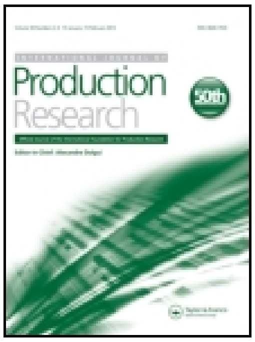


Taylor & Francis
Taylor & Francis Group

International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

Analysis of exponential reliable production lines using Kronecker descriptors

P. Fernandes , M.E.J. O'Kelly , C.T. Papadopoulos & A. Sales

International Journal of Production Research, 2013
Vol. 51, No. 14, 4240–4257, http://dx.doi.org/10.1080/00207543.2012.754550

Analysis of exponential reliable production lines using Kronecker descriptors

P. Fernandes[a], M.E.J. O'Kelly[b], C.T. Papadopoulos[c]* and A. Sales[a]

[a]*Pontificia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil;* [b]*Waterford Institute of Technology, Cork Road, Waterford, Ireland;* [c]*Aristotle University of Thessaloniki, Thessaloniki, GR 541 24 Greece*

(*Received 31 January 2012; final version received 26 November 2012*)

This paper presents a solution procedure for reliable production lines with service times distributed according to an exponential distribution, based on a Markovian formulation with a Kronecker structured representation (sum of tensor products). Specifically, structured Markovian formalisms are used to reduce the impact of the well-known state explosion problem associated with other methods of solution. Such formalisms combined with the Kronecker representation deliver memory efficiency in storing very large models, i.e. models with more than 4×10^7 states. The exact steady-state solutions of these models may be obtained using efficient existing software packages. The proposed solution procedure is illustrated with two detailed examples, and generalised with a model construction algorithm. The computed throughput for several examples of production lines with perfectly reliable machines, as well as the computational costs in terms of CPU time to solve them with PEPS2007 and GTAexpress software packages, are also presented. In effect the paper demonstrates the power of the use of the Kronecker descriptor analysis applied to the derivation of the exact solution of the particular class of production lines considered. The Kronecker descriptor methodology is well-known to analysts concerned with computer and communication systems.

Keywords: Markov modelling; stochastic models; throughput; queueing networks; production modelling; performance analysis; manufacturing systems engineering; stochastic automata networks; tensor (Kronecker) algebra

1. Introduction and literature review

Modelling formalisms are usually applied to describe real (and large) systems, capturing their behaviour and computing performance indices. A well-known modelling formalism used for this purpose is *Markov chains* (Stewart 1994, 2009). Markov chain (MC) models are present in many domains such as chemistry, bioinformatics, economics and social sciences, to cite a few. MC models use simple primitives (*states* and *transitions*) to exemplify the system's evolution and operational semantics. Associated with large systems is the well-known state explosion problem which effectively limits the use of the MC model mapping formalism for such systems using traditional methods of solution.

Over recent years a number of structured modelling formalisms have been proposed for the analysis of continuous-time MCs (CTMCs), taking advantage of the particular structure of the associated transition matrices.

Queueing networks (Kaufman 1983) is certainly the most well-known structured formalism in the performance evaluation area. The popularity of this formalism is based on a very intuitive idea of *customers* passing by *queues*. Many extensions to this formalism have been proposed, providing solution approximations and even product-form solutions for some queueing networks (QN) models (Dallery, Liu, and Towsley 1994; Levantesi, Matta, and Tolio 2003; Bariş and Gershwin 2011). However, most real-life systems do not satisfy the necessary conditions for product-type solutions, such as finite capacity queues.

The exact solution of small production lines was initiated by Hunt (1956) followed by Buzacott (1972), Gershwin and Berman (1981) and Gershwin and Schick (1981), among others. Solutions were obtained for small lines with only two to three stations with limited inter-station buffers, and methods of solution used included matrix recursive and matrix geometric methods applied to the underlying Markov chains. Initially, exponentially distributed processing times were only considered. The work of Buzacott and Kostelski (1987) extended the distribution of processing times to phase-type distributions.

When faced with the analysis of relatively large production lines, there is a need for efficient computational procedures due essentially to the large number of associated states of the underlying Markov chain of such systems. Hillier

*Corresponding author. Email: hpap@econ.auth.gr and Boling (1967) developed a numerical approach for solving reliable exponential and Erlang production lines. Papadopoulos and O'Kelly (1989), Papadopoulos, Heavey, and O'Kelly (1989, 1990) and Heavey, Papadopoulos, and Browne (1993) further developed this work by producing efficient numerical algorithms for generating the transition matrices for reliable and unreliable production lines with exponential and Erlang processing and repair time distributions and efficient solution methods. The algorithm included at the website associated with the book of Papadopoulos et al. (2009) with abbreviated name MARKOV for the generation of the transition matrix and the solution of the associated steady-state Markov equations is based on the works of Papadopoulos, O'Kelly and Heavey. Further extensions in this area are included in the book of Altiok (1997), by using the mixed generalised exponential distributions (phase-type distributions).

