STOCHASTIC AUTOMATA NETWORKS AND NEAR COMPLETE DECOMPOSABILITY[*]

OLEG GUSAK[†], TUĞRUL DAYAR[†] AND JEAN-MICHEL FOURNEAU[‡]

Abstract. Stochastic automata networks (SANs) have been developed and used in the last fifteen years as a modeling formalism for large systems that can be decomposed into loosely connected components. In this work, we extend the near complete decomposability concept of Markov chains (MCs) to SANs so that the inherent difficulty associated with solving the underlying MC can be forecasted and solution techniques based on this concept can be investigated. A straightforward approach to finding a nearly completely decomposable (NCD) partitioning of the MC underlying a SAN requires the computation of the nonzero elements of its global generator. This is not feasible for very large systems even in sparse matrix representation due to memory and execution time constraints. We devise an efficient solution algorithm to this problem that is based on analyzing the NCD structure of each component of a given SAN. Numerical results show that the given algorithm performs much better than the straightforward approach.

Key words. Markov chains, stochastic automata networks, near complete decomposability, state classification

AMS(MOS) subject classification. 60J27, 60J10, 65F30, 65F10, 65F50

1. Introduction.

The Markovian description of a system requires one to specify all possible states that the system can occupy and transitions among them. Most of the time, the complexity of the underlying system implies a very large state space. Sometimes, it is even difficult to enumerate all possible states and generate the transitions in the system. Even if we are able to do so, the size of the system remains a major obstacle in performance analysis.

Recently, a modeling paradigm called Stochastic Automata Networks (SANs) [16, 18, 14, 17, 19, 21, 22, 10, 2, 7, 13, 24, 4] has received attention. SANs provide a methodology for modeling large systems with interacting components. The main idea is to decompose the system of interest into its components and to model each component separately. Once this is done, interactions and dependencies among components can be brought into the picture and the model finalized. With this decompositional approach, the global system ends up having as many states as the product of the number of states of the individual components. The benefit of the SAN approach is twofold. First, each component can be modeled much easier compared to the global system due to state space reduction. Second, space required to store the description of components is minimal compared to the case in which transitions from each global state are stored explicitly. However, all this happens at the expense of increased analysis time [14, 22, 2, 10, 7, 13, 24, 4].

An intimately related way of coping with the state space explosion problem is to consider hierarchical decompositions arising in queueing network and superposed stochastic Petri Net formalisms [5, 3, 6]. SANs which do not have dependencies among automata are, in fact, a special case of hierarchical Markovian models. Although somewhat distant from the problem domain compared to the SAN approach, there are recent results

[*] This work is supported by TÜBİTAK-CNRS grant.

[†] Department of Computer Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey (E-mail:{gusak,tugrul}@cs.bilkent.edu.tr, Tel:(+90)(312)290-1981, Fax:(+90)(312)266-4126).

[‡] Lab. PRiSM, Université de Versailles, 45 Avenue des États-Unis, 78035 Versailles Cedex, France (E-mail:jmf@prism.uvsq.fr, Tel:(+33)(1)39-25-40-77, Fax:(+33)(1)39-25-40-57).

showing that hierarchical representations lend themselves naturally to distributed steady state analysis (see [6, p. 79]).

An important issue in choosing an efficient iterative solver for SANs is the conditioning [15] associated with the underlying Markov chain (MC). Recent numerical experiments [12] show that two-level iterative solvers perform very well with nearly completely decomposable (NCD) partitionings [9] having balanced block sizes when the MC to be solved for its steady state vector is ill-conditioned. Block iterative methods based on classical splittings (Block Jacobi, Block Gauss-Seidel, Block SOR) for SANs are introduced in [24]. Results with iterative aggregation-disaggregation [23, 20, 11, 12] type solvers for SANs appear in [2]. However, two-level iterative solvers considered so far do not exploit NCD partitionings. It should be emphasized that iterative aggregation-disaggregation based on NCD partitionings has certain rate of convergence guarantees [20] that may be useful for very large MCs.

In this paper, we extend the concept of near complete decomposability to SANs so that the inherent difficulty associated with solving the underlying MC can be forecasted and solution techniques based on this concept can be investigated. In doing this, we utilize the graph theoretic ideas for SANs given in [14]. In the next section, we review basic concepts of the SAN formalism, give an example, and introduce NCD MCs. In section 3, we make some assumptions regarding the description of a continuous-time SAN model, show what can be done if the given model does not satisfy the assumptions, and introduce the state classification (SC) algorithm whose output is used by the NCD partitioning algorithm. In section 4, we present a three step algorithm that finds an NCD partitioning of the MC underlying a SAN based on a user specified decomposability parameter without computing the global generator matrix. In doing this, we proceed step by step introducing definitions, stating propositions, making remarks, and illustrating with small examples the ideas on which our algorithm is based. The section ends with a summary of the complexity analysis of the NCD partitioning algorithm. Numerical results with the algorithm on three applications are presented in section 5. We conclude in section 6. The appendix includes more detailed description of the SC algorithm discussed in subsection 3.2, the NCD partitioning algorithm discussed in section 4 and its complexity analysis.

2. Background.

In the next two subsections, we discuss basic concepts related to the SAN formalism as a modeling paradigm and introduce NCD MCs.

2.1. SAN overview.

In a SAN (see [21, Ch. 9]), each component of the global system is modeled by a stochastic automaton. When automata do not interact (i.e., when they are independent of each other), description of each automaton consists of local transitions only. In other words, local transitions are those that affect the state of one automaton. Local transitions can be constant (i.e., independent of the state of other automata) or they can be functional. In the latter case, the transition is a nonnegative real valued function that depends on the state of other automata. Interactions among components are captured by synchronizing transitions. Synchronization among automata happens when a state change in one automaton causes a state change in other automata. Similar to local transitions, synchronizing transitions can be constant or functional.

A continuous-time system that does not have synchronizing events and dependencies among its components can be modeled by a single stochastic automaton for each component i that is expressed by the local transition rate matrix $Q_l^{(i)}$. The underlying continuous-time MC (CTMC) corresponding to the global system can then be obtained by the tensor sum of the local transition rate matrices of the automata. We refer to the tensor representation associated with the CTMC as the descriptor of the SAN.

Each synchronizing event introduces two types of matrices to the SAN formalism. These matrices are called the synchronizing event matrix and the diagonal corrector matrix. For automaton i and synchronizing event j, we have the synchronizing event matrix $Q_{e_j}^{(i)}$ and the diagonal corrector matrix $\bar{Q}_{e_j}^{(i)}$ both of order equal to that of the local transition rate matrix $Q_l^{(i)}$. The automaton that triggers a synchronizing event is called the master, the others that get affected are called slaves. Matrices associated with synchronizing events are either transition rate matrices (corresponding to master automata) or transition probability matrices (corresponding to slave automata). When they are rate matrices, each diagonal element in the diagonal corrector matrix is the negated sum of the off-diagonal elements in the corresponding synchronizing event matrix. When they are transition probability matrices, each diagonal element of the corrector matrix is the sum of the corresponding row elements in the synchronizing event matrix.

Synchronizing events introduce additional tensor products to the descriptor thereby complicating the SAN formalism. Since a tensor sum may be written as a sum of tensor products [8], it is possible to express the descriptor as a sum of Ordinary Tensor Products (OTPs) in the absence of functional transitions. When functional transitions are present, the descriptor is formed of Generalized Tensor Products (GTPs, see [13], for instance).

Summarizing the above, we can express the descriptor of a SAN as

$$Q = Q_l + Q_e + \bar{Q}_e, \tag{1}$$

where

$$Q_l = \bigoplus_{i=1}^{N} Q_l^{(i)}, \quad Q_e = \sum_{j=1}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)}, \quad \bar{Q}_e = \sum_{j=1}^{E} \bigotimes_{i=1}^{N} \bar{Q}_{e_j}^{(i)},$$

$\otimes$ is the tensor product operator, $\oplus$ is the tensor sum operator, N is the number of local automata in the system, and E is the number of synchronizing events among automata. Assuming that automaton i has n_i states, the global system has $n = \prod_{i=1}^{N} n_i$ states. When there are functional transitions, tensor operations become generalized tensor operations.

Example 1. Consider a SAN [21, pp. 470–472] that is formed of two automata (N=2) having 2 and 3 states and two synchronizing events (E=2). Automaton 1 is given by

$$Q_l^{(1)} = \begin{pmatrix} -\lambda_1 & \lambda_1 \\ 0 & 0 \end{pmatrix},$$

$$Q_{e_1}^{(1)} = \begin{pmatrix} 0 & 0 \\ \lambda_2 & 0 \end{pmatrix}, \bar{Q}_{e_1}^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & -\lambda_2 \end{pmatrix}, Q_{e_2}^{(1)} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \bar{Q}_{e_2}^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and automaton 2 is given by

$$Q_l^{(2)} = \begin{pmatrix} -\mu_1 & \mu_1 & 0 \\ 0 & -\mu_2 & \mu_2 \\ 0 & 0 & 0 \end{pmatrix},$$

$$Q_{e_1}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \bar{Q}_{e_1}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, Q_{e_2}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu_3 & 0 & 0 \end{pmatrix}, \bar{Q}_{e_2}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\mu_3 \end{pmatrix}.$$

From equation (1), the descriptor of the SAN can be obtained as

$$\begin{aligned} Q &= Q_l + Q_e + \bar{Q}_e \\ &= \bigoplus_{i=1}^{2} Q_l^{(i)} + \sum_{j=1}^{2} \bigotimes_{i=1}^{2} Q_{e_j}^{(i)} + \sum_{j=1}^{2} \bigotimes_{i=1}^{2} \bar{Q}_{e_j}^{(i)} \\ &= Q_l^{(1)} \oplus Q_l^{(2)} + Q_{e_1}^{(1)} \otimes Q_{e_1}^{(2)} + Q_{e_2}^{(1)} \otimes Q_{e_2}^{(2)} + \bar{Q}_{e_1}^{(1)} \otimes \bar{Q}_{e_1}^{(2)} + \bar{Q}_{e_2}^{(1)} \otimes \bar{Q}_{e_2}^{(2)} \\ &= \begin{pmatrix} -(\lambda_1+\mu_1) & \mu_1 & 0 & \lambda_1 & 0 & 0 \\ 0 & -(\lambda_1+\mu_2) & \mu_2 & 0 & \lambda_1 & 0 \\ \mu_3 & 0 & -(\lambda_1+\mu_3) & 0 & 0 & \lambda_1 \\ \lambda_2 & 0 & 0 & -(\lambda_2+\mu_1) & \mu_1 & 0 \\ \lambda_2 & 0 & 0 & 0 & -(\lambda_2+\mu_2) & \mu_2 \\ \lambda_2+\mu_3 & 0 & 0 & 0 & 0 & -(\lambda_2+\mu_3) \end{pmatrix} \end{aligned}$$

2.2. Nearly completely decomposable MCs. NCD MCs [15] are irreducible stochastic matrices that can be symmetrically permuted [9] to the block form

$$P_{n\times n} = \begin{pmatrix} P_{11} & P_{12} & \dots & P_{1K} \\ P_{21} & P_{22} & \dots & P_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ P_{K1} & P_{K2} & \dots & P_{KK} \end{pmatrix}$$

in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks [21, p. 286]. Hence, it is possible to represent an NCD MC as

$$P = \mathrm{diag}(P_{11}, P_{22}, \dots, P_{KK}) + E,$$

where the diagonal blocks P_{ii} are square and possibly of different order. The quantity $\|E\|_\infty$ is referred to as the degree of coupling and is taken to be a measure of the decomposability of P. When the chain is NCD, it has eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods [11, p. 290]. Hence, NCD MCs are said to be ill-conditioned [15, p. 258]. We should remark that the definition of NCDness is given through a discrete-time Markov Chain (DTMC). The underlying CTMC of a SAN can be transformed through uniformization [21, p. 24] to a DTMC for the purpose of computing its steady state vector as in

(2) $$P = I + \frac{1}{\alpha}Q,$$

where $\alpha \geq \max_{1\leq i\leq n} |Q(i,i)|$. To preserve NCDness in this transformation, α must be chosen as $\max_{1\leq i\leq n} |Q(i,i)|$.

An NCD partitioning of P corresponding to a user specified decomposability parameter ϵ can be computed as follows (see [9] for details). First, construct an undirected graph whose vertices are the states of P by introducing an edge between vertices i and j if $P(i,j) \geq \epsilon$ or $P(j,i) \geq \epsilon$, and then identify its connected components[1] (CCs). Each CC forms a subset of the NCD partitioning. Notice that for a given value of ϵ, the maximum number of subsets in a computed partitioning is unique.

There is no standard specification for the description of a SAN model. In the next section, we state definitions and propositions that enable us to transform a continuous-time SAN description to one that is more convenient to work with when developing the NCD partitioning algorithm. Then we introduce the SC algorithm for SANs.

3. On continuous-time SAN descriptions and state classification.

First, we briefly recall the parameters of our problem.

