Circulant Preconditioners for Stochastic Automata Networks *

Raymond H. Chan† Wai Ki Ching‡

March 10, 1998

Abstract

Stochastic Automata Networks (SANs) are widely used in modeling communication systems, manufacturing systems and computer systems. The SAN approach gives a more compact and efficient representation of the network when compared to the stochastic Petri nets approach. To find the steady state distribution of SANs, it requires solutions of linear systems involving the generator matrices of the SANs. Very often, direct methods such as the LU decomposition are inefficient because of the huge size of the generator matrices. An efficient algorithm should make use of the structure of the matrices. Iterative methods such as the conjugate gradient methods are possible choices. However, their convergence rates are slow in general and preconditioning is required. We note that the MILU and MINV based preconditioners are not appropriate because of their expensive construction cost. In this paper, we consider preconditioners obtained by circulant approximations of SANs. They have low construction cost and can be inverted efficiently. We prove that if only one of the automata is large in size compared to the others, then our preconditioned systems will converge very fast. Numerical results for three different SANs are given to illustrate the fast convergence of our method.

Key Words: Stochastic Automata Networks, Circulant Preconditioners, Preconditioned Conjugate Gradient Methods.

1 Introduction

Stochastic Automata Networks (SANs) are widely used in modeling queueing systems [5, 6, 7, 29], communication systems [28, 31, 39], manufacturing systems and inventory control [8] and also computer networks [33]. The SAN approach has a more compact and efficient representation when compared to the stochastic Petri nets approach [1, 33]. Moreover because of the special structure of the resulting representations, matrix-vector multiplications involving the generator matrices can usually be done very fast [35].

In analyzing the system performance of a SAN, it is required to find its steady state distribution, which can be obtained by solving a linear system involving the generator matrix of

*Dedicated to Olof Widlund on the occasion of his 60th birthday.

†Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong (rchan@math.cuhk.edu.hk). Research supported in part by the Hong Kong Research Grant Council Grant No. CUHK4207/97P and Chinese University Direct Allocation Grant Grant No. 2060143.

‡Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong and Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong (wkching@se.cuhk.edu.hk).

the SAN. In general the solution cannot be obtained efficiently by direct methods such as the LU decomposition due to the huge size of the generator matrix. Efficient numerical algorithms should make use of the special structures of the generator matrices and their fast matrix-vector multiplications. The conjugate gradient type methods [2, 4, 35] are possible choices. However, their convergence rates are slow in general. To speed up the convergence rate, we consider preconditioned conjugate gradient methods. We note that the MILU [30, 37] and MINV [25, 26] based preconditioners are not appropriate due to their expensive construction costs.

One of the early applications of preconditioned conjugate gradient methods in solving queueing networks was done by Chan [9, 10]. For Markovian overflow networks with traffic density close to 1, the generator matrices are close to the discretization matrices of elliptic equations. Using techniques from elliptic equations, such as the fast Poisson solvers and domain decomposition methods [11], Chan has constructed efficient preconditioners for these networks. These preconditioners make use of the tensor structure of the generator matrices and are easy to construct and invert.

Toeplitz matrices are matrices with constant diagonal entries. Circulant matrices are Toeplitz matrices such that each column is a cyclic shift of its preceding column. One important property of circulant matrices is that they can be diagonalized by Fast Fourier Transforms [27]. Hence their inverses can be found easily. Circulant matrices have shown to be good preconditioners for Toeplitz systems in many applications [13] and in particular in queueing networks, see for instance [12, 14, 15, 16, 20]. The main observation in queueing network applications is that most queueing networks have generator matrices that are close to Toeplitz matrices. These include sophisticated networks such as the Markov modulated Poisson processes arising in manufacturing systems and inventory control systems and also networks with more general queueing disciplines such as batch arrivals.

In this paper, we consider circulant preconditioners for networks under a more general setting, the SANs. The circulant preconditioners introduced here are easy to construct and can be inverted efficiently. We prove that if only one of the automata is large in size compared to the others, then our preconditioned systems will converge very fast. We illustrate the efficiency of our methods by applying it to three practical SANs.

The paper is organized as follows. In §2, we first introduce a two-queue overflow network which is a particular example of SANs. In §3, we give an introduction of SANs. We then construct our circulant preconditioners for SANs in §4. In §5, we give a convergence analysis for our preconditioners. In §6, we test our preconditioners for three practical examples of SANs. Finally, concluding remarks are given in §7.

2 An Overflow Queueing Network

Let us begin with some notations. We will use $\mathbf{0}$ and $\mathbf{1}$ to denote the zero column vector and the column vector of all ones of appropriate length respectively. Also we will use O and I to denote the zero and identity matrices of appropriate size respectively. For any matrix A, $z(A)$ will denote the number of nonzero columns in A. A matrix A is said to be nonnegative, denoted by $A \geq O$, if all the entries of A are nonnegative.

To introduce the terminologies and notations of SANs, let us consider a simple example of SANs with 2 automata. It is the 2-queue overflow network considered in [29, 9]. The network consists of two queues (*automata*) with exogenous Poisson arrivals and exponential servers. Whenever queue 2 is full, the arriving customers will overflow to queue 1 if it is not yet full. Otherwise the customers will be blocked and lost, see Figure 1.

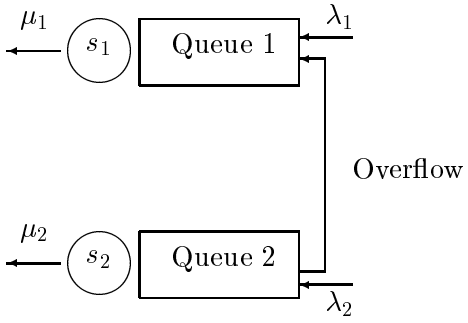

Figure 1. The Two Queue Overflow System.

For queue i, $i = 1, 2$, let λ_i be the exogenous input rate, μ_i the service rate, s_i the number of servers, and $l_i - s_i - 1$ the buffer sizes. Then the generator matrix for the queueing system is given by

$$A = Q_1 \otimes I_{l_2} + I_{l_1} \otimes Q_2 + R \otimes \mathrm{Diag}(0, \cdots, 0, 1), \tag{1}$$

where

$$Q_i = \begin{pmatrix} \lambda_i & -\mu_i & & & & & & 0 \\ -\lambda_i & \lambda_i + \mu_i & -2\mu_i & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -\lambda_i & \lambda_i + s_i\mu_i & -s_i\mu_i & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -\lambda_i & \lambda_i + s_i\mu_i & -s_i\mu_i & \\ 0 & & & & & -\lambda_i & s_i\mu_i & \end{pmatrix}, \quad i = 1, 2, \tag{2}$$

$$R = \begin{pmatrix} \lambda_2 & & & & 0 \\ -\lambda_2 & \lambda_2 & & & \\ & -\lambda_2 & \ddots & & \\ & & \ddots & \lambda_2 & \\ 0 & & & -\lambda_2 & 0 \end{pmatrix} \tag{3}$$

and I_{l_i} is the identity matrix of size l_i, see [29] for instance.