In the middle of the 19th Century, the German mathematician Leopold Kronecker (1865) proposed a new operation based on *tensors*, a generalisation of matrices where more than two dimensions could be represented. This extension of linear algebra was represented by one single operator called the *Kronecker product*. Tensor calculus was developed around 1890 by Gregorio Ricci-Curbastro under the title *absolute differential calculus* and originally presented by Ricci in 1892. It was made accessible to many mathematicians by the publication of Ricci and Tullio Levi-Civita's classic text (Ricci and Levi-Civita 1900): *Méthodes du calcul differentiel absolu et leurs applications* (Methods of absolute differential calculus and their applications). Since then many researchers, mainly physicists, have used the Kronecker products to represent operations over multi-dimensional structures. Einstein applied tensor calculus in his relativity theory.

Since the very beginning of computer science many numerical techniques have been based on linear algebra. However, as far as the authors are aware, it was in the late 1970s that computer scientists paid attention to the Kronecker extension to linear algebra. The works of Bellman (1960), Brewer (1978), Amoia, De Micheli, and Santomauro (1981), and Davio (1981) are some of the first studies applying the Kronecker product operation to computer science. In many of these works, the first references to a new operation over tensors made its appearance: *tensor sum*. Logically, tensor sum is also known as *Kronecker sum* or *direct sum*. Tensor product and tensor sum operations, and their properties, comprise *Classical Tensor Algebra* (CTA), which is an effective and efficient algebra for manipulating products and sums of matrices of the type that are embedded in generator matrices of CTMCs. Other early researchers who used tensor algebra were Graham (1981), Neuts (1981) and Kaufman (1983) (queueing models), Beounes (1985) (reliability models), Lynch, Rice, and Thomas (1964) and Birkhoff and Lynch (1984) (solution of partial differential equations), and Regalia and Mitra (1989) (signal processing applications).

Plateau (1984) proposed the first modelling formalism that uses Tensor Algebra as representation: *Stochastic Automata Networks* (SAN). Other relevant works include Plateau (1985) and her colleagues, e.g. Plateau, Fourneau, and Lee (1988), Plateau and Fourneau (1991) and Fernandes, Plateau, and Stewart (1998), among others. The SAN formalism represents the underlying CTMC by a set of tensor products (i.e. by a *Kronecker* descriptor), instead of a single matrix as is used in the queueing network formalism. This structured representation of the matrix is associated with a highly efficient computer memory storage process in mapping the underlying transition system and, additionally, facilitates the efficient solution of the model. Fernandes et al. (1998) introduced the concept of the generalised tensor product to numerically solve SAN models with functional transitions in contrast to constant transitions. The latter was achieved using sophisticated software tools, e.g. PEPS2007 (Brenne et al. 2007) and GTA EXPRESS (Czekster, Fernandes, and Webber 2009). More SAN models may be found in Farina, Fernandes, and Oliveira (2002); Baldo Baldo et al. (2005); Dotti et al. (2005); Dotti, Fernandes, and Nunes (2011); Chanin et al. (2006); Brenner et al. (2009); Czekster et al. (2011b), and Fernandes et al. (2011b).

Donatelli (1994a,b) proposed another modelling formalism using Tensor Algebra: *Superposed Generalised Stochastic Petri Nets* (SGSPN) (see also Beounes (1985) and Kemper (1996), among others). Both SAN and SGSPN formalisms are fundamental to the application of Kronecker-based representations.

Regarding the numerical solution of structured analysis techniques, Buchholz (1999) summarised various iterative methods such as the *Power* method, the *Jacobi* method, projection methods such as the *Generalised Minimal Residual* (GMRES) algorithm and the *Arnoldi* method using the Krylov subspace, the *Gauss–Seidel* or *Successive Over Relaxation* (SOR) method, the block Gauss–Seidel and the block SOR methods and approaches to accelerate convergence such as aggregation and disaggregation steps and preconditioning for both Stochastic Automata Networks (SANs) and hierarchical models (see Saad and Schultz (1986), Stewart (1994), Uysal and Dayar (1998), and Saad (2003) for a description of these numerical methods).

Other formalisms related to the SAN are the Stochastic Process Algebras (SPA), which are extended forms of stochastic automata networks proposed by Buchholz (1994), Hillston (1995), and Hermanns, Herzog, and Mertsiotakis (1998), and the PEPA formalism – *Performance Evaluation Process Algebra* (Hillston 1996). Hillston and Kloul (2007) investigated the benefits of the joint use of SAN and PEPA in the solution of CTMCs.

A Kronecker descriptor for a model with N components is a sum of tensor products with N matrices each. The number of tensor product terms in a descriptor depends on the actual formalism used but it can be explained in general terms as one single tensor sum describing all transitions that are independent within each component (transitions within partitions in (SG)SPN, or local transitions in SAN and PEPA), plus a pair of tensor product terms for each possible interaction between components (transitions between partitions in (SG)SPN, synchronising events in SAN, or co-operations in PEPA).

Comparing these Kronecker representations to the classical approach (a huge sparse matrix), it is obvious that the memory needs are dramatically reduced. The work with tensor product representation is aligned with other efforts to cope with memory demands such as the work of Tan (2003). These memory savings are quite important in reducing the impact of the classical state space explosion problem. Unfortunately, a comparable reduction in the time required to compute solutions has yet to be achieved. To cope with this problem, the research community has been working on numerical techniques to reduce the computational cost in computing exact solutions using both iterative methods (Fernandes, Plateau, and Stewart 1998) and direct methods (Buchholz and Dayar 2003).

