TUD-FI03-05 - Mai 2003

Peter Buchholz, Tuğrul Dayar

Dresden University of Technology,
Bilkent University, Ankara, Turkey

Block SOR Preconditioned Projection Methods for Kronecker Structured Markovian Representations

BLOCK SOR PRECONDITIONED PROJECTION METHODS FOR KRONECKER STRUCTURED MARKOVIAN REPRESENTATIONS*

PETER BUCHHOLZ† AND TUĞRUL DAYAR‡

Abstract. Kronecker structured representations are used to cope with the state space explosion problem in Markovian modeling and analysis. Currently an open research problem is that of devising strong preconditioners to be used with projection methods for the computation of the stationary vector of Markov chains (MCs) underlying such representations. This paper proposes a block SOR (BSOR) preconditioner for hierarchical Markovian Models (HMMs) that are composed of multiple low level models and a high level model that defines the interaction among low level models. The Kronecker structure of an HMM yields nested block partitionings in its underlying continuous-time MC which may be used in the BSOR preconditioner. The computation of the BSOR preconditioned residual in each iteration of a preconditioned projection method becomes the problem of solving multiple nonsingular linear systems whose coefficient matrices are the diagonal blocks of the chosen partitioning. The proposed BSOR preconditioner solves these systems using sparse LU or real Schur factors of diagonal blocks. The fill-in of sparse LU factorized diagonal blocks is reduced using the column approximate minimum degree algorithm (COLAMD). A set of numerical experiments are presented to show the merits of the proposed BSOR preconditioner.

Key words. Markov chains, Kronecker based numerical techniques, block SOR, preconditioning, projection methods, real Schur factorization, COLAMD ordering

AMS subject classifications. 60J27, 15A72, 65F10, 65F50, 65B99, 15A23, 65F05, 65F15

1. Introduction.

Markovian modeling and analysis is used extensively in evaluating the performance or reliability of existing and planned communication, computer, and manufacturing systems. For example, it may be used to determine the probability of rejecting a call in a mobile communication network, the effect of increasing the number of disks in a client-server system, or the throughput of a particular station in a flow shop. Compared to simulative techniques, the attraction for Markov chains (MCs) lies in that they provide exact results up to computer precision for performance or reliability measures through numerical analysis [41]. The systems of interest are becoming increasingly complex which makes their modeling and quantitative analysis difficult. The major problem associated with Markovian modeling and analysis is known as state space explosion, and it refers to the fact that the number of states required to represent a complex system grows exponentially with the number of components (or subsystems) in the system. A currently popular way of dealing with this problem is to employ Kronecker [45] (or tensor) based representations.

The concept of using Kronecker operations to define large MCs underlying structured representations appears in hierarchical Markovian models (HMMs) [8, 13, 15], or in compositional Markovian models such as stochastic automata networks (SANs) [33, 34, 35, 41] and different classes of superposed Stochastic Petri Nets (SPNs) [20, 26]. In the Kronecker based approach, the system of interest is modeled so that it is formed of smaller interacting components, and its larger underlying MC is neither generated nor stored but rather represented as a sum of Kronecker products of the smaller component matrices. In order to analyze large, structured Markovian models efficiently, various algorithms for vector-Kronecker product multiplication are devised [33, 21, 22, 14] and used as kernels in iterative solution techniques proposed for HMMs [8, 10, 12], SANs [33, 41, 42, 43, 10, 11, 14] and superposed Generalized SPNs [26].

Currently an open research problem is that of devising strong preconditioners [38, 25] to be used with projection (or Krylov subspace) methods [38] for MCs underlying Kronecker structured representations [41, 42, 10, 11]. It is known that projection methods for sparse MCs should be used with preconditioners, such as those based on incomplete LU (ILU) factorizations, to be competitive with block successive over-relaxation (BSOR) and iterative aggregation-disaggregation (IAD) [18]. However, it is not clear how to devise ILU type preconditioners for MCs that are in the form of sums of Kronecker products.

So far, various preconditioners are proposed for Kronecker representations such as those based on truncated Neumann series [41, 42], the cheap and separable preconditioner for HMMs and compositional

*This work has been carried out at Dresden University of Technology, where the second author was a research fellow of the Alexander von Humboldt Foundation.

†Department of Computer Science, Dresden University of Technology, D-01062 Dresden, Germany (p.buchholz@inf.tu-dresden.de).

‡Department of Computer Engineering, Bilkent University, TR-06800 Bilkent, Ankara, Turkey (tugrul@cs.bilkent.edu.tr).

Markovian models [10], and circulant preconditioners for a class of SANs [16]. The Kronecker product approximate preconditioner for SANs introduced recently in [28], although encouraging, is in the form of a prototype implementation.

On the other hand, results in [18] on the computation of the stationary vector of MCs show that BSOR with suitable partitionings is a very competitive solver when compared with IAD and ILU preconditioned projection methods. BSOR is developed for SANs in [43]. Therein it is shown that the Kronecker structure of the underlying continuous-time MC (CTMC) yields nested block partitionings. Recently, a more sophisticated BSOR solver is introduced for HMMs in [12]. HMMs are composed of multiple low level models (LLMs) and a high level model (HLM) that defines the interaction among LLMs. As in SANs, the Kronecker structure of an HMM yields nested block partitionings in the underlying CTMC. Diagonal blocks at a particular level of the nested partitioning are all square, but can have different orders in different HLM states. Consequently, off-diagonal blocks that correspond to a pair of different macrostates need not be square. This is different than SANs in which all (diagonal and off-diagonal) blocks at each level of nested partitioning associated with the Kronecker structure are square and have the same order. SANs in the absence of functional transition rates are HMMs having one HLM state. Furthermore, by introducing new transitions, it is possible to transform SANs that have functional transitions to SANs without functional transitions [35]. Therefore, HMMs discussed in this paper have considerable expressive power.

The particular BSOR solver for HMMs is three-level as opposed to the usual two-level solvers [30], since in addition to the outer BSOR iteration at the first level there exists an intermediate block Gauss-Seidel (BGS) iteration at the second level which solves the diagonal blocks of the BSOR partitioning using smaller nested diagonal blocks. But more importantly, in each HLM state the solver takes advantage of diagonal blocks with identical off-diagonal parts and diagonals differing from each other by a multiple of the identity matrix. Such diagonal blocks are referred to as *candidate blocks* [12] and can all utilize the same real Schur factorization [40]. Furthermore, when the candidate blocks satisfy some easy to check conditions, they are likely to possess sparse real Schur factors that can be constructed from the component matrices and their real Schur factors. This implies significant savings in storage and time during the iterative process. We remark that there are many HMMs which satisfy these conditions. Furthermore, the BSOR solver utilizes the column approximate minimum degree algorithm (COLAMD) [17] to reduce the fill-in of sparse LU factorized diagonal blocks.

Since BSOR is a preconditioned power iteration in which the preconditioning matrix (or preconditioner) is based on the block triangular part of the coefficient matrix [25], it can be used as a preconditioner with projection methods for Kronecker structured representations. The BSOR preconditioner proposed in this paper is based on the particular implementation in [12]. However, noticing that diagonal blocks of the BSOR partitioning need to be solved with high accuracy when BSOR is used as a preconditioner with projection methods, we present its two-level version in which the diagonal blocks are solved directly.

The next section introduces structured description of CTMCs using HMMs on an example. The third section presents the BSOR preconditioner and discusses how the preconditioned solve in each iteration of projection methods is performed in HMMs. The fourth section explains how diagonal blocks of the BSOR preconditioner are factorized. The fifth section describes the test problems used. The sixth section discusses results of numerical experiments. The seventh section concludes the paper.

2. Hierarchical Markovian Models. A formal definition of HMMs can be found in [10, pp. 387-392]. Since the formal HMM notation is rather complicated and difficult to follow, we introduce HMMs on an example. Hereafter, we refer to the CTMC underlying an HMM as the matrix Q. This singular matrix has nonnegative off-diagonal elements and diagonal elements that are negated row sums of its off-diagonal elements.

Example 1. We consider a model of token based scheduling in a queueing network [2] and name it as *qh_realcontrol*. Its HLM of 9 states describes the interaction among three LLMs. LLM 1 has 203 states, LLM 2 has 164 states, and LLM 3 has 151 states. All states are numbered starting from 0. We name the states of the HLM as *macrostates* and those of Q as *microstates*. The mapping between LLM states and HLM states is given in Table 1. Note that macrostates in an HLM may have different numbers of microstates when LLMs have partitioned state spaces.

Seven transitions denoted by t_0, t_{17}, t_{18}, t_{19}, t_{21}, t_{24}, and t_{27} take place in the HLM and affect the LLMs. The last six of these transitions are captured by the following (9×9) HLM matrix:

TABLE 1
Mapping between LLM states and HLM states in qh_realcontrol.