We note that the matrix Q_i in (2) corresponds to the generator matrix of a stand-alone queue i and hence the matrix $(Q_1 \otimes I_{l_2} + I_{l_1} \otimes Q_2)$ in (1) corresponds to a 2-queue network where no overflow can occur. It is called the *non-interlacing* part of the network. The last term $R \otimes \mathrm{Diag}(0, \cdots, 0, 1)$ in (1) corresponds to where the overflows (or *transitions*) occur. In general, SANs are composed of the non-interlacing part of the network together with the transitions allowed. For our 2-queue overflow network, the queueing disciplines are governed by three probabilistic rules, namely, the Markovian input-output processes of queues 1 and 2 (the non-interlacing part) and the overflow process from queue 2 to queue 1. In fact, the generator matrix A in (1) can be written in the form

$$A = \sum_{i=1}^{3} \bigotimes_{j=1}^{2} Q_{ij}$$

where $Q_{11}, Q_{21}, Q_{31}, Q_{12}, Q_{22}$ and Q_{32} are $Q_1, I_{l_1}, R, I_{l_2}, Q_2$ and $\mathrm{Diag}(0, 0 \cdots, 0, 1)$ respectively.

We note that A can be rewritten in a more complicated form, which is however the standard form of SANs:

$$A = \sum_{i=1}^{3} \left\{ \bigotimes_{j=1}^{2} D_{ij} - \bigotimes_{j=1}^{2} E_{ij} \right\}, \tag{4}$$

where

$$D_{ii} = \begin{pmatrix} \lambda_i & & & & & & & 0 \\ & \lambda_i + \mu_i & & & & & & \\ & & \ddots & & & & & \\ & & & \lambda_i + s_i\mu_i & & & & \\ & & & & \ddots & & & \\ & & & & & \lambda_i + s_i\mu_i & & \\ 0 & & & & & & & s_i\mu_i \end{pmatrix}, \quad i = 1, 2, \tag{5}$$

$$E_{ii} = \begin{pmatrix} 0 & \mu_i & & & & & & 0 \\ \lambda_i & 0 & 2\mu_i & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \lambda_i & 0 & s_i\mu_i & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \lambda_i & 0 & s_i\mu_i & \\ 0 & & & & & \lambda_i & 0 & \end{pmatrix}, \quad i = 1, 2, \tag{6}$$

$$D_{31} = \begin{pmatrix} \lambda_2 & & & & 0 \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_2 & \\ 0 & & & & 0 \end{pmatrix}, \qquad E_{31} = \begin{pmatrix} 0 & & & & 0 \\ \lambda_2 & 0 & & & \\ & \lambda_2 & \ddots & & \\ & & \ddots & 0 & \\ 0 & & & \lambda_2 & 0 \end{pmatrix}, \tag{7}$$

$$D_{12} = E_{12} = I_{l_2}, \quad D_{21} = E_{21} = I_{l_1}, \quad \text{and} \quad D_{32} = E_{32} = \mathrm{Diag}(0, \cdots, 0, 1). \tag{8}$$

We see that in this standard form, D_{ij} and $E_{ij} \geq O$, and that D_{ij} are diagonal matrices with diagonal entries equal to the column sums of E_{ij}. In particular, D_{ij} and E_{ij} have the same column sums, i.e. $\mathbf{1}^t D_{ij} = \mathbf{1}^t E_{ij}$ for $i = 1, 2, 3,\ j = 1, 2$.

To analyze the 2-queue network, we need to find its steady state distribution vector, which is the normalized right null vector of A. More precisely, the vector $\mathbf{p}$ is the nonnegative vector that satisfies $A\mathbf{p} = \mathbf{0}$ and $\mathbf{1}^t\mathbf{p} = 1$. Classical methods such as the block Gauss-Seidel method and the successive over-relaxation method are standard methods for solving this problem, see [29]. In [9, 14], the preconditioned conjugate gradient method is used and a very efficient preconditioner is constructed for networks where the traffic is about balance, i.e. $\lambda_i \approx s_i\mu_i$.

However, not all systems have balance traffic. In this paper, we are interested in the case where only one automaton is relatively large in size. In the 2-queue network above, since overflow is permitted only from queue 2 to queue 1, the performance of queue 1 is important. It is thus interesting to find $\mathbf{p}$ when the queue length of queue 1 is large. In many practical applications, only one automaton, say the one corresponding to the inventory level of finished products, is large in size. In the next three sections, we will construct and analyze preconditioners that work well under this situation.

3 Stochastic Automata Networks

A Stochastic Automata Network (SAN) consists of a number of individual stochastic automata. Each automaton is represented by a number of states and probabilistic rules that govern the transitions from one state to another. The state of an automaton at time t is just the state it occupies at time t and the state of the SAN at time t is given by the states of its constituent automata. For more details of SANs, see [5, 6, 7] for instance.

Consider a SAN with n automata and m probabilistic rules. Let the state space of the ith automaton be of size $l_i - 1$. The generator matrix A of the SAN can be written in the standard form:

$$A = \sum_{i=1}^{m} \left\{ \bigotimes_{j=1}^{n} D_{ij} - \bigotimes_{j=1}^{n} E_{ij} \right\}, \tag{9}$$

where D_{ij} and E_{ij} are of size l_j-by-l_j, see [5, 33, 35] and cf. (4). Here $E_{ij} \geq O$ are such that each term $\bigotimes_{j=1}^{n} E_{ij}$ represents the rate of certain transitions amongst the states. The D_{ij} are nonnegative diagonal matrices that contain the column sums of E_{ij}, i.e.

$$\mathbf{1}^t D_{ij} = \mathbf{1}^t E_{ij}, \quad i = 1, \ldots, m, \ j = 1, \ldots n. \tag{10}$$

For simplicity, let the non-interlacing part of the network be represented by the first n terms in (9). Thus, $D_{ii} - E_{ii}$ is the generator matrix of the ith automaton alone and $D_{ij} = E_{ij} = I$, for $1 \leq i \neq j \leq n$ (cf. the first two terms in (1)). More precisely, the first n terms of (9) can be written as

$$(D_{11} - E_{11}) \otimes I \otimes \cdots \otimes I + I \otimes (D_{22} - E_{22}) \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes (D_{nn} - E_{nn}). \tag{11}$$

For ease of discussion, we assume that the generator matrices $D_{ii} - E_{ii}$, $i = 1, \ldots, n$, of the individual automata are irreducible. As D_{ii} are diagonal matrices, E_{ii} are therefore irreducible. Clearly from (11), we see that $\sum_{i=1}^{n} \bigotimes_{j=1}^{n} E_{ij}$ is irreducible. Using the fact that $E_{ij} \geq O$, we see that $\sum_{i=1}^{m} \bigotimes_{j=1}^{n} E_{ij}$ and therefore A in (9) are irreducible too. We thus have proved the following lemma.

Lemma 1 *If the generator matrix of each of the individual automata in a SAN is irreducible then the generator matrix of the SAN (i.e. A in (9)) is also irreducible.*

To analyze the network, we need to find its steady state distribution vector $\mathbf{p}$, which is the normalized right null vector of A. We note that if $n = m$, i.e. the network consists only of the non-interlacing part (11), then $\mathbf{p}$ can be obtained easily by taking the tensor product of the steady state distribution vectors of the individual automata. For the general case where $m > n$, the existence of $\mathbf{p}$ follows from the irreducibility of the matrix A, as the following lemma shows.