Another structured approach is the hierarchical model approach with asynchronously interacting sub-models originated by Buchholz (1992), Buchholz (1994, 1999) demonstrated that there is a correspondence between this approach and the tensor algebra or Kronecker approach. In the context of manufacturing systems, very few applications exist of the use of SAN, (SG)SPN or other formalisms combined with the application of tensor algebra to solve the underlying CTMC. Mitra and Mitrani (1991) examined a simplified model of a manufacturing system with *kanban* control and Buchholz (1999) modelled this system as a (SG)SPN.

In this paper, exponential production lines consisting of a single perfectly reliable machine at each station with finite capacity buffers between any two successive stations are analysed using the SAN formalism based on a tensor (Kronecker) representation. We focus our attention on the usage of the SAN formalism, however any other structured formalism could be employed without loss of generality. In Section 2, to illustrate the traditional solution procedure, the queueing network (QN) model of a three-station line is given. In Section 3, we present the basic modelling primitives of the SAN formalism, including the basic definitions of both the classical and the generalised tensor algebra as well as the Markovian descriptor. Then the SAN equivalent model of the queueing network model of this three-station model is given, followed by the SAN equivalent model of a five-station line with finite inter-station buffers. Additionally, in Section 3 we present an algorithm for the generation of the equivalent SAN model of a production line of the class being considered. In Section 4, the throughput of the three-station production line using this new approach is given, as well as the throughputs of larger similar production lines but with more states. Comparisons are made with known results. Section 5 concludes the paper and discusses a few areas for further research.

2. The queueing network model of a three-station production line

A production line consists of workstations, materials, human resources, and inter-station storage facilities. Storage facilities have a finite capacity. Randomness is introduced by random processing times and the random behaviour of workstations in relation to failure and repair. In terms of classical queueing theory, production lines would be described as finite buffer tandem queueing systems where the workstations are the servers, storage facilities are the buffers or the waiting lines and the jobs are the customers. In Figure 1, which depicts a three-station production line, M_i represents station i and B_i denotes the buffer capacity of the buffer located in front of station M_i, where $i = 1, 2, 3$.

Jobs enter station 1 from buffer B_1 of unrestricted capacity according to a Poisson distribution with arrival rate λ. Each job enters the line at station 1, passes through all stations in order and leaves the third station (last) in finished form. All jobs at each station are processed according to the FIFO queueing discipline.

The assumptions of this Markovian model are as follows. (i) The processing or service times are exponentially distributed random variables with mean rates equal to μ_i ($i = 1, 2, 3$). In general, the service rates need not be identical (i.e. $\mu_i \neq \mu_j$ for $i \neq j$). (ii) All buffers between successive stations have finite capacities not necessarily of the same size. (iii) Blocking of a station occurs as long as the down stream buffer is full and a unit which has been serviced at the station cannot exit from the station. (iv) Stations are assumed to be perfectly reliable, i.e. they do not fail. (v) The general rule that deliberate idleness at a station is not allowed applies. (vi) A basic assumption is that the first station is never starved and the last station is never blocked. Although the arrival process is assumed to be Poisson, it is necessary to assume that the first station is never starved. This assumption characterises the *saturated* line of the saturation model. The fact that the last station is never blocked relates to the storage capacity for final products.

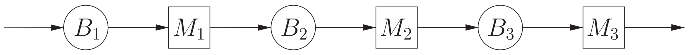


Figure 1. A three-station production line.

The system under consideration is modelled as a two-dimensional stochastic process $N(t) = [N_1(t), N_2(t)]$. See Papadopoulos, Heavey, and O'Kelly (1989, 2009) for the detailed construction of the Markovian model. The states of the three-station line are described by the vector

$$(n_2, s_2, n_3, s_3),$$

where n_i denotes the status of buffer i and s_i denotes the status of station i ($i = 2$, 3). If $s_i = 0$, station i is empty and idle (not serving/working), if $s_i = 1$, station i is occupied and may not or may be serving depending on whether it is blocked or not, respectively, and if $s_i = 2$, station i is occupied and is blocking the preceding station and may not or may be serving depending on whether it is blocked or not, respectively.

For example, in a three-station line, state $(0, 1, 1, 2)$ means that: the buffer before the second station is empty; the second station is occupied with a job that already ended its service and is waiting for room in the third station buffer, which has size one (1); the buffer of the third station has one job; and the third station is serving (busy) and blocking the previous (second) station.

It should be noted that n_1 and s_1 are not included in the state vector as it is assumed that $n_1 \geq 1$ and $s_1 = 1$, i.e. the first station is never starved and it has a buffer of unrestricted capacity.

The solution approach for solving the Markovian model of this and any other general K-station production line consists of the following steps.

- *Derivation* of the states of the system. Two formulas were given by Papadopoulos, Heavey, and O'Kelly (1989) for obtaining the number of states for equal and unequal buffer capacities, respectively.
- *Ordering* of the states. This process facilitates the determination of the structure of the transition matrix.
- *Generation* of the transition matrix (or the conservation matrix). The generating steps of a recursive algorithm are given by Papadopoulos, Heavey, and O'Kelly (1989).
- *Solution* of the resulting system of linear equations.

An appropriate method of solution of this system of equations (if the number of states is less than 300,000) is the *Successive Over Relaxation* (SOR) iterative method. The associated solution algorithm was coded in C++ by Dr. Cathal Heavey and with appropriate instructions is available at the website http://purl.oclc.org/NET/prodlinehttp://purl.oclc.org/NET/prodline associated with the text by Papadopoulos et al. (2009) with the abbreviated name MARKOV.