- A SAN consists of N automata, E synchronizing events, and possibly functional transitions.
- Automaton i is denoted by $\mathcal{A}^{(i)}$, has n_i states, and is described by:
  - local transition rate matrix $Q_l^{(i)}$;
  - $2E$ synchronizing event matrices;
    transition matrix of synchronizing event j is denoted by $Q_{e_j}^{(i)}$;
    $Q_{e_j}^{(i)}$ has the corresponding diagonal corrector matrix $\bar{Q}_{e_j}^{(i)}$;
  - (being master/slave in synchronizing event j)
    if $\mathcal{A}^{(i)}$ is the master in synchronizing event j, then $Q_{e_j}^{(i)}$ is a rate matrix;
    if $\mathcal{A}^{(i)}$ is a slave in synchronizing event j, then $Q_{e_j}^{(i)}$ is a probability matrix;
  - (not being involved in synchronizing event j)
    $Q_{e_j}^{(i)} = I_{n_i}$, where I_{n_i} is the identity matrix of order n_i.
- The descriptor of the SAN is formed of generalized tensor products and generalized tensor sums; the underlying CTMC has $n = \prod_{i=1}^{N} n_i$ global states.
- States of automata and global states are numbered starting from 1.
- The state vector of the SAN model is represented by $(s_1, s_2, \ldots, s_N)$, where $s_i \in \{1, 2, \ldots, n_i\}$ denotes the state of $\mathcal{A}^{(i)}$. The corresponding global state is given by $s = 1 + \sum_{i=1}^{N}(s_i - 1)\prod_{k=i+1}^{N} n_k$.

3.1. Description of a continuous-time SAN model.

Without loss of generality, we restrict ourselves to the case in which row sums of synchronizing transition probability matrices are either 0 or 1.

Definition 1. *A SAN description is said to be proper if and only if each synchronizing transition probability matrix has row sums of 0 or 1.*

The SAN descriptions of the three applications we consider in section 5 are proper. In other cases, one may use the following:

Proposition 1. *A given SAN description can be transformed to a SAN description that is proper.*

[1] Not to be mixed with the word component we have been using so far to mean subsystem.

Proof. Without loss of generality, consider a SAN description of N automata and one synchronizing event. There are two possible cases. In the first case, row sums of the synchronizing transition probability matrix $Q^{(k)}_{e_1}$ corresponding to slave automaton k are all equal to some constant β such that $0 < \beta < 1$. This is the trivial case; we can replace $Q^{(k)}_{e_1}$ with $\hat{Q}^{(k)}_{e_1} = \frac{1}{\beta}Q^{(k)}_{e_1}$, and $Q^{(m)}_{e_1}$ with $\hat{Q}^{(m)}_{e_1} = \beta Q^{(m)}_{e_1}$, where m is the index of the master automaton of the synchronizing event. Row sums of the transformed matrix $\hat{Q}^{(k)}_{e_1}$ are 1. In the second case, row sums of $Q^{(k)}_{e_1}$ are not equal, and some are between 0 and 1. This implies that transition rates of the master automaton m of the synchronizing event depend on the state of automaton k. Therefore, it is possible to replace $Q^{(k)}_{e_1}$ with a matrix that has row sums of 0 or 1 by introducing functional transitions to $Q^{(m)}_{e_1}$ as follows. Let β_l, $l = 1, 2, \ldots, n_k$, be the sum of row l in $Q^{(k)}_{e_1}$. We replace $Q^{(k)}_{e_1}$ with $\hat{Q}^{(k)}_{e_1}$ in which $\hat{Q}^{(k)}_{e_1}(i,j) = Q^{(k)}_{e_1}(i,j)/\beta_i$ if $0 < \beta_i < 1$, else $\hat{Q}^{(k)}_{e_1}(i,j) = Q^{(k)}_{e_1}(i,j)$, for $j = 1, 2, \ldots, n_k$. We also replace $Q^{(m)}_{e_1}$ with $\hat{Q}^{(m)}_{e_1}$ in which $\hat{Q}^{(m)}_{e_1}(i,j) = \beta_l Q^{(m)}_{e_1}(i,j)$ if $0 < \beta_l < 1$, else $\hat{Q}^{(m)}_{e_1}(i,j) = Q^{(m)}_{e_1}(i,j)$, for $i,j = 1, 2, \ldots, n_m$ when $\mathcal{A}^{(k)}$ is in state l. The transformed matrix $\hat{Q}^{(k)}_{e_1}$ has row sums of 0 or 1.

Given a synchronizing event, the above modifications must be made for each of its synchronizing transition probability matrices that has row sums between 0 and 1. This implies that nonzero elements in the synchronizing event transition rate matrix of the corresponding master automaton may need to be scaled multiple times with values that depend on the state of multiple slave automata. The generalization to E (> 1) synchronizing events is straightforward. After modifying the synchronizing event matrices, the corresponding diagonal corrector matrices must also be modified accordingly. The new SAN description has synchronizing transition probability matrices with row sums of 0 or 1, and therefore is proper. □

Now we introduce a definition which may seem to be specifying an artificial condition at first, yet the condition is satisfied by the three applications in section 5 and is very helpful in coming up with an efficient algorithm as we shall see.

Definition 2. *Synchronizations are separable from local transitions in a given SAN description if and only if for any synchronizing event t whose master is automaton m and $i,j = 1, 2, \ldots, n_m$, $Q^{(m)}_{e_t}(i,j) \neq 0$ implies $Q^{(m)}_{l}(i,j) = 0$.*

Proposition 2. *A given SAN description can be transformed to a new SAN description whose synchronizations are separable from local transitions.*

Proof. Assume that the given SAN description does not satisfy the condition in Definition 2. Without loss of generality, let t be the event, m its master, and (i,j) the indices of the problematic element. Decompose $Q^{(m)}_l$ into three terms as

$$Q^{(m)}_l = R^{(m)}_l + Q^{(m)}_l(i,j)u_i u_j^T - Q^{(m)}_l(i,j)u_i u_i^T,$$

where u_i is the ith column of the identity matrix. Here $R^{(m)}_l$ is a transition rate matrix, the second term is a matrix with a single nonzero transition rate at element (i,j) and the third term is the diagonal corrector of the second term. Now, let $R^{(m)}_l$ be the local transition rate matrix of automaton m, and introduce the new synchronizing event v with master automaton m; $Q^{(m)}_{e_v} (= Q^{(m)}_l(i,j)u_i u_j^T)$ is the rate matrix associated with automaton m and synchronizing event v, and $\bar{Q}^{(m)}_{e_v} (= -Q^{(m)}_l(i,j)u_i u_i^T)$ is its diagonal corrector. All other matrices corresponding to synchronizing event v are equal to identity. Now, recall the following identity from tensor algebra

$$A \bigoplus (Q_l^{(m)} + Q_{e_v}^{(m)} + \bar{Q}_{e_v}^{(m)}) \bigoplus B = (A \bigoplus Q_l^{(m)} \bigoplus B) + (I \bigotimes Q_{e_v}^{(m)} \bigotimes I) + (I \bigotimes \bar{Q}_{e_v}^{(m)} \bigotimes I)$$

and compare its right-hand side with equation (1). The new SAN description has separable synchronizations. $\square$

Our next definition related to the SAN description involves the number of nonzero elements in synchronizing transition rate matrices. Without loss of generality, we restrict ourselves to the case where all synchronizing events in a SAN are simple.

DEFINITION 3. *Synchronizations in a given SAN description are simple if and only if for any synchronizing event t whose master is automaton m, $Q_{e_t}^{(m)}$ has only one nonzero element.*

PROPOSITION 3. *A given SAN description can be transformed to a new SAN description whose synchronizing events are simple.*

Proof. Assume that the given SAN description does not satisfy the condition in Definition 3. Without loss of generality, let t be the event, m its master, and nz the number of nonzeros in $Q_{e_t}^{(m)}$. Decompose $Q_{e_t}^{(m)}$ into nz simple synchronizing transition rate matrices thereby creating nz new synchronizing events with master automaton m. The slave automata of the new synchronizing events are the slave automata of synchronizing event t. The transition probability matrices and their diagonal correctors associated with the new slave automata are respectively equal to the transition probability matrix and its diagonal corrector associated with the slave automata for synchronizing event t. All other matrices corresponding to the new synchronizing events are equal to identity. The new SAN description has simple synchronizations. $\square$

In the next subsection, we discuss how we proceed when we encounter an underlying MC with transient states and/or multiple essential subsets of states.

3.2. State classification in SANs. As discussed in subsection 2.2, NCD MCs are irreducible by definition. However, the MC underlying a SAN may very well be reducible. We have implemented a state classification (SC) algorithm that classifies the states in the global state space of a SAN into essential and transient subsets following [21, pp. 25–26]. In doing this, we use a strongly connected component (SCC) search algorithm on a directed graph (digraph) with edges whose presence can be taken into consideration on the fly. The SC algorithm is based on an algorithm that finds SCCs of a digraph using depth first search (DFS). The main idea of DFS is to explore all vertices of the digraph in a recursive manner. Whenever an unvisited vertex is encountered, the algorithm starts exploring its adjacent vertices. A detailed description of the SCC algorithm for digraphs can be found in [1, pp. 191–197].

Any transient states identified by the SC algorithm are omitted from further consideration when running the NCD partitioning algorithm. Furthermore, when the MC underlying the given SAN has multiple essential subsets of states, NCD analysis can be carried out on the essential subsets of states one subset at a time. The states that do not belong to the chosen essential subset should be omitted from further consideration when running the NCD partitioning algorithm. The detailed description of the SC algorithm is given in appendix A. The input parameters of the SC algorithm are local transition rate matrices and synchronizing event matrices of the SAN. The output of the algorithm is an integer array of length n in which states corresponding to the essential subset of interest are marked.

4. NCD partitioning algorithm for SANs.

The following is our proposed solution algorithm that computes NCD partitionings of the MC underlying a SAN without generating Q (or P).

Algorithm 1

NCD partitioning of MC underlying SAN for given ϵ

```
Step 1. $Q \rightarrow P$ transformation
Step 2. Preprocessing synchronizing events
Step 3. Constructing NCD connected components
```

Step 1 computes the scalar α in equation (2) that describes the transformation of the global generator Q to a DTMC P through uniformization. In the next subsection, we show how this can be achieved efficiently by inspecting the diagonal elements in local transition rate matrices and the nonzero elements in diagonal corrector matrices.

Step 2 considers the locations of off-diagonal nonzero elements in the global generator Q. Off-diagonal nonzero elements in local transition rate matrices cannot contribute to the same nonzero element in Q due to the fact that these matrices form a tensor sum. Hence, their analysis is straightforward. However, off-diagonal nonzero elements in synchronizing transition rate matrices may contribute to the same nonzero element in Q since these matrices form a sum of tensor products. Therefore, it is necessary to identify those synchronizing events that may influence the NCD partitioning of the MC underlying the SAN by contributing to the value of the same nonzero element in Q. In subsection 4.2, we explain how this is done.

Finally, Step 3 determines the NCD CCs by analyzing local transition rate matrices and matrices corresponding to synchronizing events identified in Step 2 using ϵ and the value of α computed in Step 1. This is discussed in subsection 4.3.

4.1. $Q \rightarrow P$ transformation.

The CTMC Q can be transformed to a DTMC P using equation (2) after $\alpha = \max_{1 \leq i \leq n} |Q(i,i)|$ is computed. It is Algorithm 1.1 in appendix B that computes α using local transition rate matrices, diagonal corrector matrices, and dependencies among automata.

Since Q is a CTMC, we have $Q(i,i) = -\sum_{j \neq i} Q(i,j)$ for $i = 1, 2, \ldots, n$. Note also that only the off-diagonal elements in P contribute to NCDness. Regarding the off-diagonal elements in Q, which determine the off-diagonal elements in P, we make the following observations.

Remark 1. *Each nonzero local transition rate in a SAN contributes to a different off-diagonal element in Q; two or more nonzero local transition rates cannot contribute to the same off-diagonal element in Q.*

This observation follows immediately from the term Q_l in equation (1) and the definition of tensor sum.

Remark 2. *A nonzero off-diagonal element in Q for a SAN with separable synchro-*

nizations is formed either of a nonzero local transition rate or of nonzero synchronizing transition rates but not of both.

This observation follows from the definition of the SAN descriptor in equation (1) and Definition 2.

REMARK 3. *A nonzero off-diagonal element that is formed of synchronizing transition rates in Q for a SAN can be represented as*

$$\sum_{j, e_j \in \mathcal{E}^*} q_{e_j}^{(m)} \prod_{k=1, k \neq m}^{N} q_{e_j}^{(k)}. \tag{3}$$

Here $q_{e_j}^{(m)}$ is the synchronizing transition rate in $Q_{e_j}^{(m)}$, where m is the index of the master automaton of event j; $q_{e_j}^{(k)}$ is a particular transition probability in $Q_{e_j}^{(k)}$, where k ($\neq m$) is the index of a slave automaton of event j. Finally, $\mathcal{E}^*$ is the set of synchronizing events that contribute to the off-diagonal element of interest in Q, and $|\mathcal{E}^*| \leq E$. We have omitted the row and column indices from $q_{e_j}^{(m)}$ and $q_{e_j}^{(k)}$, since only the form of equation (3) is important for the current discussion.