Lemma 2 *Let B be an irreducible matrix of the form*

$$B = \sum_{i=1}^{m} \left\{ \bigotimes_{j=1}^{n} U_{ij} - \bigotimes_{j=1}^{n} V_{ij} \right\},$$

where $V_{ij} \geq O$ and U_{ij} are diagonal matrices such that

$$\mathbf{1}^t U_{ij} = \mathbf{1}^t V_{ij}, \quad i = 1, \ldots, m, \ j = 1, \ldots n. \tag{12}$$

Then B has a 1-dimensional null space with positive null vectors.

Proof: It follows from (12) that $\mathbf{1}^t(\bigotimes_{j=1}^n U_{ij}) = \mathbf{1}^t(\bigotimes_{j=1}^n V_{ij})$ and hence $\mathbf{1}^t B = \mathbf{0}^t$, i.e. B has zero column sums. Clearly B has nonpositive off-diagonal entries. In particular, the column sum of the off-diagonal entries of any column of B cannot be positive. Since B is irreducible, these column sums cannot be zero either. Because B has zero column sums, the diagonal entries of B are therefore positive. Let D be the diagonal matrix containing the diagonal entries of B. Then $I - BD^{-1}$ is an irreducible column stochastic matrix. The lemma now follows from the Perron-Frobenius theorem, see for instance [3, p.27]. □

By Lemma 2, we see that the steady state distribution vector $\mathbf{p}$ for A in (9) exists and is unique. Moreover, all the entries in $\mathbf{p}$ are positive. Classical iterative methods such as the block Gauss-Seidel method and the successive over-relaxation method are standard methods for finding $\mathbf{p}$ [29, 31, 35]. However, in this paper, we consider preconditioned conjugate gradient methods [2, p.18]. To speed up the convergence, we need efficient preconditioners, i.e. preconditioners that can be constructed and inverted easily and can speed up the convergence rate. The choice of the preconditioners depend on what kind of systems we are considering. In many practical situations [15, 16, 20, 21, 22, 33], only one automaton will have a very large state space. A typical example is when the automaton is corresponding to the inventory level of finished products in a manufacturing system [15, 16, 17]. In this paper, we will consider SANs with only one automaton having a very large state space and without loss of generality, we assume that it is the first one. More precisely, we will analyze the convergence of our preconditioners under the limit that l_1 is large. We will use the symbol $\emptyset(l_1)$ to denote constants that are less than l_1 and are independent of l_1.

We note that because $\mathbf{p}$ exists and is unique and positive, it can be obtained by normalizing the solution $\mathbf{x}$ in the matrix equation

$$G\mathbf{x} \equiv (A + \mathbf{e}\mathbf{e}^t)\mathbf{x} = \mathbf{e}, \tag{13}$$

where $\mathbf{e} = (0, 0, \cdots, 0, 1)^t$, i.e. $\mathbf{p} = \mathbf{x}/(\mathbf{1}^t\mathbf{x})$. In the following, we will consider preconditioners for G. Clearly, we have

$$\text{rank}(G - A) = 1. \tag{14}$$

4 Circulant Preconditioners for SAN

Toeplitz matrices are matrices with constant diagonal entries. Circulant matrices are Toeplitz matrices such that each column is a cyclic shift of its preceding column. One important property of circulant matrices is that they can be diagonalized by Fast Fourier Transforms [27]. Hence their inverses can be found easily. Circulant preconditioners have been used in many applications where the Toeplitz matrices come into play, such as in image processing, partial differential equations, integral equations and in particular queueing networks, see [13] and the references therein. In this section, we consider the construction of circulant preconditioners for SANs.

The success of our preconditioners depends on the observation that in many network applications, the matrices D_{ij} and E_{ij} in (9) are low rank perturbations of Toeplitz matrices (cf. (5)–(8)). Hence they can be approximated well by circulant matrices. Because we assume that the first automaton is the one with the largest state space, our idea of constructing the preconditioners is to solve the first automaton and its related disciplines approximately and the remaining automata exactly. More precisely, we will approximate the matrices E_{i1} in (9) by nonnegative circulant matrices $c(E_{i1})$ that are low rank perturbations of E_{i1}.

For example, if E_{i1} takes the form of E_{ii} in (6), its circulant approximation will be given by

$$c(E_{ii}) = \begin{pmatrix} 0 & s_i\mu_i & & & & & \lambda_i \\ \lambda_i & 0 & s_i\mu_i & & 0 & & \\ & \ddots & \ddots & \ddots & & & \\ & & \lambda_i & 0 & s_i\mu_i & & \\ & 0 & & \ddots & \ddots & \ddots & \\ & & & & \lambda_i & 0 & s_i\mu_i \\ s_i\mu_i & & & & & \lambda_i & 0 \end{pmatrix}. \tag{15}$$

It is a rank s perturbation of E_{ii}. In fact, the number of nonzero columns of $(c(E_{ii}) - E_{ii})$, denoted by $z(c(E_{ii}) - E_{ii})$, is equal to $s + 1$. For E_{31} in (7), we define

$$c(E_{31}) = \begin{pmatrix} 0 & & & & \lambda_2 \\ \lambda_2 & 0 & & & \\ & \lambda_2 & \ddots & & \\ & & \ddots & 0 & \\ 0 & & & \lambda_2 & 0 \end{pmatrix}.$$

Hence $\mathrm{rank}(c(E_{31}) - E_{31}) = z(c(E_{31}) - E_{31}) = 1$. For the other E_{ij} in (8), we simply define $c(I) = I$ and $c(\mathrm{Diag}(0, \cdots, 0, 1)) = O$.

As another example, for automata with batch arrivals, the transition matrix E will be of the form

$$E = \begin{pmatrix} 0 & \mu & & & & & 0 \\ \lambda_1 & 0 & 2\mu & & & & \\ \vdots & \lambda_1 & 0 & \ddots & & & \\ \lambda_b & \vdots & \ddots & \ddots & s\mu & & \\ & \lambda_b & \ddots & \ddots & 0 & s\mu & \\ & & \lambda_b & \ddots & \lambda_1 & 0 & s\mu \\ 0 & & & r_b & \cdots & r_1 & 0 \end{pmatrix}, \tag{16}$$

where b is the largest possible batch size, λ_j is the arrival rate of batches with size j and $r_j = \sum_{i=j}^{b} \lambda_i$ for $j = 1, \ldots, b$. Here we define

$$c(E) = \begin{pmatrix} 0 & s\mu & & 0 & \lambda_b & \cdots & \lambda_1 \\ \lambda_1 & 0 & s\mu & & \ddots & \ddots & \vdots \\ \vdots & \lambda_1 & 0 & \ddots & & \ddots & \lambda_b \\ \lambda_b & \vdots & \ddots & \ddots & s\mu & & 0 \\ & \lambda_b & \ddots & \ddots & 0 & s\mu & \\ 0 & & \lambda_b & \ddots & \lambda_1 & 0 & s\mu \\ s\mu & 0 & & \lambda_b & \cdots & \lambda_1 & 0 \end{pmatrix}.$$

In this case $\mathrm{rank}(E - c(E)) = \max(s, b) + 1$ and $z(E - c(E)) = s + b$.

