Approximated Tensor Sum Preconditioner for Stochastic Automata Networks

Abderezak Touzene

Sultan Qaboos University
College of Science, Dept. of Computer Science
PO. Box 36, Al-khod 123, OMAN
touzene@squ.edu.om

Abstract

Some iterative and projection methods for SAN have been tested with a modest success. Several preconditioners for SAN have been developed to speedup the convergence rate. Recently Langville and Stewart proposed the Nearest Kronecker Product (NKP) preconditioner for SAN with a great success. Encouraged by their work, we propose a new preconditioning method, called Approximated Tensor Sum Preconditioner (ATSP), which uses tensor sum preconditioner rather than Kronecker product preconditioner. In ATSP, we take into account the effect of the synchronizations using an approximation technique. Our preconditioner outperforms the NKP preconditioner for the tested SAN Model.

1. Introduction

Stochastic Automata Networks (SAN) is a very powerful modeling tool for complex systems [1] [2] [3], they are particularly useful to model parallel and distributed activities. A stochastic automata network consists of a number of individual stochastic automata that operate more or less independently. Each individual automaton is represented by a number of states and rules that govern the manner in which it moves from state to the next. The state of an automaton at time t is given by the state it occupies at that time t. Considering an individual automaton is called local view of the SAN. The global view of a SAN or more precisely, the global state of a SAN at time t is the state given by the state of each automaton at time t [1]. The SAN formalism is naturally appropriate to model parallel and distributed activities, which can be viewed as a collection of components that operate more or less independently and might interact infrequently.

There are two ways in which stochastic automata interact : *Synchronized transition*, i.e a transition in one automaton may force a transition to occur in one or more other automata. *Functional transition*, i.e. the rate at which a transition occurs may be a function of the state of a set of automata.

Many methods have been proposed to solve SAN model all of them are iterative or projection method such as the power method, GMRES and Arnoldi method [2]. These methods adapt well to SAN because they are based on the product vector by the SAN descriptor, which can be carried out efficiently [4]. Direct methods for solving Markov chain, such as those based on LU factorization cannot be applicable to SAN. SAN never expands the transition matrix, which is required for the LU factorization. Classical iterative methods like Gauss-Seidel, SOR have been adapted for SAN and developed by Dayar [5]. As we just mention it, the only method easily adaptable to SAN are iterative method. It is natural then to try preconditioning methods to accelerate the convergence rate of those iterative methods.

There are several preconditioning method that have been tested on SAN. For the same reasons as discussed above, preconditioner, which showed a great success for Markov chains modeling, namely, incomplete LU factorization (ILU0,ILUTH) cannot be used for SAN preconditioning. Atif and his co-worker [6] have tried the first time Neumann series to approximate the inverse of the SAN descriptor. Overall no benefit has been drawn from that preconditioner. Other preconditioner, which are extensions of Neumann series have been applied to projection method [7] such as, BiCGSTAB, CGS, TFQMR. These preconditioners reduce the cost

1-4244-0054-6/06/$20.00 ©2006 IEEE

to compute the approximate inverse, but still they are computationally expensive for large-scale SAN.

In 2004 Langville and Stewart [8] proposed a new method called Near Kronecker Product approximation (NKP), which aims at approximate the inverse of the SAN descriptor. The NKP preconditioner first solves an optimization problem to approximate the SAN descriptor by one Kronecker product. In the second stage the inverse of this resulted Kronecker product is computed as a Kronecker product of the inverse of each matrix as shown in [9].

NKP preconditioner has been tested successfully and compare favorably with the other preconditioner. For large-scale SAN models, NKP preconditioner needs to do grouping of automata in order to reduce the cost of the preconditioner. Gouping of automata, means expanding the SAN descriptor. It is know that all the power of SAN formalism is based on keeping the SAN descriptor as compact as possible.

