Electronic Transactions on Numerical Analysis.
Volume 31, pp. 271-294, 2008.
Copyright © 2008, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

DECOMPOSITIONAL ANALYSIS OF KRONECKER STRUCTURED MARKOV CHAINS[*]

YUJUAN BAO[†], İLKER N. BOZKURT[‡], TUĞRUL DAYAR[‡], XIAOBAI SUN[§], AND KISHOR S. TRIVEDI[¶]

Abstract. This contribution proposes a decompositional iterative method with low memory requirements for the steady-state analysis of Kronecker structured Markov chains. The Markovian system is formed by a composition of subsystems using the Kronecker sum operator for local transitions and the Kronecker product operator for synchronized transitions. Even though the interactions among subsystems, which are captured by synchronized transitions, need not be weak, numerical experiments indicate that the solver benefits considerably from weak interactions among subsystems, and is to be recommended specifically in this case.

Key words. Markov chain, Kronecker representation, decomposition, iterative method, multigrid, aggregation, disaggregation.

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

1. Introduction.

In a system composed of subsystems, various events take place. Some events are constrained only to a particular subsystem and can be called local, while others require the involvement of multiple subsystems to be realized and can be called synchronized (or global). The infinitesimal generator matrix underlying Markovian systems composed by local and synchronized events can be expressed using the Kronecker sum operator for local transitions and the Kronecker product operator for synchronized transitions [24]. Since a Kronecker sum can be written as a sum of Kronecker products [31], the potentially large generator matrix of such systems can be kept in memory as a sum of Kronecker products of the smaller subsystem transition matrices without having to be generated and stored. With the help of a vector-Kronecker product algorithm [18], this enables, at the expense of increased analysis time, the iterative analysis of much larger Markovian models on a given computer than can be performed with the conventional flat, sparse matrix generation approach [28].

Throughout this work, we assume that the cross product of the state spaces of the subsystems is equal to the state space of the system. We further assume that each state of the system is reachable from every other state in the system, implying the irreducibility of the underlying generator matrix, consequently the existence of its steady-state vector. Now, letting the infinitesimal generator matrix corresponding to the continuous-time Markov chain (CTMC) underlying the Kronecker representation be denoted by Q, the objective is to solve the linear system of equations

$$\pi Q = 0 \tag{1.1}$$

for the (global) steady-state (row) vector, π, without generating Q and subject to the normalization condition $\sum_{i\in\mathcal{S}} \pi_i = 1$, where $\mathcal{S}$ is the state space of the CTMC.

Stochastic automata networks (SANs) [18, 24, 25], various classes of superposed stochastic Petri Nets (SPNs) [17, 19], and hierarchical Markovian models (HMMs) [3, 10, 11] are

[*]Received November 30, 2007. Accepted December 18, 2008. Published online on May 12, 2009. Recommended by Reinhard Nabben.

[†]Facebook, 156 University Ave., Palo Alto, CA 94301, USA (yujuan.bao@gmail.com).
[‡]Department of Computer Engineering, Bilkent University, TR-06800 Bilkent, Ankara, Turkey ({bozkurti,tugrul}@cs.bilkent.edu.tr).
[§]Department of Computer Science, Box 90291, Duke University, Durham, NC 27708-0291, USA (xiaobai@cs.duke.edu).
[¶]Department of Electrical and Computer Engineering, Box 90291, Duke University, Durham, NC 27708-0291, USA (kst@ee.duke.edu).

Markovian modeling formalisms utilizing such a Kronecker representation. In this context, the exponential growth of the size of the state space with the number of subsystems in the specification of the model is referred to as the state space explosion problem. The Kronecker based representation provides an elegant, memory conserving solution to the problem albeit not as timewise efficient as one would like to have. This has triggered much research, and currently, multilevel methods [7] and block SOR (BSOR) preconditioned projection methods [8] appear to be the strongest iterative solvers for the steady-state analysis of Kronecker based Markovian systems. The former class of solvers exhibit fast convergence on many problems, but it still has not been possible to provide a result characterizing their rate of convergence [9], and examples are known where convergence is slow. On the other hand, the latter class of solvers have many parameters that must be judiciously chosen for them to be effective, and they may yield considerable fill-in during the factorization of diagonal blocks in the BSOR preconditioner for certain problems. Hence, there is still room for research and the recent review in [16] can be consulted for issues pertinent to the analysis of Kronecker based Markovian representations.

This paper proposes a composite iterative method with low memory requirements for the steady-state analysis of Kronecker based Markovian representations. The method is based on decomposition into subsystems and is coupled with a Gauss-Seidel (GS) [30] relaxation step. In a given iteration, each subsystem is solved independently for its local steady-state vector by restricting the effects of synchronized transitions to the particular subsystem as a function of the global steady-state vector computed in the previous iteration. The Kronecker product of the local steady-state vectors constitutes the local term of the current global steady-state vector. Then the residual vector obtained by multiplying this local term with the generator matrix held in Kronecker form is used to compute a correction term for the current global steady-state vector through GS. The local term, together with the correction term, determine the current global steady-state vector. When the interactions among subsystems are weak, the Kronecker product of the local steady-state vectors is expected to produce a good approximation to the global steady-state vector and yields convergence in a small number of iterations.

As we will show later, the proposed method can be formulated using concepts from algebraic multigrid (AMG) [26] and iterative aggregation-disaggregation (IAD) [28, Ch. 6], which is originally proposed for stochastic matrices having a nearly completely decomposable block partitioning. However, we remark that the concept of weak interactions among (or near independence of) subsystems is orthogonal to the concept of near complete decomposability associated with stochastic matrices. This is so, because the former refers to the possibility of expressing an infinitesimal generator matrix as a Kronecker sum plus a term in which the nonzero values are smaller, compared to those in the Kronecker sum, whereas the latter refers to the possibility of symmetrically permuting and partitioning a stochastic matrix such that the off-diagonal blocks have smaller probabilities, compared to those in the diagonal blocks. In this sense, the two concepts can be classified as multiplicative and additive, respectively.

The fixed-point iteration presented in [13] for stochastic reward nets (SRNs) is also motivated by the concept of near independence of subsystems, but is approximative. Although not particularly geared towards Kronecker structured Markovian systems, the decomposition is inspired by the Kronecker sum operator reflecting the local evolution of subsystems. It can be considered as an extension of the work in [29], which is intended for a particular application area. There are other methods in the literature that are based especially on decomposing Kronecker structured Markovian representations. For instance, [4] provides an iterative method for SANs that bounds the solution from below and above using polyhedra theory and disag-

ETNA
Kent State University
http://etna.math.kent.edu

gregation. It is argued through two small examples that the generated bounds are satisfactory only if the interactions among subsystems are weak, or the rates of synchronized transitions are more or less independent of the states of subsystems. On the other hand, [5] introduces an approximative method for superposed GSPNs, which iteratively operates individually only on states that have higher steady-state probabilities; the remaining states are aggregated. Considerable storage savings are obtained for the global steady-state vector due to its compact representation as a Kronecker product of aggregated subsystem steady-state vectors. The method proposed in this paper is different from these methods in that it is not approximative and it aims to compute the solution exactly up to computer precision. It is coded into the APNN toolbox [1] which is developed for HMMs since, to the best of our knowledge, this is the toolbox having the largest set of steady-state solvers for Kronecker structured Markovian representations which can serve as benchmarks for our numerical experiments.

The next section introduces the Markovian model used in the paper and provides a small example. Section 3 presents the decompositional iterative method. The proposed method is compared with other iterative methods on various examples and numerical results are given in Section 4. The conclusion is drawn in Section 5.

2. Model composition.

Consider the following Kronecker representation of the CTMC underlying a Markovian system composed of multiple subsystems interacting through synchronized transitions, where $\bigoplus$ and $\bigotimes$ denote the Kronecker sum and Kronecker product operators [31], respectively.

DEFINITION 2.1. *Let K be the number of subsystems, $\mathcal{S}^{(k)} = \{0, 1, ..., |\mathcal{S}^{(k)}|-1\}$ be the state space of subsystem k for $k = 1, 2, ..., K$, t_0 be a local transition (one per subsystem), $\mathcal{T}$ be the set of synchronized transitions among subsystems, and r_{t_e} be the rate of synchronized transition $t_e \in \mathcal{T}$. Then*

$$Q = Q_{local} + Q_{\textit{synchronized}}, \tag{2.1}$$

where

$$Q_{local} = \bigoplus_{k=1}^{K} Q_{t_0}^{(k)}, \quad Q_{\textit{synchronized}} = \sum_{t_e \in \mathcal{T}} r_{t_e} \bigotimes_{k=1}^{K} Q_{t_e}^{(k)} + D,$$

D is the diagonal correction matrix that sums the rows of $Q_{\textit{synchronized}}$ to zero, $Q_{t_0}^{(k)}$ and $Q_{t_e}^{(k)}$ are matrices of order $|\mathcal{S}^{(k)}|$ capturing transitions between states of subsystem k under local transition t_0 and synchronized transition $t_e \in \mathcal{T}$, respectively.

PROPOSITION 2.2. *The matrices Q_{local} and $Q_{\textit{synchronized}}$ have zero row sums by construction.*

We remark that the state space of Q is K-dimensional and is given by $\mathcal{S} = \mathcal{S}^{(1)} \times \mathcal{S}^{(2)} \times \cdots \times \mathcal{S}^{(K)}$, where $\times$ is the cross product operator. Hence, each state in $\mathcal{S}$ can be represented by a K-tuple. Throughout the text, we denote the states in $\mathcal{S}$ by the K-tuple $s = (s_1, s_2, \ldots, s_K)$, where $s_k \in \mathcal{S}^{(k)}$ for $k = 1, 2, \ldots, K$.