With these examples in mind, we are now ready to define our circulant approximations.

Definition 1 *For $i = 1, \ldots, m$, $c(E_{i1})$ is defined to be the circulant matrix such that (i) the number of nonzero columns of $E_{i1} - c(E_{i1})$ is a constant less than l_1 and is independent of l_1, i.e.*

$$z(E_{i1} - c(E_{i1})) = \emptyset(l_1), \tag{17}$$

and (ii) $c(E_{i1})$ is a nonnegative matrix, i.e.

$$c(E_{i1}) \geq O. \tag{18}$$

We remark that requirements (17) and (18) are very general. The examples of $c(E_{i1})$ given above and also those in §6 all satisfy these two requirements provided that the queueing parameters such as s_i, λ_i, μ_i and b ($< l_1 - s$) are all independent of l_1. In general, there are many different forms of E_{i1} depending on the SAN itself. However, in many applications of the SANs [5, 6, 7, 16, 15, 20, 21, 22, 33], the transition matrix E_{11} of the main automaton takes the form of E_{ii} in (6), which is the transition matrix of an (M/M/s_i/l_i) queue. It has been shown in [32, 34] that the time spent (service time) in a jobshop is asymptotically exponentially distributed. Thus we may approximate a complex automaton with Poisson arrival process by E_{ii} of the form in (6) and $c(E_{ii})$ so defined in (15) will satisfy (17) and (18).

Next we define the circulant approximations $c(D_{i1})$ for the diagonal matrices D_{i1}. Similar to (10), we define the diagonal entries of $c(D_{i1})$ to be the column sums of $c(E_{i1})$, which are the same for all columns, as $c(E_{i1})$ are circulant matrices.

Definition 2 *For $i = 1, \ldots, m$, $c(D_{i1})$ is defined to be the constant diagonal matrix such that*

$$\mathbf{1}^t c(D_{i1}) = \mathbf{1}^t c(E_{i1}), \quad i = 1, \ldots, m. \tag{19}$$

Recall that the diagonal entries of D_{i1} are the column sums of E_{i1}. Hence by (17), we see that

$$z(D_{i1} - c(D_{i1})) = \emptyset(l_1), \quad i = 1, \ldots m. \tag{20}$$

To construct our preconditioner, let us first define (cf (9))

$$c(A) = \sum_{i=1}^{m} \left\{ c(D_{i1}) \bigotimes_{j=2}^{n} D_{ij} - c(E_{i1}) \bigotimes_{j=2}^{n} E_{ij} \right\}, \tag{21}$$

We claim that $c(A)$ is irreducible.

Lemma 3 *If the generator matrix $D_{ii} - E_{ii}$, $i = 1, \ldots, n$, of each stand-alone automaton is irreducible and that $c(E_{i1})$ satisfy (17) and (18) for $i = 1, \ldots, m$. Then $c(A)$ is irreducible. In particular, $c(A)$ has a one dimensional null space with positive null vectors.*

Proof: By assumption, E_{ii} are irreducible for $i = 1, \ldots, n$. We claim that $c(E_{11})$ is irreducible. For if not, then since it is a circulant matrix, it can only be a constant diagonal matrix. Then by (17), E_{11} is the sum of a diagonal matrix and a matrix with $\emptyset(l_1)$ nonzero columns. Since $\emptyset(l_1) < l_1$, E_{11} cannot be irreducible, a contradiction. Thus $c(E_{11})$ is irreducible.

By the definition of $c(A)$ in (21) and the fact that the first n terms of A are given in (11), we see that the first n terms of $c(A)$ will be of the form

$$(c(D_{11}) - c(E_{11})) \otimes I \otimes \cdots \otimes I + I \otimes (D_{22} - E_{22}) \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes (D_{nn} - E_{nn})$$

which is clearly irreducible. In particular, $\sum_{i=1}^{n} \{c(E_{i1}) \bigotimes_{j=2}^{n} E_{ij}\}$ is also irreducible. Since E_{ij} and $c(E_{ij}) \geq O$ for all i and j, $\sum_{i=1}^{m} \{c(E_{i1}) \bigotimes_{j=2}^{n} E_{ij}\}$ and hence $c(A)$ are irreducible too. By

applying Lemma 2 to $c(A)$, we see that $c(A)$ has a one dimensional null space with positive null vectors. □

Since $c(A)$ is singular, we cannot use it as a preconditioner. Our preconditioner is constructed by perturbing $c(A)$ by a rank one matrix, similar to what we did in (13). In order to do it systematically, let us first look closely to the eigenvalues of $c(E_{i1})$ and $c(D_{i1})$. Recall that $c(D_{i1})$ are constant diagonal matrices, thus we may write

$$c(D_{i1}) = d_i I, \qquad i = 1, \ldots, m.$$

Lemma 4 *Let F be the Fourier matrix of size l_1, i.e. the (j,k)th entry of F is given by $\exp(2\pi\sqrt{-1}jk/l_1)/\sqrt{n}$. For $i = 1, \ldots, m$, $c(E_{i1})$ can be diagonalized by F:*

$$F^* c(E_{i1}) F = \mathrm{Diag}(t_{i1}, t_{i2}, \cdots, t_{il_1}). \tag{22}$$

Moreover,

$$t_{il_1} = d_i, \quad i = 1, \ldots, m. \tag{23}$$

Proof: Equation (22) follows from the fact that any circulant matrices can be diagonalized by the Fourier matrix of the same size, see [27]. To get (23), we first note that the last column of F is $\frac{1}{\sqrt{n}}\mathbf{1}$, i.e. $F\mathbf{e} = \frac{1}{\sqrt{n}}\mathbf{1}$, where $\mathbf{e} = (0, 0, \cdots, 0, 1)^t$. Thus by (19), we have

$$t_{il_1} = \mathbf{e}^t F^* c(E_{i1}) F = \frac{1}{\sqrt{n}}\mathbf{1}^t c(E_{i1}) F = \frac{1}{\sqrt{n}}\mathbf{1}^t c(D_{i1}) F = \frac{d_i}{\sqrt{n}}\mathbf{1}^t F = d_i, \quad i = 1, \ldots, m.$$

□

Using (22), we then have

$$\begin{aligned} & (F^* \otimes I) c(A) (F \otimes I) \\ = & \sum_{i=1}^{m} \left\{ d_i I \bigotimes_{j=2}^{n} D_{ij} - \mathrm{Diag}(t_{i1}, t_{i2}, \cdots, t_{il_1}) \bigotimes_{j=2}^{n} E_{ij} \right] \\ = & \mathrm{Diag}\left[\sum_{i=1}^{m} \left\{ d_i \bigotimes_{j=2}^{n} D_{ij} - t_{i1} \bigotimes_{j=2}^{n} E_{ij} \right\}, \cdots, \sum_{i=1}^{m} \left\{ d_i \bigotimes_{j=2}^{n} D_{ij} - t_{il_1} \bigotimes_{j=2}^{n} E_{ij} \right\} \right], \end{aligned} \tag{24}$$

which is a diagonal block matrix. Using (23), the last diagonal block in (24) becomes

$$\sum_{i=1}^{m} d_i \left\{ \bigotimes_{j=2}^{n} D_{ij} - \bigotimes_{j=2}^{n} E_{ij} \right\}.$$

Premultiplying $\mathbf{1}^t$ to this matrix we get,

$$\mathbf{1}^t \cdot \sum_{i=1}^{m} d_i \left\{ \bigotimes_{j=2}^{n} D_{ij} - \bigotimes_{j=2}^{n} E_{ij} \right\} = \sum_{i=1}^{m} d_i \left\{ \mathbf{1}^t \cdot \bigotimes_{j=2}^{n} D_{ij} - \mathbf{1}^t \cdot \bigotimes_{j=2}^{n} E_{ij} \right\} = \mathbf{0}^t,$$

where the last equality follows from (10). Thus the last diagonal block in (24) is a singular matrix.