From Remarks 1–3 and equations (1)–(3), P without its main diagonal, denoted by P^*, can be computed as

$$P^* = \bigoplus_{i=1}^{N} \left(\frac{1}{\alpha} Q_l^{(i)} \right) + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} \hat{Q}_{e_j}^{(i)}, \tag{4}$$

where

$$\hat{Q}_{e_j}^{(i)} = \begin{cases} \frac{1}{\alpha} Q_{e_j}^{(i)} & \text{if } \mathcal{A}^{(i)} \text{ is the master of event } j \\ Q_{e_j}^{(i)} & \text{otherwise} \end{cases}.$$

Note that only rate matrices are scaled. What remains to be done is to compute α. To that effect, we state two propositions.

PROPOSITION 4. *The diagonal element with maximum magnitude of Q for a SAN that does not have synchronizations and functional dependencies is given by the sum of diagonal elements with maximum magnitude of the local transition rate matrices. Thus,*

$$\alpha = \sum_{i=1}^{N} \max_{1 \leq k \leq n_i} |Q_l^{(i)}(k,k)|.$$

Proof. Recall Remark 2. Also note that the rates of transitions out of state k of automaton i all appear in the same row of Q, and consequently they contribute the value $Q_l^{(i)}(k,k)$ to the diagonal element of Q in that row. Since the state space of the global system is all possible combinations of the automata states, there is a global state in Q for which the off-diagonal row sums of all automata are maximized. □

REMARK 4. *Dependencies among automata may arise either as explicit functions whose values depend on the states of automata other than the ones in which they are defined or implicitly by the existence of zero rows in synchronizing event matrices associated with slave automata. The latter case corresponds to the disabling of the synchronized transition when the slave automaton is in local state corresponding to the zero row.*

From now on, by dependencies we refer to both explicit and implicit dependencies as discussed in Remark 4. As an extension to Proposition 4, we have the next one.

PROPOSITION 5. *The diagonal element with maximum magnitude of Q for a SAN that does not have dependencies can be obtained from*

$$\alpha = \sum_{i=1}^{N} \max_{1 \le k \le n_i} \left| \left(Q_l^{(i)}(k,k) + \sum_{j, e_j \in \mathcal{M}_i} \bar{Q}_{e_j}^{(i)}(k,k) \right) \right|, \tag{5}$$

where $\mathcal{M}_i$ is the set of synchronizing events whose master is automaton i.

Proof. Observe that local and synchronizing transitions of a master automaton that emanate from the same local state appear in the same row of Q. The rest of the proof follows from an argument similar to that for Proposition 4. Note that equation (5) is also valid for the case in which synchronizing transition probabilities are less than 1. In this case, the synchronizing transition rate in the master automaton is split according to transition probabilities in slave automata. However, these fractional synchronizing transition rates still appear in the same row of Q; that is, they contribute to a diagonal element of Q the corresponding diagonal element of the diagonal corrector matrix. □

Example 2. Now let us show the computation of the diagonal element with maximum magnitude of Q for the SAN in Example 1 with $(\lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3) = (2, 3, 3, 2, 1)$. According to Proposition 5, we have

$$\begin{aligned} \alpha &= \max_{1 \le k \le 2} |Q_l^{(1)}(k,k) + \bar{Q}_{e_1}^{(1)}(k,k)| + \max_{1 \le k \le 3} |Q_l^{(2)}(k,k) + \bar{Q}_{e_2}^{(2)}(k,k)| \\ &= \max\{\lambda_1, \lambda_2\} + \max\{\mu_1, \mu_2, \mu_3\} = \lambda_2 + \mu_1 = 6, \end{aligned}$$

which can be verified on

$$Q = \begin{pmatrix} -5 & 3 & 0 & 2 & 0 & 0 \\ 0 & -4 & 2 & 0 & 2 & 0 \\ 1 & 0 & -3 & 0 & 0 & 2 \\ 3 & 0 & 0 & -6 & 3 & 0 \\ 3 & 0 & 0 & 0 & -5 & 2 \\ 4 & 0 & 0 & 0 & 0 & -4 \end{pmatrix}.$$

Example 3. Consider the SAN presented in Example 2 with the following modification: $Q_{e_1}^{(2)}(1,1) = 0$ and $\bar{Q}_{e_1}^{(2)}(1,2) = 0$. Note that the modified SAN still does not have functional transitions defined explicitly. However, the rate of synchronizing event 1 is in fact a function, and it depends on the state of automaton 2 (see Remark 4). When automaton 2 is in state 1, the synchronizing transition rate is 0 since it is disabled due to the modification, else it is 3. It is not possible to apply Proposition 5 to this SAN, since it would produce the incorrect result $\alpha = 6$ which can be seen from

$$Q = \begin{pmatrix} -5 & 3 & 0 & 2 & 0 & 0 \\ 0 & -4 & 2 & 0 & 2 & 0 \\ 1 & 0 & -3 & 0 & 0 & 2 \\ 0 & 0 & 0 & -3 & 3 & 0 \\ 3 & 0 & 0 & 0 & -5 & 2 \\ 4 & 0 & 0 & 0 & 0 & -4 \end{pmatrix}.$$

For SANs having dependencies, equation (5) cannot be used. A naive solution is to compute explicitly each diagonal element of Q and to find the element with maximum magnitude. However, this is expensive. To reduce the complexity, we propose to partition automata into dependency sets.

Definition 4. *Let $G(\mathcal{V},\mathcal{E})$ be a digraph in which v_i corresponds to $\mathcal{A}^{(i)}$ and $(v_i, v_j) \in \mathcal{E}$ if transitions in $\mathcal{A}^{(i)}$ depend on the state of $\mathcal{A}^{(j)}$ either explicitly or implicitly as discussed in Remark 4. Then, the dependency sets of a SAN, denoted by $\mathcal{D}_k$, $k = 1, 2, \dots, N_{\mathcal{D}}$, are the connected components of the dependency graph G.*

We assume that for each automaton of the SAN, the set of automata with which it is involved in a functional dependency relationship is known. Regarding implicit dependencies that originate from synchronizations (see Remark 4), we make the following observation. If the diagonal corrector matrix $\bar{Q}_{e_j}^{(i)}$ of slave automaton i has at least one row with a zero diagonal element, then the master automaton of event j is dependent on the state of automaton i. Note that we do not need to scan each synchronizing event diagonal corrector matrix to detect such dependencies. Each row of a diagonal corrector matrix that is a probability transition matrix can have at most one nonzero element per row. Hence, if the number of nonzeros in $\bar{Q}_{e_j}^{(i)}$ is less than the order of the matrix (i.e., n_i), then the master automaton of event j depends on automaton i.

We refer to

$$\max\_\mathcal{D}_k = \max \left| \bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} \mathrm{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} \mathrm{diag}(\bar{Q}_{e_j}^{(i)}) \right| \tag{6}$$

as the maximum of the dependency set $\mathcal{D}_k$, where diag returns a vector consisting of the diagonal elements of its matrix argument, and $\mathcal{M}_{\mathcal{D}_k}$ is the set of synchronizing events whose masters are in $\mathcal{D}_k$.

Proposition 6. *The diagonal element with maximum magnitude of Q for a SAN can be obtained from*

$$\alpha = \sum_{k=1}^{N_{\mathcal{D}}} \max\_\mathcal{D}_k. \tag{7}$$

Proof. From the definition of dependency sets, automata in a dependency set are independent of automata in other dependency sets. Now, consider a new SAN description of $N_{\mathcal{D}}$ automata and E synchronizing events. In the new description, $\mathcal{A}^{(\mathcal{D}_k)}$, $k = 1, 2, \dots, N_{\mathcal{D}}$, corresponds to the dependency set $\mathcal{D}_k$, $Q_l^{(\mathcal{D}_k)} = \bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} Q_l^{(i)}$, $Q_{e_j}^{(\mathcal{D}_k)} = \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} Q_{e_j}^{(i)}$, and $\bar{Q}_{e_j}^{(\mathcal{D}_k)} = \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} \bar{Q}_{e_j}^{(i)}$, where $j = 1, 2, \dots, E$. We define $\mathcal{A}^{(\mathcal{D}_k)}$ as the master of synchronizing event j, if the master automaton of event j in the original SAN description is a member of $\mathcal{D}_k$. By construction, the new SAN does not have dependencies. Hence, we can apply Proposition 5. Substituting in equation (5) the matrices $Q_l^{(\mathcal{D}_k)}$ and $\bar{Q}_{e_j}^{(\mathcal{D}_k)}$, we obtain equation (7). □

Propositions 4, 5, and 6 are valid for irreducible MCs underlying SANs. When transient states and/or multiple essential subsets of states are present, the diagonal element with maximum magnitude given by equation (7) may not belong to the essential subset of interest (see subsection 3.2). In the following, we refer to the states other than the ones in the essential subset of interest as uninteresting. In the presence of uninteresting states, we can compute α by finding the maximums of all $N_{\mathcal{D}}$ dependency sets (see equations (6) and (7)). For dependency set $\mathcal{D}_k$, this task amounts to the enumeration of $\prod_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} n_i$ states and an equal number of floating-point comparisons. Now, observe that to max_$\mathcal{D}_k$ of the dependency set $\mathcal{D}_k$ corresponds a state S_k. Hence, if the global state s that corresponds to $S_1, S_2, \ldots, S_{N_{\mathcal{D}}}$ maps into the essential subset of interest, then α given by equation (7) is taken as the diagonal element with maximum magnitude. However, if s is an uninteresting state, we omit from further consideration the element that correspond to max_$\mathcal{D}_k$ for $k = 1, 2, \ldots, N_{\mathcal{D}}$, and proceed as in the while-loop of Algorithm 1.1 in appendix B.

In the first step, for $k = 1, 2, \ldots, N_{\mathcal{D}}$ we find the next largest value denoted by next_max_$\mathcal{D}_k$ from equation (6) and the corresponding state $\tilde{S}_k$. In order to find next_max_$\mathcal{D}_k$ rapidly, the vectors $\left|\bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} \mathrm{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} \mathrm{diag}(\bar{Q}_{e_j}^{(i)})\right|$, $k = 1, 2, \ldots, N_{\mathcal{D}}$, should be stored as sorted. In the second step, we find t such that next_max_$\mathcal{D}_t \geq$ next_max_$\mathcal{D}_k$ for $k = 1, 2, \ldots, N_{\mathcal{D}}$. Finally, we replace max_$\mathcal{D}_t$ with next_max_$\mathcal{D}_t$, S_t with $\tilde{S}_t$, and omit the element corresponding to next_max_$\mathcal{D}_t$ from further consideration. If the updated global state s maps to a state in the essential subset of interest, then α given by equation (7) is taken as the diagonal element with maximum magnitude. Else we go back to the first step. Since finite MCs have at least one recurrent state in each essential subset, the algorithm is terminating.

Our final remark is about the special case of a SAN with a single dependency set; that is, $N_{\mathcal{D}} = 1$ and $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(N)}\}$. In this case, finding $\alpha =$ max_$\mathcal{D}_1$ amounts to enumerating all diagonal elements of Q since we have the equality $\bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_1} \mathrm{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_1} \mathrm{diag}(\bar{Q}_{e_j}^{(i)}) = \mathrm{diag}(Q)$. Therefore, for a SAN with a single dependency set, there is no need to sort and store diag(Q) as suggested. When finding the maximum of diag(Q), we test an element of diag(Q) only if its index corresponds to a state in the essential subset of interest. Although it is implemented, this case is omitted from the presentation of Algorithm 1.1 in appendix B.

Example 4. This example shows the computation of the diagonal element with maximum magnitude of Q for the following SAN that has functional and synchronizing transitions. The parameters are $N = 3$, $E = 2$, $n_1 = 2$, $n_2 = 3$, $n_3 = 2$; $f = 3$ when $\mathcal{A}^{(1)}$ is in state 1, and $f = 5$ when $\mathcal{A}^{(1)}$ is in state 2. The master of synchronizing event 1 is $\mathcal{A}^{(3)}$, and the master of synchronizing event 2 is $\mathcal{A}^{(2)}$. The matrices are

$$Q_l^{(1)} = \begin{pmatrix} -2 & 2 \\ 1 & -1 \end{pmatrix}, Q_l^{(2)} = \begin{pmatrix} -2 & 2 & 0 \\ 2 & -5 & 3 \\ 1 & 3 & -4 \end{pmatrix}, Q_l^{(3)} = \begin{pmatrix} -f & f \\ 0 & 0 \end{pmatrix},$$

$$Q_{e_1}^{(1)} = \bar{Q}_{e_1}^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, Q_{e_2}^{(1)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \bar{Q}_{e_2}^{(1)} = I,$$

$$Q_{e_1}^{(2)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \bar{Q}_{e_1}^{(2)} = I, Q_{e_2}^{(2)} = \begin{pmatrix} 0 & 0 & 5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \bar{Q}_{e_2}^{(2)} = \begin{pmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$Q_{e1}^{(3)} = \begin{pmatrix} 0 & 0 \\ 5 & 0 \end{pmatrix}, \bar{Q}_{e1}^{(3)} = \begin{pmatrix} 0 & 0 \\ 0 & -5 \end{pmatrix}, Q_{e2}^{(3)} = \bar{Q}_{e2}^{(3)} = I.$$