HLM	LLM 1	LLM 2	LLM 3	# of microstates						
0	171:202	138:163	0:56	32	.	26	.	57	=	47,424
1	0:76	138:163	127:150	77	.	26	.	24	=	48,048
2	77:123	100:137	127:150	47	.	38	.	24	=	42,864
3	77:123	138:163	92:126	47	.	26	.	35	=	42,770
4	124:170	62:99	127:150	47	.	38	.	24	=	42,864
5	124:170	138:163	57:91	47	.	26	.	35	=	42,770
6	171:202	0:61	127:150	32	.	62	.	24	=	47,616
7	171:202	62:99	92:126	32	.	38	.	35	=	42,560
8	171:202	100:137	57:91	32	.	38	.	35	=	42,560

(1)
$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} & & & t_{24} & & t_{19} & & & \\ & & t_{18} & & t_{21} & & & & \\ & & & t_{17} & & & t_{21} & & \\ & t_{19} & & & & & & t_{21} & \\ & & & & & t_{27} & t_{18} & & \\ & t_{24} & & & & & & & t_{18} \\ & & & & & & & t_{17} & t_{27} \\ t_{27} & & & & t_{19} & & & & \\ t_{17} & & t_{24} & & & & & & \end{pmatrix} \end{array},$$

To each transition in the HLM matrix corresponds a Kronecker product of three (i.e., number of LLMs) LLM matrices. The matrices associated with those LLMs that do not participate in a transition are all identity. LLM 1 participates in t_{18}, t_{19}, t_{21} and t_{24} respectively with the matrices $Q^{(1)}_{t_{18}}$, $Q^{(1)}_{t_{19}}$, $Q^{(1)}_{t_{21}}$ and $Q^{(1)}_{t_{24}}$; LLM 2 participates in t_{17}, t_{18}, t_{21}, t_{27} respectively with the matrices $Q^{(2)}_{t_{17}}$, $Q^{(2)}_{t_{18}}$, $Q^{(2)}_{t_{21}}$ and $Q^{(2)}_{t_{27}}$; and LLM 3 participates in t_{17}, t_{19}, t_{24}, and t_{27} respectively with the matrices $Q^{(3)}_{t_{17}}$, $Q^{(3)}_{t_{19}}$, $Q^{(3)}_{t_{24}}$ and $Q^{(3)}_{t_{27}}$. In general, these matrices are very sparse and therefore held in row sparse format [41]. In this example, each of the transitions t_{17}, t_{18}, t_{19}, t_{21}, t_{24}, t_{27} affects exactly two LLMs. For instance, the Kronecker product associated with t_{24} in element (0,3) of the HLM matrix in equation (1) is

$$Q^{(1)}_{t_{24}}(171:202, 77:123) \otimes I_{26} \otimes Q^{(3)}_{t_{24}}(0:56, 92:126),$$

where $Q^{(1)}_{t_{24}}(171:202, 77:123)$ denotes the submatrix of $Q^{(1)}_{t_{24}}$ that lies between states 171 through 202 rowwise and states 77 through 123 columnwise, I_{26} denotes the identity matrix of order 26, $Q^{(3)}_{t_{24}}(0:56, 92:126)$ denotes the submatrix of $Q^{(3)}_{t_{24}}$ that lies between states 0 through 56 rowwise and states 92 through 126 columnwise, and $\otimes$ is the Kronecker product operator [45]. Hence, this particular Kronecker product yields a $(47,424 \times 42,770)$ matrix. The rates associated with the 18 transitions in (1) are all 10,000. The transition rates are scalars that multiply the corresponding Kronecker products.

Other than Kronecker products due to the transitions in (1), there is a Kronecker sum implicitly associated with each diagonal element of the HLM matrix. Each Kronecker sum is formed of three LLM matrices corresponding to *local transition* t_0. For instance, the Kronecker sum associated with element (4,4) of the HLM matrix is

$$Q^{(1)}_{t_0}(124:170, 124:170) \oplus Q^{(2)}_{t_0}(62:99, 62:99) \oplus Q^{(3)}_{t_0}(127:150, 127:150),$$

where $\oplus$ is the Kronecker sum operator. Each Kronecker sum is a sum of three Kronecker products in which all but one of the matrices are identity. The non-identity matrix in each Kronecker product appears in the same position as in the Kronecker sum. That state changes do not take place in any but one of the LLM matrices with t_0 in each such Kronecker product is the reason behind naming t_0 a local transition. The particular Kronecker sum associated with element (4,4) of the HLM matrix is $(42,864 \times 42,864)$.

In the HLM matrix of *qh_realcontrol* in (1), there do not exist any non-local transitions along the diagonal. In general, this need not be so. Therefore, we introduce the following definition [12].

TABLE 2
Three nested partitionings along the diagonal in qh_realcontrol.

HLM	Level 0		Level 1		Level 2	
state	blks	ordr	blks	ordr	blks	ordr
0	1	47,424	32	1,482	832	57
1	1	48,048	77	624	2,002	24
2	1	42,864	47	912	1,786	24
3	1	42,770	47	910	1,222	35
4	1	42,864	47	912	1,786	24
5	1	42,770	47	910	1,222	35
6	1	47,616	32	1,488	1,984	24
7	1	42,560	32	1,330	1,216	35
8	1	42,560	32	1,330	1,216	35
$\Sigma =$	9		393		13,266	

DEFINITION 2.1. *Let the diagonal block (j,j) of Q corresponding to element (j,j) of the HLM matrix be denoted by $Q_{j,j}$. Then*

$$Q_{j,j} = \bigoplus_{k=1}^{K} Q_{t_0}^{(k)}(\mathcal{S}_j^{(k)}, \mathcal{S}_j^{(k)}) + \sum_{t_e \in \mathcal{T}_{j,j}} rate_{t_e}(j,j) \bigotimes_{k=1}^{K} Q_{t_e}^{(k)}(\mathcal{S}_j^{(k)}, \mathcal{S}_j^{(k)}) + D_j, \tag{2}$$

where K is the number of LLMs, $\mathcal{S}_j^{(k)}$ is the subset of states of LLM k mapped to macrostate j, $\mathcal{T}_{j,j}$ is the set of LLM non-local transitions in element (j,j) of the HLM matrix, $rate_{t_e}(j,j)$ is the rate associated with transition $t_e \in \mathcal{T}_{j,j}$, and D_j is the diagonal (correction) matrix that sums the rows of Q corresponding to macrostate j to zero.

When there are multiple macrostates, Q is a block matrix having as many blocks in each dimension as the number of macrostates (i.e., order of the HLM matrix). The diagonal blocks of this partitioning are the $Q_{j,j}$ matrices defined in equation (2). The diagonal of Q is formed of its negated off-diagonal row sums, and may be stored explicitly or can be generated as needed.

In Example 1, the second term in equation (2) is missing. Although Q in *qh_realcontrol* is of order 399,476 and has 1,871,004 nonzeros, the Kronecker representation associated with the HMM needs to store 1 HLM matrix having 18 nonzeros and 15 LLM matrices (since identity matrices are not stored) having a total of 1,486 nonzeros.

In Table 2, we provide three nested partitionings along the diagonal of Q defined by the Kronecker structure of the HMM in *qh_realcontrol*. The columns blks and ordr list respectively number and order of blocks in each macrostate for the partitionings. Since the HLM has multiple macrostates, there exists a partitioning at level 0. The diagonal blocks at level 0 can be partitioned further as defined by LLM 1 at level 1 (i.e., one block is defined for each state of LLM 1) and LLM 2 defines the next level of partitioning (i.e., one block is defined for each pair of states of LLM 1 and LLM 2).

The next section introduces the BSOR preconditioner for Kronecker structured representations.

3. BSOR as a preconditioner for HMMs. Our aim is to solve the singular linear system $\pi Q = 0$ subject to the normalization condition $||\pi||_1 = 1$, where π is the (row) stationary probability vector of Q. We assume that Q is irreducible; hence, the stationary vector of Q is also its steady state vector.

Projection methods (or Krylov subspace methods) [38] are state-of-the-art iterative solvers developed mostly in the last twenty years that may also be used to solve for the stationary vector of Markov chains [36, 32, 23, 41, 37, 18, 6]. A concise discussion on popular projection methods and the motivation behind preconditioning may be found in [3]. A recent survey of preconditioning techniques for large sparse linear systems appears in [5]. The objective in preconditioning is to transform the linear system so that it becomes easier to solve with the iterative method at hand. To provide effective solvers, projection methods are used with preconditioners. This requires the preconditioning matrix (or preconditioner) to approximate the coefficient matrix of the original system in some sense and the solution of linear systems involving the preconditioner to be cheap. The need for a preconditioner becomes vital when the problem of interest is especially difficult to solve. Various types of preconditioners have been and are still being developed [38, 25]. Their efficiency is highly dependent on the system to be solved and it is quite difficult to forecast which preconditioner is the best for a given system.