Since by Lemma 3, $c(A)$ has only a one-dimensional null space, the last diagonal block in (24) is the only singular block. All the other diagonal blocks in (24) are nonsingular. Similar to

the proof in Lemma 1, we can easily prove that this last diagonal block is an irreducible matrix. Hence by Lemma 2, it also has a one dimensional null space with positive null vectors. To get our nonsingular preconditioner, we replace this last block by a nonsingular matrix using a rank 1 perturbation, as we did in (13):

$$H \equiv \left[\sum_{i=1}^{m} d_i \left\{ \bigotimes_{j=2}^{n} D_{ij} - \bigotimes_{j=2}^{n} E_{ij} \right\} + \mathbf{e}\mathbf{e}^t \right].$$

Thus our preconditioner is defined as

$$\begin{aligned} C \quad \equiv \quad (F \otimes I)\mathrm{Diag} \Bigg[& \sum_{i=1}^{m} \left\{ d_i \bigotimes_{j=2}^{n} D_{ij} - t_{i1} \bigotimes_{j=2}^{n} E_{ij} \right\}, \cdots, \\ & \sum_{i=1}^{m} \left\{ d_i \bigotimes_{j=2}^{n} D_{ij} - t_{i(l_1-1)} \bigotimes_{j=2}^{n} E_{ij} \right\}, H \Bigg] (F^* \otimes I). \end{aligned} \tag{25}$$

By the above arguments, C is clearly nonsingular. Moreover,

$$\mathrm{rank}(c(A) - C) = 1. \tag{26}$$

In the following, we will use C in (25) to precondition (13), i.e. instead of solving (13), we solve

$$(GC^{-1})\mathbf{y} = \mathbf{e} \quad \text{where} \quad \mathbf{x} = C^{-1}\mathbf{y}, \tag{27}$$

by conjugate gradient type methods [2, 36]. The cost per iteration of these methods depends on the cost of the matrix-vector multiplications of the form $G\mathbf{u}$ and also the cost of solving $C\mathbf{v} = \mathbf{u}$. In multiplying $G\mathbf{u}$, we can make use of the tensor structure of A as given in (9) and also the special structure of the transition matrices E_{ij}. Usually, E_{ij} are either sparse or near-Toeplitz matrices, cf. (5)–(8) and (16). The cost is therefore either of order $O(l_1)$ or $O(l_1 \log l_1)$.

The main cost for solving the preconditioner system $C\mathbf{v} = \mathbf{u}$ comes from (i) the matrix-vector multiplications by the Fast Fourier Transform (see (25)) and (ii) solving the diagonal block systems in (25). The cost for (i) is of $O(l_1 \log l_1)$. The cost for (ii) depends on the structure of the individual blocks in (25) which are of size $\prod_{i=2}^{n} l_i$-by-$\prod_{i=2}^{n} l_i$. Again fast algorithms for solving the block systems should make use of the sparse or near-Toeplitz structure of the blocks. In any case, the cost of solving each block system will be independent of l_1. As there are l_1 diagonal blocks in (25), the total cost will be of order $O(l_1)$. Hence the total cost for solving the preconditioner system is of $O(l_1 \log l_1 + l_1)$. Clearly, one can speed up (ii) by solving the diagonal block systems in parallel.

Take the 2-queue overflow network in §2 as an example again. Each individual diagonal block in (25) is a tridiagonal matrix which can be solved in $O(l_2)$ operations. Thus the cost per iteration is $O(l_1 \log l_1 + l_1 l_2)$ operations.

5 Convergence Analysis

In this section, we prove that if all the system parameters such as n, m, λ_i, μ_i, s_i, $(i = 1, \cdots, n)$, and l_j, $(j = 2, \cdots, n)$ are fixed and independent of l_1, then the preconditioned system GC^{-1} in (27) has singular values clustered around 1 as l_1 tends to infinity. Hence when conjugate gradient type methods are applied to solving the preconditioned system (27), we expect fast convergence. Numerical examples will be given in §6 to illustrate this fast convergence.

Theorem 1 *Suppose all the system parameters such as n, m, λ_i, μ_i, s_i, $(i = 1, \cdots, n)$, and l_j, $(j = 2, \cdots, n)$ are fixed and independent of l_1, and that $c(E_{i1})$ satisfy (17) and (18) for $i = 1, \ldots, m$. Then the preconditioned matrix GC^{-1} has at most $\{4 + 4(\sum_{i=1}^{m} \emptyset(l_1)) \prod_{i=2}^{n} l_i\}$ singular values not equal to 1.*

Proof: We first notice that by (9) and (21),

$$A - c(A) = \sum_{i=1}^{m} \left\{ (D_{i1} - c(D_{i1})) \bigotimes_{j=2}^{n} D_{ij} - (E_{i1} - c(E_{i1})) \bigotimes_{j=2}^{n} E_{ij} \right\}.$$

Therefore by (17) and (20),

$$\mathrm{rank}(A - c(A)) \leq \sum_{i=1}^{m} (z(D_{i1} - c(D_{i1})) \prod_{i=2}^{n} l_i + \sum_{i=1}^{m} (z(E_{i1} - c(E_{i1})) \prod_{i=2}^{n} l_i \leq 2(\sum_{i=1}^{m} \emptyset(l_1)) \prod_{i=2}^{n} l_i.$$

Hence by (14) and (26), we have

$$\mathrm{rank}(G - C) \leq \mathrm{rank}(G - A) + \mathrm{rank}(A - c(A)) + \mathrm{rank}(c(A) - C) \leq 2 + 2(\sum_{i=1}^{m} \emptyset(l_1)) \prod_{i=2}^{n} l_i.$$

If we write

$$GC^{-1} = I + (G - C)C^{-1} \equiv I + L,$$

then $\mathrm{rank}(L) \leq 2 + 2(\sum_{i=1}^{m} \emptyset(l_1)) \prod_{i=2}^{n} l_i$. Therefore

$$C^{-*}G^{*}GC^{-1} - I = L^{*}(I + L) + L,$$

is a matrix of rank at most $4 + 4(\sum_{i=1}^{m} \emptyset(l_1)) \prod_{i=2}^{n} l_i$. □

Thus the number of singular values of GC^{-1} that are distinct from 1 is a constant independent of l_1. In order to show fast convergence of preconditioned conjugate gradient type methods with the preconditioner C, one still needs an estimate of $\sigma_{\min}(GC^{-1})$, the smallest singular value of GC^{-1}. If $\sigma_{\min}(GC^{-1})$ is uniformly bounded away from zero independent of l_1, then the method converges in $O(1)$ iterations; and if $\sigma_{\min}(GC^{-1})$ decreases like $O(l_1^{-\alpha})$ for some $\alpha > 0$, then the method converges in at most $O(\log l_1)$ steps, see [38] or [9, Lemma 3.8.1].