The given SAN has two dependency sets: $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(3)}\}$ and $\mathcal{D}_2 = \{\mathcal{A}^{(2)}\}$. Note that $\mathcal{A}^{(3)}$ functionally depends on the state of $\mathcal{A}^{(1)}$ due to functional transition f as well as due to synchronizing event 1 (see $\bar{Q}_{e_1}^{(1)}$). Hence, the diagonal element with maximum magnitude of Q is comprised of two terms. The maximum of $\mathcal{D}_1$ is given by

$$\begin{aligned}
\max\_\mathcal{D}_1 &= \max \left| \operatorname{diag}(Q_l^{(1)}) \bigoplus \operatorname{diag}(Q_l^{(3)}) + \operatorname{diag}(\bar{Q}_{e1}^{(1)}) \bigotimes \operatorname{diag}(\bar{Q}_{e1}^{(3)}) \right| \\
&= \max \left| \begin{pmatrix} -2-f \\ -2-0 \\ -1-f \\ -1-3 \end{pmatrix} + \begin{pmatrix} 0 \\ -5 \\ 0 \\ 0 \end{pmatrix} \right| = \max \left| \begin{pmatrix} -5 \\ -2 \\ -6 \\ -4 \end{pmatrix} + \begin{pmatrix} 0 \\ -5 \\ 0 \\ 0 \end{pmatrix} \right| = 7.
\end{aligned}$$

On the other hand, $\mathcal{D}_2$ is a singleton, and therefore the maximum of $\mathcal{D}_2$ is given by

$$\max\_\mathcal{D}_2 = \max \left| \operatorname{diag}(Q_l^{(2)}) + \operatorname{diag}(\bar{Q}_{e_2}^{(2)}) \right| = \max \left| \begin{pmatrix} -2 \\ -5 \\ -4 \end{pmatrix} + \begin{pmatrix} -5 \\ 0 \\ 0 \end{pmatrix} \right| = 7.$$

Since the underlying MC is irreducible, $\alpha = \max\_\mathcal{D}_1 + \max\_\mathcal{D}_2 = 14$ as verified on

$$Q = \left(\begin{array}{cccccc|cccccc}
-12 & 3 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 5 & 0 \\
0 & -14 & 0 & 2 & 5 & 0 & 0 & 2 & 0 & 0 & 0 & 5 \\
2 & 0 & -10 & 3 & 3 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
5 & 2 & 0 & -12 & 0 & 3 & 0 & 0 & 0 & 2 & 0 & 0 \\
1 & 0 & 3 & 0 & -9 & 3 & 0 & 0 & 0 & 0 & 2 & 0 \\
5 & 1 & 0 & 3 & 0 & -11 & 0 & 0 & 0 & 0 & 0 & 2 \\
1 & 0 & 0 & 0 & 5 & 0 & -13 & 5 & 2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 5 & 0 & -8 & 0 & 2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & -11 & 5 & 3 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & -6 & 0 & 3 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 3 & 0 & -10 & 5 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 3 & 0 & -5
\end{array} \right).$$

As pointed out at the beginning of this subsection, an NCD partitioning of P that corresponds to a user specified decomposability parameter ϵ is determined by the off-diagonal elements in P. Having found α, we can obtain P^* by scaling each transition rate matrix of the SAN with $1/\alpha$ (see equation (4)).

4.2. Preprocessing synchronizing events. As it is mentioned in Remark 3, transition rates from different synchronizing event matrices may sum up to form a nonzero in the generator matrix Q. Hence, in some cases it may not be possible to determine the value of an off-diagonal element in Q by inspecting each automaton separately. The aim of Step 2 in Algorithm 1 is to find sets of those synchronizing events that may influence the NCD partitioning of Q. We name these sets as potential sets of synchronizing events. The potential sets are disjoint, and their union is a subset of the set of synchronizing events. It is Algorithm 1.2 in appendix C that finds these potential sets using synchronizing event matrices, ϵ, and α computed in Step 1. The output of Algorithm 1.2 is $N_{\mathcal{P}}$ potential sets denoted by $\mathcal{P}_r$, $r = 1, 2, \ldots, N_{\mathcal{P}}$.

There are two cases in which synchronizing events may influence the NCD partitioning of Q. First, a simple synchronizing event has the corresponding transition rate greater than or equal to $\alpha\epsilon$. Second, a set of synchronizing events contribute to the same element in Q, and the sum of the synchronizing transition rates of the events in the set is greater than or equal to $\alpha\epsilon$.

In the first case, each synchronizing event with transition rate greater than or equal to $\alpha\epsilon$ forms a potential set that is a singleton. When the transition rate of a synchronizing event is a function, its value can be evaluated only on the global state space. This can be done in Step 3 of Algorithm 1 when NCD CCs of the SAN are formed. Hence, if the synchronizing transition rate is a function and the maximum value of the function is not known in advance, then the corresponding synchronizing event also forms a potential set that is a singleton. Regarding the second case, we make the following observation. The position of a synchronizing transition rate in Q is uniquely determined by all synchronizing transition matrices that correspond to the synchronizing event. This can be seen from equation (1). Hence, we have the following proposition.

PROPOSITION 7. *In a SAN with simple synchronizations, the set $\mathcal{E}^*$ of synchronizing events contribute to the same nonzero element of Q if and only if there exists at least one nonzero element with the same indices in the matrices $Q_{e_j}^{(i)}$ for all $e_j \in \mathcal{E}^*$ and $i = 1, 2, \ldots, N$.*

Proof. The proof follows from equation (1), definition of tensor product, Definitions 2 and 3. □

Example 5. Consider Example 1. We remark that synchronizing events 1 and 2 are simple. By inspecting the synchronizing event matrices of $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$, we see that $Q_{e_1}^{(1)}$ and $Q_{e_2}^{(1)}$ have a nonzero element with the same indices (2,1), and $Q_{e_1}^{(2)}$ and $Q_{e_2}^{(2)}$ have a nonzero element with the same indices (3,1). Hence, transition rates that correspond to e_1 (i.e., λ_2) and e_2 (i.e., μ_3) contribute to the same element of Q (see element $Q(6, 1)$).

Those synchronizing events that are not classified as potential sets of singletons must be tested for the condition in Proposition 7. The test of two events, t and u, for the condition requires the comparison of the indices of nonzero elements in $Q_{e_t}^{(i)}$ and $Q_{e_u}^{(i)}$ for $i = 1, 2, \ldots, N$; that is, we test N pairs of matrices. For k events, the number of matrix pairs that need to be tested is $Nk(k-1)/2$. Note that for three events, t, u, and v, the fact that the pairs $(Q_{e_t}^{(i)}, Q_{e_u}^{(i)})$ and $(Q_{e_u}^{(i)}, Q_{e_v}^{(i)})$ each have at least one nonzero element with the same indices for $i = 1, 2, \ldots, N$ does not imply that the events t and v also satisfy the condition. In other words, the condition is not transitive. This further complicates the test for the condition in Proposition 7.

In order to avoid excessive computation associated with the test, we consider the set of synchronizing events $\mathcal{P}$ as a potential set if for all $e_u \in \mathcal{P}$ there exists $e_v \in \mathcal{P}$ such that the condition in Proposition 7 is satisfied for synchronizing events u and v, and the sum of transition rates of synchronizing events in $\mathcal{P}$ is greater than or equal to $\alpha\epsilon$. According to this definition, we form potential sets as follows. Let $\mathcal{L}$ be the set of synchronizing events that are not classified as potential sets of singletons. We choose event $e_v \in \mathcal{L}$, remove it from $\mathcal{L}$, and test e_v with each event in $\mathcal{L}$ for the condition in Proposition 7. Let $\mathcal{K}$ be the set of events that satisfy this condition. Then, if the sum of the transition rates of synchronizing event v and those in $\mathcal{K}$ is greater than or equal to $\alpha\epsilon$, we remove the events that are in $\mathcal{K}$ from $\mathcal{L}$ and form the potential set $\mathcal{P} = \{e_v\} \cup \mathcal{K}$. We repeat this procedure for all events in $\mathcal{L}$ until $\mathcal{L} = \emptyset$.

Example 4 (continued). Let us consider the application of Algorithm 1.2 to the SAN in Example 4 for which $\alpha = 14$. Let $\epsilon = 0.3$ implying $\alpha\epsilon = 4.2$. The transition rate of the master automaton of simple synchronizing event 1 is 5 and greater than $\alpha\epsilon$. See $Q_{e_1}^{(3)}(2,1)$ in Example 4. Hence, the first potential set, $\mathcal{P}_1$, consists of synchronizing event 1 only. The second synchronizing event of the SAN also forms a potential set. See $Q_{e_2}^{(2)}(1,3)$ for justification. Thus, $\mathcal{P}_1 = \{e_1\}$ and $\mathcal{P}_2 = \{e_2\}$. Now, consider the case in which $\epsilon = 0.4$ implying $\alpha\epsilon = 5.6$. Both transition rates of synchronizing events 1 and 2 are less than $\alpha\epsilon$. Hence, we have to test these two events for the condition in Proposition 7; that is, we check if each of the three pairs of matrices $(Q_{e_1}^{(1)}, Q_{e_2}^{(1)})$, $(Q_{e_1}^{(2)}, Q_{e_2}^{(2)})$, and $(Q_{e_1}^{(3)}, Q_{e_2}^{(3)})$ have at least one nonzero element with the same indices. However, the condition in Proposition 7 is not satisfied. Thus, the number of potential sets for the case of $\epsilon = 0.4$ is zero. This implies that neither of the synchronizing events influence the NCD partitioning of the underlying MC. Therefore, synchronizing events of the SAN are omitted from further consideration in Step 3 of Algorithm 1 when $\epsilon = 0.4$.

4.3. Constructing NCD connected components. As indicated in Remark 2, a nonzero element in the global generator of a SAN originates either from a local transition rate or from one or more synchronizing transition rates. Hence, NCD CCs of the underlying MC are determined by (i) constant local transition rates that are greater than or equal to $\alpha\epsilon$, (ii) functional local transition rates that can take values greater than or equal to $\alpha\epsilon$, or (iii) transition rates of synchronizing events that are in the potential sets $\mathcal{P}_r$, $r = 1, 2, \ldots, N_{\mathcal{P}}$. These three different possibilities are implemented in Algorithm 1.3 that appears in appendix D. The input parameters of the algorithm are local transition rate and synchronizing event matrices, ϵ, α computed by Algorithm 1.1, and potential sets formed by Algorithm 1.2. The output of Algorithm 1.3 is the set of NCD CCs of the underlying MC.

First, we consider possibility (i) in which local transition rates are constant, and assume that $Q = Q_l$ (see equation (1)). Using $\alpha\epsilon$, we can find the NCD CCs of $Q_l^{(i)}$, $i = 1, 2, \ldots, N$. Let $\mathcal{C}^{(i)}$ be the set of NCD CCs of $Q_l^{(i)}$, where a member of $\mathcal{C}^{(i)}$, denoted by $\mathtt{c}^{(i)}$, is a partition of states from $\mathcal{A}^{(i)}$. Let $\mathcal{B}$ and $\mathcal{H}$ be sets in which each member of either set is also a set. In other words, $\mathcal{B}$ as well as $\mathcal{H}$ is a set of sets. We define the binary operator $\odot$ between the two sets $\mathcal{B}$ and $\mathcal{H}$ as $\mathcal{B} \odot \mathcal{H} = \{\mathtt{b} \times \mathtt{h} \mid \mathtt{b} \in \mathcal{B}, \mathtt{h} \in \mathcal{H}\}$, where $\times$ is the ordinary Cartesian product operator. Then, based on the graph interpretation of the tensor sum operator discussed in [14], the set of NCD CCs is given by $\mathcal{C} = \mathcal{C}^{(1)} \odot \mathcal{C}^{(2)} \odot \cdots \odot \mathcal{C}^{(N)}$. Observe that if $\mathcal{C}^{(i)}$, $i = 1, 2, \ldots, N$, are singletons, then $\mathcal{C}$ is a singleton as well; that is, the underlying MC is not NCD for given ϵ. One can take advantage of the same property when there are only K $(< N)$ $\mathcal{C}^{(i)}$ that are singletons. In this case, we renumber the automata so that these K sets assume indices from $(N - K + 1)$ to N. Then these K sets can be replaced with the set $\mathcal{C}^{[N-K+1]} = \{\{1, 2, \ldots, n_{N-K+1}\} \times \{1, 2, \ldots, n_{N-K}\} \times \cdots \times \{1, 2, \ldots, n_N\}\}$.

Now we bring into the picture functional local transition rates and consider possibility (ii). Let us assume that the automata of the given SAN can be reordered and renumbered so that transitions of automaton i depend (if at all) on the states of higher indexed automata, but they do not depend on the states of lower indexed automata (see [13] for details). Since Cartesian product is associative, $\odot$ is also associative, and one can rewrite the expression for $\mathcal{C}$ as

$$\mathcal{C} = \left(\mathcal{C}^{(1)} \odot \left(\mathcal{C}^{(2)} \odot \cdots \odot \left(\mathcal{C}^{(N-1)} \odot \mathcal{C}^{(N)}\right) \cdots\right)\right). \tag{8}$$

Given $\mathcal{C}^{[k]} = \left(\mathcal{C}^{(k)} \odot \left(\mathcal{C}^{(k+1)} \odot \cdots \odot \left(\mathcal{C}^{(N-1)} \odot \mathcal{C}^{(N)}\right) \cdots\right)\right)$, the union of all members of $\mathcal{C}^{[k]}$ is a set that is equivalent to the product state space of $\mathcal{A}^{(k)}, \mathcal{A}^{(k+1)}, \dots, \mathcal{A}^{(N)}$. Therefore, taking into account the assumed ordering of automata, functional transition rates of $\mathcal{A}^{(k)}$ can be evaluated and NCD CCs of $\mathcal{C}^{[k]}$ can be updated accordingly. More formally, let $Q_l^{(k)}(s_k, \tilde{s}_k)$ be a functional element, i.e., $Q_l^{(k)}(s_k, \tilde{s}_k) = f$. Then the NCD CCs $\mathsf{c}^{[k]}, \tilde{\mathsf{c}}^{[k]} \in \mathcal{C}^{[k]}$ must be joined if $(s_k, s_{k+1}, \dots, s_N) \in \mathsf{c}^{[k]}$, $(\tilde{s}_k, s_{k+1}, \dots, s_N) \in \tilde{\mathsf{c}}^{[k]}$, and $f(s_k, s_{k+1}, \dots, s_N) \geq \alpha\epsilon$.