EXAMPLE 2.3. We illustrate these concepts with a simple system which consists of K subsystems interacting through synchronized transitions. The kth subsystem has n_k redundant components (implying $|\mathcal{S}^{(k)}| = n_k + 1$) only one of which is working (i.e., operating) at a given time instant for $k = 1, 2, \ldots, K$. The working component in subsystem k fails independently of the working components in the other subsystems, with an exponentially distributed time having rate λ_k. When a working component in a subsystem fails, it is replaced with one of the intact redundant components in that subsystem in no time and one again has a working subsystem. Furthermore, there is one repairman in subsystem k who can repair a failed component independently of the failed components in the other subsystems with an exponentially distributed time having rate μ_k. Hence, the considered model is that of a system with redundant components, each of which does not have to be highly reliable. It improves steady-state availability by using a larger number of redundant components, and not by employing highly reliable individual components.

For this system, local transition rate matrices have the tridiagonal form in

$$Q_{t_0}^{(k)} = \begin{bmatrix} -\lambda_k & \lambda_k & & & \\ \mu_k & -(\lambda_k+\mu_k) & \lambda_k & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_k & -(\lambda_k+\mu_k) & \lambda_k \\ & & & \mu_k & -\mu_k \end{bmatrix}.$$

To center the discussion around the solution method rather than the model, at this point we consider the existence of a single synchronized transition, t_1, representing a global reset to the initial state (or a global repair corresponding to complete recovery by repairing all failed redundant components) with rate μ in which all redundant components are intact and all subsystems are functioning, when the system is in the state of total failure. Such synchronized transition rate matrices can be expressed as $Q_{t_1}^{(k)} = e_{n_k+1}e_1^T$, where e_j represents the jth column of the identity matrix $I_{|\mathcal{S}^{(k)}|}$ of order $|\mathcal{S}^{(k)}|$.

When $K = 2$ in this system, we have

$$Q = Q_{t_0}^{(1)} \bigoplus Q_{t_0}^{(2)} + \mu Q_{t_1}^{(1)} \bigotimes Q_{t_1}^{(2)} + D. \tag{2.2}$$

Furthermore, if $n_1 = n_2 = 2$, then

$$Q_{t_0}^{(1)} = \begin{bmatrix} -\lambda_1 & \lambda_1 & 0 \\ \mu_1 & -(\lambda_1+\mu_1) & \lambda_1 \\ 0 & \mu_1 & -\mu_1 \end{bmatrix}, \quad Q_{t_0}^{(2)} = \begin{bmatrix} -\lambda_2 & \lambda_2 & 0 \\ \mu_2 & -(\lambda_2+\mu_2) & \lambda_2 \\ 0 & \mu_2 & -\mu_2 \end{bmatrix},$$

$$Q_{t_1}^{(1)} = Q_{t_1}^{(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad D = \text{diag}(0,0,0,0,0,0,0,0,-\mu),$$

where diag$(\cdot)$ denotes a diagonal matrix which has its vector argument along its diagonal. Besides, the state space of Q is given by

$$\mathcal{S} = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\},$$

whereas the state spaces of the subsystems are given by

$$\mathcal{S}^{(1)} = \mathcal{S}^{(2)} = \{0,1,2\}.$$

Now, in order to see that weak interaction between the two subsystems has nothing to do with near complete decomposability, consider the uniformized stochastic matrix $P(\alpha) = I + Q/\alpha$

corresponding to

$$Q = \left[\begin{array}{ccc|ccc|ccc} * & \lambda_2 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ \mu_2 & * & \lambda_2 & 0 & \lambda_1 & 0 & 0 & 0 & 0 \\ 0 & \mu_2 & * & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ \hline \mu_1 & 0 & 0 & * & \lambda_2 & 0 & \lambda_1 & 0 & 0 \\ 0 & \mu_1 & 0 & \mu_2 & * & \lambda_2 & 0 & \lambda_1 & 0 \\ 0 & 0 & \mu_1 & 0 & \mu_2 & * & 0 & 0 & \lambda_1 \\ \hline 0 & 0 & 0 & \mu_1 & 0 & 0 & * & \lambda_2 & 0 \\ 0 & 0 & 0 & 0 & \mu_1 & 0 & \mu_2 & * & \lambda_2 \\ \mu & 0 & 0 & 0 & 0 & \mu_1 & 0 & \mu_2 & * \end{array}\right] \begin{array}{c} (0,0) \\ (0,1) \\ (0,2) \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array}, \tag{2.3}$$

where $*$ denote the negated off-diagonal row sums and $\alpha \in [\max_s |q_{s,s}|, \infty)$ for the values $\mu_1 = \mu_2 = \lambda_1 = \lambda_2 = 1$ and $\mu = 0.001$. Q is composed of two subsystems, each having three states that interact with a rate of 0.001; see (2.2). Hence, they are weakly interacting with regards to their local evolutions. On the other hand, no matter which one of the $9!$ reorderings of the nine states in $\mathcal{S}$ is used, $P(4)$ cannot be symmetrically permuted to nearly completely decomposable form with a decomposability parameter of 0.001; see (2.3). In fact, the current ordering of states with the block partitioning of $P(4)$ having three diagonal blocks of order three yields a degree of coupling (i.e., maximum block off-diagonal probability mass) of 0.5. As α approaches ∞, $P(\alpha)$ will end up possessing a nearly completely decomposable partitioning with nine diagonal blocks, each of order one.

In passing, we remark that the subject of this paper is not the steady-state solution of the uniformized stochastic matrix $P(\alpha)$ by a decompositional iterative method using aggregation-disaggregation based on a block partitioning, but it is the solution of the CTMC underlying a sum of Kronecker products by decomposition into subsystems. Now, we are in a position to introduce the proposed method.

3. Decompositional method.

Our decompositional solution approach is built upon two key components: systems of local equations obtained from the local transition rate matrices of subsystems, and a system of global equations for the correction of the Kronecker product of local solutions. In each iteration, the systems of local equations are solved first. The right-hand sides of the systems of local equations depend on the global solution. After solving the systems of local equations, the Kronecker product of local solutions is computed and used to find the new correction and, hence, the new global solution. In summary, the local solutions are used to improve the global solution, the global solution is used to improve the local solutions, and the systems of local equations and the system of global equations are solved alternatingly until a stopping criterion is met.

3.1. Local solutions and global correction.

We express the global solution π in (1.1) as the sum of two terms, one of which is the Kronecker product of the local solutions, $\pi^{(k)}$ for $k = 1, 2, \ldots, K$, and the other is the global correction y, as in

$$\pi = \bigotimes_{k=1}^{K} \pi^{(k)} - y. \tag{3.1}$$

This expression assumes that the local solutions are normalized (i.e., $\|\pi^{(k)}\|_1 = 1$); in other words, $\pi^{(k)}$ represents the local steady-state vector of subsystem k, for $k = 1, 2, \ldots, K$.

We substitute global state variables in the right-hand sides of the systems of local equations, $v^{(k)}(\pi)$, for $k = 1, 2, \ldots, K$, using local state variables and the correction variables, without altering the synchronization policy, and obtain

$$\pi^{(k)} Q_{t_0}^{(k)} = v^{(k)}(\pi), \qquad \text{subject to } \|\pi^{(k)}\|_1 = 1, \quad \text{for } k = 1, 2, \ldots, K. \tag{3.2}$$

Here, $v^{(k)}(\pi)$ are associated with synchronized transitions, through functions of local variables, and global correction variables appearing in the definition of π in (3.1). The π in parentheses indicates the dependence of the right-hand side on the global solution, that is, $v^{(k)}$ is a function of π. Specifically, one term is added to (subtracted from) the s_kth element of $v^{(k)}$ for every synchronized transition that moves subsystem k out of (into) the local state $s_k \in \mathcal{S}^{(k)}$. In contrast, $y = 0$ would be imposed in (3.1) if the subsystems were independent, and $v^{(k)}(\pi) = 0$ would be obtained for $k = 1, 2, \ldots, K$.

Now, let us proceed to show how the right-hand side vectors $v^{(k)}(\pi)$, for $k = 1, 2, \ldots, K$, of the systems of local equations are obtained on our running example, and then formalize our observations. In the following, π_s refers to the steady-state probability of the global state $s = (s_1, s_2, \ldots, s_K) \in \mathcal{S}$, whereas $\pi_{s_k}^{(k)}$ refers to the steady-state probability of the local state $s_k \in \mathcal{S}^{(k)}$ of subsystem k, for $k = 1, 2, \ldots, K$. Hence,

$$\pi_{s_k}^{(k)} = \sum_{l \neq k, s_l \in \mathcal{S}^{(l)}} \pi_{(s_1, s_2, \ldots, s_K)}. \tag{3.3}$$

EXAMPLE 2.3 (CONTINUED). Consider the nine global balance equations

$$\begin{aligned}
-(\lambda_1 + \lambda_2)\pi_{(0,0)} + \mu_2\pi_{(0,1)} + \mu_1\pi_{(1,0)} + \mu\pi_{(2,2)} &= 0\\
\lambda_2\pi_{(0,0)} - (\lambda_1 + \lambda_2 + \mu_2)\pi_{(0,1)} + \mu_2\pi_{(0,2)} + \mu_1\pi_{(1,1)} &= 0\\
\lambda_2\pi_{(0,1)} - (\lambda_1 + \mu_2)\pi_{(0,2)} + \mu_1\pi_{(1,2)} &= 0\\
\lambda_1\pi_{(0,0)} - (\lambda_1 + \lambda_2 + \mu_1)\pi_{(1,0)} + \mu_2\pi_{(1,1)} + \mu_1\pi_{(2,0)} &= 0\\
\lambda_1\pi_{(0,1)} + \lambda_2\pi_{(1,0)} - (\lambda_1 + \lambda_2 + \mu_1 + \mu_2)\pi_{(1,1)} + \mu_2\pi_{(1,2)} + \mu_1\pi_{(2,1)} &= 0\\
\lambda_1\pi_{(0,2)} + \lambda_2\pi_{(1,1)} - (\lambda_1 + \mu_1 + \mu_2)\pi_{(1,2)} + \mu_1\pi_{(2,2)} &= 0\\
\lambda_1\pi_{(1,0)} - (\lambda_2 + \mu_1)\pi_{(2,0)} + \mu_2\pi_{(2,1)} &= 0\\
\lambda_1\pi_{(1,1)} + \lambda_2\pi_{(2,0)} - (\lambda_2 + \mu_1 + \mu_2)\pi_{(2,1)} + \mu_2\pi_{(2,2)} &= 0\\
\lambda_1\pi_{(1,2)} + \lambda_2\pi_{(2,1)} - (\mu + \mu_1 + \mu_2)\pi_{(2,2)} &= 0
\end{aligned}$$