2.1. *The matrix representation for the three-station example*

Applying the recursive algorithm described by Papadopoulos et al. (2009) for a reliable exponential production line with $K = 3$ stations in series and intermediate buffers $B_2 = 1$ and $B_3 = 0$ (see Figure 1) the following conservative matrix of order 11, equal to the number of states, is obtained:

$$\begin{array}{ccccccccccc|c}
(0,0,0,0) & (0,0,0,1) & (0,1,0,2) & (0,1,0,0) & (0,1,0,1) & (1,1,0,2) & (1,1,0,0) & (1,1,0,1) & (1,2,0,2) & (1,2,0,0) & (1,2,0,1) & \\
-\mu_1 & \mu_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0,0,0,0) \\
0 & -(\mu_1+\mu_3) & \mu_3 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0,0,0,1) \\
0 & 0 & -(\mu_1+\mu_3) & 0 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & (0,1,0,2) \\
\mu_1 & 0 & 0 & -(\mu_1+\mu_2) & \mu_3 & 0 & 0 & 0 & 0 & 0 & 0 & (0,1,0,0) \\
0 & \mu_1 & 0 & 0 & -\sum_{i=1}^{3}\mu_i & \mu_3 & \mu_2 & 0 & 0 & 0 & 0 & (0,1,0,1) \\
0 & 0 & \mu_1 & 0 & 0 & -(\mu_1+\mu_3) & 0 & \mu_2 & 0 & 0 & 0 & (1,1,0,2) \\
0 & 0 & 0 & \mu_1 & 0 & 0 & -(\mu_1+\mu_2) & \mu_3 & 0 & 0 & 0 & (1,1,0,0) \\
0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\sum_{i=1}^{3}\mu_i & \mu_3 & \mu_2 & 0 & (1,1,0,1) \\
0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_3 & 0 & \mu_2 & (1,2,0,2) \\
0 & 0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_2 & \mu_3 & (1,2,0,0) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -(\mu_2+\mu_3) & (1,2,0,1)
\end{array}$$

For this matrix the order of states according to the previously mentioned notation is (0,0,0,0) (0,0,0,1) (0,1,0,2) (0,1,0,0) (0,1,0,1) (1,1,0,2) (1,1,0,0) (1,1,0,1) (1,2,0,2) (1,2,0,0) (1,2,0,1), as indicated above and on the right-hand side. The reader interested in further details of this matrix derivation will find extensive material in Papadopoulos et al. (1989, 2009).

For the balanced case (with identical mean service rates, $\mu_i = 1$, where $i = 1, 2, 3$), by applying the above solution algorithm, it may be shown that the throughput of this line is equal to the throughput of its symmetrical line, i.e. with intermediate buffers of interchanged sizes, $B_2 = 0$ and $B_3 = 1$ and is equal to 0.61333.

3. The equivalent SAN model

The basic idea of the SAN formalism is to represent the whole system by a set of subsystems where each subsystem may behave independently of the other members of the set, with occasional interdependent behaviour between subsystems. Each subsystem is described as a *stochastic automaton*, composed of a set of local states and transitions between them. The Cartesian product of the local states of all automata defines the *product state space* (PSS) of a SAN model. However, from any given initial state only a subset of PSS can be reachable, composing the *reachable state space* (RSS) of the model.

Transitions of a SAN model are triggered by *events* that may affect one single automaton (local event) or many automata (synchronising events). Events have rates describing their frequency (probability) of occurrence and these rates are *constant*, if the event always occurs according to the same exponential distribution, or *functional* if the distribution mean changes according to the current states of the other automata. For convenience, in developing expressions for functional rates, non-typed language is used and evaluated, i.e. where each comparison is evaluated to 1 (*one*) if it is *true* and to 0 (*zero*) if it is *false*, where the automata states (e.g., denoted by '*state A*' for the state of automaton A) are variable according to the current model global state. The reader interested in a formal description of SAN and detailed examples can consult previous publications (Fernandes, Plateau, and Stewart 1998; Brenner, Fernandes, and Sales 2005).

3.1. *Three-station example*

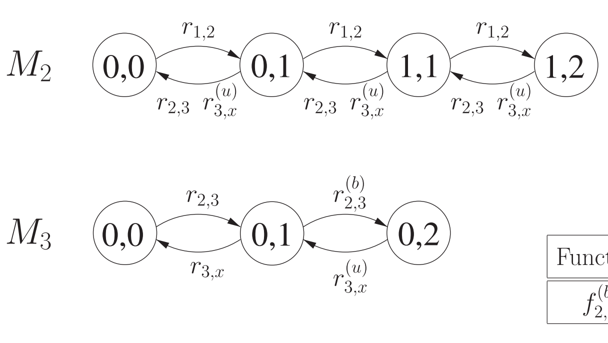
Based on the assumptions presented in Section 2 ($K = 3$ stations and intermediate buffers $B_2 = 1$ and $B_3 = 0$), we propose a SAN model where each station M_i, $i = 2, 3$ and its corresponding buffer B_i is modelled as an *automaton* (named M_i). The number of states of each automaton is determined by the combination of n_i, s_i of each station, i.e. the combination of the buffer occupancy and server state for station i. Hence, station M_2 with buffer $B_2 = 1$ has four states (0,0; 0,1; 1,1; 1,2), whereas station M_3 with $B_3 = 0$ has three states (0,0; 0,1; 0,2). Figure 2 presents the SAN model equivalent to the QN model previously described.