Results with preconditioned projection methods on MCs underlying Kronecker structured representations are reported in a number of papers [42, 10, 11, 16, 28]. The preconditioner based on truncated Neumann series [41, 42] is computationally expensive to be effective and therefore impractical, whereas the cheap and separable preconditioner [10, 11] that forms (the inverse of) the preconditioner using the LLM non-local transition submatrices and the inverses of LLM local transition submatrices is not consistently effective. The circulant preconditioner in [16] can be used only with a certain class of SANs.

The Kronecker product approximate preconditioner for SANs introduced recently in [28], although encouraging, is in the form of a prototype implementation. In [27, pp. 100–113], numerical results with the preconditioner are presented using Matlab for nine problems all of which are feed-forward queueing networks; two of the larger problems consider models having independent subsystems each of which can be analyzed separately, in isolation. Yet, all test problems can be thought of as being HMMs with one macrostate and having K LLMs (see Definition 2.1), a total of E non-local transitions, and specific nonzero structure in LLM matrices. Assuming that $T = K + 2E$, the proposed preconditioner [27, pp. 99–100] requires the computation of $KT(T+1)/2$ traces of the products of pairs of LLM matrices, the solution of a nonlinear minimization problem of KT variables, the computation of K smaller matrices each of which is a weighted sum of T LLM matrices, and the inversion of the K smaller matrices that are computed. The Kronecker product of the inverted smaller matrices forms (the inverse of) the proposed preconditioner for SANs.

Results in [27] indicate that in terms of reducing the number of iterations to convergence of projection methods, such as Generalized Minimum RESidual (GMRES) [39] and BiConjugate Gradient STABilized (BICGSTAB) [44], the Kronecker product approximate preconditioner demonstrates similar behavior to that of the cheap and separable preconditioner in [10]. The difference in the number of iterations with GMRES and BICGSTAB between the two preconditioners is not more than a few iterations in any of the nine test problems. Furthermore, there are cases in which the cheap and separable preconditioner yields fewer iterations. We also remark that in general the inverted smaller matrices in the proposed preconditioner are likely to be less sparse than the inverted LLM local transition matrices in the cheap and separable preconditioner since each inverted smaller matrix is a weighted sum of T matrices one of which is an LLM local transition matrix. Still, there seems to be some timing advantages that may be gained with the Kronecker product approximate preconditioner since the preconditioning step at each iteration with it involves a single vector-Kronecker product multiplication, whereas with the cheap and separable preconditioner it involves two multiplications. The first multiplication is with a Kronecker product having the inverses of the LLM local transition matrices as factors (which are likely to be sparser than their counterparts in the Kronecker product approximate preconditioner), and the second multiplication is with a sum of Kronecker products due to LLM non-local transition matrices each of which is almost always sparse. The excess setup time of the proposed Kronecker product approximate preconditioner over the cheap and separable preconditioner is dictated by the time to solve the nonlinear minimization problem of KT variables. In conclusion, it is not evident what results the Kronecker product approximate preconditioner will yield on a full-fledged implementation in sparse storage suitable for larger and more complex models.

The SOR method and its block version are preconditioned power iterations (see [41, p. 144] or [25, p. 26, pp. 147-149]), and therefore can also be used with projection methods as preconditioners. Although generally inferior to incomplete LU (ILU) factorization type preconditioners ([36, p. 467], [32] and [18]), this study shows that the proposed BSOR implementation results in an effective preconditioner for MCs underlying large and complex Kronecker structured representations.

Since we work with row vectors, we consider a right BSOR preconditioner with relaxation parameter $w \in (0, 2)$. Given a block partitioning of Q, let Q be split in block form according to the partitioning as

$$Q = \left(\frac{1}{w}D - U\right) - \left(\frac{1-w}{w}D + L\right), \tag{3}$$

where D, $-U$, and $-L$ are square matrices respectively formed of the block diagonal, block strictly upper-triangular, and block strictly lower-triangular parts of Q. Then the BSOR preconditioning matrix is given by

$$M_{BSOR} = \omega^{-1}D - U. \tag{4}$$

In other words, it is the first term in equation (3) (see [25, p. 149]).

At each iteration of the underlying solver, the (row) residual vector, r, (which may have been computed explicitly or implicitly) is used as the right-hand side of the linear system

$$zM_{BSOR} = r \tag{5}$$

to compute the preconditioned (row) residual vector, z [25, pp. 25-26]. The objective of this preconditioning step is to correct the error in the approximate solution vector at that iteration. Note that if M_{BSOR} were the identity matrix, the preconditioned residual would be equal to the residual computed at that iteration. However, M_{BSOR} is not the identity matrix, but rather used to obtain hopefully an improved solution. For instance, a partitioning that may be used with the *qh_realcontrol* problem in forming a BSOR preconditioner is the one having 393 diagonal blocks at level 1 (see Table 2).

ALGORITHM 1
BSOR Preconditioned Solve for HMMs: $zM_{BSOR} = r$

For each macrostate j, sequentially:

(a) Compute negated right-hand side b:
- Set b by $-r_j$; add to b product of z_i with $Q_{i,j}$ for all $i < j$;

(b) Solve block upper-triangular part at level $l(j)$ of $Q_{j,j}$ for z_j using b as right-hand side:
- For each diagonal block k at level $l(j)$ of $Q_{j,j}$, sequentially:
  i. Solve diagonal block k at level $l(j)$ in $Q_{j,j}$ for subvector k of z_j with precomputed factors using negated subvector k of b as right-hand side;
  ii. If $(w \neq 1)$, set subvector k of z_j by w times subvector k of z_j;
  iii. Add to b product of subvector k of z_j with corresponding blocks in block upper-triangular part at level $l(j)$ of $Q_{j,j}$.

Algorithm 1 is a high-level description of the preconditioned solve in equation (5) for HMMs. Note that it is possible to employ different partitioning levels in different macrostates (see the parameter $l(j)$). This provides considerable flexibility in choosing favorable partitionings. The (row) vectors z_j and r_j denote respectively the subvectors of z and r corresponding to macrostate j. Note that the vector b needs to be as long as z_j when macrostate j is considered; hence, b is allocated so that it is as long as the maximum number of microstates among all macrostates (see Table 1). The negated right-hand side b is used in Algorithm 1 since the vector-Kronecker product multiplication routine is coded so as to add onto an input vector. Therefore, right before solving a diagonal block, the appropriate subvector of b is negated and used as the right-hand side. Note that if one has multiple macrostates and a level 0 partitioning, then there is only one block per macrostate and step (b) of Algorithm 1 simplifies accordingly.

Next, following section 3 in [12], we provide a summary of the implementation details of the particular BSOR preconditioner and explain how the diagonal blocks of the chosen partitioning are factorized.

4. BSOR preconditioner implementation. The diagonal blocks that correspond to a partitioning of an irreducible CTMC have negative diagonal elements and nonnegative off-diagonal elements. Such diagonal blocks are nonsingular [7]. Algorithm 2 in [12] describes how we set up the BSOR preconditioner for HMMs which can be used to accelerate the convergence of projection methods for solving the underlying CTMC. As we next explain, it may be possible to reduce the number of factorized diagonal blocks.

4.1. Benefiting from real Schur factorization. In HMMs, Kronecker sums contribute only to the diagonal of the HLM matrix. Furthermore, the contribution of a Kronecker sum associated with a macrostate is the same to all diagonal blocks in that macrostate. Therefore, under certain conditions, it is possible to have diagonal blocks with identical off-diagonal parts and diagonals differing from each other by a multiple of the identity matrix. We name such diagonal blocks as *candidate blocks* [12] in that macrostate. To detect candidate blocks, one must check conditions related to transitions that appear in the HLM matrix. We use Algorithm 1 in [12] to detect candidate blocks in each macrostate.

Recall that the real Schur factorization of a real nonsymmetric square matrix B exists [40, p. 114] and can be written as $B = ZTZ^T$. The matrix T is quasi-triangular meaning it is block triangular with blocks of order 1 or 2 along the diagonal; the blocks of order 1 contain the real eigenvalues of B and the blocks of order 2 contain the pairs of complex conjugate eigenvalues of B. On the other hand, the matrix Z is orthogonal and contains the real Schur vectors of B. When both T and Z are requested, the cost of

factorizing B of order m into real Schur form, assuming it is full, is $25m^3$ [19, p. 185]. Note that T and Z are unique up to a permutation P since $B = (ZP)(P^TTP)(ZP)^T$. We assume without loss of generality that T is quasi-upper-triangular.