obtained by using (2.3) in (1.1). Summing up the first three global balance equations yields

$$-\lambda_1(\pi_{(0,0)} + \pi_{(0,1)} + \pi_{(0,2)}) + \mu_1(\pi_{(1,0)} + \pi_{(1,1)} + \pi_{(1,2)}) + \mu\pi_{(2,2)} = 0,$$

which is equivalent to

$$-\lambda_1\pi_0^{(1)} + \mu_1\pi_1^{(1)} + \mu\pi_{(2,2)} = 0,$$

since, from (3.3), the steady-state probability of subsystem 1 being in local state $s_1 \in \mathcal{S}^{(1)}$ is given by $\pi_{s_1}^{(1)} = \sum_{s_2 \in \mathcal{S}^{(2)}} \pi_{(s_1, s_2)}$. Adding the next three and the last three global balance equations in a similar manner yields, respectively,

$$\begin{aligned}
\lambda_1\pi_0^{(1)} - (\lambda_1 + \mu_1)\pi_1^{(1)} + \mu_1\pi_2^{(1)} &= 0,\\
\lambda_1\pi_1^{(1)} - \mu_1\pi_2^{(1)} - \mu\pi_{(2,2)} &= 0.
\end{aligned}$$

Now, observe that the three equations resulting from the addition of specific mutually exclusive global balance equations, and use of local steady-state probabilities, can be expressed as a linear system of the form

$$Q_{t_0}^{(1)}\pi^{(1)} = v^{(1)}(\pi), \qquad \text{subject to } \|\pi^{(1)}\|_1 = 1,$$

where

$$v^{(1)}(\pi) = (-\mu\pi_{(2,2)}, 0, \mu\pi_{(2,2)}).$$

Following the same line of argument, but this time adding the global balance equations one, four and seven, two, five and eight, and three, six and nine, one obtains

$$Q^{(2)}_{t_0}\pi^{(2)} = v^{(2)}(\pi), \qquad \text{subject to } \|\pi^{(2)}\|_1 = 1,$$

where

$$v^{(2)}(\pi) = (-\mu\pi_{(2,2)}, 0, \mu\pi_{(2,2)}).$$

The proposed method is related to iterative aggregation-disaggregation (IAD) [28, Ch. 6] for stochastic matrices, and algebraic multigrid (AMG) [26] for general systems of equations. Although there are variations of IAD, all of them combine the solution of the aggregated stochastic matrix, whose elements correspond to blocks in a block partitioning of the stochastic matrix, with pre- and/or post-iteration steps over the global system of equations. The solution of the aggregated system distributes the steady-state probability mass over state space partitions, whereas for each block the probability mass is distributed inside the corresponding state space partition. On the other hand, AMG solves a system of equations by performing iterations on systems of equations of decreasing size, where each system of equations is obtained by aggregation. We return to this point after we present the algorithm in the next section. Let us first formalize our observations. In doing that, we follow the approach taken in [9].

DEFINITION 3.1. *Let the surjective (i.e., onto) mapping $f_k : \mathcal{S} \longrightarrow \mathcal{S}^{(k)}$, which satisfies*

$$\exists s \in \mathcal{S} \ \text{ s.t. } \ f_k(s) = s_k \ \text{ for each } \ s_k \in \mathcal{S}^{(k)},$$

represent the transformation of states in $\mathcal{S}$ to states in $\mathcal{S}^{(k)}$, for $k = 1, 2, \ldots, K$, during the decomposition into subsystems.

Since the normalization of the local steady-state vector of each subsystem can be performed after it is computed, we introduce the K *restriction* (or aggregation) operators that are used to transform global steady-state probability variables to local steady-state probability variables of K different subsystems, as in the next definition.

DEFINITION 3.2. *The $(|\mathcal{S}| \times |\mathcal{S}^{(k)}|)$ restriction operator $R^{(k)}$ for the mapping*

$$f_k : \mathcal{S} \longrightarrow \mathcal{S}^{(k)},$$

$k = 1, 2, \ldots, K$, has its (s, s_k)th element given by

$$r^{(k)}_{s,s_k} = \begin{cases} 1, & \text{if } f_k(s) = s_k, \\ 0, & \text{otherwise}, \end{cases}$$

for $s \in \mathcal{S}$ and $s_k \in \mathcal{S}^{(k)}$. In Kronecker representation, $R^{(k)}$ is written as

$$R^{(k)} = \left(\bigotimes_{l=1}^{k-1} I_{|\mathcal{S}^{(l)}|} e \right) \bigotimes I_{|\mathcal{S}^{(k)}|} \bigotimes \left(\bigotimes_{l=k+1}^{K} I_{|\mathcal{S}^{(l)}|} e \right), \tag{3.4}$$

where e is the vector of ones of appropriate dimension.

The postmultiplication by e of each identity matrix $I_{|\mathcal{S}^{(l)}|}$, except $l = k$, in (3.4) corresponds to the aggregation of each dimension, except k, in the decomposition process. Equivalently, (3.4) can be interpreted as $I_{|\mathcal{S}^{(k)}|}$ being pre-Kronecker (post-Kronecker) multiplied by a vector of ones of length $\prod_{l=1}^{k-1} |\mathcal{S}^{(l)}|$ ($\prod_{l=k+1}^{K} |\mathcal{S}^{(l)}|$).

PROPOSITION 3.3. *The restriction operator $R^{(k)}$, for $k = 1, 2, \ldots, K$, is nonnegative (i.e., $R^{(k)} \geq 0$), has only a single nonzero with the value 1 in each row, and therefore row sums of 1, i.e., $R^{(k)}e = e$. Furthermore, since there is at least one nonzero in each column of $R^{(k)}$ (i.e., $e^T R^{(k)} > 0$), it is also the case that $\operatorname{rank}(R^{(k)}) = |\mathcal{S}^{(k)}|$.*

For each local state $s_k \in \mathcal{S}^{(k)}$, the global balance equations that are mapped to the same state $s_k \in \mathcal{S}^{(k)}$, for $k = 1, 2, \ldots, K$, are summed by the K *prolongation* (or disaggregation) operators defined next.

DEFINITION 3.4. *The $(|\mathcal{S}^{(k)}| \times |\mathcal{S}|)$ prolongation operator $P^{(k)}(\pi)$ for the mapping $f_k : \mathcal{S} \longrightarrow \mathcal{S}^{(k)}$, $k = 1, 2, \ldots, K$, has its (s_k, s)th element given by*

$$p^{(k)}_{s_k,s}(\pi) = \begin{cases} \pi_s/\pi^{(k)}_{s_k}, & \text{if } f_k(s) = s_k, \\ 0, & \text{otherwise}, \end{cases}$$

for $s \in \mathcal{S}$ and $s_k \in \mathcal{S}^{(k)}$.

Observe the dependency of the prolongation operator of each subsystem on the steady-state vector π.

PROPOSITION 3.5. *The prolongation operator $P^{(k)}(\pi)$ is nonnegative (that is, $P^{(k)}(\pi) \geq 0$), has the same nonzero structure as the transpose of $R^{(k)}$ (i.e., $(R^{(k)})^T$), has a single nonzero in each column (i.e., $e^T P^{(k)}(\pi) > 0$), and has at least one nonzero in each row, implying $\operatorname{rank}(P^{(k)}(\pi)) = |\mathcal{S}^{(k)}|$. Furthermore, each row of $P^{(k)}(\pi)$ is a probability vector, implying that $P^{(k)}(\pi)$ has row sums of 1 just like $R^{(k)}$.*

Now, we state three results that follow from the definitions of the particular restriction and prolongation operators.

LEMMA 3.6. *The prolongation operator $P^{(k)}(\pi)$ and the restriction operator $R^{(k)}$ satisfy*

$$P^{(k)}(\pi)R^{(k)} = I_{|\mathcal{S}^{(k)}|}, \quad \text{for } k = 1, 2, \ldots, K.$$

Proof. The identity follows from Propositions 3.3 and 3.5, as $P^{(k)}(\pi) \geq 0$, $R^{(k)} \geq 0$, $P^{(k)}(\pi)$ has the same nonzero structure as $(R^{(k)})^T$, $P^{(k)}(\pi)e = e$, and $e^T(R^{(k)})^T = e^T$. □

LEMMA 3.7. *The local steady-state vector of subsystem k is given by*

$$\pi^{(k)} = \pi R^{(k)}, \quad \text{for } k = 1, 2, \ldots, K,$$

and it satisfies

$$\pi = \pi^{(k)} P^{(k)}(\pi), \quad \text{for } k = 1, 2, \ldots, K.$$

Proof. The first part of the result follows from (3.3) and Definition 3.2, and the second part follows from (3.3) and Definition 3.4. □

LEMMA 3.8. *The $(|\mathcal{S}^{(k)}| \times |\mathcal{S}^{(k)}|)$ matrix*

$$Q^{(k)}(\pi) = P^{(k)}(\pi)QR^{(k)}, \quad \text{for } k = 1, 2, \ldots, K, \tag{3.5}$$

is the irreducible infinitesimal generator underlying the CTMC associated with subsystem k.

Proof. The result follows from the assumption that Q is an irreducible CTMC, $\pi > 0$, and similar arguments used in the proof of Lemma 4.1 in [9, p. 1038]. □

Observe the dependency on π in (3.5). The next result enables us to form the linear system of local equations to be solved for each subsystem.