Example 4 (continued). We illustrate possibilities (i) and (ii) on the SAN of Example 4 by omitting synchronizing events 1 and 2. Synchronizing events are treated in possibility (iii). We set $\epsilon = 0.3$ implying $\alpha\epsilon = 4.2$ and assume that the automata are ordered as $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. First, we find the NCD CCs of all local transition rate matrices as in possibility (i) by treating functional transition rates as zero. Inspection of local transition rate matrices shows that local transition rates of all automata are less than $\alpha\epsilon$. Hence, we have $\mathcal{C}^{(1)} = \{\{1_1\}, \{2_1\}\}$, $\mathcal{C}^{(2)} = \{\{1_2\}, \{2_2\}, \{3_2\}\}$, and $\mathcal{C}^{(3)} = \{\{1_3\}, \{2_3\}\}$. The subscripts in the states enable us to distinguish between states with identical indices but that belong to different automata. According to equation (8), we form the NCD CCs of $Q_l^{(3)} \bigoplus Q_l^{(1)}$, i.e., $\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)} = \{\{(1_3, 1_1)\}, \{(1_3, 2_1)\}, \{(2_3, 1_1)\}, \{(2_3, 2_1)\}\}$. Then we continue with possibility (ii). The value of the functional transition rate $Q_l^{(3)}(1,2)$ $(= f)$ depends on the state of $\mathcal{A}^{(1)}$ only. Hence, we can evaluate f when $\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)}$ is formed. The functional transition rate f evaluates to 5, which is larger than $\alpha\epsilon$, when $\mathcal{A}^{(1)}$ is in state 2. Therefore, we join $\{(1_3, 2_1)\}$ and $\{(2_3, 2_1)\}$. Finally, the NCD CCs of Q_l are given by

$$\begin{aligned}
\mathcal{C} &= \mathcal{C}^{(2)} \odot (\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)}) \\
&= \{\{1_2\}, \{2_2\}, \{3_2\}\} \odot \{\{(1_3, 1_1)\}, \{(1_3, 2_1), (2_3, 2_1)\}, \{(2_3, 1_1)\}\} \\
&= \{\{(1_2, 1_3, 1_1)\}, \{(1_2, 2_3, 1_1)\}, \{(2_2, 1_3, 1_1)\}, \{(2_2, 2_3, 1_1)\}, \\
&\quad\ \{(3_2, 1_3, 1_1)\}, \{(3_2, 2_3, 1_1)\}, \{(1_2, 1_3, 2_1), (1_2, 2_3, 2_1)\} \\
&\quad\ \{(2_2, 1_3, 2_1), (2_2, 2_3, 2_1)\}, \{(3_2, 1_3, 2_1), (3_2, 2_3, 2_1)\}\}
\end{aligned}$$

Now we consider possibility (iii). When possibilities (i) and (ii) are handled, the union of all members in $\mathcal{C}$ is a set that corresponds to the global state space of the SAN. The transition rate of synchronizing event t can be taken into account as follows. Let $(s_1, s_2, \dots, s_N) \in \mathsf{c}$ and $(\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_N) \in \tilde{\mathsf{c}}$, where $\mathsf{c}, \tilde{\mathsf{c}} \in \mathcal{C}$. Then c and $\tilde{\mathsf{c}}$ must be joined if $\prod_{i=1}^{N} Q_{e_t}^{(i)}(s_i, \tilde{s}_i) \geq \alpha\epsilon$. Since the global state space of the SAN is usually very large, it may take a significant amount of time to find all pairs c and $\tilde{\mathsf{c}}$ that satisfy this condition.

Fortunately, the situation can be improved. Let p, $1 < p \leq N$, be the smallest index among automata involved in event t, i.e., $Q_{et}^{(i)} = I_{n_i}$ for $i = 1, 2, \dots, p-1$. We rewrite the first two terms of equation (1) as

$$(9)\quad \bigoplus_{i=1}^{N} Q_l^{(i)} + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)} = \left(\bigoplus_{i=1}^{p-1} Q_l^{(i)}\right) \bigoplus Q_l^{[p]} + \left(\bigotimes_{i=1}^{p-1} I_{n_i}\right) \bigotimes Q_{et}^{[p]} + \sum_{j=1, j\neq t}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)},$$

where $Q_l^{[p]} = \bigoplus_{i=p}^{N} Q_l^{(i)}$, and $Q_{et}^{[p]} = \bigotimes_{i=p}^{N} Q_{et}^{(i)}$. From the definition of tensor sum, the first two terms of expression (9) can be written as

$$(10)\qquad \left(\bigoplus_{i=1}^{p-1} Q_l^{(i)}\right) \bigoplus Q_l^{[p]} + \left(\bigotimes_{i=1}^{p-1} I_{n_i}\right) \bigotimes Q_{et}^{[p]} = \left(\bigoplus_{i=1}^{p-1} Q_l^{(i)}\right) \bigoplus \left(Q_l^{[p]} + Q_{et}^{[p]}\right).$$

From (10), it can be seen that the transition rate of synchronizing event t can be taken into account on the smaller state space $\mathcal{C}^{(p)} \odot \mathcal{C}^{(p+1)} \odot \cdots \odot \mathcal{C}^{(N)}$. The same idea can be extended to the potential sets formed in Step 2. In other words, if for $\mathcal{P}_r$, there exists σ_r, $1 < \sigma_r \leq N$, such that $Q_{e_j}^{(i)} = I_{n_i}$ for $i = 1, 2, \dots, \sigma_r - 1$ and all $e_j \in \mathcal{P}_r$, then transition rates of synchronizing events in $\mathcal{P}_r$ can be taken into account when the set $\mathcal{C}^{[\sigma_r]} = \mathcal{C}^{(\sigma_r)} \odot \mathcal{C}^{(\sigma_r+1)} \odot \cdots \odot \mathcal{C}^{(N)}$ is formed. We remark that for the assumed ordering of automata, all functional transitions that may be present in synchronizing transition matrices of events in $\mathcal{P}_r$ can be evaluated when $\mathcal{C}^{[\sigma_r]}$ is formed.

Example 4 (continued). We continue with the illustration of Algorithm 1 on the SAN of Example 4. For $\epsilon = 0.3$, each of the two synchronizing events of the SAN is classified as a potential set. We assume the same ordering of automata, i.e., $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. After renumbering the automata, let the new indices of the automata be $\tilde{1}$, $\tilde{2}$, $\tilde{3}$, respectively. For the given ordering of automata, the smallest index among automata involved in event 1 as well as in event 2 is $\tilde{1}$. Hence, the transition rates of events 1 and 2 can be taken into account when $\mathcal{C}^{[\tilde{1}]} = \mathcal{C}$ is formed. Due to the transition rate of synchronizing event 1, we join the NCD CCs that have the members $(1_2, 2_3, 1_1)$ and $(3_2, 1_3, 1_1)$, $(2_2, 2_3, 1_1)$ and $(1_2, 1_3, 1_1)$, $(3_2, 2_3, 1_1)$ and $(1_2, 1_3, 1_1)$. Similarly, due to synchronizing event 2, we join the NCD CCs that have the members $(1_2, 1_3, 1_1)$ and $(3_2, 1_3, 2_1)$, $(1_2, 1_3, 2_1)$ and $(3_2, 1_3, 1_1)$, $(1_2, 2_3, 1_1)$ and $(3_2, 2_3, 2_1)$, $(1_2, 2_3, 2_1)$ and $(3_2, 2_3, 1_1)$. For justification, see $\mathcal{C}$ formed in the example following possibility (ii), and the SAN in Example 4.

Our next remark is about cyclic dependencies. When the automata of a SAN have cyclic dependencies, they cannot be ordered as discussed. Such cases can be handled as follows. Let $G(\mathcal{V}, \mathcal{E})$ be the digraph in which v_i corresponds to $\mathcal{A}^{(i)}$ and $(v_i, v_j) \in \mathcal{E}$ if transitions in $\mathcal{A}^{(i)}$ depend on the state of $\mathcal{A}^{(j)}$ (see Definition 4). Let G_{SCC} be the digraph obtained by collapsing each SCC of G to a single vertex. This graph is acyclic and the automata of the SAN can be ordered topologically with respect to G_{SCC}. Assuming that the automata are in this order, let p be the smallest index among cyclically dependent automata. Then we can evaluate all functional transitions in the cyclically dependent automata when $\mathcal{C}^{[p]}$ is formed. This observation is omitted from Algorithm 1.3 presented in appendix D. The special case in which a cyclic dependency is created by transitions in the synchronizing transition matrices of a particular event can be handled in the same way as discussed in possibility (iii). There, the potential set $\mathcal{P}_r$, $r \in \{1, 2, \dots, N_{\mathcal{P}}\}$, is taken into account when $\mathcal{C}^{[\sigma_r]}$ is formed. Assuming that the automata are ordered topologically with respect to G_{SCC}, all functions in the matrices of synchronizing events that belong to $\mathcal{P}_r$ can be evaluated when $\mathcal{C}^{[\sigma_r]}$ is formed.

Our final remark is about a SAN with more that one essential subset of states and/or transient states. As in subsection 4.2, we refer to the states other than the ones in the essential subset of interest as uninteresting. For $1 < i \leq N$, we do not have a one-to-one mapping between the global state space and the union of all members in $\mathcal{C}^{[i]}$. Hence, we cannot say whether a member of $\mathsf{c}^{[i]} \in \mathcal{C}^{[i]}$ maps to a state in the essential subset of interest or to an uninteresting state. Therefore, the decomposition of $\mathcal{C}$ as in (8) that allows us to handle functional local transition rates and synchronizing transition rates on a smaller state space cannot be used. This is because one or both of the members that belong to the joined NCD CCs may map to an uninteresting state. For a SAN with uninteresting states, possibilities (ii) and (iii) should be considered on the global state space. Hence, the NCD CCs $\mathsf{c}, \tilde{\mathsf{c}} \in \mathcal{C}$ should be joined only if the members under consideration from each of the two sets map into the essential subset of interest. When we compute $\mathcal{C} = \mathcal{C}^{(1)} \odot \mathcal{C}^{(2)} \odot \cdots \odot \mathcal{C}^{(N)}$, uninteresting states must also be omitted from consideration. From the definition of the binary operator $\odot$, if s_i and $\tilde{s}_i$ are in the same NCD CC of $\mathcal{C}^{(i)}$, then it must be that $(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1} \ldots, s_N)$ and $(s_1, s_2, \ldots, s_{i-1}, \tilde{s}_i, s_{i+1} \ldots, s_N)$ are in the same NCD CC of $\mathcal{C}$. When uninteresting states are present, we exercise the additional constraint that $(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1} \ldots, s_N)$ and $(s_1, s_2, \ldots, s_{i-1}, \tilde{s}_i, s_{i+1} \ldots, s_N)$ must belong to the essential subset of interest.

In the next subsection, we summarize for Algorithm 1 the detailed space and time complexity analysis that appears in appendix E, and apply the results to Example 4.

4.4. Complexity analysis of Algorithm 1. The core operation performed by an algorithm that finds the NCD CCs of a MC is floating-point comparison. Hence, we provide the number of floating-point comparisons performed in Algorithm 1. Regarding the algorithm's storage requirements, we remark that its three steps are executed sequentially. Hence, the maximum amount of memory required by Algorithm 1 is upper bounded by an integer array of length $O(n)$.

As in appendix E, we assume that the MC underlying the SAN is irreducible. In Step 1, the number of floating-point comparisons is given by $\sum_{k=1}^{N_D} \prod_{i, \mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$. For the best case in which each dependency set is a singleton, the number of floating-point comparisons reduces to $\sum_{i=1}^{N} n_i$. On the other hand, if all automata form a single dependency set, we have the upper bound $\prod_{i=1}^{N} n_i = n$. In Step 2, the lower bound on the number of floating-point comparisons is E, and it corresponds to the case in which the transition rate of each simple synchronizing event is greater than or equal to $\alpha\epsilon$. The upper bound is equal to $\frac{1}{2}E(E+1)$ floating-point comparisons. This number of floating-point comparisons is achieved when the transition rate of each simple synchronizing event is less than $\alpha\epsilon$ and the transition rates of synchronizing events do not sum up in Q. The number of floating-point comparisons in Step 3 depends strongly on the number of functional transitions and synchronizing events as well as the automata ordering. Assuming that in Step 2 of Algorithm 1 synchronizing event r is classified as the potential set $\mathcal{P}_r$, $r = 1, 2, \ldots, E$, and the automata are ordered as discussed in possibility (iii) in subsection 4.3, the number of floating-point comparisons in Step 3 is given by

$\sum_{i=1}^{N} nz_l^{(i)} + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$, where $nz_l^{(i)}$ is the number of nonzero off-diagonal elements in $Q_l^{(i)}$, nf_i is the number of functional transitions in $Q_l^{(i)}$, $nz_{e_r}^{(j)}$ is the number of nonzeros in $Q_{e_r}^{(i)}$, and m_r is the index of the master automaton of event r. Finally, the number of floating-point comparisons performed in Algorithm 1 is given by $E + \sum_{i=1}^{N}(n_i + nz_l^{(i)}) + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ in the best case, and $n + \frac{1}{2}E(E+1) + \sum_{i=1}^{N} nz_l^{(i)} + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ in the worst case.