Let $B_1 = ZTZ^T$ be the real Schur factorization of the first candidate block in the macrostate under consideration. Let $B_i = B_1 + \lambda_i I$, $i > 1$, represent its ith candidate block. Then $B_i = Z(T + \lambda_i I)Z^T$. Hence, all candidate blocks in the same macrostate can utilize the T and Z factors of the first candidate block. Consequently, the solution of a nonsingular linear system whose coefficient matrix is a candidate block requires two vector-matrix multiplications and one quasi-triangular solve. All needs to be done is to store λ_i for each candidate block and the real Schur factors T and Z in each macrostate. When the computed real Schur factors are sparse this implies significant storage savings and in some cases a reduction in solution time.

The real Schur factors of a candidate block may be obtained using the CLAPACK routine `dgees` [19, p. 185] available at [31]. This routine effectively uses two two-dimensional double precision arrays the first of which has the particular matrix on input and its T factor on output, whereas the second has its Z factor on output. The returned factors can be compacted and stored as sparse matrices to be used in the iterative part of a solver. However, this approach is not feasible for large candidate blocks due to time and space requirements associated with the `dgees` routine. The next subsection states a proposition which enables one to construct the real Schur factors from smaller submatrices so that the expensive real Schur factorization of the larger candidate blocks can be circumvented.

4.2. Candidate blocks having real eigenvalues. The following proposition in [12] specifies sufficient conditions for a candidate block to have real eigenvalues (i.e., upper-triangular T factor).

PROPOSITION 4.1. *Let the real Schur factorization of the local transition submatrix of LLM k in element (j, j) of the HLM matrix be given by (see Definition 2.1)*

$$Q_{t_0}^{(k)}(\mathcal{S}_j^{(k)}, \mathcal{S}_j^{(k)}) = Z_k T_k {Z_k}^T,$$

where T_k is its (quasi-)upper-triangular factor and Z_k is its orthogonal factor. Also let $\tilde{D}_j$ denote the diagonal block of D_j associated with the candidate block at level $l(j)$ in macrostate j. If for macrostate j:

(a) each T_k for $k > l(j)$ is upper-triangular, and

(b) each $\bigotimes_{k>l(j)}({Z_k}^T Q_{t_e}^{(k)}(\mathcal{S}_j^{(k)}, \mathcal{S}_j^{(k)})Z_k)$ that contributes to the candidate block at level $l(j)$ for all $e \in \mathcal{T}_{j,j}$ is upper-triangular, and

(c) $(\bigotimes_{k>l(j)} {Z_k}^T)\tilde{D}_j(\bigotimes_{k>l(j)} Z_k)$ is diagonal,

then the candidate block at level $l(j)$ in macrostate j has real eigenvalues.

We remark that part (a) of Proposition 4.1 is satisfied, for instance, when the LLM local transition submatrices that are mapped to macrostate j for LLMs $(l(j) + 1)$ and higher are triangular. Its part (b) is satisfied by all macrostates along the diagonal of the HLM matrix in many HMMs arising from closed queueing networks as in Example 1 which do not have any non-local transitions along the diagonal of their HLM matrices (i.e., $\mathcal{T}_{j,j} = \emptyset$ for all j). Note also that it suffices for the first nondiagonal factor in the Kronecker product of part (b) to be an upper-triangular matrix to satisfy the condition for the particular $e \in \mathcal{T}_{j,j}$ (see Appendix A in [43, pp. 181–183]). Checking part (b) of the proposition requires one to have previously computed the multipliers that multiply each Kronecker product in forming the candidate block when $l(j) > 0$. However, this is something we already do in detecting candidate blocks. Finally, [12] also shows how one can check part (c) of Proposition 4.1 and build the product using orthogonal real Schur factors of LLM local transition submatrices and $\tilde{D}_j$.

As indicated in [12], Proposition 4.1 also suggests an approach to construct the T and Z factors of the candidate block that is to be real Schur factorized at level $l(j)$ in macrostate j from the real Schur factors of the LLM local transition submatrices, the LLM non-local transition submatrices, and D_j.

4.3. Reordering LLMs. When Proposition 4.1 does not apply to the original ordering of LLMs, it may apply to a reordering of LLMs as in the next *kanban* problem.

Example 2. Consider the following smaller model associated with a manufacturing system having Kanban control [29] with the submatrices

$$Q_{t_0}^{(1)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ Q_{t_1}^{(1)} = \begin{pmatrix} 0 & 0 \\ 10 & 0 \end{pmatrix},\ Q_{t_2}^{(1)} = I,\ Q_{t_3}^{(1)} = I,$$

$$Q_{t_0}^{(2)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ Q_{t_1}^{(2)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ Q_{t_2}^{(2)} = I,\ Q_{t_3}^{(2)} = I,$$

$$Q_{t_0}^{(3)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ Q_{t_1}^{(3)} = I,\ Q_{t_2}^{(3)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ Q_{t_3}^{(3)} = \begin{pmatrix} 0 & 0 \\ 10 & 0 \end{pmatrix},$$

$$Q_{t_0}^{(4)} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},\ Q_{t_1}^{(4)} = I,\ Q_{t_2}^{(4)} = I,\ Q_{t_3}^{(4)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Let the corresponding HLM matrix have one state with the three transitions, t_1, t_2 and t_3, in element (0,0) and the rates of all transitions be 1. Then the corresponding CTMC may be obtained from (see Definition 2.1)

$$Q = \bigoplus_{k=1}^{4} Q_{t_0}^{(k)} + \sum_{t_e \in \{1,2,3\}} \bigotimes_{k=1}^{4} Q_{t_e}^{(k)} + D,$$

where D is the diagonal correction matrix that sums the rows of Q to zero, as

(6) $Q =$

$$\left(\begin{array}{cccc|cccc|cccc|cccc}
-3 & & 1 & & 1 & & & & 1 & & & & & & & \\
1 & -4 & & 1 & & 1 & & & & 1 & & & & & & \\
 & 10 & -12 & & & & 1 & & & & 1 & & & & & \\
 & & 1 & -3 & & & & 1 & & & & 1 & & & & \\
\hline
 & & 10 & & -12 & & 1 & & & & & & 1 & & & \\
 & & & 10 & 1 & -13 & & 1 & & & & & & 1 & & \\
 & & & & & 10 & -11 & & & & & & & & 1 & \\
 & & & & & & 1 & -2 & & & & & & & & 1 \\
\hline
 & & & & 10 & & & & -12 & & 1 & & 1 & & & \\
 & & & & & 10 & & & 1 & -13 & & 1 & & 1 & & \\
 & & & & & & 10 & & & 10 & -21 & & & & 1 & \\
 & & & & & & & 10 & & & 1 & -12 & & & & 1 \\
\hline
 & & & & & & & & & & 10 & & -11 & & 1 & \\
 & & & & & & & & & & & 10 & 1 & -12 & & 1 \\
 & & & & & & & & & & & & & 10 & -10 & \\
 & & & & & & & & & & & & & & 1 & -1
\end{array}\right).$$

At level 2 there are 4 blocks of order 4 along the diagonal of Q. Observe that $Z_3 = I_2$ (since $Q_{t_0}^{(3)}$ is strictly upper-triangular), $Q_{t_3}^{(3)}$ is strictly lower-triangular, and the 4 multipliers associated with the contribution of $({Z_3}^T Q_{t_3}^{(3)} Z_3) \otimes ({Z_4}^T Q_{t_3}^{(4)} Z_4)$ to the 4 diagonal blocks of Q at level 2 are 1 (since $Q_{t_3}^{(1)} = Q_{t_3}^{(2)} = I_2$). Hence, Q does not satisfy Proposition 4.1 at level 2 since $Z_3^T Q_{t_3}^{(3)} Z_3$ is strictly lower-triangular and this contradicts part (b).

Now consider the version of the *kanban* model in which LLMs are reordered as (3 4 2 1). This results in the symmetric permutation of Q given by

(7) $Q' =$

$$\left(\begin{array}{cccc|cccc|cccc|cccc}
-3 & 1 & 1 & & & & & & 1 & & & & & & & \\
 & -12 & 10 & 1 & & & & & & 1 & & & & & & \\
 & & -12 & 1 & & & 1 & & 10 & & 1 & & & & & \\
 & & & -11 & & & & 1 & & 10 & & 1 & & & & \\
\hline
1 & & & & -4 & 1 & 1 & & & & & & 1 & & & \\
 & 1 & & & & -13 & 10 & 1 & & & & & & 1 & & \\
 & & 1 & & & & -13 & 1 & & & & & 10 & & 1 & \\
 & & & 1 & & & & -12 & & & & & & 10 & & 1 \\
\hline
 & & & & 10 & & & & -12 & 1 & 1 & & & & & \\
 & & & & & 10 & & & & -21 & 10 & 1 & & & & \\
 & & & & & & 10 & & & & -11 & 1 & & & & \\
 & & & & & & & 10 & & & & -10 & & & & \\
\hline
 & & & & & & & & 1 & & & & -3 & 1 & 1 & \\
 & & & & & & & & & 1 & & & & -12 & 10 & 1 \\
 & & & & & & & & & & 1 & & & & -2 & 1 \\
 & & & & & & & & & & & 1 & & & & -1
\end{array}\right).$$

When LLMs are reordered as (3 4 2 1), transitions t_1 and t_3 do not pose any problems for Proposition 4.1. Regarding the 4 multipliers associated with the contribution of $({Z_2}^T Q_{t_2}^{(2)} Z_2) \otimes ({Z_1}^T Q_{t_2}^{(1)} Z_1)$ to the 4

TABLE 3
Problems.