Suppose it turns out that $\sigma_{\min}(GC^{-1})$ decreases in an order faster than $O(l_1^{-\alpha})$ for any $\alpha > 0$, such as like $O(e^{-l_1})$. Then (27) is ill-posed. However, we can still have a fast convergence rate. The key step is to consider a regularized equation of (27):

$$C^{-*}(G^{*}G + l_1^{-4-\beta} I)C^{-1}\mathbf{u} = C^{-*}G^{*}\mathbf{e} \tag{28}$$

where β is any positive constant. We can prove that the regularized preconditioned matrix

$$C^{-*}(G^{*}G + l_1^{-4-\beta} I)C^{-1}$$

has eigenvalues clustered around 1 and its smallest eigenvalue decreases at a rate no faster than $O(l_1^{-4-\beta})$. Hence preconditioned conjugate gradient type methods will converge in at most $O(\log l_1)$ steps when applied to solving the preconditioned linear system (28). Moreover, we can prove that the 2-norm of the error introduced by the regularization tends to zero at a rate of $O(l_1^{-\beta})$. The proof for this regularization technique is similar to that given in [15, 20] and is therefore omitted here. We remark that in all the examples in §6, and also in many other problems, see for instance [15, 16, 20, 21, 22], we can obtain fast convergence without using the regularization technique.

6 Practical Examples

In this section, we consider three practical SANs and compare our preconditioning method discussed in §4 with a classical method, the block Gauss-Seidel (BGS) method [3, p.174]. The examples come from queueing systems, communication systems and manufacturing systems.

Since all the generator matrices considered here are non-symmetric, we employ a generalized conjugate gradient method, the Conjugate Gradient Squared (CGS) method [36], to solve the system (27). The method does not require the transpose of the iteration matrix GC^{-1}. The stopping criteria for all iterative methods is $||A\mathbf{x}^k||_2 \leq 10^{-10}$, where $\mathbf{x}^k$ is the approximated steady state distribution vector obtained in the kth iteration. The initial guess for all methods is $\mathbf{e} = (0, 0, \cdots, 0, 1)^t$. The symbols I, C and BGS in the tables below represent the CGS method without preconditioning, with preconditioner C in (25) and the block Gauss-Seidel method respectively. The N and M stands for the size of the generator matrix of the given SAN (i.e. A in (9)) and the size of the diagonal blocks in (25) respectively. The symbol $**$ signifies more than 1000 iterations. All the computations were done on an HP 712/80 workstation with MATLAB.

6.1 Overflow Queueing Systems

We first consider the 2-queue overflow networks discussed in §2 where overflow is permitted from queue 2 to queue 1 when queue 2 is full. Thus the performance of queue 1 is important. We are interested in finding the steady state distribution vector when the queue length of the first queue increases. In the tests, we fix $l_2 = 32$ and arbitrarily set $s_1 = s_2$, $\lambda_1 = \lambda_2 = 1$ and $\mu_1 = \mu_2 = 2$. Tables 1a and 1b give the cost per iteration and the number of iterations required for convergence for each method respectively.

I	C	BGS
$O(l_1 l_2)$	$O(l_2 l_1 \log l_1)$	$O(l_1 l_2)$

Table 1a: Cost Per Iteration.

$l_2 = 32$			$s_1 = s_2 = 1$			$s_1 = s_2 = 2$		
l_1	N	M	I	C	BGS	I	C	BGS
32	1024	32	10	8	21	70	10	27
64	2048	32	145	10	42	196	12	46
128	4096	32	445	12	150	533	14	153
256	8192	32	**	14	433	**	16	441
512	16384	32	**	14	**	**	16	**

Table 1b: Numbers of Iterations for Convergence.

6.2 Telecommunication Systems

In this section, we present an (MMPP/M/s/$s+m$) network arising in telecommunication [31]. A Markov Modulated Poisson Process (MMPP) is a Poisson process whose instantaneous rate varies according to an irreducible Markov chain, see for instance [31]. Let us first define the system parameters, see Figure 2. We let $1/\lambda$ be the mean arrival time of the exogenously originating calls of the main queue, $1/\mu$ the mean service time of each server of the main queue, s the number of servers in the main queue, $l - s - 1$ the number of waiting spaces in the main

queue, n the number of overflow queues, and finally (Q_j, Λ_j), $1 \le j \le n$, the parameters of the MMPP's modeling overflow parcels, where

$$Q_j = \begin{pmatrix} \sigma_{j1} & -\sigma_{j2} \\ -\sigma_{j1} & \sigma_{j2} \end{pmatrix} \qquad \text{and} \qquad \Lambda_j = \begin{pmatrix} \lambda_j & 0 \\ 0 & 0 \end{pmatrix}.$$

Here σ_{j1}, σ_{j2} and λ_j, $1 \le j \le n$, are positive MMPP parameters.

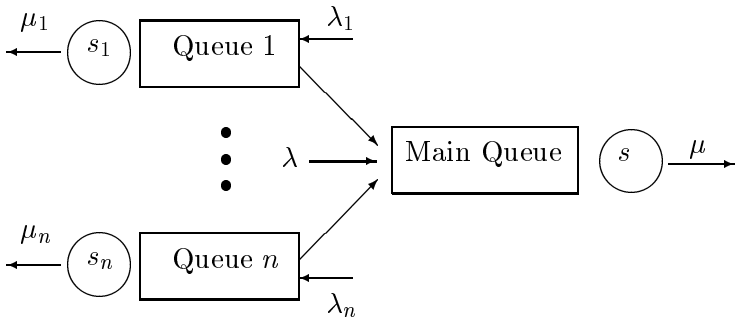

Figure 2. The Telecommunication System.

The input of the main queue comes from its own exogenous arrivals and the superposition of several independent MMPPs, which is still an MMPP and is parameterized by two $2^n \times 2^n$ matrices (Q, Γ). Here

$$Q = (Q_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes Q_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes Q_n),$$

$$\Lambda = (\Lambda_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes \Lambda_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes \Lambda_n)$$

and $\Gamma = \Lambda + \lambda I_{2^n}$, where I_2 and I_{2^n} are the 2×2 and $2^n \times 2^n$ identity matrices respectively.

We can regard the (MMPP/M/s/l) queue as a Markov process on the state space

$$\{(i, j) \mid 0 \le i \le l-1, 1 \le j \le 2^n\}.$$

The number i corresponds to the number of calls at the destination, while j corresponds to the state of the Markov process with generator matrix Q. Hence the generator matrix of the queueing process is given by the following $l2^n \times l2^n$ tridiagonal block matrix:

$$A = \begin{pmatrix} Q+\Gamma & -\mu I & & & & & & 0 \\ -\Gamma & Q+\Gamma+\mu I & -2\mu I & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -\Gamma & Q+\Gamma+s\mu I & -s\mu I & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -\Gamma & Q+\Gamma+s\mu I & -s\mu I & \\ & & & & & -\Gamma & Q+s\mu I & \\ 0 & & & & & & & \end{pmatrix}.$$

It can be rewritten as

$$A = I_l \otimes Q + P_1 \otimes I_{2^n} + P_2 \otimes \Gamma.$$

where P_1 and P_2 take the form of Q_i and R in (2) and (3) respectively.