In this model, local event $r_{1,2}$ has a rate μ_1 and represents the completion of service at station M_1 and the availability of the job to enter station M_2 (i.e. in automaton M_2, consisting of buffer B_2 and station M_2). Local event $r_{3,x}$ (with rate μ_3) represents the departure from station M_3 to the exterior of the production line. *Synchronizing event* $r_{2,3}$ is the passage of jobs from M_2 to M_3 and it has rate μ_2.

Additionally, we consider two other events $r_{2,3}^{(b)}$ and $r_{3,x}^{(u)}$ that reflect the status of the relationship between stations (automata) M_2 and M_3. Basically, these events indicate, respectively, that:

- station M_3 is busy and it blocks station M_2 ($r_{2,3}^{(b)}$); or
- station M_3 completes a service and it unblocks M_2 ($r_{3,x}^{(u)}$).

Additionally, event $r_{2,3}^{(b)}$ has a functional rate $f_{2,3}^{(b)}$ which is evaluated at each global state. Function $f_{2,3}^{(b)}$ has a μ_2 rate for global states where M_2 station is busy, or a *zero* rate (i.e. event $r_{2,3}^{(b)}$ does not happen) for global states where server



Type	Event	Rate
loc	$r_{1,2}$	μ_1
syn	$r_{2,3}$	μ_2
loc	$r_{2,3}^{(b)}$	$f_{2,3}^{(b)}$
loc	$r_{3,x}$	μ_3
syn	$r_{3,x}^{(u)}$	μ_3

Function	Definition
$f_{2,3}^{(b)}$	$(state\ M_2 \neq 0,0) * \mu_2$

Figure 2. Equivalent SAN model to the QN model with $B_2 = 1$ and $B_3 = 0$.

M_2 is empty (i.e. M_2 is in state $0,0$). Thus, function $f_{2,3}^{(b)}$ is defined by (*state* $M_2 \neq 0,0) * \mu_2$. In other words, if the expression '*state* $M_2 \neq 0,0$' is true (i.e. M_2 station is busy), it is evaluated to value 1 and multiplied by rate μ_2. Otherwise, if it is false (i.e. M_2 is empty), it is evaluated to value 0 and hence the event $r_{2,3}^{(b)}$ does not happen.

If, as in the states of M_2 in Figure 2, there are two events on the same transition arc, the occurrence of either event will fire the transition in question. In this model, we have 11 valid (reachable) states of a PSS of $4 \times 3 = 12$ states. Note that global state (0,0; 0,2) is an unreachable state by definition.

3.1.1. *Tensor (Kronecker) representation of the three-station example in SAN*

To facilitate the understanding of the tensor representation of serial production lines the following definitions from both classical tensor algebra (CTA) and generalised tensor algebra (GTA) are required. Two illustrative examples of the application of these definitions, one for the CTA and one for the GTA, are also given.

3.1.1.1. *Classical tensor algebra (CTA).* The definition of tensor product ($\otimes$) and tensor sum ($\oplus$).

Definition 3.1: (CTA-1) Let A be a matrix of size $m_1 \times n_1$ and B a matrix of size $m_2 \times n_2$. The tensor product of A and B, denoted as $A \otimes B$, is a matrix D of size $m_1m_2 \times n_1n_2$ defined as

$$D = A \otimes B \quad \text{and} \quad D = (d_{ij}), \quad i \in [1 \ldots m_1m_2],\ j \in [1 \ldots n_1n_2],$$

where $d_{ij} = a_{i_1j_1}b_{i_2j_2}$, $i = (i_1, i_2)$ and $j = (j_1, j_2)$. It may be considered that the tensor product consists of m_1n_1 blocks each having dimensions $m_2 \times n_2$, i.e. the dimensions of B. To specify a particular element, it suffices to specify the block to which the element belongs and the position within the block of the element under consideration.

Due to an associativity property (Fernandes, Plateau, and Stewart 1998), the tensor product operation can generally be defined for N matrices $A^{(k)}$ ($k \in 1 \ldots N$), respectively, with size $m_k \times n_k$, resulting in a matrix D as follows:

$$D = \bigotimes_{k=1}^{N} A^{(k)} \quad \text{and} \quad D = (d_{ij}), \quad i \in \left[1 \ldots \prod_{k=1}^{N} m_k\right],\ j \in \left[1 \ldots \prod_{k=1}^{N} n_k\right],$$

where $d_{ij} = \prod_{k=1}^{N} a_{i_kj_k}^{(k)}$. Therefore, each element of D will be the Cartesian product of one element from each of the matrices $A(k)$ according to its row (i_k) and column (j_k) indexes.

