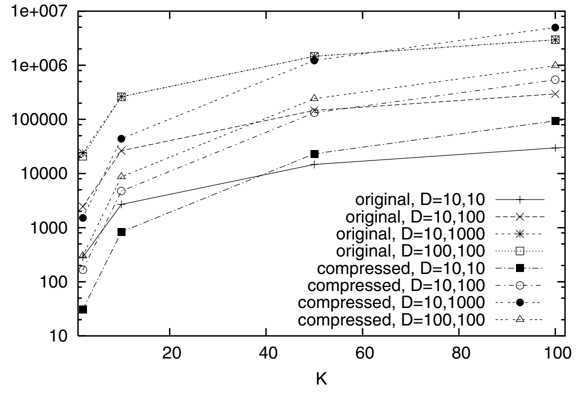

Figure 9. Number of non-zero entries in the DTMC with two servers

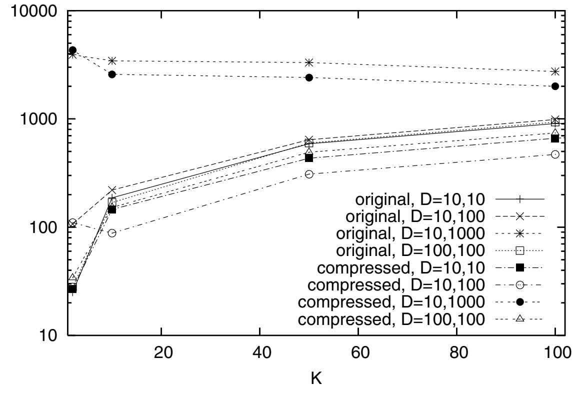

Figure 10. Number of iterations to achieve the results with two servers

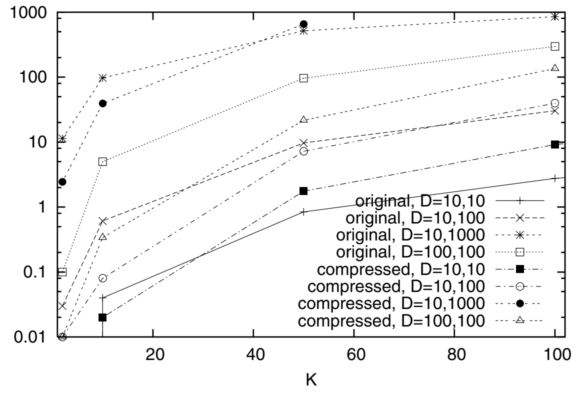

Figure 11. Time to achieve the results with DTMC stored in memory and two servers

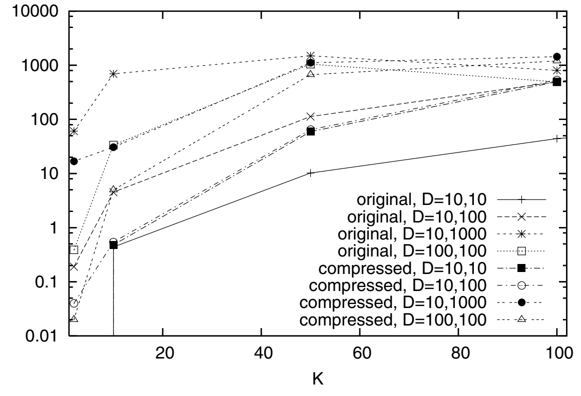

Figure 12. Time to achieve the results without storing the DTMC and with two servers

The original and the reduced matrices with service times (10,10) and $K = 10$ are depicted in Figures 13 and 14. Among the states in which with both servers active, only those are not eliminated in which one of the servers is in its starting phase.

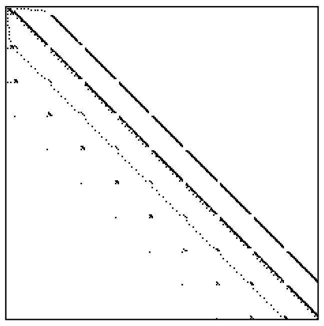

Figure 13. Structure of the original DTMC with service times (10,10) and $K = 10$

Two further cases are reported in Tables 1 and 2. The model parameters are given in the first three columns: number of servers, service times, buffer size. Columns with header COMP contain results for the complete DTMC, those with REDU for the reduced model. Missing entries indicate that the computation could not be performed for lack of memory or was terminated because took more than 1e+05 seconds.

5 Future work

The present version of the algorithm can handle the preemptive repeat different memory policy only,

Model parameters			# States		# Non-zeros		# Iterations		Time to build	
S	D	K	COMP	REDU	COMP	REDU	COMP	REDU	COMP	REDU
2	10000,10000	5	4.00e+08	1.00e+05	1.21e+09	7.80e+05		4.40e+01		2.58e+01
3	100,100,100	10	8.03e+06	2.68e+05	2.31e+07	4.13e+06	1.58e+02	1.53e+02	1.71e+01	1.55e+00

Table 1. Number of states, number of non-zeros in the transition matrix, number of iterations and time to build the transition matrix

Model parameters			Sec. per iteration				Sec. per solution			
			Matrix Built		On the fly		Matrix Built		On the fly	
S	D	K	COMP	REDU	COMP	REDU	COMP	REDU	COMP	REDU
2	10000,10000	5		9.75e-02		7.80e-01		1.06e+02		6.74e+01
3	100,100,100	10		4.34e-01	2.11e+01	7.26e+00		6.79e+01		1.11e+03

Table 2. Time per iteration and time per solution

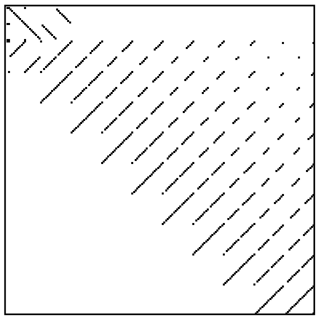

Figure 14. Structure of the reduced DTMC with service times (10,10) and $K = 10$

i.e. when a task is preempted before completion the amount of work done is lost. We plan to investigate how to include preemptive repeat resume and preemptive repeat identical policies [5].