Step 3 of Algorithm 1 also incurs floating-point multiplications when synchronizing events are handled. Computation of a single nonzero transition originating from synchronizing event r requires $(N - \sigma_r)$ floating-point multiplications. For synchronizing event r, we compute $\prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ elements. Hence, the maximum number of floating-point multiplications in Algorithm 1.3 is $\sum_{r=1}^{E}[(N - \sigma_r) \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}]$. Observe that this expression is almost the same as the last term of the expression for the number of floating-point comparisons performed in Algorithm 1. Hence, assuming that the time it takes to perform floating-point multiplication and floating-point comparison are of the same order, the time complexity of Algorithm 1 is roughly the number of floating-point comparisons.

Example 4 (continued). As an example, we calculate the number of floating-point comparisons performed by Algorithm 1 to find an NCD partitioning of the MC underlying the SAN in Example 4. We use the same input parameters for Algorithm 1 as in subsection 4.3; that is, $\epsilon = 0.3$ and the automata are ordered as $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. The SAN in Example 4 has two dependency sets, $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(3)}\}$ and $\mathcal{D}_2 = \{\mathcal{A}^{(2)}\}$. Hence, Step 1 of Algorithm 1 takes $n_1 n_3 + n_2 = 7$ floating-point comparisons. The diagonal element with maximum magnitude of the SAN is 14 and $\alpha\epsilon = 4.2$. This SAN has two simple synchronizing events. Transition rates of the master automata of these events are greater than $\alpha\epsilon$. Hence, each synchronizing event is classified as a potential set, and the number of floating-point comparisons in Step 2 is 2. In Step 3, we first find the NCD CCs of local transition rate matrices. This operation takes 7 floating-point comparisons and corresponds to the number of off-diagonal nonzero elements in the local transition rate matrices. The value of the functional transition rate in $Q_l^{(3)}$ depends on the state of $\mathcal{A}^{(1)}$. Hence, for the given ordering of automata, the value of the function can be evaluated when $\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)}$ is formed (see Algorithm 1.3). The number of these evaluations is equal to the number of states in $\mathcal{A}^{(1)}$. Hence, the number of floating-point comparisons due to the functional transition rate is 2. Recall that in Step 2, each synchronizing event is classified as a potential set. Hence, transition rates of both events must be taken into consideration in Step 3. For the given ordering of automata, let the new indices of the automata be $\tilde{1}$, $\tilde{2}$, and $\tilde{3}$, respectively. For potential set 1, we have $\sigma_1 = \tilde{1}$, and for potential set 2, we have $\sigma_2 = \tilde{1}$. Hence, the number of floating-point comparisons due to synchronizing events of the SAN is $nz_{e_1}^{(1)} nz_{e_1}^{(2)} + nz_{e_2}^{(1)} nz_{e_2}^{(3)} = 7$. Finally, the total number of floating-point comparisons performed in Algorithm 1 is 25. The number of floating-point multiplications performed to process synchronizing events 1 and 2 is $(N-1)(nz_{e_1}^{(1)} nz_{e_1}^{(2)} + nz_{e_2}^{(1)} nz_{e_2}^{(3)}) = 14$. When the global generator is stored in sparse format, the total number of floating-point comparisons performed by the straightforward algorithm that finds the NCD CCs is 57, which is almost two times as large as the corresponding value of Algorithm 1.

5. Numerical results. We implemented the SC algorithm and Algorithm 1 in C++ as part of the software package PEPS [18]. We ran all the experiments on a SUN UltraSparcstation 10 with 128 MBytes of RAM. To verify the NCD partitionings obtained for a given SAN, we compared our results with the straightforward approach of generating in core the submatrix of Q corresponding to the essential subset of states obtained using the SC algorithm and finding its NCD CCs. We remark that the same data structure for NCD CCs is used in Algorithm 1 and the straightforward approach.

The input parameters of Algorithm 1 are the user specified decomposability parameter ϵ, the vector output by the SC algorithm in which states corresponding to the essential subset of interest are marked, and a file in PEPS format that contains the description of the SAN under consideration. The only modification we introduced to the PEPS descriptor file format is a matrix appended to the end which describes the dependency among automata. This matrix is in the same sparse matrix format used for automata matrices in PEPS. Note that the new descriptor file format is compatible with the previous version of PEPS. We remark that the first step of Algorithm 1 (i.e., $Q \rightarrow P$ transformation) does not introduce any explicit changes to the original input description of the SAN. In other words, ϵ is multiplied by α once and rates are compared with $\alpha\epsilon$ on the fly. Hence, upon termination of the algorithm, the description of the SAN remains unchanged and can be used in further processing. The only modification that we make on the SAN is the transformation of each synchronizing event to the simple form (if the SAN is not already in that form). Note that this transformation is taken into account in the reported results.

As test problems, we use three SAN models [24]. We name them resource sharing, three queues, and mass storage. The first problem is a resource sharing model with U processes of which at most S can be simultaneously accessing the resource. Each process alternates between sleeping and resource using states. The transition rates between these two states for process i are characterized respectively by λ_i and μ_i, $i = 1, 2, \ldots, U$. We consider the case $S < U$; otherwise, all processes are independent from each other and the problem can be modeled as a trivial multidimensional MC. In the SAN model [13], each process is represented by a two-state automaton. There are U such automata implying a state space size of $n = 2^U$. There is a single subset of essential states whose size depends on S with respect to U. Since $S < U$, process i cannot acquire the resource if it is already being used by S processes. Hence, λ_i is a functional transition rate, and the value of the function (i.e., whether the transition is enabled or disabled) depends on the state of all other automata. Observe that there is a single dependency set in the SAN model. In our experiments, we used $\lambda_i = 0.04$ and $\mu_i = 0.03$ for $i = 1, 2, \ldots, U$.

The three queues problem is an open queueing network that consists of three queues with capacities $(C_1 - 1)$, $(C_2 - 1)$, and $(C_3 - 1)$. Customers from queues 1 and 2, called respectively as type 1 and type 2 customers, try to join queue 3. When type 1 customers try to join queue 3 and find it full, they are blocked, whereas type 2 customers in the same situation are lost. Queue i has arrival rate λ_i and service rate μ_i, $i = 1, 2$. The service rate of queue 3 depends on the type of customer and is either μ_{3_1} or μ_{3_2}; moreover, type 1 customers have priority over type 2 customers in service. The system is modeled by 2 synchronizing events and 4 automata $\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \mathcal{A}^{(3_1)}, \mathcal{A}^{(3_2)}$ with respectively C_1, C_2, C_3, C_3 states. The state space size is given by $n = C_1 C_2 C_3^2$ and there is a single subset of $C_1C_2C_3(C_3+1)/2$ essential states. Functional transition rates appear in local transition rate and synchronizing event matrices. There are two dependency sets, $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(3_1)}, \mathcal{A}^{(3_2)}\}$ and $\mathcal{D}_2 = \{\mathcal{A}^{(1)}\}$. In our experiments, we used $\lambda_1 = 0.4$, $\lambda_2 = 0.3$, $\mu_1 = 0.6$, $\mu_2 = 0.5$, $\mu_{3_1} = 0.7$, and $\mu_{3_2} = 0.2$. Detailed description of the three queues problem can be found in [13].

The model of the mass storage system considered consists of two layers of storage. The first layer provides fast access based on magnetic disks. The second layer (i.e., near-line storage) utilizes a robotic tape library. The system which describes the interaction between read/write requests and these two layers is modeled by a SAN of 3 synchronizing events and 5 automata with functional transitions. The automata are denoted by $\mathcal{A}^{(\tilde{C})}, \mathcal{A}^{(n_1)}, \mathcal{A}^{(n_2)}, \mathcal{A}^{(n_3)}\mathcal{A}^{(erl)}$, and they respectively have $\lceil (H-L)(C-1) \rceil + 1$, N_1, N_2, N_3, R states, where $H(L)$ is the high (low) water-mark for the disk cache. The state space size is $n = (\lceil (H-L)(C-1) \rceil + 1)N_1N_2N_3R$ and the corresponding MC is irreducible. The SAN model has three dependency sets, $\mathcal{D}_1 = \{\mathcal{A}^{(n_1)}, \mathcal{A}^{(n_2)}, \mathcal{A}^{(n_3)}\}$, $\mathcal{D}_2 = \{\mathcal{A}^{(\tilde{C})}\}$, and $\mathcal{D}_3 = \{\mathcal{A}^{(erl)}\}$. Since the parameters are too many to discuss, we refer the reader to [10] for detailed information. We used $R = 5$, $H = 0.95$, $L = 0.75$, and the values of the other parameters in [10] except C, N_1, N_2, N_3.

Results of experiments for the resource sharing, three queues, and mass storage problems are presented respectively in Tables 1, 2, and 3. All timing results are in seconds. In these tables, n denotes the number of states in the global state space of the particular SAN under consideration, n_{ess} denotes the number of states in the essential subset when the underlying MC is reducible, nz_{ess} denotes the number of nonzero elements in the submatrix of Q corresponding to the essential subset of states, and SC denotes the time for state classification. For each problem, we indicate, in parentheses under n the values of the integer parameters used. The column ϵ denotes the value of the decomposability parameter used and $|CCs|$ denotes the number of NCD CCs corresponding to ϵ when transient states are removed. The column NCD_S contains timing results for Algorithm 1.

TABLE 1
Results of the resource sharing problem (U, S).

n	n_{ess}	nz_{ess}	SC	ϵ	$\|CCs\|$	NCD_S	Gen.	NCD_N
32,768	16,384	210,664	0.69	0.04	1	0.46	0.61	0.04
(15,7)				0.08	16,384	0.44		0.04
65,536	39,203	563,491	1.57	0.04	1	1.06	1.64	0.13
(16,8)				0.08	39,203	1.03		0.12
131,072	65,536	960,858	3.29	0.04	1	2.15	3.06	0.22
(17,8)				0.08	65,536	2.12		0.21
262,144	155,382	2,514,678	7.24	0.04	1	4.96	8.02	0.57
(18,9)				0.08	155,382	4.83		0.55
524,288	262,144	4,319,100	15.31	0.04	1	9.76	15.03	0.85
(19,9)				0.08	262,144	9.83		0.91
1,048,576	616,666	11,102,426	33.43	0.04	1	22.03		
(20,10)				0.08	616,666	22.31		
2,097,152	1,048,576	19,188,796	70.58	0.04	1	44.98		
(21,10)				0.08	1,048,576	45.45		
4,194,304	2,449,868	48,587,212	152.56	0.04	1	100.79		
(22,11)				0.08	2,449,868	101.71		
8,388,608	4,194,304	84,438,360	319.53	0.04	1	205.24		
(23,11)				0.08	4,194,304	206.98		

TABLE 2
Results of the three queues problem (C_1, C_2, C_3).

n	n_{ess}	nz_{ess}	SC	ϵ	$\|CCs\|$	NCD_S	Gen.	NCD_N
37,800	20,160	112,242	0.44	0.10	1	0.06	0.21	0.02
(12,14,15)				0.22	364	0.12		0.04
				0.25	2,520	0.07		0.04
				0.35	20,160	0.05		0.04
68,850	36,720	207,279	0.82	0.10	1	0.10	0.39	0.05
(18,17,15)				0.22	544	0.22		0.09
				0.25	4,590	0.13		0.07
				0.35	36,720	0.11		0.06
202,400	106,260	608,474	2.63	0.10	1	0.30	1.24	0.17
(23,22,20)				0.22	924	0.68		0.28
				0.25	10,120	0.38		0.24
				0.35	106,260	0.33		0.23
390,000	202,800	1,168,676	4.91	0.10	1	0.56	2.40	0.32
(26,24,25)				0.22	1,200	1.37		0.54
				0.25	15,600	0.71		0.47
				0.35	202,800	0.58		0.45
756,000	390,600	2,264,460	9.83	0.10	1	1.03	4.58	0.62
(30,28,30)				0.22	1,652	2.46		1.04
				0.25	25,200	1.37		0.92
				0.35	390,600	1.12		0.90
1,414,875	727,650	4,239,795	19.04	0.10	1	1.88	8.37	1.16
(35,33,35)				0.22	2,277	4.60		1.94
				0.25	40,425	2.46		1.71
				0.35	727,650	2.02		1.56
4,050,000	2,070,000	12,143,950	56.91	0.10	1	5.02		
(40,50,45)				0.22	4,200	12.66		
				0.25	90,000	6.74		
				0.35	2,070,000	5.53		
6,875,000	3,506,250	20,632,250	96.37	0.10	1	8.63		
(50,55,50)				0.22	5,445	21.85		
				0.25	137,500	11.57		
				0.35	3,506,250	9.18		
9,150,625	4,658,500	27,445,825	131.34	0.10	1	11.25		
(55,55,55)				0.22	5,995	33.04		
				0.25	166,375	14.24		
				0.35	4,658,500	12.44		

The columns Gen. and NCD_N respectively contain timing results to generate in core the submatrix of Q corresponding to the essential subset of states and to naively compute its NCD partitioning for given ϵ after the SC algorithm is executed. We have varied the value of ϵ in each problem to see how the performance of Algorithm 1 changes for different number of NCD CCs.