For example, the tensor product of

$$A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \qquad B = \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{vmatrix}, \qquad \text{and} \qquad C = \begin{vmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{vmatrix}$$

will result in the matrix $D = A \otimes B \otimes C$ as follows:

$$D = \left|\begin{array}{cc|cc||cc|cc}
a_{1,1}b_{1,1}c_{1,1} & a_{1,1}b_{1,1}c_{1,2} & a_{1,1}b_{1,2}c_{1,1} & a_{1,1}b_{1,2}c_{1,2} & a_{1,2}b_{1,1}c_{1,1} & a_{1,2}b_{1,1}c_{1,2} & a_{1,2}b_{1,2}c_{1,1} & a_{1,2}b_{1,2}c_{1,2} \\
a_{1,1}b_{1,1}c_{2,1} & a_{1,1}b_{1,1}c_{2,2} & a_{1,1}b_{1,2}c_{2,1} & a_{1,1}b_{1,2}c_{2,2} & a_{1,2}b_{1,1}c_{2,1} & a_{1,2}b_{1,1}c_{2,2} & a_{1,2}b_{1,2}c_{2,1} & a_{1,2}b_{1,2}c_{2,2} \\
\hline
a_{1,1}b_{2,1}c_{1,1} & a_{1,1}b_{2,1}c_{1,2} & a_{1,1}b_{2,2}c_{1,1} & a_{1,1}b_{2,2}c_{1,2} & a_{1,2}b_{2,1}c_{1,1} & a_{1,2}b_{2,1}c_{1,2} & a_{1,2}b_{2,2}c_{1,1} & a_{1,2}b_{2,2}c_{1,2} \\
a_{1,1}b_{2,1}c_{2,1} & a_{1,1}b_{2,1}c_{2,2} & a_{1,1}b_{2,2}c_{2,1} & a_{1,1}b_{2,2}c_{2,2} & a_{1,2}b_{2,1}c_{2,1} & a_{1,2}b_{2,1}c_{2,2} & a_{1,2}b_{2,2}c_{2,1} & a_{1,2}b_{2,2}c_{2,2} \\
\hline\hline
a_{2,1}b_{1,1}c_{1,1} & a_{2,1}b_{1,1}c_{1,2} & a_{2,1}b_{1,2}c_{1,1} & a_{2,1}b_{1,2}c_{1,2} & a_{2,2}b_{1,1}c_{1,1} & a_{2,2}b_{1,1}c_{1,2} & a_{2,2}b_{1,2}c_{1,1} & a_{2,2}b_{1,2}c_{1,2} \\
a_{2,1}b_{1,1}c_{2,1} & a_{2,1}b_{1,1}c_{2,2} & a_{2,1}b_{1,2}c_{2,1} & a_{2,1}b_{1,2}c_{2,2} & a_{2,2}b_{1,1}c_{2,1} & a_{2,2}b_{1,1}c_{2,2} & a_{2,2}b_{1,2}c_{2,1} & a_{2,2}b_{1,2}c_{2,2} \\
\hline
a_{2,1}b_{2,1}c_{1,1} & a_{2,1}b_{2,1}c_{1,2} & a_{2,1}b_{2,2}c_{1,1} & a_{2,1}b_{2,2}c_{1,2} & a_{2,2}b_{2,1}c_{1,1} & a_{2,2}b_{2,1}c_{1,2} & a_{2,2}b_{2,2}c_{1,1} & a_{2,2}b_{2,2}c_{1,2} \\
a_{2,1}b_{2,1}c_{2,1} & a_{2,1}b_{2,1}c_{2,2} & a_{2,1}b_{2,2}c_{2,1} & a_{2,1}b_{2,2}c_{2,2} & a_{2,2}b_{2,1}c_{2,1} & a_{2,2}b_{2,1}c_{2,2} & a_{2,2}b_{2,2}c_{2,1} & a_{2,2}b_{2,2}c_{2,2}
\end{array}\right|.$$

Definition 3.2: (CTA-2) Let A be a square matrix of order n_1 and B a square matrix of order n_2. The tensor sum of A and B, denoted as $A \oplus B$, is a matrix D of size $n_1n_2 \times n_1n_2$ defined as

$$D = A \oplus B = A \otimes I_B + I_A \otimes B,$$

where, I_N is the identity matrix with the dimension of matrix N and '+' represents the usual operation of matrix addition. Note that tensor addition is defined for square matrices only.

For example, the tensor sum of

$$A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \quad \text{and} \quad B = \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{vmatrix}$$

will result in the matrix $C = A \oplus B$ as follows:

$$C = \begin{vmatrix} a_{11}I_B & a_{12}I_B \\ a_{21}I_B & a_{22}I_B \end{vmatrix} + \begin{vmatrix} B & 0 \\ 0 & B \end{vmatrix} = \begin{vmatrix} a_{11}+b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11}+b_{22} & 0 & a_{12} \\ a_{21} & 0 & a_{22}+b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{22}+b_{22} \end{vmatrix}.$$

As for the tensor product operator, an associative property (Fernandes, Plateau, and Stewart 1998) allows us to express the tensor sum of N matrices $A^{(k)}$ ($k \in 1 \ldots N$) in a multiple operator as follows:

$$\bigoplus_{k=1}^{N} A^{(k)} = \sum_{k=1}^{N} I_{n_1} \otimes \cdots \otimes I_{n_{k-1}} \otimes A^{(k)} \otimes I_{n_{k+1}} \otimes \cdots \otimes I_{n_N},$$

where n_k is the order of the matrix $A^{(k)}$ and I_{n_k} is the identity matrix of order n_k.

3.1.1.2. *Generalised tensor algebra (GTA).* The definition of generalised tensor product ($\otimes_g$) and generalised tensor sum ($\oplus_g$).