In several cases, as suggested by Figure 14, the resulting reduced matrix is still sparse, and since further application of Gaussian elimination could be performed. We plan to increase the set of eliminated states. This must be done in a way that the entries of the matrix describing the reduced DTMC can still be calculated on the fly.

6 Conclusions

In this paper we have presented a solution technique for discrete time models with geometric and finite support activity duration. DTMC of such model can be built by expansion. This DTMC is subject to state space explosion. The proposed steady state solution technique combines Gaussian elimination and an iterative technique. Gaussian elimination is performed exploiting the structure of the DTMC in such a way that entries of the reduced DTMC can be computed on the fly. This way the steady state solution of the DTMC can be obtained by performing computations on vectors with significantly less number of entries than the number of states of the DTMC. As illustrated by an example, the procedure results in significant memory saving without causing drastic slow down.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-15:832–846, 1989.

[2] A. Bobbio and A. Horváth. Petri nets with discrete phase type timing: A bridge between stochastic and functional analysis. In *Proc. of 2nd Workshop on Models for Time-Critical Systems*, volume 52 No. 3 of *Electronic Notes in Theoretical Computer Science*, Aalborg, Denmark, Aug. 2001.

[3] A. Bobbio, A. Horváth, M. Scarpa, and M. Telek. Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm. *Performance Evaluation*, 54(1):1–32, 2003.

[4] A. Bobbio, A. Horváth, and M. Telek. The scale factor: A new degree of freedom in phase type approximation. *Performance Evaluation*, 56(1–4):121–144, 2004.

[5] A. Bobbio, V.G. Kulkarni, A. Puliafito, M. Telek, and K. Trivedi. Preemptive repeat identical transitions in Markov Regenerative Stochastic Petri Nets. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 113–122. IEEE Computer Society, 1995.

[6] P. Buchholz. Iterative decomposition and aggregation of labelled GSPNs. In *Application and Theory of Petri Nets 1998: 19th International Conference, ICATPN'98*, 1998.

[7] P. Buchholz. Structured analysis approaches for large Markov chains. *Applied Numerical*, 31(4):375–404, 1999.

[8] P. Buchholz. Numerical analysis approaches for large Markov chains: Experiments, observations, and some new results. In *Tutorials at the 2003 Multiconf. on Measurement, Modelling, and Evaluation of Computer-Communication Systems*, 2003. Available at `http://ls4-www.cs.uni-dortmund.de/QM/MA/pb/pb_public.html`.

[9] P. Buchholz. An adaptive decomposition approach for the analysis of stochastic Petri nets. *Performance Evaluation*, 56(1–4):23–52, 2004.

[10] P. Buchholz, G. Ciardo, P. Kemper, and S. Donatelli. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 13(3):203–222, 2000.

[11] J. Campos, M. Silva, and S. Donatelli. Structured solution of asynchronously communicating. *IEEE Transactions on Software Engineering*, 25(2):147–165, 1999.

[12] G. Ciardo, A. S. Miner, and S. Donatelli. Using the exact state space of a model to compute approximate stationary measures. In *Proc. ACM Sigmetrics*, 2000.

[13] D.R. Cox. The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. *Proc. of the Cambridge Phylosophical Society*, 51:433–440, 1955.

[14] J.A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[15] R. German. *Performance analysis of communication systems: Modeling with non-Markovian stochastic Petri nets.* John Wiley and Sons, Chichester, 2000.

[16] M. Gribaudo, M. Sereno, A. Horváth, and A. Bobbio. Fluid stochastic Petri nets augmented with flush-out arcs: Modelling and analysis. *Discrete Event Dynamic Systems*, 11:97–111, 2001.

[17] A. Horváth and M. Telek. Phfit: A general phase-type fitting tool. In *Proc. of 12th Performance TOOLS*, volume 2324 of *Lecture Notes in Computer Science*, Imperial College, London, April 2002.

[18] A. Horváth and M. Telek. Time domain analysis of non-Markovian stochastic Petri nets with pri transitions. *IEEE Transactions On Software Engineering*, 28(20):933–943, 2002.

[19] Jim Kohl. *MatView: Scalable Sparse Matrix Viewer.* `http://www.csm.ornl.gov/~kohl/MatView/` .

[20] C. Lindemann, A. Reuys, and A. Thummler. The DSPNexpress 2.000 performance and dependability modeling environment. In *Proc. of the 29th Int. Symp. on Fault Tolerant Computing*, 1999.

[21] B. Plateau. On the stochastic structure of parallelism and synchronisation models for distributed. *Performance Evaluation Review*, 13:142–154, 1985.

[22] M. Scarpa and A. Bobbio. Kronecker representation of Stochastic Petri nets with discrete PH distributions. In *International Computer Performance and Dependability Symposium - IPDS98*, pages 52–61. IEEE CS Press, 1998.

[23] Tomas Skalicky. *LASPack Reference Manual.* `http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html` .

[24] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* 1994.

[25] M. Telek and A. Horváth. Transient analysis of age-MRSPNs by the method supplementary variables. *Performance Evaluation*, 45(4):205–221, 2001.

[26] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal of Algebraic and Discrete Mathematics*, 2(1):77–79, 1981.