We note that in this example there are $(n+1)$ individual automata and $m = (2n+2)$ probabilistic rules. The cost per iteration of the CGS method with preconditioner C is of $O(n2^n l \log l)$, see [15], whereas for the BGS method, it is of $O(2^{2n} l)$, see [15, 31]. In the tests, we have tried the number of overflow queues n to be 1 and 4. The number of servers s is set to 2 in all cases. The MMPP parameters are arbitrarily chosen to be $\sigma_{j1} = 2/3$, $\sigma_{j2} = 1/3$, for $j = 1, \cdots, n$. The other queueing parameters are $\mu = 2$, $\lambda = 1$, $\lambda_j = 1/n$, for $j = 1, \cdots, n$. Tables 2a and 2b give the cost per iteration and the number of iterations required for convergence for each method.

I	C	BGS
$O(n2^n l)$	$O(n2^n l \log l)$	$O(2^{2n} l)$

Table 2a: Cost Per Iteration.

$s=2$	$n=1$					$n=4$				
l	N	M	I	C	BGS	N	M	I	C	BGS
32	64	2	155	8	171	512	16	161	13	110
64	128	2	**	7	242	1024	16	**	13	199
128	256	2	**	8	366	2048	16	**	14	317
256	512	2	**	8	601	4096	16	**	14	530
512	1024	2	**	8	**	8192	16	**	14	958

Table 2b: Numbers of Iterations for Convergence.

6.3 The Manufacturing System

In this subsection, we consider a manufacturing system of two machines in tandem under the hedging point product policy, see [22] and Figure 3. The system parameters are: $1/\lambda$, the mean inter-arrival time of a demand; $1/\mu_1$, the mean unit processing time of the first machine; $1/\mu_2$, the mean unit processing time of the second machine; b_1, the size of the buffer B_1 for the first machine; b_2, the maximum size of the buffer B_2 for the finished products; h, the hedging point; and m, the maximum allowable backlog. We note that the inventory level of the first buffer cannot be negative or exceed the buffer size b_1. Thus the total number of possible inventory levels in the first buffer is $(b_1 + 1)$. For the second buffer, under the hedging point policy, the maximum possible inventory level is h with $h \leq b_2$. Since we allow a maximum backlog of m in the system, the total number of possible inventory levels in the second buffer is $l = (m + h + 1)$. In practice the value of l can easily go up to thousands.

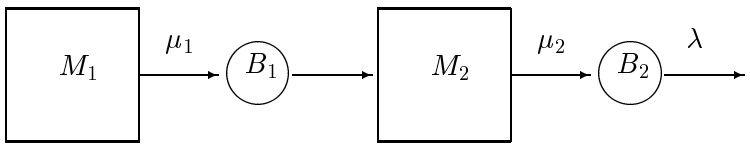

Figure 3. Two Machines in Tandem.

We let $z_1(t)$ and $z_2(t)$ be the inventory levels of the first and second buffers at time t respectively. Then $z_1(t)$ and $z_2(t)$ take integer values in $[0, b_1]$ and $[-m, h]$ respectively. Thus

the joint inventory process $\{(z_1(t), z_2(t)), t \geq 0\}$ is a continuous time Markov chain taking values in the state space

$$S = \{(z_1(t), z_2(t)) : z_1 = 0, \cdots, b_1,\ z_2 = -m, \cdots, h.\}.$$

We order the inventory states lexicographically, with $z_1(t)$ first and then $z_2(t)$. Then we obtain the tridiagonal block generator for the joint inventory system

$$A = \begin{pmatrix} \Lambda + \mu_1 I_l & \Sigma & & & & 0 \\ -\mu_1 I_l & \Lambda + D + \mu_1 I_l & \Sigma & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\mu_1 I_l & \Lambda + D + \mu_1 I_l & \Sigma & \\ 0 & & & -\mu_1 I_l & \Lambda + D & \end{pmatrix},$$

where

$$\Lambda = \begin{pmatrix} 0 & -\lambda & & 0 \\ & \lambda & \ddots & \\ & & \ddots & -\lambda \\ 0 & & & \lambda \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0 & & & 0 \\ -\mu_2 & \ddots & & \\ & \ddots & \ddots & \\ 0 & & -\mu_2 & 0 \end{pmatrix},$$

and $D = \mathrm{Diag}(\mu_2, \cdots, \mu_2, 0)$ is an $l \times l$ diagonal matrix. We note that A can be written as

$$A = I_{b_1+1} \otimes \Lambda + W_1 \otimes I_l + \mathrm{Diag}(0, 1, \cdots, 1) \otimes D + W_2 \otimes \Sigma, \tag{29}$$

where

$$W_1 = \begin{pmatrix} \mu_1 & 0 & & 0 \\ -\mu_1 & \mu_1 & & \\ & \ddots & \ddots & \\ 0 & & -\mu_1 & 0 \end{pmatrix} \quad \text{and} \quad W_2 = \begin{pmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & 0 \end{pmatrix}.$$

Here we are interested in the system performance when l is large.

Similar to the discussion in §4, our preconditioner is obtained by taking the circulant approximations of the matrices Λ, Σ and D in (29). It is easy to check that our preconditioner is unitary similar to a diagonal block matrix with each block being a tridiagonal matrix. The cost per iteration in the CGS algorithm is therefore of $O(l \log l)$. In our tests, we let $\lambda = 1$, $\mu_1 = 3/2$ and $\mu_2 = 3$. Tables 3a and 3b give the cost per iteration and the number of iterations required for convergence for each method respectively.

I	C	BGS
$O(l)$	$O(l \log l)$	$O(l)$

Table 3a: Cost Per Iteration.

	$b_1 = 1$					$b_1 = 4$				
l	N	M	I	C	BGS	N	M	I	C	BGS
32	64	2	34	5	72	160	5	64	10	72
64	128	2	129	7	142	320	5	139	11	142
128	256	2	**	8	345	640	5	**	12	401
256	512	2	**	8	645	1280	5	**	12	**
512	1024	2	**	8	**	2560	5	**	12	**

Table 3b: Numbers of Iterations for Convergence

We conclude from the above three applications that the CGS method without preconditioning converges very slowly. Although the cost of the preconditioned CGS method is larger than that of the nonpreconditioned one or the BGS method by an order of $O(\log l_1)$, the fast convergence of the method can cover this overhead in all the examples tested.

7 Concluding Remarks

In this paper, we discuss circulant preconditioners for stochastic automata newtorks. Our preconditioners are constructed by taking circulant approximations of the generator matrices of the networks. Convergence rate of the preconditioned conjugate gradient method is proven in some practical situations. We test our method for systems from three different applications and they all give very fast convergence when compared with the block Gauss-Seidel method.

We remark that our preconditioners and convergence proof can be applied to manufacturing systems of more than two machines (jobshops) in tandem, see [22] for instance. It will be interesting to extend our results to other sophisticated Markovian models [18, 19, 23, 24].