Attribute	*qh_realcontrol*	*msmq_medium*	*msmq_large*
HLM states	$\{0:8\}$	$\{0:14\}$	$\{0:34\}$
nz_{HLM}	18 ($rates \in \{10,000\}$)	25 ($rates \in \{1\}$)	75 ($rates \in \{10\}$)
LLM 1 states	$\{0:76, 77:123, 124:170, 171:202\}$	$\{0:5, 6:16, 17:31\}$	$\{0:6, 7:19, 20:37, 38:59\}$
LLM 2 states	$\{0:61, 62:99, 100:137, 138:163\}$	$\{0:5, 6:16, 17:31\}$	$\{0:6, 7:19, 20:37, 38:59\}$
LLM 3 states	$\{0:56, 57:91, 92:126, 127:150\}$	$\{0:5, 6:16, 17:31\}$	$\{0:6, 7:19, 20:37, 38:59\}$
LLM 4 states	None	$\{0:5, 6:16, 17:31\}$	$\{0:6, 7:19, 20:37, 38:59\}$
LLM 5 states	None	$\{0:5, 6:16, 17:31\}$	$\{0:6, 7:19, 20:37, 38:59\}$
LLM matrices	15	15	15
nz_{LLMs}	1,486	370	790
$\mathcal{T}(i,j), i \neq j$	$\{t_{17}, t_{18}, t_{19}, t_{21}, t_{24}, t_{27}\}$	$\{t_1, t_2, t_3, t_4, t_5\}$	$\{t_{14}, t_{15}, t_{16}, t_{17}, t_{18}\}$
$\mathcal{T}(j,j)$	None	None	None
n	399,476	358,560	2,945,880
nz	1,871,004	2,135,160	19,894,875

Attribute	*kanban_medium*	*kanban_large*	*kanban_fail*
HLM states	$\{0\}$	$\{0\}$	$\{0:7\}$
nz_{HLM}	3 ($rates \in \{1\}$)	3 ($rates \in \{1\}$)	40 ($rates \in \{0.00001, 0.0001, 10\}$)
LLM 1 states	$\{0:10\}$	$\{0:19\}$	$\{0:35, 36:80\}$
LLM 2 states	$\{0:65\}$	$\{0:65\}$	$\{0:35, 36:80\}$
LLM 3 states	$\{0:65\}$	$\{0:65\}$	$\{0:35, 36:80\}$
LLM 4 states	$\{0:10\}$	$\{0:19\}$	$\{0:2, 3:6, 7:10, 11:15,$ $16:19, 20:24, 25:29, 30:35\}$
LLM matrices	10	10	20
nz_{LLMs}	370	406	792
$\mathcal{T}(i,j), i \neq j$	None	None	$\{t_4, t_5, t_6, t_7, t_{12}, t_{13}\}$
$\mathcal{T}(j,j)$	$\{t_1, t_2, t_3\}$	$\{t_1, t_2, t_3\}$	$\{t_{10}, t_{14}\}$
n	527,076	1,742,400	2,302,911
nz	3,001,405	10,183,360	14,313,663

Attribute	*courier_medium*	*courier_large*
HLM states	$\{0:9\}$	$\{0:12\}$
nz_{HLM}	47 ($rates \in \{1\}$)	65 ($rates \in \{1, 1449.3, 4821.6, 8771.9\}$)
LLM 1 states	$\{0:14\}$	$\{0:29\}$
LLM 2 states	$\{0:1, 2, 3:5, 6:18,$ $19:22, 23:74, 75:216\}$	$\{0, 1:140, 141, 142:201, 202,$ $203:222, 223, 224:227, 228\}$
LLM 3 states	$\{0, 1, 2:5, 6, 7:26, 27, 28:87\}$	$\{0:14\}$
LLM 4 states	$\{0:29\}$	$\{0:321, 322:326, 327:470, 471:474,$ $475:526, 527:529, 530:542, 543:544, 545\}$
LLM matrices	14	14
nz_{LLMs}	845	2,333
$\mathcal{T}(i,j), i \neq j$	$\{t_0, t_2, t_3, t_4\}$	$\{t_0, t_{28}, t_{29}, t_{30}\}$
$\mathcal{T}(j,j)$	$\{t_1, t_5\}$	$\{t_{17}, t_{23}\}$
n	419,400	1,632,600
nz	2,281,620	9,732,330

diagonal blocks of Q' that are of order 4, they are all 0 (since both diagonal elements of $Q^{(3)}_{t_2}$ are 0). In fact, all (4×4) diagonal blocks of Q' are upper-triangular. Hence, the reordered CTMC satisfies Proposition 4.1 for diagonal blocks of order 4.

As Example 2 shows, reordering LLMs may help in satisfying Proposition 4.1. It is our experience that there is considerable sparsity and structure in Kronecker structured Markovian representations which result in Proposition 4.1 being satisfied in many cases (with sparse real Schur factors for candidate blocks).

4.4. When all else fails. We use the column approximate minimum degree ordering (COLAMD) [17] on those diagonal blocks that are not candidates and that do not satisfy Proposition 4.1 to reduce the fill-in produced by their sparse LU factorizations. See subsection 3.4 in [12] for more information.

In the next section we provide the test problems used in numerical experiments.

5. Test problems. We experiment with eight problems. The characteristics of these problems are given in Table 3. For each problem, we provide the macrostates (HLM states), the number of nonzeros in the HLM matrix (nz_{HLM}) and their values (rates), the state space partition of each LLM (LLM states), the number of LLM matrices (LLM matrices), the total number of nonzeros in LLM matrices (nz_{LLMs}), the transitions in the off-diagonal part ($\mathcal{T}(i,j), i \neq j$) and the diagonal part ($\mathcal{T}(j,j)$) of the HLM matrix,

TABLE 4
Ordering of LLMs and partitionings used.

Problem	Ordering	Level	blks	cdts	ordr	nz_{LU}	nz_{Schur}	
qh_realcontrol	(1 2 3)	1	393	393	624–1,488	0	50,725	(399,476)
msmq_medium	(1 2 3 4 5)	2	913	913	216–726	0	40,425	(358,560)
msmq_large	(1 2 3 4 5)	2	3,621	3,621	343–2,197	0	214,629	(2,945,880)
kanban_medium	(3 4 2 1)	2	726	121	726	1,537,305	3,267	(87,846)
kanban_large	(3 4 2 1)	2	1,320	220	1,320	5,190,900	6,039	(290,400)
kanban_fail	(1 2 3 4)	2	13,122	2,349	135–225	11,177,577	6,248	(431,568)
courier_medium	(1 2 3 4)	2	4,245	4,245	30–1,800	0	30,436	(419,400)
courier_large	(1 2 4 3)	2	13,590	13,590	15–4,830	0	66,137	(1,632,600)
	(2 4 1 3)	2	3,628	2,464	450	10,325,844	33,618	(1,108,800)

the number of states (n) and the number of nonzeros (nz) of the underlying CTMC.

The *qh_realcontrol* problem is introduced in Example 1 and the smaller version of the *kanban_medium* and *kanban_large* problems is introduced in Example 2. We also consider another version of the *kanban* problem in which the machines can fail and name it as *kanban_fail*. We consider two versions of the multiserver multiqueue problem discussed in [1] and name them as *msmq_medium* and *msmq_large*. Finally, we consider two problems associated with the Courier protocol in [46] named *courier_medium* and *courier_large* (see also [9]). The CTMCs underlying all problems are irreducible.