Generalised tensor algebra (GTA) is an extension of classical tensor algebra (CTA) that considers more complex matrices. In fact, the main distinction of GTA with respect to CTA is the addition of the concept of the *functional elements*.

In CTA, every matrix is composed using constant elements, i.e. values belonging to $\mathbb{R}$. In GTA, however, the matrices can be composed using functional elements. A functional element is a function evaluated in $\mathbb{R}$ according to a set of parameters composed by the rows of one or more matrices.

Thus, a matrix in GTA contains a family of functions that are evaluated according to the rows of matrix B. Formally, we will define the following:

$A(\mathcal{B})$ as the matrix A composed of functional elements that may depend on the rows of matrix B;
$a_{ij}(\mathcal{B})$ as the unevaluated functional element in the ith row and jth column of matrix $A(\mathcal{B})$, i.e. a function of rows of matrix B;
a_i as the ith row of matrix A;
$a_{ij}(b_k)$ as the evaluated (according to row b_k) functional element in the ith row and jth column of matrix $A(\mathcal{B})$, i.e. the value in $\mathbb{R}$ corresponding to the functional element $a_{ij}(\mathcal{B})$ evaluated for row b_k .

It is important to note that the GTA definition allows the expression of functional elements as depending only on the matrices rows, not columns. Such a restriction is due to the use of the matrices as an expression of transition matrices of Markovian models that represent transition rates from the state represented in the row to the state represented in the column. Therefore, a functional element that depends on the row index of a given matrix A means: "*according to the current state represented by matrix A row, a specific rate of transition will be considered*". Note that such a restriction is very intuitive in a Markovian environment, since it implies that the transitions may be affected by the current state, and not the destination state.

Definition 3.3: (GTA-1) Let A be a matrix of size $m_1 \times n_1$ and B a matrix of size $m_2 \times n_2$. The generalised tensor product of $A(\mathcal{B})$ and $B(\mathcal{A})$, denoted as $A(\mathcal{B}) \otimes_g B(\mathcal{A})$, is a matrix D of size $m_1m_2 \times n_1n_2$ defined as

$$D = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) \quad \textit{and} \quad D = (d_{ij}), \quad i \in [1 \ldots m_1m_2],\ j \in [1 \ldots n_1n_2],$$

where $d_{ij} = a_{i_1 j_1}(b_{i_2}) b_{i_2 j_2}(a_{i_1})$, $i = (i_1, i_2)$ and $j = (j_1, j_2)$. It may be considered that the generalised tensor product consists of $m_1 n_1$ blocks each having dimensions $m_2 \times n_2$, i.e. the dimensions of B.

For example, the generalised tensor product of two matrices A and B, each composed by a family of functions that depend on the rows of the other matrix, i.e. the matrices

$$A(\mathcal{B}) = \left| \begin{array}{cc} a_{11}(\mathcal{B}) & a_{12}(\mathcal{B}) \\ a_{21}(\mathcal{B}) & a_{22}(\mathcal{B}) \end{array} \right| \quad \text{and} \quad B(\mathcal{A}) = \left| \begin{array}{cc} b_{11}(\mathcal{A}) & b_{12}(\mathcal{A}) \\ b_{21}(\mathcal{A}) & b_{22}(\mathcal{A}) \end{array} \right|,$$

is the matrix $C = A(\mathcal{B}) \otimes_g B(\mathcal{A})$ as follows:

$$C = \left| \begin{array}{cc|cc} a_{11}(b_1)b_{11}(a_1) & a_{11}(b_1)b_{12}(a_1) & a_{12}(b_1)b_{11}(a_1) & a_{12}(b_1)b_{12}(a_1) \\ a_{11}(b_2)b_{21}(a_1) & a_{11}(b_2)b_{22}(a_1) & a_{12}(b_2)b_{21}(a_1) & a_{12}(b_2)b_{22}(a_1) \\ \hline a_{21}(b_1)b_{11}(a_2) & a_{21}(b_1)b_{12}(a_2) & a_{22}(b_1)b_{11}(a_2) & a_{22}(b_1)b_{12}(a_2) \\ a_{21}(b_2)b_{21}(a_2) & a_{21}(b_2)b_{22}(a_2) & a_{22}(b_2)b_{21}(a_2) & a_{22}(b_2)b_{22}(a_2) \end{array} \right|.$$

Definition 3.4: (GTA-2) Let A be a square matrix of order n_1 and B a square matrix of order n_2. The generalised tensor sum of A and B, denoted as $A \oplus_g B$, is a matrix D of size $n_1 n_2 \times n_1 n_2$ defined as

$$D = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A}) = \left(A(\mathcal{B}) \underset{g}{\otimes} I_B \right) + \left(I_A \underset{g}{\otimes} B(\mathcal{A}) \right),$$

where I_A is the identity matrix with the dimension of matrix A and '+' represents the usual linear algebra operation of matrix addition.