In this paper we present a new approximate inverse preconditioner method based on tensor sum approximation for the SAN descriptor, it is called Approximated Tensor Sum Preconditioner (ATSP). ATSP technique is more scalable than NKP, it does not require to group automata. The rest of this paper is organized as follows: In Section 2, we present the background and related works. Our preconditioner methodology is discussed in Section 3. In Section 4, we compare our preconditioner and NKP for some SAN models. Section 5 concludes the paper.

2. Backgroud and Related Works

In this paper we focus only on computing the stationary probability vector denoted π corresponding to a SAN model. As discussed in [6], the SAN descriptor, i.e. the infinitesimal generator of the SAN model has the following compact structure:

$$Q = \sum_{j=1}^{T} \otimes_i^N \ Q_j^i \qquad (1)$$

The power method is one of the basic iterative method that has been applied to SAN [6] [7] [10]. The power method is applied to the matrix $P = I + \Delta Q$, where $\Delta = \frac{1}{max|q_{ii}|}$. For SAN formalism the power method iteration is given by:

$$x^{(k+1)} = x^{(k)}(I+\Delta Q) = x^{(k)}+\Delta x^{(k)}(\sum_{j=1}^{T} \otimes_i^N \ Q_j^i) \quad (2)$$

The power method has the merit to be very simple and it cost per iteration is cheap (product vector by the SAN descriptor) comparatively with others. The disadvantage of the power method is its very slow convergence rate. It is still used as a baseline to compare new iterative methods.

Preconditioning an iterative method offers a good improvement in the convergence rate of the specific iterative method. Since the convergence rate of iterative methods depends on the distribution of the eigenvalues of the iteration matrix. The goal of preconditioning is to modify the eigenvalues distribution of the iteration matrix so that it will converge rapidly. In the general case of a system of equations $Ax = B$, preconditioning is the introduction of a matrix M called the preconditioner, such that the new system becomes $MAx = Mb$. The matrix M is selected to be as close as possible to the inverse of A. The difficulty is how to construct M as an approximation to A^{-1} efficiently. Popular preconditioner like Incomplete LU factorization ILU (ILU0, ILUTH [11]) are among the best preconditioner for Markov chains. Unfortunately these method cannot be adapted to SAN formalism because of the compact form of the SAN descriptor. Another preconditioner developed by Stewart and his co-workers [6], uses Neumann series. This preconditioner is accurate but it is computationally very expensive. Buchholz [7] , has tested a preconditioner based on the inverse of the individual automata, but unfortunately with a very moderate success in some cases.

Recently Langville and Stewart [8] proposed the Nearest Kronecker Product preconditioner, which is base on approximating the inverse of the SAN descriptor. Their idea is to approximate the SAN descriptor $Q = (\sum_{j=1}^{T} \otimes_i^N Q_j^i)$ by a Kronecker producted of matrices $A_1 \otimes A_2 \otimes ... \otimes A_n$. This can be done by solving the following minimization problem: $||Q - A_1 \otimes A_2 \otimes ... \otimes A_N||_F^2$. Once the matrice $A_i, i = 1..N$ are calculated, the NKP preconditioner is $M = A_1^{-1} \otimes A_2^{-1} \otimes ... \otimes A_N^{-1}$. The cost of this preconditioner is given in [8], and it depend on the number of variables NT. Clearly if N the number of automata is high and/or the number of synchronizing event is high, both the cost of forming the traces and also solving the nonlinear optimization problem including NT variables will be very expensive. As mentioned in [8], this drawback can be circumvented by using grouping of automata as it done in [10]. As we discussed in the introduction the grouping may limit the advantages gained through keeping the SAN descriptor in its compact form. Grouping means expanding the descriptor in some sens.

In the next section we present our preconditioner, which is also an approximate inverse but instead of using a Kronecker product as an approximation as in the NKP method, we use tensor sum preconditionner.

3. Tensor Term Preconditioner for SAN

The are two motivations for the use of tensor sum preconditioner rather than Kronecker product preconditionner:

1. In SAN modeling the automata act independently and they may synchronize from time to time. Synchronization rate may be small.