Acknowledgment: The first author would like to thank Prof. Olof Widlund for introducing him to the subject of queueing networks and his guidance and help all through the years.

References

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, Wiley, New York, 1995.

[2] O. Axelsson and V. Barker, *Finite Element Solution of Boundary Value Problems: Theory and Computation*, Academic Press, Orlando, Fl., 1983.

[3] A. Berman and R. Plemmons, *Nonnegative Matrices in Mathematical Sciences*, SIAM, Philadelphia, 1994.

[4] P. Berlard, Y. Saad and W. Stewart, *Numerical Methods in Markov Chain Modeling*, Operat. Res., 40 (1992), pp. 1156–1179.

[5] P. Buchholz, *A Class of Hierarchical Queueing Networks and Their Analysis*, Queueing Systems, 15 (1994), pp. 59–80.

[6] P. Buchholz, *Hierarchical Markovian Models: Symmetries and Aggregation*, Performance Evaluation, 22 (1995), pp. 93–110.

[7] P. Buchholz, *Equivalence Relations for Stochastic Automata Networks*, Computations of Markov Chains: Proc. of the 2nd International Workshop On Numerical Solutions of Markov Chains, Kluwer, 1995, pp. 197–216.

[8] J. Buzacott and J. Shanthikumar, *Stochastic Models of Manufacturing Systems*, Prentice-Hall International Editions, New Jersey, 1993.

[9] R. Chan, *Iterative Methods for Overflow Queueing Models I*, Numer. Math., 51 (1987), pp. 143–180.

[10] R. Chan, *Iterative Methods for Overflow Queueing Models II*, Numer. Math., 54 (1988), pp. 57–78.

[11] R. Chan, *Iterative Methods for Queueing Networks with Irregular State-Spaces*, Proceedings to the IMA Workshop on Linear Algebra, Markov Chains and Queueing Models, Minneapolis, January 13–17, 1992, pp. 89–110, Eds: C. Meyer and R. Plemmons, Springer-Verlag, 1993.

[12] R. Chan and W. Ching, *Toeplitz-circulant Preconditioners for Toeplitz Systems and Their Applications to Queueing Networks with Batch Arrivals*, SIAM J. Sci. Comput., 17 (1996), pp. 762–772.

[13] R. Chan and K. Ng, *Conjugate Gradient Methods for Toeplitz Systems*. SIAM Review, 38 (1996), pp. 427–482.

[14] R. Chan, C. Wong and W. Ching *Optimal Trigonometric Preconditioners for Elliptic Problems and Queueing Problems*, SEA Bull. Math., 3 (1996), pp. 117–124.

[15] W. Ching, R. Chan and X. Zhou, *Circulant Preconditioners for Markov Modulated Poisson Processes and Their Applications to Manufacturing Systems*, SIAM J. Matrix Anal. Appl., 17 (1997), pp. 452–467.

[16] W. Ching and X. Zhou, *Matrix Methods for Production Planning in Failure Prone Manufacturing Systems*, Lecture Notes in Control and Information Sciences, Springer-Verlag, 1996, pp. 3–30.

[17] W. Ching and X. Zhou, *Optimal (S,s) Production Policies with Delivery Time Guarantee*, Lectures in Applied Mathematics, Mathematics of Stochastic Manufacturing Systems, The American Mathematical Society, 1997, pp. 71–82.

[18] W. Ching and X. Zhou, *Machine Repairing Models for Manufacturing Systems*, The 5^{th} IEEE Conference on Emerging Technologies and Factory Automation, November, 1996.

[19] W. Ching and X. Zhou, *Optimal (S,s) Policies for Manufacturing Systems with Buffers Holding Costs*, The 15^{th} World Congress of Scientific Computation, Modeling and Applied Mathematics, Berlin, Germany, August, 1997.

[20] W. Ching, *Circulant Preconditioners for Failure Prone Manufacturing Systems*, Lin. Alg. Appl., 266 (1997), pp. 161–180.

[21] W. Ching, *Markov Modulated Poisson Processes for Multi-location Inventory Problems*, Inter. J. Prod. Econ., 53 (1997), pp. 232–239.

[22] W. Ching, *Iterative Methods for Manufacturing Systems of Two Stations in Tandem*, Inter. J. Appl. Math. Letters, 11 (1998), pp. 7–12.

[23] W. Ching, *Markov Modulated Poisson Processes and Production Planning in Manufacturing Systems,* To Appear in the WMC'97 International Symposium on Manufacturing Systems, Auckland, New Zealand, 1997.

[24] W. Ching, *Optimal (S,s) Policies for Manufacturing Systems of Unreliable Machines in Tandem*, International Symposium on Product Quality and Integrity, Anaheim, U.S.A., January, 1998.

[25] P. Concus, G. Golub and G. Meurant, *Block Preconditioning for Conjugate Gradient Method*, SIAM J. Statist. Comput., 6 (1985), pp. 220–252.

[26] P. Concus and G. Meurant, *On Computing INV Block Preconditionings for Conjugate Gradient Method*, BIT, 26 (1986), pp. 493–504.

[27] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.

[28] H. Heffes and D. Lucantoni, *A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance*, IEEE J. Select. Areas Commun., SAC-4 (1986), pp. 856–868.

[29] L. Kaufman, *Matrix Methods for Queueing Problems*, SIAM J. Sci. Statist. Comput., 4 (1982), pp. 525–552.

[30] T. Manteuffel, *An Incomplete Factorization Techniques for Positive Definite Linear Systems*, Math. Comp., 34 (1980), pp. 473–497.

[31] K. Meier-Hellstern, *The Analysis of a Queue Arising in Overflow Models*, IEEE Trans. Commun., 37 (1989), pp. 367–372.

[32] H. Mendelson and S. Whang, *Optimal Incentive-Compatible Priority Pricing for the M/M/1 Queue*, Operat. Res., 38 (1990), pp. 870–883.

[33] B. Plateau and K. Atif, *Stochastic Automata Network for Modelling Parallel Systems*, IEEE Tran. Software Engineering, 12 (1991), pp. 370–389.

[34] P. Shanthikumar, S. Datar and R. Akella, *Approximations for the time spent in a dynamic job shop with applications to due date assignment*, Inter. J. Prod. Res., 26 (1988), pp. 1329–1352.

[35] W. Stewart, K. Atif and B. Plateau, *The Numerical Solution of Stochastic Automata Networks*, Euro. J. Operat. Res., 86 (1995), 503–525.

[36] P. Sonneveld, *CGS, A Fast Lanczos-type Solver for Non-symmetric Linear Systems*, SIAM J. Sci. Comput., 10 (1989), pp. 36–52.

[37] W. Takumi and K. Hayami, *Overlapping Multicolor MILU preconditioning*, SIAM J. Sci. Comput., 16 (1995), pp. 636–650.

[38] H. Van der Vorst, *Preconditioning by Incomplete Decomposition*, Ph. D Thesis, Rijksuniversiteit te Utrecht, 1982.

[39] H. Young, B. Byung and K. Chong, *Performance Analysis of Leaky-bucket Bandwidth Enforcement Strategy for Bursty Traffics in an ATM Network*, Comput. Net. ISDN Syst., 25 (1992), pp. 295–303.