COROLLARY 3.9. *The irreducible infinitesimal generator underlying the CTMC associated with subsystem k can be written as*

$$Q^{(k)}(\pi) = Q_{t_0}^{(k)} + P^{(k)}(\pi) Q_{\textit{synchronized}} R^{(k)}, \quad \textit{for } k = 1, 2, \ldots, K.$$

Proof. Observe from Lemma 3.8 and (2.1) that

$$Q^{(k)}(\pi) = P^{(k)}(\pi) Q_{local} R^{(k)} + P^{(k)}(\pi) Q_{\text{synchronized}} R^{(k)}.$$

Writing

$$Q_{local} = \bigoplus_{k=1}^{K} Q_{t_0}^{(k)} = \sum_{k=1}^{K} \left(\bigotimes_{l=1}^{k-1} I_{|\mathcal{S}^{(l)}|} \right) \bigotimes Q_{t_0}^{(k)} \bigotimes \left(\bigotimes_{l=k+1}^{K} I_{|\mathcal{S}^{(l)}|} \right)$$

and premultiplying Q_{local} by $P^{(k)}(\pi)$ and postmultiplying by $R^{(k)}$ yields the matrix $Q_{t_0}^{(k)}$, of size $(|\mathcal{S}^{(k)}| \times |\mathcal{S}^{(k)}|)$, which has row sums of zero from Proposition 2.2. □

EXAMPLE 2.3 (CONTINUED). Corollary 3.9 suggests for our running example that

$$Q^{(1)}(\pi) = Q_{t_0}^{(1)} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu\pi_{(2,2)}/\pi_2^{(1)} & 0 & -\mu\pi_{(2,2)}/\pi_2^{(1)} \end{bmatrix},$$

$$Q^{(2)}(\pi) = Q_{t_0}^{(2)} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu\pi_{(2,2)}/\pi_2^{(2)} & 0 & -\mu\pi_{(2,2)}/\pi_2^{(2)} \end{bmatrix}.$$

Note that although $Q^{(k)}(\pi)$ is irreducible for $k = 1, 2, \ldots, K$, $Q_{t_0}^{(k)}$ need not be.

The next definition introduces the projector which is used to prove that $\pi^{(k)}$ is the local steady-state vector of $Q^{(k)}(\pi)$, for $k = 1, 2, \ldots, K$.

DEFINITION 3.10. *The $(|\mathcal{S}| \times |\mathcal{S}|)$ matrix*

$$H^{(k)}(\pi) = R^{(k)} P^{(k)}(\pi), \quad \textit{for } k = 1, 2, \ldots, K,$$

defines a nonnegative projector (i.e., $H^{(k)}(\pi) \geq 0$ and $(H^{(k)}(\pi))^2 = H^{(k)}(\pi)$) which satisfies $H^{(k)}(\pi)e = e$.

LEMMA 3.11. *The steady-state vector π satisfies*

$$\pi H^{(k)}(\pi) = \pi, \quad \textit{for } k = 1, 2, \ldots, K.$$

Proof. The result follows from Definitions 3.2 and 3.4 and the fact that the restricted and then prolonged row vector is π. □

COROLLARY 3.12. *The local steady-state vector $\pi^{(k)}$ of subsystem k satisfies*

$$\pi^{(k)} Q^{(k)}(\pi) = 0, \quad \textit{for } k = 1, 2, \ldots, K. \tag{3.6}$$

Proof. We have

$$\begin{aligned}\pi^{(k)}Q^{(k)}(\pi) &= (\pi R^{(k)})(P^{(k)}(\pi)QR^{(k)}) = (\pi R^{(k)}P^{(k)}(\pi))QR^{(k)} \\ &= (\pi H^{(k)}(\pi))QR^{(k)} = (\pi)QR^{(k)} = (\pi Q)R^{(k)} = 0,\end{aligned}$$

from the first part of Lemma 3.7, Lemma 3.8, Definition 3.10, Lemma 3.11, and (1.1). □

THEOREM 3.13. *The linear system of local equations to be solved for subsystem k can be written as in (3.2), where*

$$v^{(k)}(\pi) = -\pi Q_{\text{synchronized}}R^{(k)}, \quad \text{for } k = 1, 2, \ldots, K, \tag{3.7}$$

and $v^{(k)}(\pi)$ satisfies $v^{(k)}(\pi)e = 0$.

Proof. Writing (3.6) from Corollary 3.9 as

$$\pi^{(k)}Q_{t_0}^{(k)} = -\pi^{(k)}P^{(k)}(\pi)Q_{\text{synchronized}}R^{(k)}$$

and using the second part of Lemma 3.7 yields the result. □

In the global solution, numerically significant correction to the Kronecker product of local solutions requires the solution of the system of global equations

$$yQ = \left(\bigotimes_{k=1}^{K}\pi^{(k)}\right)Q \tag{3.8}$$

for the global correction y. Note that Q is the Kronecker structured generator matrix in (2.1) for the non-altered, original Markovian system, and the nonzero right-hand side is the residual computed by multiplying the Kronecker product of the local solutions with Q. The systems of local equations in (3.2) and the system of global equations in (3.8) together are equivalent to the original system of equations for π in (1.1); they are linear in the local variables of each subsystem and in y.

In our method, we recompute the local solutions and the global correction alternatingly in each iteration starting with initial approximations until a predetermined stopping criterion is met. At first sight, there may seem to be no advantage in this. However, exploiting the Kronecker structure when solving the systems of local equations and the system of global equations speeds up the convergence to the global solution over other methods when the subsystems are weakly interacting, as we show in the section on numerical experiments. Now we present the solution algorithm.

3.2. Algorithm.

Let

$$Q = U - L$$

be the forward GS splitting of the generator matrix in Kronecker form, where U corresponds to its upper-triangular part and L contains the rest, as discussed in [16, pp. 287–289]. Note that one can also consider the more general SOR splitting with relaxation parameter ω — GS is SOR with $\omega = 1$ — from which we refrain in order not to clutter the discussion. The algorithm is stated in Algorithm 1, for a user-specified stopping tolerance tol, where subscripts within square brackets denote iteration numbers.

In step 0, the global correction vector is set to zero and the global solution vector is set to the uniform probability distribution. Note that a positive initial global solution vector is sufficient in exact arithmetic for the irreducibility of the aggregated matrices in the first iteration if the local transition rate matrices are reducible. In the current implementation, each

ETNA
Kent State University
http://etna.math.kent.edu

ALGORITHM 1 (Decompositonal iterative method with GS correction step).

```
0. Initial step:
       Set $it = 0,\ y_{[it]} = 0,\ \pi_{[it]} = e^T/n$.
1. Compute local solutions and normalize:
       If $Q_{t_0}^{(k)}$ is irreducible, solve $\pi_{[it+1]}^{(k)} Q_{t_0}^{(k)} = v^{(k)}(\pi_{[it]})$,
       else solve $\pi_{[it+1]}^{(k)} Q^{(k)}(\pi_{[it]}) = 0$,
       subject to $\|\pi_{[it+1]}^{(k)}\|_1 = 1$, for $k = 1, 2, \dots, K$.
2. Compute global correction:
       $y_{[it+1]} U = y_{[it]} L + \left(\bigotimes_{k=1}^{K} \pi_{[it+1]}^{(k)}\right) Q$.
3. Compute global solution, normalize, and check for termination:
       $\pi_{[it+1]} = \bigotimes_{k=1}^{K} \pi_{[it+1]}^{(k)} - y_{[it+1]}$, subject to $\|\pi_{[it+1]}\|_1 = 1$,
       exit if $\|\pi_{[it+1]} Q\|_\infty < tol$, otherwise $it = it + 1$ and return to step 1.
```

system of local equations is solved in step 1 using Gaussian elimination (GE), and the local solution is normalized. The use of GE is justified by the reasonably small number of states in each subsystem arising in practical applications. There are two cases. In the former case, as shown in Theorem 3.13, each linear system of local equations to be solved has a zero sum right-hand side vector due to the particular way in which synchronized transition rate matrices are specified in the composed model. If $Q_{t_0}^{(k)}$ is a singular negated M-matrix [2, p. 156] of rank $(|\mathcal{S}^{(k)}|-1)$ (this requires the irreducibility of $Q_{t_0}^{(k)}$), then a unique positive solution $\pi_{[it+1]}^{(k)}$ up to a multiplicative constant can be computed using the normalization condition. In the latter case, from Lemma 3.8, the coefficient matrix $Q^{(k)}(\pi_{[it]})$ is irreducible if Q is irreducible and $\pi_{[it]} > 0$. Hence, the existence of a unique positive solution up to a multiplicative constant is guaranteed.

Observe from Definition 2.1 that in each term $r_{t_e} \bigotimes_{k=1}^{K} Q_{t_e}^{(k)}$ of the global synchronized transition rate matrix $Q_{\text{synchronized}}$, the rates of nonzero transitions are obtained by multiplying the products of nonzero elements in $Q_{t_e}^{(k)}$ with the rate r_{t_e}. Hence, each global synchronized transition obtained in this way from some global state $(s_1, s_2, \dots, s_K)$ to some global state $(s_1', s_2', \dots, s_K')$ is due to synchronized transition from local state s_k to local state s_k' in subsystem k. Since the synchronized transition rate matrices $Q_{t_e}^{(k)}$ are in general very sparse, the enumeration process associated with the nonzeros in the global synchronized transition rate matrix to form the right-hand side vectors of the local systems of equations in (3.7), or the aggregated coefficient matrices in Corollary 3.9, can be handled in a systematic manner.

Note that there are differences from a computational point of view between using (3.2) versus (3.6) in step 1 of the proposed iterative method. In the former case, the local transition rate matrix is constant and already available in sparse format, meaning it can be factorized once at the outset. In the latter case, the aggregated coefficient matrix needs to be reconstructed and factorized at each iteration. Furthermore, in the former case, it is the right-hand side vector that is dependent on the current global solution $\pi_{[it]}$, whereas, in the latter case, it is the coefficient matrix that is dependent on $\pi_{[it]}$. The two approaches for obtaining the new local steady-state vector are therefore not equivalent except at steady-state (i.e., $\pi_{[it]} = \pi_{[it+1]} = \pi$), since the former case uses only the new local steady-state vector, whereas the latter case uses both the new and the current local steady-state vector in the left-hand side. The new local steady-state vector premultiplies the aggregated matrix, but the elements of the aggregated matrix are computed using the elements of the current global and

ETNA
Kent State University
http://etna.math.kent.edu

local steady-state vectors (see Definition 3.4).