2. The use of Kronecker product structure to approximate a dominant tensor sum structure may not be appropriate. The places of the non zero elements may be very different for the two different structures. This may explain why the NKP preconditioner is really performing bad for the general Markov chains as discussed in [8]. The performance of the NKP preconditioner does not converge as expected even for for SAN models, may be for the same reasons.

Preconditioning of the power method is done by introducing the preconditioner matrix M as seen in the previous section. The preconditioned power method for SAN is given by the following formula:

$$\pi^{(k+1)} = \pi^{(k)} + \Delta\pi^{(k)}(\sum_{j=1}^{T} \otimes_i^N Q_j^i)M^{-1}) \qquad (3)$$

If we denote by $y^{(k)} = \Delta\pi^{(k)}(\sum_{j=1}^{T} \otimes_i^N Q_j^i)$ the preconditioned iteration will look like:

$$\pi^{(k+1)} = \pi^{(k)} + y^{(k)}M^{-1} \qquad (4)$$

In practice we do not compute the inverse of M. It is easier to modify the problem of computing the inverse of a matrix to solving a system of equation involving $M : y^{(k)}M^{-1} = x$, and the equation to be solved is then

$$xM = y, \qquad (5)$$

where $M$is kept in its SAN compact form.

To simplify the presentation of our method, we first present the case where the SAN descriptor is a pure tensor sum (it contains only local terms). Then we show how to extend the method to the case where the SAN is not a pure tensor sum, which is in fact the more general case.

3.1. SAN Descriptor with Pure Tensor Sums

For clarity reasons we consider the case where the SAN descriptor consists of a tensor sum with only two factors : $M = (A_1 \oplus A2)$, where A_1 is of dimension $m \times m$, and A_2 is of dimension $n \times n$. The generalization of this method will be discussed later. We recall that our aim is to solve the system of equations (5). The basic algorithm was proposed by Bartels and Stewart [12], the system (5) is equivalent to the system:

$$A_1^T X + X A_2 = Y \qquad (6)$$

where X is an $m \times n$ matrix. The matrix X can be seen as a matrix of columns : $X = (x_1, x_2, ..., x_n)$, where x_j denotes the j^{th} column of X. The matrix Y which represents y is structured in a similar way. The algorithm is as follows:

Algorithm

1. Compute a unitary transformation U and V such that

$$\bar{A_1} = U^T A_1 U \quad \text{and} \bar{A_2} = V^T A_2 V, \qquad (7)$$

   where $\bar{A_1}$ and $\bar{A_2}$ are upper triangular matrices obtained using a Shur decomposition algorithm.

2. Transform the original system (6) using step 1 as follows:

$$\bar{A_1}^T \bar{X} + \bar{X}\bar{A_2} = \bar{Y} \qquad (8)$$

   where

$$\bar{X} = U^T X V$$

   and

$$\bar{Y} = U^T Y V.$$

3. The first column of $\bar{X}$ is given by solving the following lower triangular system:

$$(\bar{A_1}^T + \bar{A_2}(1,1) \times I)\bar{x_1} = \bar{y_1}. \qquad (9)$$

4. The computation of the k^{th} column of the system is given by:

$$(\bar{A_1}^T + \bar{A_2}(k,k) \times I)\bar{x_k} = \bar{y_k} - \sum_{i=1}^{k-1} \bar{A_2}(i,k)\bar{x_i}. \qquad (10)$$

5. Form the solution X using $\bar{X} : X = U\bar{X}V^T$.

Note that in this algorithm the Shur decomposition is not necessary for the first matrix A_1. This helps the generalization from the 2 matrices case to the N matrices case. The generalization of this method can be seen as a recursive process, which is described as follows: Any system $x(A_1 \oplus A_2 ... \oplus A_N) = y$, where the matrices A_i are of dimension $n_i, i = 1..N$, can be decomposed as $x(A \oplus A_N) = y$. At this stage we do require only the Shur decomposition for the matrix A_N. The matrix A has still a tensor sum structure but with only $N-1$ matrices. steps 1, 2 of the basic algorithm are the same except that U^T and U matrices are taken to be identity matrices. The steps 3 and 4 will be:

- The first column of $\bar{X}$ is given by solving the above system involving A:

$$(\bar{A}^T + \bar{A_N}(1,1)I)\bar{x_1} = \bar{y_1}. \qquad (11)$$

- The computation of the k^{th} column of the system is given by:

$$(\bar{A}^T + \bar{A_N}(k,k)I)\bar{x_k} = \bar{y_k} - \sum_{i=1}^{k-1} \bar{A_N}(i,k)\bar{x_i}. \qquad (12)$$

For both steps we need to solve a system of equation involving the matrix A, which has a tensor sum structure with the following form:

$$(\bar{A}^T + aI)\bar{x_k} = \bar{f_k}, \ or \ (\bar{B}^T)\bar{x_k} = \bar{f_k}, \qquad (13)$$

where the matrix $B = (A_1 \oplus A_2 ... \oplus A_{N-2} \oplus (A_{N-1} + aI))$, and a a real number. Using the transpose operation the system (13) can be written as $\bar{x_k}^T(\bar{B}) = \bar{f_k}^T$. To compute $\bar{x_k}^T$ we need to solve a similar problem as the original one but hopefull with smaller size. We apply recursively this process till having a system with only two matrices in the tensor sum matrix B and then we use the basic procedure. It is easy to see that the cost of solving the above system of equations (excluding the Shur decomposition of the matrices) is given by

$$Cost = \frac{5}{2}(n_1 n_2 ... n_N)(n_1 + n_2 + ... + n_N).$$

In general, matrices $A_i, i = 1..N$ are small matrices. The cost of their Shur decomposition is negligible and hopefully they are calculated only one time for all iterations. The cost to solve the preconditioned system is then equal to 2.5 time the cost of the multiplication vector by one term of the SAN descriptor.

Let us focus now on practical implementation issues. Since M is singular (generator), the inverse of M does not exist. The idea is to alter slightly the system (8) to overcome the singularity problem. One sure way to alter M lies within the following property of the tensor sum: $(A_1^T + \alpha I)X + X(A_2 - \alpha I) = Y$, for any real number α. The resulting system is equivalent to the first one, but the singularity is still present. The second way is to apply a shift to $M = (A_1 \oplus A_2 + \alpha I)$. The new system is not equivalent, but fortunately any shift combined with the power method converge to the right solution.

3.2. SAN Descriptor with Synchronization Terms

We recall that the SAN descriptor has the form given in formula (1). In general, it has two parts, the local terms, which are pure tensor sums, plus the synchronization terms, which are Kronecker products. In this case our algorithm cannot be applied directly, since it handles only pure tensor sums. We present two techniques based on how to handle the synchronization:

1. One solution is to consider only the local terms as a preconditioner. In this case the synchronization rate are not included within the preconditioner, which may not lead to an accurate preconditioner.

2. A second way of dealing with the synchronization is approximating the synchronization by independent events. Each synchronization event is broken as local independent events with the same rate in each automaton where the synchronization event act on. In other words, the rates of the synchronization events will be treated as local transition and thus easy to be added to the ATSP preconditioner.

In the next section we will test the quality of our ATSP preconditioner using the last methods.

4. ATSP Preconditioner and SAN Models

In this section we present the SAN models tested and the results obtained by our ATSP preconditioner comparatively with the power method without preconditioning(None); power method with NKP preconditioner (NKP). For all the experiments the shifting factor α (to remove the singularity) is chosen from the interval $[.001, .1]$. The current version of ATSP is implemented using MATLAB software. The tests have being running on a Pentium 4, 1MGHz.