The *qh_realcontrol*, *msmq_medium*, *kanban_medium*, and *courier_medium* problems have in the order of 100,000 states, whereas the other problems have in the order of 1,000,000 states. The *kanban_medium* and *kanban_large* problems have one macrostate; the other problems have multiple macrostates. The *qh_realcontrol*, *msmq_medium*, and *msmq_large* problems do not have any non-local transitions along the diagonal of their HLM matrices. Regarding non-local transitions, each LLM in *qh_realcontrol* participates in four transitions, whereas each of those in *msmq_medium* and *msmq_large* participate in two transitions. In *kanban_medium* and *kanban_large* LLM 2 and 3 each participates in two transitions while each of the other two LLMs participate in one transition. In *kanban_fail* LLM 4 participates in six transitions, LLM 1 participates in four transitions and each of the other two LLMs participate in three transitions. In *courier_medium* LLM 2 and 3 each participates in four transitions while each of the other two LLMs participates in one transition. Similar to *courier_medium*, in *courier_large* LLM 2 and 4 each participate in four transitions while each of the other two LLMs participate in one transition. Observe that the number of LLM matrices in each problem is the sum of the number of LLMs (since there is a local transition matrix that implicitly contributes to the diagonal of the HLM matrix per LLM) and the total number of non-local transitions in which LLMs participate. The *qh_realcontrol* and *kanban_fail* problems are especially difficult to solve owing it to the existence of nonzeros in their HLM and LLM matrices that have considerably different orders of magnitude.

In Table 4 we specify the ordering of LLMs (Ordering) and the associated partitionings (Level, blks, ctds, ordr, nz_{LU}, nz_{Schur}) used (with BSOR) in all problems. The column ordr gives the minimum and maximum order of diagonal blocks and the columns nz_{LU} and nz_{Schur} give respectively the number of nonzeros in the sparse LU and real Schur factors of the corresponding partitionings. The number in parentheses in column nz_{Schur} indicates the nonzeros used by the reciprocals of the diagonals of the T factors of candidate blocks which we store explicitly.

In five of the problems, we employ the original ordering of LLMs. When the original ordering does not yield a favorable partitioning in terms of macrostates having a suitable number of candidate blocks that satisfy Proposition 4.1, or the number and order of blocks, it is possible to consider different orderings of LLMs. That we do in *kanban_medium*, *kanban_large*, and *courier_large*. In all problems with the indicated ordering of LLMs in Table 4, there are some candidate blocks. With the original ordering of LLMs in *qh_realcontrol*, *msmq_medium*, *msmq_large*, *courier_medium* and with the ordering (1 2 4 3) of LLMs in *courier_large*, all diagonal blocks at the specified partitioning level are candidates and they satisfy Proposition 4.1. Hence, in these five cases there are no nonzeros in the nz_{LU} column and the amount of storage required by the real Schur factors is quite modest. Note that when $nz_{LU} = 0$, the number in parentheses in the nz_{Schur} column is equal to n as expected. In *courier_large*, we also consider the ordering (2 4 1 3) of LLMs to show that sometimes at the expense of extra storage one may be better off in terms of solution time. We use the real Schur factorization approach only in those candidate blocks

TABLE 5
Results of qh_realcontrol: (1 2 3), $w = 1.0$, $l = 1$.

Solver	it	res	Setup	Solve
STR_SOR	3,720	10^{-9}	0	1,670
STR_RSOR	3,610	10^{-9}	0	1,929
STR_GMRES(20)	4,280	10^{-2}	0	5,005
STR_BICGSTAB	3,827	10^{-9}	0	2,342
STR_TFQMR	5,000	10^{-2}	0	3,004
PRE_GMRES(20)	2,920	10^{-2}	0	5,025
PRE_BICGSTAB	4,382	10^{-8}	0	5,003
PRE_TFQMR	4,356	10^{-8}	0	5,002
STR_BSOR	3,430	10^{-9}	6	1,859
BSOR_GMRES(20)	3,080	10^{-1}	6	5,017
BSOR_BICGSTAB	276	10^{-9}	6	274
BSOR_TFQMR	246	10^{-8}	6	**241**

TABLE 6
Results of msmq_medium: (1 2 3 4 5), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	360	10^{-9}	0	117
STR_RSOR	240	10^{-9}	0	106
STR_GMRES(20)	5,000	10^{-4}	0	4,983
STR_BICGSTAB	325	10^{-9}	0	147
STR_TFQMR	398	10^{-9}	0	185
PRE_GMRES(20)	3,320	10^{-4}	0	5,028
PRE_BICGSTAB	222	10^{-10}	0	221
PRE_TFQMR	226	10^{-10}	0	223
STR_BSOR	120	10^{-9}	2	60
BSOR_GMRES(20)	119	10^{-10}	2	177
BSOR_BICGSTAB	47	10^{-9}	2	43
BSOR_TFQMR	46	10^{-10}	2	**41**

TABLE 7
Results of msmq_large: (1 2 3 4 5), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	360	10^{-9}	3	1,652
STR_RSOR	190	10^{-9}	3	965
STR_GMRES(20)	540	10^{-4}	3	5,027
STR_BICGSTAB	401	10^{-9}	3	2,020
STR_TFQMR	484	10^{-10}	3	2,385
PRE_GMRES(20)	300	10^{-5}	3	5,097
PRE_BICGSTAB	388	10^{-9}	3	5,013
PRE_TFQMR	278	10^{-11}	3	3,554
STR_BSOR	120	10^{-9}	24	771
BSOR_GMRES(20)	52	10^{-9}	24	747
BSOR_BICGSTAB	46	10^{-9}	24	486
BSOR_TFQMR	46	10^{-10}	24	**471**

that satisfy Proposition 4.1. Hence, even though there are non-candidate blocks in *kanban_medium* and *kanban_large* with the ordering (3 4 2 1) of LLMs, in *kanban_fail* with the original ordering of LLMs, and in *courier_large* with the ordering (2 4 1 3) of LLMs, the real Schur factors of candidate blocks are also sparse and require modest storage in these cases. Sparse LU factorizations are performed using COLAMD.

In the next section we present results of experiments with the eight problems in Table 3 using the LLM orderings and partitionings in Table 4.

6. Numerical results. We implemented the BSOR preconditioner as discussed in the previous section in C as part of the APNN toolbox [4, 2]. In this study, we consider BSOR preconditioned versions of the projection methods GMRES, BICGSTAB, and (Transpose Free) Quasi-Minimal Residual (TFQMR) [24], which are respectively named BSOR_GMRES, BSOR_BICGSTAB, and BSOR_TFQMR. We compare all results with those of other HMM solvers available in the APNN toolbox. In particular, we compare BSOR_GMRES, BSOR_BICGSTAB, and BSOR_TFQMR with STR_SOR, STR_RSOR, STR_GMRES, STR_BICGSTAB, STR_TFQMR, PRE_GMRES, PRE_BICGSTAB, PRE_TFQMR, and STR_BSOR. The STR_SOR solver implements a BSOR like method which uses diagonal blocks at level 0 with relaxation parameter w but does not attempt to solve diagonal blocks. When there is a single macrostate, STR_SOR becomes the point Jacobi over-relaxation (JOR) method. The STR_RSOR solver implements a point SOR method with relaxation parameter w similar to the one discussed in [43]. The STR_GMRES solver implements restarted GMRES with a fixed number of vectors for the Krylov subspace as discussed in [41, p. 198]. We use a Krylov subspace size of 20. The STR_BICGSTAB solver implements BiCGStab as discussed in [3, pp. 27–28]. The STR_TFQMR solver implements TFQMR as discussed in [24]. The PRE_GMRES, PRE_BICGSTAB, and PRE_TFQMR solvers are respectively preconditioned versions of STR_GMRES, STR_BICGSTAB, and STR_TFQMR using the cheap and separable preconditioner mentioned in section 3 [11]. Finally, STR_BSOR is the two-level version of the block SOR solver with relaxation parameter w proposed for HMMs in [12].

All experiments are performed on a 550 MHz Pentium III processor and a 1 GBytes main memory under Linux. The large main memory is necessary due to the large number of vectors of length n used in

TABLE 8
Results of kanban_medium: (3 4 2 1) $w = 0.9$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	3,200	10^{-9}	1	3,268
STR_RSOR	1,240	10^{-9}	1	2,937
STR_GMRES(20)	1,380	10^{-9}	1	2,674
STR_BICGSTAB	959	10^{-9}	1	1,092
STR_TFQMR	4,390	10^{-8}	1	5,003
PRE_GMRES(20)	440	10^{-9}	1	1,554
PRE_BICGSTAB	1,880	10^{-7}	1	5,005
PRE_TFQMR	544	10^{-11}	1	1,462
STR_BSOR	1,110	10^{-9}	29	1,035
BSOR_GMRES(20)	178	10^{-10}	29	409
BSOR_BICGSTAB	164	10^{-9}	29	**259**
BSOR_TFQMR	188	10^{-10}	29	288