In step 2, the global correction is computed by solving a triangular linear system in Kronecker form. The situation in this step is somewhat better compared to that in step 1, in that Q is already a singular M-matrix of rank $(|\mathcal{S}|-1)$ since it is an irreducible infinitesimal generator matrix by assumption. Hence, the global correction $y_{[it+1]}$ is obtained through a GS relaxation on Q with a zero sum (see (3.8)), but nonzero right-hand side as long as the method has not converged. Step 3 subtracts the global correction obtained in step 2 from the Kronecker product of local solutions obtained in step 1, to compute the new global solution vector. Steps 1 through 3 are repeated until the infinity norm of the residual falls below tol.

Two of the earlier papers which analyze iterative methods based on aggregation-disaggregation for linear systems with nonsingular coefficient matrices, using successive substitution together with restriction and prolongation operators, are [12] and [21]. The latter provides a local convergence proof. Convergence analysis of a two-level IAD method for Markov chains and its equivalence to AMG is provided in [20]. Another paper that investigates the convergence of a two-level IAD method for Markov chains using concepts from multigrid is [22]. Recently, in [23], the results from [22] have been improved, and an asymptotic convergence result is provided for a two-level IAD method which uses post-smoothings of the power iteration type. However, fast convergence cannot be guaranteed in a general setting even when there are only two-levels [23, p. 340].

Now, we take a look at what goes on during one iteration of the proposed method in more detail, and remark that the situation is different from that of the ML method [9] in two ways. First, the proposed method works in two levels, whereas the ML method utilizes K levels. Second, the proposed method solves K systems of local equations at the second level and these systems are obtained from a well-defined decomposition of a Kronecker structured Markov chain, whereas the ML method solves only one aggregated system of equations at each level and the aggregated system does not have to arise from a Kronecker decomposition.

Let

$$\tilde{\pi}_{[it+1]} = \bigotimes_{k=1}^{K} \pi_{[it+1]}^{(k)} \quad \text{and} \quad T_{GS} = LU^{-1}$$

represent the GS iteration matrix. Then, after some algebraic manipulation on the equation in step 2 using $Q = U - L$ and the definition of T_{GS}, we have

$$y_{[it+1]} = y_{[it]}T_{GS} + \tilde{\pi}_{[it+1]}(I - T_{GS}), \quad \text{for } it = 0, 1, \ldots.$$

Substituting this in the equation of step 3, we obtain

$$\pi_{[it+1]} = (\tilde{\pi}_{[it+1]} - y_{[it]})T_{GS}, \quad \text{for } it = 0, 1, \ldots.$$

Using $y_{[0]} = 0$, yields

$$\pi_{[1]} = \tilde{\pi}_{[1]}T_{GS}.$$

Continuing in this manner, we obtain

$$\pi_{[2]} = \tilde{\pi}_{[2]}T_{GS} - \pi_{[1]}(I - T_{GS}),$$

and eventually,

$$\pi_{[it+1]} = \tilde{\pi}_{[it+1]}T_{GS} - \left(\sum_{i=1}^{it} \pi_{[i]}\right)(I - T_{GS}), \quad \text{for } it = 0, 1, \ldots,$$

which implies

$$\pi_{[it+1]} = \pi_{[it]} T_{GS} + (\tilde{\pi}_{[it+1]} - \tilde{\pi}_{[it]}) T_{GS}, \quad \text{for } it = 0, 1, \ldots \tag{3.9}$$

(assuming that $\tilde{\pi}_{[0]} = \pi_{[0]}$). Equation (3.9) reveals that the proposed method computes the new global solution vector by summing two terms, the first of which is the GS iterated current global solution vector and the second of which is the GS iterated difference between the current and the previous Kronecker products of local solution vectors. The method will be advantageous only if the second term brings the new global solution vector closer to π than the first term alone.

Now, let us write $Q = A + B$, where $A = e\pi$ (that is, A is the stochastic matrix having the steady-state vector along its rows). Then, we have

$$A > 0, \quad A^2 = A, \quad Ae = e, \quad \pi A = \pi, \quad \pi(B + I) = 0, \quad A(B + I) = (B + I)A = 0,$$

$$H^{(k)}(\pi_{[it]})A = A \quad \text{and} \quad \pi^{(k)}_{[it+1]} P^{(k)}(\pi_{[it]})A = \pi, \quad \text{for } \pi^{(k)}_{[it+1]} > 0,\ \pi_{[it]} > 0.$$

These results follow from the definition of A, $Q = A + B$, Definition 3.10, and Proposition 3.5. Observing that A is a positive projector and using proof by contradiction as in [23, p. 330], it is possible to show that $P^{(k)}(\pi_{[it]})BR^{(k)}$ is nonsingular.

Let us consider the homogeneous linear systems with coefficient matrices $Q^{(k)}(\pi_{[it]})$ to be solved in step 2 of Algorithm 1. Then, from $Q = A + B$, the kth linear system can be reformulated as

$$\pi^{(k)}_{[it+1]} P^{(k)}(\pi_{[it]}) A R^{(k)} = -\pi^{(k)}_{[it+1]} P^{(k)}(\pi_{[it]}) B R^{(k)},$$

which implies

$$\pi R^{(k)} = \pi^{(k)} = -\pi^{(k)}_{[it+1]} P^{(k)}(\pi_{[it]}) B R^{(k)},$$

from $\pi^{(k)}_{[it+1]} P^{(k)}(\pi_{[it]})A = \pi$ and the first part of Lemma 3.7; thus,

$$\pi^{(k)}_{[it+1]} = -\pi^{(k)} (P^{(k)}(\pi_{[it]}) B R^{(k)})^{-1},$$

and, consequently,

$$\tilde{\pi}_{[it+1]} = \bigotimes_{k=1}^{K} -\pi^{(k)} (P^{(k)}(\pi_{[it]}) B R^{(k)})^{-1}.$$

From the compatibility of the Kronecker product with matrix multiplication and matrix inversion [31, pp. 85–86], this can be rewritten as

$$\tilde{\pi}_{[it+1]} = -\pi \left(P(\pi_{[it]}) \left(\bigotimes_{k=1}^{K} B \right) R \right)^{-1}, \tag{3.10}$$

where

$$\pi = \bigotimes_{k=1}^{K} \pi^{(k)}, \quad P(\pi_{[it]}) = \bigotimes_{k=1}^{K} P^{(k)}(\pi_{[it]}), \quad \text{and} \quad R = \bigotimes_{k=1}^{K} R^{(k)}.$$

We remark that

$$H(\pi_{[it]}) = RP(\pi_{[it]})$$

is a nonnegative projector, and (3.10) expresses the Kronecker product of the new local solution vectors in terms of the global solution vector, an $(n \times n^K)$ prolongation operator associated with the current global solution vector, the K-fold Kronecker product of part of the generator matrix, and an $(n^K \times n)$ restriction operator. It follows from Propositions 3.3 and 3.5 that R and $P(\pi_{[it]})$ are restriction and prolongation operators, respectively.

After substituting (3.10) in (3.9), we obtain

$$\pi_{[it+1]} = \pi_{[it]}T_{GS} - \pi \left[\left(P(\pi_{[it]}) \left(\bigotimes_{k=1}^{K} B \right) R \right)^{-1} - \left(P(\pi_{[it-1]}) \left(\bigotimes_{k=1}^{K} B \right) R \right)^{-1} \right] T_{GS},$$

which may be rewritten, using $B = Q - e\pi$ and (2.1), as

$$\pi_{[it+1]} = \left[\pi_{[it]} - \pi \bigotimes_{k=1}^{K} \left(Q_{t_0}^{(k)} + P^{(k)}(\pi_{[it]})Q_{\text{synchronized}}R^{(k)} - e\pi R^{(k)} \right)^{-1} \right.$$
$$\left. + \pi \bigotimes_{k=1}^{K} \left(Q_{t_0}^{(k)} + P^{(k)}(\pi_{[it-1]})Q_{\text{synchronized}}R^{(k)} - e\pi R^{(k)} \right)^{-1} \right] T_{GS}.$$

Observe that the new solution vector depends on both the current and the previous solution vectors.

In the next section, we present results of numerical experiments for some benchmark problems, larger versions of the running example with varying number and rates of synchronized transitions, and some randomly generated problems.

4. Numerical experiments. Experiments are performed on a PC with an Intel Core2 Duo 1.83GHz processor having 4 Gigabytes of main memory, running Linux. The large main memory is necessary to store the large number of vectors of length $|\mathcal{S}|$ used in some of the benchmark solvers in the APNN toolbox. The existence of two cores in the CPU is not exploited for parallel computing in the implementation. Hence, only one of the two cores is busy running solvers in the experiments.

The proposed decompositional method (D) is compared with the following iterative solvers: Jacobi (J), GS, block GS (BGS), generalized minimum residual with a Krylov subspace size of 20 (GMRES(20)), transpose-free quasi-minimal residual (TFQMR), biconjugate gradient stabilized (BICGSTAB), BGS preconditioned GMRES(20), TFQMR, BICGSTAB (that is, BGS_GMRES(20), BGS_TFQMR, BGS_BICGSTAB), multilevel with one pre- and one post-smoothing using GS, W-cycle, and cyclic order of aggregating subsystems in each cycle (ML_GS(W,C)). More information can be obtained on these methods in [27] except ML_GS(W,C) for which [9] can be consulted. In passing, we remark that BGS preconditioned projection methods and multilevel methods are state-of-the-art iterative solvers for Kronecker based Markovian representations [7].

The solvers are compared in terms of number of iterations to converge to a user-specified stopping tolerance, elapsed CPU time, and amount of allocated main memory. In the tables, the columns labelled as Iteration, Residual, Time, and Memory, report the number of iterations to converge, the infinity norm of the residual upon stopping, the CPU time taken in seconds, and the amount of memory, in megabytes, allocated by the solvers, respectively. An asterisk superscript over an iteration number indicates that convergence has not taken place

TABLE 4.1
Numerical results for the Overflow_large problem with $K = 6$, $|\mathcal{S}^{(1)}| = 6$, $|\mathcal{S}^{(2)}| = |\mathcal{S}^{(3)}| = \cdots = |\mathcal{S}^{(6)}| = 11$, $\mathcal{T} = \{t_1, t_2, \ldots, t_{30}\}$, $r_{t_1} = r_{t_2} = \cdots = r_{t_{30}} = 1$.