4.1. Independent Queues Model

In this model we want to measure the quality of our preconditioner at its full power, i.e. models where the SAN descriptor is a pure tensor sum. In this model we consider a system of six independent $M/M/1/c_i$ queues. Arrivals to the queues are independent of rate $\lambda_i, i = 1..6$. The service rate at each queue i is μ_i. The SAN descriptor will have the following form: $Q = \bigoplus_{i=1}^{N} A_i$ (pure tensor sums), where A_i is

$$A_i = \begin{pmatrix} -\lambda_i & \lambda_i & 0 & \dots & \dots \\ \mu_i & -\mu_i - \lambda_i & \lambda_i & 0 & \dots \\ 0 & \mu_i & -\mu_i - \lambda_i & \lambda_i & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_i & -\mu_i \end{pmatrix}$$

The results of testing this model are summariazed in table 1, the model parameters are $\lambda_i = i, i = 1..6$, and $\mu_i = 2i, i = 1..6$. The time units in the tables is seconds.

$\prod c_i$	Prec.	Power Nb. Iter.	Power Time
2^6	None	153	4.97
	NKP	101	8.63
	ATSP	2	0.36
3^6	None	303	74.33
	NKP	167	81.60
	ATSP	2	1.33
4^6	None	331	364.69
	NKP	282	682.02
	ATSP	2	4.9

Table 1. Independent queues.

We can see clearly the very good quality of our ATSP preconditioner. It converges in only two iterations when NKP takes several iterations.

4.2. Tandem Queue Model

In this example we consider two queues in tandem where customers enter the system from the first queue with rate λ, get their service with rate μ_1. Customers leaving the first queue may join immediately the second queue with probability p or leave the system with probability $(1-p)$. This model contains two local terms, one synchronization term and one normalization term for the synchronization:

- Local term one L_1:

$$L_1 = \begin{pmatrix} * & \lambda & 0 & \dots & \dots \\ \mu_1(1-p) & * & \lambda & 0 & \dots \\ 0 & \mu_1(1-p) & * & \lambda & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_1(1-p) & * \end{pmatrix} \otimes I_{c_2},$$

  where $*$ denotes the negative of the sum of the elements of the corresponding row.

- Local term two L_2:

$$L_2 = I_{c_1} \otimes \begin{pmatrix} 0 & \dots & \dots & \dots & \dots \\ \mu_2 & -\mu_2 & 0 & \dots & \dots \\ 0 & \mu_2 & -\mu_2 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_2 & -\mu_2 \end{pmatrix}$$

- Synchronization term S:

$$S = \begin{pmatrix} 0 & \dots & \dots & \dots & \dots \\ \mu_1 p & -\mu_1 p & 0 & \dots & \dots \\ 0 & \mu_1 p & -\mu_1 p & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_1 p & -\mu_1 p \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & \dots & \dots & \dots & \\ 0 & 0 & 1 & \dots & \dots & \\ 0 & 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & \dots & \dots & \dots & \dots & \end{pmatrix}$$

- Normalization term NS:

$$NS = \begin{pmatrix} 0 & \dots & \dots & \dots & \dots \\ 0 & -\mu_1 p & 0 & \dots & \dots \\ 0 & 0 & -\mu_1 p & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & -\mu_1 p \end{pmatrix} \otimes \begin{pmatrix} 0 & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & \dots & \dots \\ 0 & 0 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}$$

The SAN descriptor is the sum of four terms: $Q = L_1 + L_2 + S + NS$. The local terms are $L = L_1 \oplus L_2$. We can use only the local term as an approximation to Q, the effect of the synchronization is lost. This approximation will depend on how small is the rate of the

synchronization event ($p\mu_1$). We can use approximation technique 2: the synchronization events are considered as independent events (not a synchronization event). In this case, the arrival at the second queue is considered as a Poisson process of parameter $p\mu_1$. The local terms will be modified as follows:

$$ModL_1 = \begin{pmatrix} -\lambda & \lambda & 0 & \dots & \dots \\ \mu_1 & -\mu_1 - \lambda & \lambda & 0 & \dots \\ 0 & \mu_1 & -\mu_1 - \lambda & \lambda & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_1 & -\mu_1 \end{pmatrix} \otimes I_{c_2}$$

and,

$$ModL_2 = I_{c_1} \otimes \begin{pmatrix} * & \mu_1 p & \dots & \dots & \dots \\ \mu_2 & * & \mu_1 p & \dots & \dots \\ 0 & \mu_2 & * & \mu_1 p & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \mu_2 & * \end{pmatrix}$$