TABLE 9
Results of kanban_large: (3 4 2 1) $w = 0.9$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	1,430	10^{-7}	2	5,014
STR_RSOR	650	10^{-7}	2	5,022
STR_GMRES(20)	800	10^{-7}	2	5,062
STR_BICGSTAB	1,143	10^{-9}	2	4,355
STR_TFQMR	1,362	10^{-5}	2	5,012
PRE_GMRES(20)	420	10^{-8}	2	5,048
PRE_BICGSTAB	548	10^{-7}	2	5,020
PRE_TFQMR	544	10^{-9}	2	5,008
STR_BSOR	1,542	10^{-8}	171	5,008
BSOR_GMRES(20)	220	10^{-10}	171	1,753
BSOR_BICGSTAB	195	10^{-9}	171	**1,086**
BSOR_TFQMR	222	10^{-9}	171	1,224

TABLE 10
Results of kanban_fail: (1 2 3 4), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	950	10^{-8}	4	5,039
STR_RSOR	440	10^{-9}	4	4,319
STR_GMRES(20)	560	10^{-5}	4	5,085
STR_BICGSTAB	856	10^{-3}	4	5,013
STR_TFQMR	864	10^{-5}	4	5,013
PRE_GMRES(20)	320	10^{-5}	4	5,299
PRE_BICGSTAB	384	10^{-4}	4	5,036
PRE_TFQMR	380	10^{-9}	4	5,029
STR_BSOR	254	10^{-7}	29	5,022
BSOR_GMRES(20)	200	10^{-7}	29	5,205
BSOR_BICGSTAB	218	10^{-8}	29	5,024
BSOR_TFQMR	120	10^{-10}	29	**2,747**

TABLE 11
Results of courier_medium: (1 2 3 4), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	1,190	10^{-9}	1	1,239
STR_RSOR	360	10^{-9}	1	556
STR_GMRES(20)	2,800	10^{0}	1	5,035
STR_BICGSTAB	339	10^{-9}	1	394
STR_TFQMR	4,378	10^{-1}	1	5,004
PRE_GMRES(20)	1,440	10^{-1}	1	5,006
PRE_BICGSTAB	203	10^{-11}	1	559
PRE_TFQMR	180	10^{-11}	1	496
STR_BSOR	60	10^{-10}	4	154
BSOR_GMRES(20)	38	10^{-9}	4	164
BSOR_BICGSTAB	35	10^{-9}	4	**131**
BSOR_TFQMR	40	10^{-9}	4	142

projection methods. All times are reported as seconds of CPU time. In Tables 5 through 13, we report the times spent in setup and iterative parts of the solvers respectively under columns Setup and Solve, and indicate the fastest solvers in bold. The it column indicates the number of iterations it takes the solvers to stop and the res column indicates the infinity norm of the residual upon stopping. In the caption, l stands for level in Table 4. We use a stopping tolerance of 10^{-8} on the residual norm (or of its approximation). The maximum number of permissible iterations is 5,000 and the maximum permissible CPU time is 5,000 seconds. In solvers involving BiCGStab and TFQMR, each pass through the body of the code counts as two iterations rather than one. We choose to normalize the solution vector and compute the residual every 10 iterations in the solvers STR_SOR, STR_RSOR, and STR_BSOR. The relaxation parameter w is set to 1.0 except the *kanban_medium* and *kanban_large* problems in which it is set to 0.9 due to the fact that the reordered smaller version of this problem in Example 2 does not converge using STR_RSOR and STR_BSOR with $w = 1.0$ for the chosen partitioning. Although the BSOR preconditioner in the toolbox has the flexibility to sparse LU factorize the diagonal blocks at level 0 corresponding to macrostates that have a small number of microstates, or to use different partitioning levels at different macrostates, these features are turned off. In that sense, the results provided in this section may not be the best results that can be obtained with BSOR and BSOR preconditioned projection methods.

For the problems in which convergence is observed due to the stopping tolerance of 10^{-8} but the norm of the residual is found to be larger than 10^{-8}, we continued the iterative process by decreasing the stopping tolerance one order of magnitude at a time until we encountered a residual norm less then 10^{-8}. Such a situation is witnessed among BSOR preconditioned projection methods since we work with unnormalized solution vectors and the underlying CTMCs are not scaled. Recall that the system we solve is singular and a non-scaled coefficient matrix with considerably large entries may result in the residual norm being larger than what the (unnormalized) solution vector actually implies (see [18, p. 1697]) especially when convergence takes place rapidly. In only one of the problems we are not able to reduce the residual norm below 10^{-8} by iterating in this manner, and that happens to be with the BSOR_TFQMR solver in *qh_realcontrol* where the residual norm is computed to be 1.2×10^{-8}.

TABLE 12
Results of courier_large: (1 2 4 3), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	940	10^{-7}	3	5,006
STR_RSOR	130	10^{-9}	3	**899**
STR_GMRES(20)	640	10^{-1}	3	5,115
STR_BICGSTAB	379	10^{-9}	3	2,079
STR_TFQMR	926	10^{-2}	3	5,014
PRE_GMRES(20)	320	10^{-2}	3	5,216
PRE_BICGSTAB	254	10^{-10}	3	3,367
PRE_TFQMR	246	10^{-11}	3	3,258
STR_BSOR	60	10^{-10}	20	1,287
BSOR_GMRES(20)	41	10^{-9}	20	1,178
BSOR_BICGSTAB	43	10^{-9}	20	1,071
BSOR_TFQMR	42	10^{-9}	20	1,023

TABLE 13
Results of courier_large: (2 4 1 3), $w = 1.0$, $l = 2$.

Solver	it	res	Setup	Solve
STR_SOR	960	10^{-7}	3	5,043
STR_RSOR	130	10^{-9}	3	841
STR_GMRES(20)	640	10^{-1}	3	5,052
STR_BICGSTAB	379	10^{-9}	3	2,039
STR_TFQMR	926	10^{-2}	3	5,020
PRE_GMRES(20)	320	10^{-2}	3	5,175
PRE_BICGSTAB	257	10^{-10}	3	3,388
PRE_TFQMR	246	10^{-11}	3	3,246
STR_BSOR	110	10^{-9}	23	985
BSOR_GMRES(20)	77	10^{-9}	23	1,086
BSOR_BICGSTAB	75	10^{-10}	23	867
BSOR_TFQMR	68	10^{-9}	23	**784**

Nearly in all experiments the setup time of the BSOR preconditioner turns out to be a relatively small fraction of the total solution time with the BSOR preconditioned solver. This is mostly due to the fact that in all cases in Table 4 it is possible to construct the real Schur factors of candidate blocks from the real Schur factors of component matrices. However, even in those cases where a large fraction of the diagonal blocks are sparse LU factorized using COLAMD ordering, as in *kanban_medium*, *kanban_large*, and *kanban_fail*, the setup time is acceptable (see Tables 8-10). The difference between the setup time of *kanban_large* and *kanban_fail* is due to the difference between the order of diagonal blocks that get sparse LU factorized in each case.

In all problems there are at least two BSOR preconditioned projection methods (i.e., BSOR_TFQMR and BSOR_BICGSTAB) among the fastest five solvers (see Tables 5-13). If we exclude the *courier_large* problem with the ordering (1 2 4 3) of LLMs, the winner in all problems is either BSOR_TFQMR (five times) or BSOR_BICGSTAB (three times). The BSOR_GMRES(20) solver appears among the fastest five solvers six out of nine cases considered in Table 4. STR_BSOR and STR_RSOR are among the fastest five solvers respectively seven and six times. Finally, STR_SOR, STR_BICGSTAB and PRE_TFQMR are among the fastest five solvers respectively three, three, and two times.

STR_RSOR is the winner in the *courier_large* problem with the ordering (1 2 4 3) of LLMs with BSOR_TFQMR and BSOR_BICGSTAB coming respectively close second and third. Observe that in this case the orders of diagonal blocks in the BSOR preconditioner are relatively non-uniform compared to other cases. By reordering the LLMs as (2 4 1 3) we obtain diagonal blocks of order 450. Even though not all diagonal blocks are candidates in this alternative ordering, by investing more storage in the factorization of diagonal blocks one is able to obtain a stronger BSOR preconditioner, which makes BSOR_TFQMR the winner.

STR_BSOR is faster than STR_RSOR when there is sufficient decrease in the number of iterations to convergence with STR_BSOR. We remark that the right-hand side update that takes place due to the (block) strictly upper-triangular part at each iteration in STR_RSOR and STR_BSOR is detrimental to the efficient vector-Kronecker product multiplication algorithm. In *kanban_medium* and *kanban_large*, which are two problems with one macrostate, nonzeros of the underlying CTMCs are constrained mostly within the diagonal blocks of the chosen partitionings. Furthermore, diagonal blocks in these two problems are still relatively sparse after being factorized; therefore, even if the decrease in the number of iterations in these two problems with STR_BSOR is marginal, STR_BSOR performs much better than STR_RSOR timewise. These cannot be said for the *kanban_fail* and *courier_large* problems in which the two-level version of STR_BSOR is at a disadvantage. Nevertheless, it is possible to consider the three-level version of the STR_BSOR solver in [12] as well.