Method	Iteration	Residual	Time	Memory
J	1,120	$9.7e-09$	1,440	37
GS	590	$9.3e-09$	1,563	37
BGS	340	$9.1e-09$	699	1,685
GMRES(20)	160	$6.9e-09$	214	184
TFQMR	224	$3.2e-10$	257	88
BICGSTAB	126	$8.6e-09$	**145**	74
BGS_GMRES(20)	60	$5.8e-09$	237	1,869
BGS_TFQMR	66	$5.8e-09$	237	1,773
BGS_BICGSTAB	50	$1.4e-08$	194	1,758
ML_GS(W,C)	1,894*	$8.2e-03$	10,010	70
D	330	$8.8e-09$	1,339	52

for the particular solver in the allotted CPU time. Bold font in the Time column indicates the fastest solver. In all solvers, the maximum number of iterations is set to 5,000, maximum CPU time is set to 10,000 seconds, and $tol = 10^{-8}$ is enforced on the infinity norm of the residual. We remark that the stopping test is executed every 10 iterations in the proposed solver just like in the J, GS, and BGS solvers. This explains why all numbers of iterations to converge are multiples of 10 with these solvers, unless the solver stops due to the CPU time limit. Now, we turn to the numerical experiments.

4.1. Some benchmark problems. We have run experiments with the proposed solver on some benchmark problems such as *Kanban_medium* and *Kanban_large* [8], arising in a manufacturing system with Kanban control, *Availability* [9], arising in a system availability model with subsystems working at different time scales, and *Overflow_large* [1], arising in an overflow queueing network. We must remark that all local transition rate matrices in the *Kanban* problems are triangular, meaning they are reducible. Those in the *Availability* problem are reducible, and those in the *Overflow_large* problem are tridiagonal and therefore irreducible.

Nonzero values in the local transition rate matrices of the *Kanban* problems are 1, whereas those in the *Availability* problem are in $\{1\} \cup 10^{1-k}\{0.01, 0.02, 0.09, 0.18\}$ for subsystems $k = 1, 2, \ldots, 6$, and those in *Overflow_large* are in $\{1, 1.5\}$ for subsystem 1 and in $\{1\} \cup \{1.1 - 0.1k\}$ for subsystems $k = 2, 3, \ldots, 6$. Hence, the nonzero values in the local transition matrices are about the same order in the *Kanban* and *Overflow_large* problems, but not in the *Availability* problem. On the other hand, nonzero values in the transition rate matrices of the *Kanban* problems are in $\{1, 10\}$ for subsystems $k = 1, 2, 4$ and 1 for subsystem 3, whereas those in the *Availability* problem are 1, and those in *Overflow_large* are in $\{1, 1.5\}$ for subsystem 1 and in $\{1\} \cup \{1.1 - 0.1k\}$ for subsystems $k = 2, 3, \ldots, 6$.

Table 4.1 shows the performance of the proposed solver and the other solvers for one of these four problems, namely *Overflow_large*. In the set of experiments reported, the diagonal blocks associated with the BGS solvers and the BGS preconditioners for projection methods at level 3 are LU factorized [6] using the column approximate minimum degree (COLAMD) ordering [14, 15]. The number of nonzeros generated during the LU factorization with the COLAMD ordering of the 726 diagonal blocks of order 1,331 for *Overflow_large* is 132,500,082. The equivalent of these numbers in megabytes is accounted for in the memory consumed by solvers utilizing BGS.

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 4.2
Effect of near independence on the decompositional method for the Fail-Repair problem with $K = 2$, $n_1 = n_2 = 19$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\mathcal{T} = \{t_1, t_2\}$.

$r_{t_1} = r_{t_2}$	Iteration	Residual
1.0	270	$9.1e-09$
0.7	260	$8.3e-09$
0.6	250	$9.0e-09$
0.5	240	$9.4e-09$
0.1	90	$1.0e-09$
0.05	80	$8.4e-09$
0.01	60	$8.7e-09$
0.005	50	$8.8e-09$
0.001	10	$6.3e-09$

The memory requirement of the proposed solver is the smallest after J and GS solvers in the two problems. Observe that the rates of synchronized transitions in the *Overflow_large* problem are 1. We also remark that the *Overflow_large* problem uses the local transition rate matrices as coefficient matrices in the systems of local equations. However, the behavior of the proposed method does not change for the *Overflow_large* problem even if aggregated transition rate matrices are used. It is interesting to note that the decompositional iterative solver is not timewise competitive with the fastest solver although its number of iterations to converge can be smaller than that of the respective relaxation method. This is not surprising since the interactions among the subsystems in this problem are relatively strong. Now we turn to a problem in which it is advantageous to use the decompositional method.

4.2. Fail-Repair problem. First, we investigate the effect of changing the rates of synchronized transitions. We do this on an instance of the fail-repair problem discussed earlier as an example. The particular system has two subsystems each with 20 states, i.e., $n_1 = n_2 = 19$. Hence, we have a system of 400 states. There are two synchronized transitions, the first which takes the system into global state $(0, 0)$ with rate r_{t_1} when subsystems 1 and 2 are each in their local state 4, and the second which takes the system into global state $(5, 5)$ with rate r_{t_2} when subsystems 1 and 2 are in their local state 9. These synchronized transitions can be considered corresponding to batch repairs of four failed redundant components in each subsystem. We remark that the two subsystems are not identical since their local fail and repair rates are different.

Table 4.2 shows the number of iterations to converge to the solution with the proposed solver for various values of the two synchronized transition rates, which are taken to be identical. When the synchronized transition rates are small, convergence becomes very fast because the subsystems are nearly independent and the Kronecker product of the local solutions yields a very good approximation to the global solution early in the iteration.

In the next set of experiments, we consider a larger version of the fail-repair problem with five subsystems each having 20 states, resulting in a system of 3,200,000 states. There are four synchronized transitions in this system, the first takes the system into global state $(0, 0, 0, 0, 0)$ with rate r_{t_1} when all subsystems are in their local state 4, the second takes the system into global state $(5, 5, 5, 5, 5)$ with rate r_{t_2} when all subsystems are in their local state 9, the third takes the system into global state $(10, 10, 10, 10, 10)$ with rate r_{t_3} when all subsystems are in their local state 14, and the fourth takes the system into global state $(15, 15, 15, 15, 15)$ with rate r_{t_4} when all subsystems are in their local state 19. Local failure and repair rates of subsystems are not identical.

TABLE 4.3
Numerical results for the Fail-Repair problem with $K = 5$, $n_k = 19$ *for* $k = 1, 2, \ldots, K$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\lambda_3 = 0.6$, $\mu_3 = 0.5$, $\lambda_4 = 0.7$, $\mu_4 = 0.6$, $\lambda_5 = 0.8$, $\mu_5 = 0.7$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.5$.

Method	Iteration	Residual	Time
J	1,890	$9.8e-09$	1,722
GS	910	$9.4e-09$	1,237
BGS	244*	$1.6e-07$	10,060
GMRES(20)	580	$1.8e-09$	677
TFQMR	242	$4.3e-10$	219
BICGSTAB	117	$8.3e-09$	**104**
BGS_GMRES(20)	60	$4.2e-09$	2,647
BGS_TFQMR	112	$2.3e-10$	4,662
BGS_BICGSTAB	46	$1.2e-08$	1,969
ML_GS(W,C)	36	$9.3e-09$	142
D	60	$9.1e-09$	142

TABLE 4.4
Numerical results for the Fail-Repair problem with $K = 5$, $n_k = 19$ *for* $k = 1, 2, \ldots, K$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\lambda_3 = 0.6$, $\mu_3 = 0.5$, $\lambda_4 = 0.7$, $\mu_4 = 0.6$, $\lambda_5 = 0.8$, $\mu_5 = 0.7$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.05$.

Method	Iteration	Residual	Time
J	1,910	$9.8e-09$	1,734
GS	910	$9.9e-09$	1,254
BGS	244*	$1.7e-07$	10,060
GMRES(20)	800	$4.7e-09$	928
TFQMR	258	$1.5e-10$	231
BICGSTAB	153	$3.1e-09$	135
BGS_GMRES(20)	60	$4.3e-09$	2,645
BGS_TFQMR	112	$2.7e-10$	4,658
BGS_BICGSTAB	55	$3.5e-08$	2,342
ML_GS(W,C)	34	$8.6e-09$	135
D	30	$5.1e-09$	**72**

This problem is solved for three different values of the synchronized transition rates, which are taken to be identical in a given instance of the problem. The results are reported in Tables 4.3, 4.4, and 4.5. The rates of the four synchronized transitions are decreased gradually from 0.5 to 0.05 and then to 0.005. In this set of experiments, the diagonal blocks associated with the BGS solver, as well as and the BGS preconditioner for projection methods at level 3 (meaning 8,000 diagonal blocks of order 400) are LU factorized using the COLAMD ordering as in the benchmark problems. We remark that the number of nonzeros generated during the LU factorization of the 8,000 diagonal blocks of order 400 with the COLAMD ordering is 77,184,000. The equivalent of this number in megabytes is accounted for in the memory consumed by solvers utilizing BGS.

In none of the instances of the fail-repair problem considered, GMRES(20), BICGSTAB, and TFQMR benefit from BGS preconditioning. Although the iteration counts of the preconditioned projection methods decrease over those of the unpreconditioned ones, the decrease is not offset by the increase in time per iteration. The performances of the J, GS, and BGS solvers are insensitive to the rates of synchronized transitions. BGS performs very poorly

TABLE 4.5
Numerical results for the Fail-Repair problem with $K = 5$, $n_k = 19$ *for* $k = 1, 2, \ldots, K$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\lambda_3 = 0.6$, $\mu_3 = 0.5$, $\lambda_4 = 0.7$, $\mu_4 = 0.6$, $\lambda_5 = 0.8$, $\mu_5 = 0.7$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.005$.