The results of testing this model are shown in table 2, the model parameters are $\lambda = 13, p = .7, \mu_1 = 20, \mu_2 = 30$.

c_1	c_2	Prec.	Power Nb. Iter.	Power Time
20	20	None	567	6.50
		NKP	346	5.29
		ATSP	22	0.32
50	50	None	504	28.14
		NKP	421	31.02
		ATSP	169	12.02
71	71	None	458	48.84
		NKP	401	57.01
		ATSP	235	35.14

Table 2. Tandem Queue Example.

From table 2 we can see the good performance of ATSP compared to NKP. For the largest model tested, ATSP is almost two times faster than NKP.

5. conclusion

In this paper we presented a new approximated tensor sum preconditioning method for solving stochastic automata network models. The cost of this preconditioner is 2.5 time the product vector by one term of the SAN, which is relatively cheap. Testing our preconditioner on a SAN model without synchronization gives very good results, only two iterations are needed to solve the SAN models.

We presented also a new technique to handle synchronizations in the ATSP preconditioner if they exist in the SAN model. This approximation technique is based on breaking the synchronization event into local events with the same rate. Experimental results the superiority of our preconditioner over NKP preconditioner for the tested SAN models. In our future work we will investigate other technique to handle synchronizations to make our preconditioner more accurate.

References

[1] B. Plateau. On the Stochastic Structure of Parallelism and Synchronization Models for Distributed Algorithms. *Proc. ACM Sigmitrics Conference on Measurement and Modeling of Computer Systems, Austin, Texas, Aug. 1985.*

[2] K. Atif. Modelisation du Parallelism et de la Synchronisation. *These de Docteur de l'Institut National Polytechnique de Grenoble, France, Sep. 1991.*

[3] K. Atif, B. Plateau. Stochastic Automata Network for Modeling Parallel Systems. *IEEE Trans. On Soft. Eng., 17, 10, Oct. 1991* .

[4] P. Fernandes, B. Plateau, W. J. Stewart. Efficient descriptor-vector multiplication in stochastic automata networks. *J. ACM, (3), 381-414, 1998.*

[5] E.Uysal, T. Dayar. Iterative methods based on splitting for stochastic automata network. *European Journal of Operation Research. 110:166-186, 1998.*

[6] K. Atif, B. Plateau, and W. Stewart. The numerical solution of stochastic automata network. *European Journal of Operation Research, 86(3), Nov. 1995.*

[7] P. Busholz. ; Projection methods for the analysis of stochastic automata networks. *In B. Plateau, W. Stewart, and M. Silva, eds. Numerical Solutions for Markov Chains, pp. 149-168. Zaragoza :Prensas Universitarias de Zaragoza, 1999.*

[8] A. Langville, W. Stewart. A Kronecker product approximate preconditioner for SANs. *Numerical Linear Algebra with Application:11,723-752, 2004* .

[9] CF. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics:123, 85-100, 2000.*

[10] B. Plateau, W. J. Stewart, P. Fernandes. On the benefits of using functional transition in Kronecker modelling. *Submitted to Performance Evaluation, 201.*

[11] W. J. Stewart An Introduction to the Numerical Solution of Markov Chains. *Princeton University Press, NJ. 1994.*

[12] R. H. Bartels and G. W. Stewart. Algorithm 432: The solution of the matrix equation AX-BX=C. *Communication of ACM, 8:820-826, 1972.*