We remark that the difference between the time required to generate in core the submatrix of Q corresponding to the essential subset of states for a given SAN and the time to find the corresponding NCD partitionings using Algorithm 1 is noticeable. Compare columns Gen. and NCD_S in Tables 1–3, and also compare the sum of columns Gen. and NCD_N with column NCD_S. Moreover, there are cases in each of the three tables for which it is not possible to generate in core the submatrix of Q corresponding to

TABLE 3
Results of the mass storage problem (C, N_1, N_2, N_3).

$n(=n_{ess})$	nz_{ess}	SC	ϵ	$\|CCs\|$	NCD_S	Gen.	NCD_N
30,240	199,440	0.46	0.01	1	0.05	0.43	0.05
(10,12,14,12)			0.05	5	0.03		0.05
			0.15	10	0.02		0.05
			0.22	720	0.03		0.05
			0.25	22,680	0.04		0.06
			0.40	30,240	0.05		0.08
69,120	462,720	1.08	0.01	1	0.13	1.05	0.12
(10,16,18,16)			0.05	5	0.07		0.12
			0.15	10	0.06		0.13
			0.22	960	0.07		0.14
			0.25	56,160	0.14		0.16
			0.40	69,120	0.14		0.15
184,320	1,232,640	2.95	0.01	1	0.16	2.78	0.35
(38,16,16,16)			0.05	5	0.11		0.33
			0.15	10	0.11		0.37
			0.22	2,880	0.12		0.34
			0.25	149,760	0.27		0.41
			0.40	184,320	0.30		0.43
372,680	2,524,060	6.21	0.01	1	0.45	6.12	0.67
(30,22,22,22)			0.05	5	0.27		0.66
			0.15	10	0.26		0.69
			0.22	3,080	0.33		0.69
			0.25	304,920	0.62		0.79
			0.40	372,680	0.71		0.85
760,000	5,130,000	13.09	0.01	1	0.45	12.35	1.38
(90,20,20,20)			0.05	5	0.37		1.33
			0.15	10	0.34		1.40
			0.22	7,600	0.58		1.32
			0.25	646,000	1.21		1.71
			0.40	760,000	1.22		1.82
1,572,160	10,773,920	28.90	0.01	1	2.19		
(35,34,34,34)			0.05	5	1.34		
			0.15	10	1.32		
			0.22	5,440	1.88		
			0.25	1,340,960	3.11		
			0.40	1,572,160	3.48		
3,510,000	23,985,000	65.93	0.01	1	2.08		
(126,30,30,30)			0.05	5	1.73		
			0.15	10	1.72		
			0.22	15,600	3.31		
			0.25	3,042,000	5.78		
			0.40	3,510,000	6.29		
5,573,750	38,220,000	108.92	0.01	1	3.53		
(126,35,35,35)			0.05	5	2.88		
			0.15	10	2.84		
			0.22	18,200	5.80		
			0.25	4,777,500	9.32		
			0.40	5,573,750	10.01		
9,280,000	63,800,000	189.80	0.01	1	5.91		
(140,40,40,40)			0.05	5	5.14		
			0.15	10	5.07		
			0.22	23,200	10.18		
			0.25	8,120,000	17.31		
			0.40	9,280,000	17.96		

the essential subset of states on the particular architecture. Hence, the straightforward approach of finding NCD partitionings is relatively more restricted with memory and is slower than using Algorithm 1.

The time spent for state classification does not involve any floating-point operations, whereas the time spent to generate in core the submatrix of Q corresponding to the essential subset of states primarily involves floating-point arithmetic operations. The overhead associated with evaluating functions slows down both tasks dramatically. Compare columns SC and Gen. in Tables 1–3 with columns NCD_S and NCD_N. The time spent by the SC algorithm is larger than the time spent by Algorithm 1 in all experiments. This is not surprising since the former is based on finding SCCs while the latter is based on finding CCs. The difference is more pronounced when there are multiple dependency sets for which Algorithm 1 can bring in considerable savings.

The resource sharing problem is the most difficult of the three problems considered since it has a single dependency set, is reducible, and contains a significant number of functional transitions. Hence, the time to find its NCD CCs using Algorithm 1 is the largest for a given problem size. Compare column NCD_S in Table 1 with those in Tables 2 and 3. However, even for this problem, Gen. is larger than NCD_S since we are able to take advantage of the constant transition values in automata matrices which makes Algorithm 1 worthwhile to use.

The case of $|CCs| = 1$ corresponds to smaller ϵ and implies the largest number of nonzeros taken into account from automata matrices in Algorithm 1 and from the submatrix of Q corresponding to the essential subset of states in the naive NCD partitioning algorithm. The case of $|CCs| = n_{ess}$ corresponds to larger ϵ and implies larger temporary data structures being used by both algorithms when determining NCD CCs. Hence, for increasing ϵ, the results in columns NCD_S and NCD_N either increase then decrease (Table 2) or they decrease then increase (Table 3). NCD_S for intermediate ϵ values for the mass storage example seem to have benefited significantly from its larger number of dependency sets, irreducibility, and the improvements introduced by possibilities (ii) and (iii) discussed in subsection 4.3.

6. Conclusion.

In this work, we have considered the application of the near complete decomposability concept to SANs. The definitions, propositions, and remarks presented in sections 3 and 4 have enabled us to devise an efficient algorithm that computes NCD partitionings of the MC underlying a SAN. The approach is based on determining the NCD connected components of a SAN from the description of individual automata without generating the global transition rate matrix. We have also implemented a state classification algorithm for SANs that classifies each state in the global state space as essential or transient. The output of the state classification algorithm is used in the NCD partitioning algorithm for SANs. The time and space complexities of the NCD partitioning algorithm depend on the number of automata, the number of synchronizing events, the number of functions, the number of essential states of interest, the sparsity of automata matrices, the dependency sets, and the ordering of automata. Future work should focus on taking advantage of the partitionings computed by the devised algorithms in two-level iterative solvers.

REFERENCES

[1] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, Addison-Wesley, Reading, Massachusetts, 1988.

[2] P. Buchholz, *An aggregation\disaggregation algorithm for stochastic automata networks*, Probability in the Engineering and Informational Sciences, 11 (1997), pp. 229–253.

[3] P. Buchholz, *An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models*, European Journal of Operational Research, 116 (1999), pp. 545–564.

[4] P. Buchholz, *Projection methods for the analysis of stochastic automata networks*, in the Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains, B. Plateau, W. J. Stewart, and M. Silva, eds., Prensas Universitarias de Zaragoza, Spain, 1999, pp. 149–168.

[5] P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper, *Complexity of Kronecker operations on sparse matrices with applications to solution of Markov models*, NASA/ICASE Report No.97-66, Langley Research Center, Virgina, December 1997.

[6] P. Buchholz, M. Fischer, and P. Kemper, *Distributed steady state analysis using Kronecker Algebra*, in the Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains, B. Plateau, W. J. Stewart, and M. Silva, eds., Prensas Universitarias de Zaragoza, Spain, 1999, pp. 76–95.

[7] R. Chan and W. Ching, *Circulant preconditioners for stochastic automata networks*, Technical Report 98-05 (143), Department of Mathematics, The Chinese University of Hong Kong, April 1998.

[8] M. Davio, *Kronecker products and shuffle algebra*, IEEE Transactions on Computers, C-30 (1981), pp. 116–125.

[9] T. Dayar, *Permuting Markov chains to nearly completely decomposable form*, Technical Report BU-CEIS-9808, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, August 1998;
Available online at http://www.cs.bilkent.edu.tr/tech-reports/1998/BU-CEIS-9808.ps.z.

[10] T. Dayar, O. I. Pentakalos, and A. B. Stephens, *Analytical modeling of robotic tape libraries using stochastic automata*, Technical Report TR-97-189, CESDIS, NASA/GSFC, Greenbelt, Maryland, January 1997.

[11] T. Dayar and W. J. Stewart, *On the effects of using the Grassmann-Taksar-Heyman method in iterative aggregation-disaggregation*, SIAM Journal on Scientific Computing, 17 (1996), pp. 287–303.

[12] 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.

[13] 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.

[14] J.-M. Fourneau and F. Quessette, *Graphs and stochastic automata networks*, in Computations with Markov Chains: Proceedings of the 2nd International Workshop on the Numerical Solution of Markov Chains, W. J. Stewart, ed., Kluwer, Boston, Massachusetts, 1995, pp. 217–235.

[15] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems*, SIAM Review, 31 (1989), pp. 240–272.

[16] B. Plateau, *On the stochastic structure of parallelism and synchronization models for distributed algorithms*, in the Proceedings of the SIGMETRICS Conference on Measurement and Modelling of Computer Systems, Texas, 1985, pp. 147–154.

[17] B. Plateau and K. Atif, *Stochastic automata network for modeling parallel systems*, IEEE Transactions on Software Engineering, 17 (1991), pp. 1093–1108.

[18] B. Plateau, J.-M. Fourneau, and K.-H. Lee, *PEPS: A package for solving complex Markov models of parallel systems*, in Modeling Techniques and Tools for Computer Performance Evaluation, R. Puigjaner and D. Ptier, eds., Spain, 1988, pp. 291–305.

[19] 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.

[20] G. W. Stewart, W. J. Stewart, and D. F. McAllister, *A two-stage iteration for solving*

nearly completely decomposable Markov chains, in Recent Advances in Iterative Methods, IMA Vol. Math. Appl. 60, G. H. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, New York, 1994, pp. 201–216.

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

[22] 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.

[23] W. J. Stewart and W. Wu, *Numerical experiments with iteration and aggregation for Markov chains*, ORSA Journal on Computing, 4 (1992), pp. 336–350.

[24] E. Uysal and T. Dayar, *Iterative methods based on splittings for stochastic automata networks*, European Journal of Operational Research, 110 (1998), pp. 166–186.

Appendix

A. State classification algorithm for SANs. The SC algorithm is based on an algorithm that finds SCCs of a digraph using DFS (see subsection 3.2). When the digraph corresponds to the MC underlying a SAN, for each global state s of the SAN, we have to find all states $\tilde{s}(\neq s)$ such that $Q(s,\tilde{s}) \neq 0$. Recall that the global state s can be represented as a vector $(s_1, s_2, \ldots, s_N)$, where s_k is the state of automaton k. According to Remarks 1 and 2, a nonzero element in Q can originate either from a local transition or from one or more synchronizing transitions. If $Q(s,\tilde{s})$ originates from the local transition $Q_l^{(k)}(s_k, \tilde{s}_k)$, then it must be that $\tilde{s}$ corresponds to the global state vector $(s_1, \ldots, s_{k-1}, \tilde{s}_k, s_{k+1}, \ldots, s_N)$. If $Q(s,\tilde{s})$ is formed of transitions of synchronizing event j, then $Q_{e_j}^{(k)}(s_k, \tilde{s}_k) \neq 0$ for $k = 1, 2, \ldots, N$ (see Remark 3). The implementation of a DFS algorithm for SANs is straightforward once these observations are made. The output of the SC algorithm is an integer array of length n with states corresponding to the essential subset of interest marked. Assuming that the matrices of the SAN description are stored in sparse format, the number of times global states are visited by the SC algorithm is $\sum_{i=1}^{N} nz_l^{(i)} \prod_{k=1,k\neq i}^{N} n_k + \sum_{j=1}^{E} \prod_{i=1}^{N} nz_{e_j}^{(i)}$, where $nz_l^{(i)}$ is the number of off-diagonal nonzero elements in $Q_l^{(i)}$, and $nz_{e_j}^{(i)}$ is the number of nonzero elements in $Q_{e_j}^{(i)}$.