There is significant decrease in the number of iterations to convergence when BSOR is used as a preconditioner with projection methods. In fact, for those cases in which both unpreconditioned and BSOR preconditioned projection methods converge within the experimental framework, the number of iterations with the BSOR preconditioned solver is at most one fifth that of the unpreconditioned solver. Furthermore, there are a few cases in which the ratio is about one tenth. Although the cheap and separable preconditioner performs well on some problems, it is clearly inferior to the BSOR preconditioner. In conclusion, BSOR preconditioned BICGSTAB and TFQMR solvers can be recommended for HMMs.

7. Conclusion. CTMCs in the form of sums of Kronecker products have considerable structure that may be exploited in devising effective preconditioners for projection methods. A two-level BSOR preconditioner that exploits this structure is presented for HMMs. The idea of using one real Schur factorization per macrostate for the diagonal blocks of the BSOR preconditioner that differ from each other by a multiple of the identity (that is, candidate blocks) and COLAMD ordering in the remaining diagonal blocks tend to reduce storage taken by the BSOR preconditioner. When there is a relatively large number of candidate blocks and they all meet certain conditions (as in Proposition 4.1), the setup time of the BSOR preconditioner is expected to be relatively small compared to the total solution time with the BSOR preconditioned solver. To improve the situation for the BSOR preconditioner, one may consider different orderings of LLMs. Numerical experiments on a representative set of problems demonstrate that BSOR preconditioned BICGSTAB and TFQMR using these ideas are potentially effective solvers for Kronecker based Markovian representations.

REFERENCES

[1] M. Ajmone-Marsan, S. Donatelli, and F. Neri, *GSPN models of Markovian multiserver multiqueue systems*, Performance Evaluation, 11 (1990), pp. 227–240.
[2] APNN-Toolbox case studies. http://www4.cs.uni-dortmund.de/APNN-TOOLBOX/case_studies/
[3] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems*, SIAM Press, Philadelphia, Pennslyvania, 1994.
[4] F. Bause, P. Buchholz, and P. Kemper, *A toolbox for functional and quantitative analysis of DEDS*, in Quantitative Evaluation of Computing and Communication Systems, Lecture Notes in Computer Science 1469, R. Puigjaner, N. N. Savino, and B. Serra, eds., Springer Verlag, 1998, pp. 356–359.
[5] M. Benzi, *Preconditioning techniques for large linear systems: a survey*, Journal of Computational Physics, 182 (2002), pp. 418–477.
[6] M. Benzi and M. Tuma, *A parallel solver for large-scale Markov chains*, Applied Numerical Mathematics, 41 (2002), pp. 135–153.
[7] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM Press, Philadelphia, Pennslyvania, 1994.
[8] P. Buchholz, *A class of hierarchical queueing networks and their analysis*, Queueing Systems, 15 (1994), pp. 59–80.
[9] P. Buchholz, *Hierarchical structuring of superposed GSPNs*, IEEE Transactions on Software Engineering, 25 (1999), pp. 166–181.
[10] P. Buchholz, *Structured analysis approaches for large Markov chains*, Applied Numerical Mathematics, 31 (1999), pp. 375–404.
[11] P. Buchholz, *Projection methods for the analysis of stochastic automata networks*, in Numerical Solution of Markov Chains, B. Plateau, W.J. Stewart, and M. Silva, eds., Prensas Universitarias de Zaragoza, Zaragoza, Spain, 1999, pp. 149–168.
[12] P. Buchholz and T. Dayar, *Block SOR for Kronecker structured representations*, Technical Report TUD-FI03-02, Department of Computer Science, Dresden University of Technology, Dresden, Germany, 2003. http://www.inf.tu-dresden.de/Fak/berichte.html
[13] P. Buchholz and P. Kemper, *On generating a hierarchy for GSPN analysis*, Performance Evaluation Review, 26 (1998), pp. 5–14.
[14] P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper, *Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models*, INFORMS Journal on Computing, 12 (2000), pp. 203–222.
[15] J. Campos, S. Donatelli, and M. Silva, *Structured solution of asynchronously communicating stochastic models*, IEEE Transactions on Software Engineering, 25 (1999), pp. 147–165.
[16] R. H. Chan and W. K. Ching, *Circulant preconditioners for stochastic automata networks*, Numerische Mathematik, 87 (2000), pp. 35–57.
[17] COLAMD, *Column Approximate Minimum Degree Ordering*. http://www.cise.ufl.edu/research/sparse/colamd/
[18] T. Dayar and W. J. Stewart, *Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1691–1705.
[19] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM Press, Philadelphia, Pennslyvania, 1997.
[20] S. Donatelli, *Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and*

distributed state space, Performance Evaluation, 18 (1993), pp. 21–26.
[21] P. Fernandes, B. Plateau, and W. J. Stewart, *Efficient descriptor-vector multiplications in stochastic automata networks*, Journal of the ACM, 45 (1998), pp. 381–414.
[22] P. Fernandes, B. Plateau, and W. J. Stewart, *Optimizing tensor product computations in stochastic automata networks*, RAIRO Operations Research, 32 (1998), pp. 325–351.
[23] R. W. Freund and M. Hochbruck, *On the use of two QMR algorithms for solving singular systems and applications in Markov chain modelling*, Numerical Linear Algebra with Applications, 1 (1994), pp. 403–420.
[24] R. W. Freund, *A transpose-free quasi-minimal residual method for non-Hermitian linear systems*, SIAM Journal on Scientific Computing, 14 (1993), pp. 470–482.
[25] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM Press, Philadelphia, Pennslyvania, 1997.
[26] P. Kemper, *Numerical analysis of superposed GSPNs*, IEEE Transactions on Software Engineering, 22 (1996), pp. 615–628.
[27] A. N. Langville *Preconditioning for Stochastic Automata Networks*, Ph.D. Thesis, Operations Research, North Carolina State University, Raleigh, North Carolina, 2002. http://www.lib.ncsu.edu/etd/public/etd-232913131021900/etd-title.html
[28] A. N. Langville and W. J. Stewart, *A Kronecker product approximate preconditioner for SANs*, Numerical Linear Algebra with Applications, to appear.
[29] D. Mitra and I, Mitrani, *Analysis of a Kanban discipline for cell coordination in production lines II: stochastic demands*, Operations Research, 39 (1991), pp. 807–823.
[30] V. Migallón, J. Penadés, and D. B. Szyld, *Block two-stage methods for singular systems and Markov chains*, Numerical Linear Algebra with Applications, 3 (1996), pp. 413–426.
[31] Netlib, *A collection of mathematical software, papers, and databases.* http://www.netlib.org
[32] B. Philippe, Y. Saad, and W. J. Stewart, *Numerical methods in Markov chain modelling*, Operations Research, 40 (1992), pp. 1156–1179.
[33] B. Plateau, *On the stochastic structure of parallelism and synchronization models for distributed algorithms*, in Proceedings of the ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems, Austin, Texas, 1985, pp. 147–154.
[34] B. Plateau and K. Atif, *Stochastic automata network for modeling parallel systems*, IEEE Transactions on Software Engineering, 17 (1991), pp. 1093–1108.
[35] B. Plateau and J.-M. Fourneau, *A methodology for solving Markov models of parallel systems*, Journal of Parallel and Distributed Computing, 12 (1991), pp. 370–387.
[36] Y. Saad, *Projection methods for the numerical solution of Markov chain models*, in Numerical Solution of Markov Chains, W. J. Stewart, ed., M. Dekker, Inc., New York, 1991, pp. 455–471.
[37] Y. Saad, *Preconditioned Krylov subspace methods for the numerical solution of Markov chains*, in Computations with Markov Chains: Proceedings of the Second International Workshop on the Numerical Solution of Markov Chains, W. J. Stewart, ed., Kluwer, Boston, Massachusetts, 1995, pp. 49–64.
[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*. PWS Publishing, New York, 1996.
[39] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.
[40] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM Press, Philadelphia, Pennslyvania, 2002.
[41] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, New Jersey, 1994.
[42] W. J. Stewart, K. Atif, and B. Plateau, *The numerical solution of stochastic automata networks*, European Journal of Operational Research, 86 (1995), pp. 503–525.
[43] E. Uysal and T. Dayar, *Iterative methods based on splittings for stochastic automata networks*, European Journal of Operational Research, 110 (1998), pp. 166–186.
[44] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 631–644.
[45] C. F. Van Loan, *The ubiquitious Kronecker product*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 85-100.
[46] C. M. Woodside and Y. Li, *Performance Petri net analysis of communications protocol software by delay equivalent aggregation*, in Proceedings of the 4th International Workshop on Petri Nets and Performance Models, IEEE CS-Press, 1991, pp. 64–73.