Method	Iteration	Residual	Time
J	1,910	$9.9e-09$	1,733
GS	920	$9.4e-09$	1,255
BGS	244*	$1.7e-07$	10,060
GMRES(20)	800	$8.8e-09$	929
TFQMR	322	$9.2e-11$	289
BICGSTAB	143	$2.9e-09$	127
BGS_GMRES(20)	60	$4.3e-09$	2,647
BGS_TFQMR	116	$8.8e-10$	4,828
BGS_BICGSTAB	71	$2.7e-08$	3,002
ML_GS(W,C)	24	$9.5e-09$	99
D	10	$4.0e-09$	**25**

TABLE 4.6
Memory requirements of solvers in megabytes for the Fail-Repair problem with $K = 5$, $n_k = 19$ *for* $k = 1, 2, \ldots, K$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$.

Method	Memory
J	122
GS	122
BGS	717
GMRES(20)	610
BICGSTAB	244
TFQMR	293
BGS_GMRES(20)	1,327
BGS_TFQMR	961
BGS_BICGSTAB	1,010
ML_GS(W,C)	232
D	171

due to the large time per iteration and ML_GS(W,C) is the fastest solver when compared to J, GS, and BGS, and improves slightly as the synchronized transition rates become smaller. In Table 4.3, BICGSTAB is the fastest solver. However, when the rates of four synchronized transitions decrease to 0.05 in Table 4.4, the proposed solver becomes the fastest. In Table 4.5, the proposed solver exhibits the smallest number of iterations to converge and is also the fastest solver. The time per iteration taken by the proposed solver is larger than that of GS but less than twice that of GS. As the rates of the synchronized transitions decrease, we see that the number of iterations taken by the proposed solver to converge decreases similarly as in Table 4.2. The problem seems to become easier to solve for the proposed solver as the subsystems become nearly independent.

As it is shown in Table 4.6, BGS and BGS preconditioned projection methods require considerably more memory than the other methods, because of the need to store factors of diagonal blocks and, in the latter case, also a larger number of vectors. Memorywise, the proposed solver requires about 1.5 times that of J and GS, but less than ML_GS(W,C), and therefore can be considered to be memory efficient.

In Table 4.7, we investigate the scalability of the proposed solver for increasing number

TABLE 4.7
Performance of the decompositional iterative solver on the Fail-Repair problem for increasing number of subsystems K with $n_k = 19$ for $k = 1, 2, \ldots, K$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\lambda_3 = 0.6$, $\mu_3 = 0.5$, $\lambda_4 = 0.7$, $\mu_4 = 0.6$, $\lambda_5 = 0.8$, $\mu_5 = 0.7$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.005$.

K	Iteration	Residual	Time
2	70	$9.3e-09$	0
3	70	$7.1e-09$	0
4	30	$7.4e-09$	2
5	10	$4.0e-09$	25

TABLE 4.8
Performance of the decompositional iterative solver on the Fail-Repair problem for increasing number of synchronized transitions in $\mathcal{T}$ with $K = 5$, $n_k = 19$, for $k = 1, 2, \ldots, K$, $\lambda_1 = 0.4$, $\mu_1 = 0.3$, $\lambda_2 = 0.5$, $\mu_2 = 0.4$, $\lambda_3 = 0.6$, $\mu_3 = 0.5$, $\lambda_4 = 0.7$, $\mu_4 = 0.6$, $\lambda_5 = 0.8$, $\mu_4 = 0.7$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.005$.

$\mathcal{T}$	Iteration	Residual	Time
$\{t_1\}$	10	$3.3e-14$	15
$\{t_1, t_2\}$	10	$2.7e-14$	18
$\{t_1, t_2, t_3\}$	10	$2.0e-12$	21
$\{t_1, t_2, t_3, t_4\}$	10	$4.0e-09$	25

of subsystems when the four synchronized transition rates are relatively small compared to those in the local transition rate matrices. We see that the number of iterations to converge decreases as subsystems are added to the system at hand. This is due to the decrease in the throughputs of synchronized transitions for a larger number of subsystems (because the steady-state probabilities of global states in which synchronized transitions can be trigged become smaller), leading to more independent subsystems. This is different from the behavior of the multilevel method, which takes more or less the same number of iterations to converge as the number of subsystems increases.

In Table 4.8, we investigate the scalability of the proposed solver for increasing number of synchronized transitions when there are 5 subsystems and the rates of synchronized transitions are relatively small compared to those in the local transition rate matrices. As expected, the results indicate that the time the proposed solver takes to converge is affected linearly by an increase in the number of synchronized transitions.

In Table 4.9, we investigate the effects of using a larger number of synchronizations and smaller local failure rates, meaning that the redundant components in each subsystem are more reliable and therefore fail less often. The sixteen synchronized transitions are from global state (i, i, i, i) to $(i-4, i-4, i-4, i-4)$, for $i = 4, 5, \ldots, 19$. It seems that the asymmetry created among the nonzeros of the generator matrix due to the one order of magnitude difference between local repair and failure rates does not have a noticeable effect on the proposed solver (other than the fact that convergence takes place in one iteration but cannot be witnessed from the results due to the residual norm test every 10 iterations), but improves the situation with the J, GS, BGS, and ML_GS(W,C) solvers, and worsens the performance of others.

4.3. Some randomly generated problems. We consider randomly generated test cases, which have sparse transition rate matrices with nonzero elements following either the standard uniform distribution (i.e., nonzero values are chosen uniformly from the interval (0,1)) or the folded unit normal distribution (i.e., nonzero values are absolute values of samples chosen normally with mean 0 and standard deviation 1) and user-specified degrees of sparsity.

TABLE 4.9
Numerical results for the Fail-Repair problem with $K = 5$, $n_k = 19$ *for* $k = 1, 2, \ldots, K$, $\lambda_1 = 0.04$, $\mu_1 = 0.3$, $\lambda_2 = 0.05$, $\mu_2 = 0.4$, $\lambda_3 = 0.06$, $\mu_3 = 0.5$, $\lambda_4 = 0.07$, $\mu_4 = 0.6$, $\lambda_5 = 0.08$, $\mu_5 = 0.7$, $\mathcal{T} = \{t_1, t_2, \ldots, t_{16}\}$, $r_{t_1} = r_{t_2} = \cdots = r_{t_{16}} = 0.005$.

Method	Iteration	Residual	Time
J	230	$7.8e-09$	265
GS	160	$2.1e-09$	499
BGS	100	$6.0e-09$	4,138
GMRES(20)	5,000*	$1.8e-05$	6,951
TFQMR	5,000*	$1.9e-05$	5,706
BICGSTAB	241	$1.7e-09$	271
BGS_GMRES(20)	240*	$6.2e-05$	10,560
BGS_TFQMR	152	$1.1e-09$	6,391
BGS_BICGSTAB	240*	$9.9e-06$	10,030
ML_GS(W,C)	12	$5.9e-09$	84
D	10	$1.2e-09$	**64**

To this end, we have written a Matlab script, which generates the APNN toolbox input files randomly for user-specified test cases composed of K components having $n_1, n_2, \ldots, n_K$ states, $|\mathcal{T}|$ synchronized transition rates, and sparse transition rate matrices with prescribed degrees of sparsity. We have run many experiments, but here we discuss the results of just a few that are indicative of the performance of the proposed solver. We remark that due to randomness, the sparsity patterns of local and synchronized transition rate matrices in the experiments are arbitrary. In the following, $E[\cdot]$ is the expectation operator, and $E[nz(Q_{t_0}^{(k)})]$ and $E[nz(Q_{t_e}^{(k)})]$ denote the average number of nonzero entries in local and synchronized transition rate matrices, respectively.

In the first set of experiments, we consider a randomly generated problem with six subsystems each having 10 states resulting in a system of 1,000,000 states and four synchronized transitions. The values of the synchronized transition rates $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4}$ are chosen from the set $\{0.1, 0.01, 0.001, 0.0001\}$; thus, we experimented with four versions of the problem. The local and synchronized transition rate matrices, respectively, have an average of 43.5 (excluding the diagonal) and 8.3 nonzero elements randomly generated using the standard uniform distribution. Hence, the local transition rate matrices are about 54% full and the synchronized transition rate matrices are about 8% full. In this set of experiments, the diagonal blocks associated with the BGS solver and the BGS preconditioner for projection methods at level 4 (meaning 10,000 diagonal blocks of order 100) are LU factorized using the COLAMD ordering as before. It was not possible to consider the block partitioning at level 3 due to memory limitations. The number of nonzero entries generated during the LU factorization of the 10,000 diagonal blocks of order 100 is 59,650,000.

In Table 4.10, we present the results of experiments with synchronized transition rate values of 0.001, and remark that the number of iterations and time to convergence for the other solvers do not change except for TFQMR, which takes 94 iterations and 69 seconds to converge for rate values of 0.1, and BICGSTAB, which takes 54 iterations and 40 seconds to converge for rate values of 0.1 and 0.01, and 59 iterations and 43 seconds to converge for rate values of 0.0001. However, the proposed solver takes 40, 20 iterations and 68, 35 seconds to converge for rate values of 0.1, 0.01, respectively.

In the second set of experiments, we consider the same randomly generated problem as in the first set of experiments, but with nonzero elements in the transition rate matrices following

TABLE 4.10
Numerical results for $K = 6$, $n_k = 10$ *for* $k = 1, 2, \ldots, K$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.001$, *standard uniform,* $E[nz(Q_{t_0}^{(k)})] = 53.5$, *and* $E[nz(Q_{t_e}^{(k)})] = 8.3$.