B. Algorithm 1.1. Computing α.

Input:
- $Q_l^{(i)}$, $\bar{Q}_{e_j}^{(i)}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, E$
- $\mathcal{D}_k$ is kth dependency set, $k = 1, 2, \ldots, N_{\mathcal{D}}$, as in Definition 4
- integer array of length n with essential subset of interest marked by SC algorithm

Output:
- α as discussed in subsection 4.1

Notation:
- S_k: state that corresponds to $\max\_\mathcal{D}_k$
- $\mathcal{M}_{\mathcal{D}_k}$: set of synchronizing events whose masters are in $\mathcal{D}_k$
- s: global state that corresponds to $S_1, S_2, \ldots, S_{N_{\mathcal{D}}}$
- $Diag_k$: double precision array of length $\prod_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$

```
. for $k = 1, 2, \ldots, N_D$,
.     $Diag_k = \left| \bigoplus_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} \mathrm{diag}(Q_l^{(i)}) + \sum_{j,e_j \in \mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} \mathrm{diag}(\bar{Q}_{e_j}^{(i)}) \right|$
.     sort elements of $Diag_k$ in non-increasing order
.     find $\max\_\mathcal{D}_k = \max_i Diag_k(i)$, and corresponding state $S_k$
.     set element corresponding to $\max\_\mathcal{D}_k$ to 0 in $Diag_k$
.   while global state $s$ does not map into essential subset of interest,
.       for $k = 1, 2, \ldots, N_{\mathcal{D}}$, next_max_$\mathcal{D}_k = \max_i Diag_k(i)$
.       find $t$ such that next_max_$\mathcal{D}_t \geq$ next_max_$\mathcal{D}_k$ for $k = 1, 2, \ldots, N_D$
.       let $\tilde{S}_t$ be state corresponding to next_max_$\mathcal{D}_t$
.       max_$\mathcal{D}_t$ =next_max_$\mathcal{D}_t$
.       set element corresponding to max_$\mathcal{D}_t$ to 0 in $Diag_t$, and $S_t = \tilde{S}_t$
.   $\alpha = \sum_{i=1}^{N_{\mathcal{D}}}$ max_$\mathcal{D}_k$
```

C. Algorithm 1.2. Finding potential sets.

Input:
- $Q_{e_j}^{(i)}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, E$
- ϵ
- α computed by Algorithm 1.1

Output:
- $\mathcal{P}_r$ is rth potential set, $r = 1, 2, \ldots, N_{\mathcal{P}}$, as discussed in subsection 4.2

Notation:
- $\|\mathcal{P}_r\|$: sum of transition rates of synchronizing events in $\mathcal{P}_r$
- $Q_{e_i}^{(k)} \equiv Q_{e_j}^{(k)}$ holds if $Q_{e_i}^{(k)}$ and $Q_{e_j}^{(k)}$ have at least one nonzero element with same indices (see second if-statement in innermost for-loop)

Remark:
- $r = N_{\mathcal{P}}$ upon termination

```
.  for $i = 1, 2, \ldots, E$, set synchronizing event $e_i$ to unmarked
.  $r = 0$
.  for $i = 1, 2, \ldots, E$,
.      if $e_i$ is unmarked then
.          $r = r + 1$
.          $\mathcal{P}_r = \{e_i\}$
.          if $\|\mathcal{P}_r\| < \alpha\epsilon$ then
.              for $j = i + 1, i + 2, \ldots, E$,
.                  if $e_j$ is unmarked and $\|\{e_j\}\| < \alpha\epsilon$ then
.                      if $Q_{e_i}^{(k)} \equiv Q_{e_j}^{(k)}$ for $k = 1, 2, \ldots, N$ then $\mathcal{P}_r = \mathcal{P}_r \cup \{e_j\}$
.              if $\|\mathcal{P}_r\| < \alpha\epsilon$ then
.                  $\mathcal{P}_r = \emptyset$
.                  $r = r - 1$
.              else mark each $e_p \in \mathcal{P}_r$
.          else mark $e_i$
```

D. Algorithm 1.3. Constructing NCD CCs.

Input:
- $Q_l^{(i)}$, $Q_{e_j}^{(i)}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, E$
  with $\mathcal{A}^{(i)}$ reordered/renumbered as discussed in subsection 4.3
- ϵ
- α computed by Algorithm 1.1
- $\mathcal{P}_r$ is rth potential set, $r = 1, 2, \ldots, N_{\mathcal{P}}$, computed by Algorithm 1.2

Output:
- $\mathcal{C}$: NCD CCs of MC underlying SAN as discussed in subsection 4.3

Notation:

- $\mathcal{C}^{(i)}$: set of NCD CCs of $Q^{(i)}$
- σ_r: smallest index among automata involved in synchronizing events in $\mathcal{P}_r$
- $\odot$: binary operator as defined in subsection 4.3

```
. for $i = 1, 2, \ldots, N$,
.     using $\alpha\epsilon$, find $\mathcal{C}^{(i)}$ by treating all functional transition rates as 0
. let $K$ be the number of $\mathcal{C}^{(i)}$s that are singletons
. reorder/renumber $\mathcal{A}^{(i)}$ such that $\mathcal{C}^{(i)}$, $i = N-K+1, N-K, \ldots, N$, are singletons
. if $K > 0$ then $\mathcal{C} = \{\{1, 2, \ldots, n_{N-K+1}\} \times \{1, 2, \ldots, n_{N-K}\} \times \ldots \times \{1, 2, \ldots, n_N\}\}$
. else
.     $\mathcal{C} = \mathcal{C}^{(N)}$
.     $K = 1$
. for $k = N-K, N-K-1, \ldots, 1$,
.     $\mathcal{C} = \mathcal{C}^{(k)} \odot \mathcal{C}$
.     for each functional rate $f = Q_l^{(k)}(s_k, \tilde{s}_k)$
.         for each pair $\mathsf{c}, \tilde{\mathsf{c}} \in \mathcal{C}$ such that
.             $(s_k, s_{k+1}, \ldots, s_N) \in \mathsf{c}$ and $(\tilde{s}_k, s_{k+1}, \ldots, s_N) \in \tilde{\mathsf{c}}$ and $f(s_k, s_{k+1}, \ldots, s_N) \geq \alpha\epsilon$
.                 join $\mathsf{c}$ with $\tilde{\mathsf{c}}$
.     if there exists $r$ such that $\sigma_r = k$ then
.         for each pair $\mathsf{c}, \tilde{\mathsf{c}} \in \mathcal{C}$ such that
.             $(s_k, s_{k+1}, \ldots, s_N) \in \mathsf{c}$ and $(\tilde{s}_k, \tilde{s}_{k+1}, \ldots, \tilde{s}_N) \in \tilde{\mathsf{c}}$ and $\sum_{p \in \mathcal{P}_r} \prod_{i=k}^{N} Q_{e_p}^{(i)}(s_i, \tilde{s}_i) \geq \alpha\epsilon$
.                 join $\mathsf{c}$ and $\tilde{\mathsf{c}}$
```

E. Complexity analysis. The time complexity analysis of Algorithm 1 is complicated in that it requires one to make assumptions regarding the number of nonzero elements in automata matrices, the dependency sets, the ordering of automata, the number of functions, and the number of states in the essential subset of interest. Nevertheless, in the following three subsections, we provide some results concerning time and space complexity. In order to simplify the analysis of the algorithm, we assume that the MC underlying the given SAN description is irreducible. We remark that the algorithm and automata matrices are coded in sparse format. Most diagonal corrector matrices in applications turn out to be identity. To reduce storage requirements further and to improve the complexity of operations with identity matrices, we do not store identity matrices.

E.1. Algorithm 1.1. According to our simplifying assumption, the MC underlying the given SAN description is irreducible. For an irreducible SAN, the diagonal element with maximum magnitude is given by equation (7), which is the sum of the maximums of dependency sets. The operation of finding the maximum of a dependency set requires $\prod_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$ floating-point comparisons. Since this operation is performed N_D times, the total number of floating-point comparisons is given by $\sum_{k=1}^{N_D} \prod_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$. For the best case in which each dependency set is a singleton, the total number of floating-point comparisons reduces to $\sum_{i=1}^{N} n_i$. On the other hand, if all automata form a single dependency set, we have the upper bound $\prod_{i=1}^{N} n_i = n$.

When the MC underlying the SAN has uninteresting states, the vectors $\left|\bigoplus_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} \mathrm{diag}(Q_l^{(i)}) + \sum_{j,e_j \in \mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} \mathrm{diag}(\bar{Q}_{e_j}^{(i)})\right|$, $k = 1, 2, \ldots, N_{\mathcal{D}}$, are stored as sorted. For a dependency set $\mathcal{D}_k$, sorting of an array of length $n_{\mathcal{D}_k} = \prod_{i,\mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$ requires

$O(n_{\mathcal{D}_k} \log n_{\mathcal{D}_k})$ floating-point comparisons. Thus, the total number of floating-point comparisons performed before the while-loop in Algorithm 1.1 is $\sum_{k=1}^{N_{\mathcal{D}}} O(n_{\mathcal{D}_k} \log n_{\mathcal{D}_k})$. The number of iterations of the while-loop depends on the number of essential states of interest and their global state indices. If the number of essential states is n_{ess}, the number of iterations performed of the while-loop is less than or equal to $(n - n_{ess})$. The memory requirement of Algorithm 1.1 consists of a double precision array of length $\sum_{k=1}^{N_{\mathcal{D}}} n_{\mathcal{D}_k}$ $(\leq n)$.

E.2. Algorithm 1.2. Each potential set $\mathcal{P}_r$ is described by its member synchronizing events, the sum of scaled transition rates $||\mathcal{P}_r||$, and σ_r, which is the smallest index among automata involved in the synchronized events in $\mathcal{P}_r$. Since we can have at most E potential sets, Algorithm 1.2 requires at most two integer arrays of length E and one double precision array of length E.

Algorithm 1.2 does not involve any floating-point arithmetic. Therefore, its time complexity depends on the relation between $\alpha\epsilon$ and the transition rate of each simple synchronizing event in the SAN description. See the comparison between $||\mathcal{P}_r||$ and $\alpha\epsilon$ in the second if-statement of Algorithm 1.2. Hence, we give lower and upper bounds on the total number of floating-point comparisons.

The lower bound corresponds to the case where the transition rate of each simple synchronizing event is greater than or equal to $\alpha\epsilon$. In this case, the SAN under consideration has E potential sets and each rate is compared against $\alpha\epsilon$ only once since the algorithm never enters the innermost for-loop. Thus, the smallest total number of floating-point comparisons is E.

The upper bound is achieved when the transition rate of each simple synchronizing event is less than $\alpha\epsilon$ and the transition rates of synchronizing events do not sum up in Q. See the innermost if-statement in Algorithm 1.2. In this case, the outermost for-loop is executed E times, and for each value of i, the innermost for-loop is executed $(E - i)$ times. Hence, the total number of floating-point comparisons with $\alpha\epsilon$ is $\frac{1}{2}E(E + 1)$.

E.3. Algorithm 1.3. It is clear that Algorithm 1.3 requires the largest amount of storage since it works with the global state space of the SAN. The computed NCD partitioning is kept in the structure $\mathcal{C}$ which requires an integer array of length $O(n)$. The exact amount of storage for $\mathcal{C}$ depends on the particular implementation which must be sophisticated enough so that operations of the form "join c and $\tilde{\mathsf{c}}$" can be executed rapidly. This data structure must also grant fast access to a specific element in a particular NCD CC. On the other hand, the data structure $\mathcal{C}^{(i)}$ requires an integer array of length at most $(2n_i + 1)$. The first n_i elements of this array contain a permutation of the state indices of $\mathcal{A}^{(i)}$, its $(n_i + 1)$st element contains the number of NCD CCs, and its last n_i elements contain the number of states in each NCD CC since we may have n_i NCD CCs. Thus for N automata, we need a maximum of $N + 2\sum_{i=1}^{N} n_i$ integer locations for $\mathcal{C}^{(i)}$. The total amount of storage is then upper bounded by $O(n)$ integer locations.

For the time complexity of Algorithm 1.3, we consider floating-point comparisons and also count floating-point multiplications performed to compute transition rates. Floating-point comparisons take place when $\mathcal{C}^{(i)}$ are determined. For $\mathcal{A}^{(i)}$, this means $nz_l^{(i)}$ comparisons are performed, where $nz_l^{(i)}$ is the number of off-diagonal nonzeros in $Q_l^{(i)}$. Hence, we have $\sum_{i=1}^{N} nz_l^{(i)}$ floating-point comparisons for N automata. Floating-point comparisons also take place when evaluating functions. The number depends on the ordering of automata and the number of functions to evaluate in each automaton. The number of floating-point comparisons performed due to one functional transition in $Q_l^{(i)}$ is $\prod_{j=i+1}^{N} n_j$. The total number of such floating-point comparisons is $\sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j$, where nf_i is the number of functional transitions in $Q_l^{(i)}$. Based on our assumption regarding dependency among automata and their ordering, the Nth automaton cannot contain functional transitions and is excluded from the summation. To estimate the number of floating-point comparisons due to synchronizing events, we assume that each synchronizing event in the SAN is classified as a potential set $\mathcal{P}_r$, where r corresponds to the index of the event in $\mathcal{P}_r$, implying E potential sets. Let m_r be the master automaton of event r. Then, for synchronizing event r, we have $\prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ floating-point comparisons, where $nz_{e_r}^{(j)}$ is the number of nonzeros in $Q_{e_r}^{(j)}$. The total number of such comparisons is $\sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$. Hence, the total number of floating-point comparisons in Algorithm 1.3 is given by $\sum_{i=1}^{N} nz_l^{(i)} + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$. This expression depends strongly on the number of functional transitions and synchronizing events in a SAN as well as the automata ordering. Thus, an optimal ordering that minimizes the total number of floating-point comparisons from the perspective of synchronizing events is one in which there is no automaton with index greater than σ_r uninvolved in event r.

For floating-point arithmetic operations incurred when handling synchronizing events, we make the following observations. Computation of a single nonzero transition originating from synchronizing event r requires $(N - \sigma_r)$ floating-point multiplications. For synchronizing event r, we compute $\prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ elements. Hence, the maximum number of floating-point multiplications in Algorithm 1.3 is $\sum_{r=1}^{E}[(N - \sigma_r) \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}]$.