Method	Iteration	Residual	Time
J	220	$7.1e-09$	168
GS	110	$6.8e-09$	99
BGS	70	$8.8e-09$	4,474
GMRES(20)	60	$5.7e-09$	51
TFQMR	5,000*	$3.0e-08$	3,611
BICGSTAB	60	$1.1e-09$	44
BGS_GMRES(20)	19	$3.0e-09$	1,356
BGS_TFQMR	26	$1.5e-09$	1,741
BGS_BICGSTAB	18	$2.1e-08$	1,228
ML_GS(W,C)	12	$2.3e-09$	33
D	10	$1.7e-09$	**18**

TABLE 4.11
Numerical results for $K = 6$, $n_k = 10$ *for* $k = 1, 2, \ldots, K$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.0001$, *folded unit normal,* $E[nz(Q_{t_0}^{(k)})] = 53.8$, *and* $E[nz(Q_{t_j}^{(k)})] = 8.7$.

Method	Iteration	Residual	Time
J	450	$8.3e-09$	360
GS	230	$6.6e-09$	228
BGS	160	$5.5e-09$	9,762
GMRES(20)	140	$2.8e-09$	122
TFQMR	5,000*	$5.4e-05$	3,774
BICGSTAB	75	$7.8e-09$	58
BGS_GMRES(20)	24	$5.5e-09$	1,664
BGS_TFQMR	30	$6.4e-09$	1,848
BGS_BICGSTAB	25	$1.2e-09$	1,602
ML_GS(W,C)	12	$5.3e-09$	33
D	10	$3.3e-09$	**19**

the folded unit normal distribution. The local and synchronized transition rate matrices respectively have an average of 43.8 (excluding the diagonal) and 8.7 nonzero elements. Hence, the local transition rate matrices are about 54% full and the synchronized transition rate matrices are about 9% full. The number of nonzeros generated during the LU factorization of the 10,000 diagonal blocks of order 100 with COLAMD ordering is 59,190,000.

In Table 4.11, we present the results of experiments with synchronized transition rate values of 0.0001, and remark that the number of iterations and time to convergence for the other solvers either do not change or do change slightly as in the first set of experiments except for TFQMR, which took 116 iterations and 89 seconds to converge for rate values of 0.1. However, the proposed solver takes 60, 40, 20 iterations and 109, 73, 36 seconds to converge for rate values of 0.1, 0.01, 0.001, respectively.

In the first two sets of experiments in which standard uniformly and folded unit normally distributed nonzero elements are used in the transition rate matrices, we see that the proposed solver performs better as the synchronized transition rate values become smaller. Furthermore, in both sets of experiments there is a value of synchronized transition rates for which the proposed solver becomes the fastest solver.

TABLE 4.12
Numerical results for $K = 6$, $n_k = 10$ *for* $k = 1, 2, \ldots, K$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.001$, *standard uniform, 3 sets of 30 test matrices each.*

$E[nz(Q^{(k)}_{t_0})]$	$E[nz(Q^{(k)}_{t_j})]$	E[Iteration]	E[Residual]	E[Time]
53.2	4.4	14	$3.9e-10$	20
53.1	8.7	36	$4.0e-09$	63
53.1	12.6	103	$6.8e-09$	285

TABLE 4.13
Numerical results for $K = 6$, $n_k = 10$ *for* $k = 1, 2, \ldots, K$, $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, $r_{t_1} = r_{t_2} = r_{t_3} = r_{t_4} = 0.001$, *folded unit normal, 3 sets of 30 test matrices each.*

$E[nz(Q^{(k)}_{t_0})]$	$E[nz(Q^{(k)}_{t_j})]$	E[Iteration]	E[Residual]	E[Time]
53.1	4.4	57	$2.6e-09$	79
53.2	8.6	83	$5.8e-09$	143
53.0	12.6	104	$6.4e-09$	292

In the next two sets of experiments, we investigate the effect of changing the sparsity of the synchronized transition rate matrices on the proposed solver for the same problem considered in the first two sets of experiments. To increase our confidence in the results, the experiments are performed on 30 randomly generated matrices for each degree of sparsity, and average results are presented for those 30 matrices. Hence, 180 test matrices are considered in these sets of experiments.

The results in Tables 4.12 and 4.13 indicate that indeed the performance of the proposed solver is adversely affected by an increasing average number of nonzeros in the synchronized transition rate matrices. The situation with standard uniformly distributed nonzero elements is better compared to the situation with folded unit normally distributed nonzero elements when the synchronized transition rate matrices are relatively sparser. But, as the sparsity decreases, there seems to be a point beyond which the distribution does not make much difference. Perhaps, this can be explained by an effective weaking of interactions among subsystems, as sparsity of synchronized transition rate matrices increase for a constant synchronized transition rate.

5. Conclusion.

A decompositional iterative method for obtaining the steady-state solution of Kronecker structured Markov chains is presented using disaggregation and aggregation operators. Currently, the method is applicable to systems that do not have unreachable states, but have state spaces equal to the cross products of the state spaces of their subsystems. The interactions among subsystems are not assumed to be weak, but the method works particularly well when there are weak interactions among subsystems. Numerical experiments show that as the interactions among subsystems weaken, the subsystems become nearly independent and the method benefits considerably from this near independence. Future work should concentrate on extending the method to Kronecker structured Markov systems with state spaces smaller than the cross products of the state spaces of their subsystems.

Acknowledgement. This research is partially supported by TÜBİTAK and the Turkish Academy of Sciences grant TÜBA-GEBİP. We thank the anonymous referees for their constructive remarks and suggesting some of the references.

REFERENCES

[1] F. Bause, P. Buchholz, and P. Kemper, *A toolbox for functional and quantitative analysis of DEDS*, in Quantitative Evaluation of Computing and Communication Systems, Lect. Notes Comput. Sci. 1469, R. Puigjaner, N. N. Savino, and B. Serra, eds., Springer, Heidelberg, 1998, pp. 356–359.

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

[3] P. Buchholz, *A class of hierarchical queueing networks and their analysis*, Queueing Syst., 15 (1994), pp. 59–80.

[4] P. Buchholz, *An iterative bounding method for stochastic automata networks*, Performance Evaluation, 49 (2002), pp. 211–226.

[5] P. Buchholz, *Adaptive decomposition and approximation for the analysis of stochastic Petri nets*, Performance Evaluation, 56 (2004), pp. 23–52.

[6] P. Buchholz and T. Dayar, *Block SOR for Kronecker structured Markovian representations*, Linear Algebra Appl., 386 (2004), pp. 83–109.

[7] P. Buchholz and T. Dayar, *Comparison of multilevel methods for Kronecker structured Markovian representations*, Computing, 73 (2004), pp. 349–371.

[8] P. Buchholz and T. Dayar, *Block SOR preconditioned projection methods for Kronecker structured Markovian representations*, SIAM J. Sci. Comput., 26 (2005), pp. 1289–1313.

[9] P. Buchholz and T. Dayar, *On the convergence of a class of multilevel methods for large sparse Markov chains*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1025–1049.

[10] P. Buchholz and P. Kemper, *On generating a hierarchy for GSPN analysis*, Performance Evaluation, 26 (1998), pp. 5–14.

[11] J. Campos, S. Donatelli, and M. Silva, *Structured solution of asynchronously communicating stochastic models*, IEEE Trans. Software Engrg., 25 (1999), pp. 147–165.

[12] F. Chatelin and W. L. Miranker, *Acceleration by aggregation of successive approximations*, Linear Algebra Appl., 43 (1982), pp. 17–47.

[13] G. Ciardo and K. S. Trivedi, *A decomposition approach for stochastic reward net models*, Performance Evaluation, 18 (1993), pp. 37–59.

[14] T. A. Davis, J. R. Gilbert, S. Larimore, and E. Ng, *A column approximate minimum degree ordering algorithm*, ACM Trans. Math. Software, 30 (2004), pp. 353–376.

[15] T. A. Davis, J. R. Gilbert, S. Larimore, and E. Ng, *Algorithm 836: COLAMD, a column approximate minimum degree ordering algorithm*, ACM Trans. Math. Software, 30 (2004), pp. 377–380.

[16] T. Dayar, *Analyzing Markov chains based on Kronecker products*, MAM 2006: Markov Anniversary Meeting, A. N. Langville, W. J. Stewart, eds., Boson Books, Raleigh, 2006, pp. 279–300.

[17] 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.

[18] P. Fernandes, B. Plateau, and W. J. Stewart, *Efficient descriptor-vector multiplications in stochastic automata networks*, J. ACM, 45 (1998), pp. 381–414.

[19] P. Kemper, *Numerical analysis of superposed GSPNs*, IEEE Trans. Software Engrg., 22 (1996), pp. 615–628.

[20] U. Krieger, *Numerical solution of large finite Markov chains by algebraic multigrid techniques*, in Computations with Markov Chains, W. J. Stewart, ed., Kluwer Academic Publishers, Boston, 1995, pp. 403–424.

[21] J. Mandel and B. Sekerka, *A local convergence proof for the iterative aggregation method*, Linear Algebra Appl., 51 (1983), pp. 163–172.

[22] I. Marek and P. Mayer, *Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices*, Numer. Linear Algebra Appl., 5 (1998), pp. 253–274.

[23] I. Marek and I. Pultarova, *A note on local and global convergence analysis of iterative aggregation-disaggregation methods*, Linear Algebra Appl., 413 (2006), pp. 327–341.

[24] 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, 1985, pp. 147–154.

[25] B. Plateau and J.-M. Fourneau, *A methodology for solving Markov models of parallel systems*, J. Parallel Distrib. Comput., 12 (1991), pp. 370–387.

[26] J. Ruge and K. Stuben, *Algebraic Multigrid*, in Frontiers in Applied Mathematics: Multigrid Methods, S. McCormick, ed., SIAM, Philadelphia, 1987, pp. 473–485.

[27] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003.

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

[29] L. A. Tomek and K. S. Trivedi, *Fixed point iteration in availability modelling*, in Proceedings of the 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems, Tests, Diagnosis, Fault

Treatment, Informatik-Fachberichte 283, M. D. Cin and W. Hohl, eds., Springer, London, 1991, pp. 229–240.

[30] E. UYSAL AND T. DAYAR, *Iterative methods based on splittings for stochastic automata networks*, European J. Oper. Res., 110 (1998), pp. 166–186.

[31] C. F. VAN LOAN, *The ubiquitious Kronecker product*, J. Comput. Appl. Math., 123 (2000), pp. 85–100.