For example, for matrices

$$A(\mathcal{B}) = \left| \begin{array}{cc} a_{11}(\mathcal{B}) & a_{12}(\mathcal{B}) \\ a_{21}(\mathcal{B}) & a_{22}(\mathcal{B}) \end{array} \right| \quad \text{and} \quad B(\mathcal{A}) = \left| \begin{array}{cc} b_{11}(\mathcal{A}) & b_{12}(\mathcal{A}) \\ b_{21}(\mathcal{A}) & b_{22}(\mathcal{A}) \end{array} \right|,$$

the generalised tensor sum $D = A(\mathcal{B}) \oplus_g B(\mathcal{A})$ is expressed as follows:

$$D = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A}) = \left| \begin{array}{cc} a_{11}(\mathcal{B})I_B & a_{12}(\mathcal{B})I_B \\ a_{21}(\mathcal{B})I_B & a_{22}(\mathcal{B})I_B \end{array} \right| + \left| \begin{array}{cc} B(\mathcal{A}) & 0 \\ 0 & B(\mathcal{A}) \end{array} \right|$$

$$= \left| \begin{array}{cccc} a_{11}(b_1) + b_{11}(a_1) & b_{12}(a_1) & a_{12}(b_1) & 0 \\ b_{21}(a_1) & a_{11}(b_2) + b_{22}(a_1) & 0 & a_{12}(b_2) \\ a_{21}(b_1) & 0 & a_{22}(b_1) + b_{11}(a_2) & b_{12}(a_2) \\ 0 & a_{21}(b_2) & b_{21}(a_2) & a_{22}(b_2) + b_{22}(a_2) \end{array} \right|.$$

The reader is addressed to Davio (1981) and Fernandes, Plateau, and Stewart (1998), respectively, for properties that apply to classical and generalised versions of tensor product and sum, the main operators of CTA and GTA.

3.1.1.3. *Markovian descriptor*. Q (or the Kronecker descriptor) is an algebraic formula that allows us to store in a compact form the infinitesimal generator of the underlying Markov chain of a SAN model using a mathematical formula (Plateau 1985, Fernandes, Plateau, and Stewart 1998):

$$Q = \bigoplus_{i=1^g}^{N} Q_l^{(i)} + \sum_{e \in \mathcal{E}_s} \left(\bigotimes_{i=1^g}^{N} Q_{e^+}^{(i)} + \bigotimes_{i=1^g}^{N} Q_{e^-}^{(i)} \right), \tag{1}$$

where N is the number of automata, $Q_l^{(i)}$ is the local matrix that represents the local transitions of the ith automaton, $Q_{e^+}^{(i)}$ and $Q_{e^-}^{(i)}$ are, respectively, the matrices that represent the synchronising transitions and the diagonal adjustment of the synchronising transitions of event e, and $\mathcal{E}_s$ is the set of synchronising events of the SAN model.

Once a tensor sum is equivalent to a sum of particular product tensors, the Kronecker descriptor may be represented as

$$Q = \sum_{j=1}^{(N+2|\mathcal{E}_s|)} \bigotimes_{i=1^g}^{N} Q_j^{(i)}, \quad \text{where } Q_j^{(i)} = \begin{cases} \text{Identity matrix}, & \text{if } j \leq N \text{ and } j \neq i, \\ Q_l^{(i)}, & \text{if } j \leq N \text{ and } j = i, \\ Q_{e^+_{(j-N)}}^{(i)}, & \text{if } N < j \leq (N + |\mathcal{E}_s|), \\ Q_{e^-_{(j-(N+|\mathcal{E}_s|))}}^{(i)}, & \text{if } j > (N + |\mathcal{E}_s|). \end{cases} \tag{2}$$

For this three-station example, applying the formula presented in (2), we have

$$Q = \left(Q_l^{(M_2)} \underset{g}{\otimes} I_{M_3}\right) + \left(I_{M_2} \underset{g}{\otimes} Q_l^{(M_3)}\right) + \left(Q_{r^+_{2,3}}^{(M_2)} \underset{g}{\otimes} Q_{r^+_{2,3}}^{(M_3)}\right) + \left(Q_{r^-_{2,3}}^{(M_2)} \underset{g}{\otimes} Q_{r^-_{2,3}}^{(M_3)}\right) + \left(Q_{r^{(u)+}_{3,x}}^{(M_2)} \underset{g}{\otimes} Q_{r^{(u)+}_{3,x}}^{(M_3)}\right) + \left(Q_{r^{(u)-}_{3,x}}^{(M_2)} \underset{g}{\otimes} Q_{r^{(u)-}_{3,x}}^{(M_3)}\right),$$

where

$$Q_l^{(M_2)} = \begin{array}{|cccc|c} (0,0) & (0,1) & (1,1) & (1,2) & \\ -\mu_1 & \mu_1 & 0 & 0 & (0,0) \\ 0 & -\mu_1 & \mu_1 & 0 & (0,1) \\ 0 & 0 & -\mu_1 & \mu_1 & (1,1) \\ 0 & 0 & 0 & 0 & (1,2) \end{array} \qquad Q_l^{(M_3)} = \begin{array}{|ccc|c} (0,0) & (0,1) & (0,2) & \\ 0 & 0 & 0 & (0,0) \\ \mu_3 & -(\mu_3 + f_{2,3}^{(b)}) & f_{2,3}^{(b)} & (0,1) \\ 0 & 0 & 0 & (0,2) \end{array}$$

$$Q_{r^+_{2,3}}^{(M_2)} = \begin{array}{|cccc|c} (0,0) & (0,1) & (1,1) & (1,2) & \\ 0 & 0 & 0 & 0 & (0,0) \\ \mu_2 & 0 & 0 & 0 & (0,1) \\ 0 & \mu_2 & 0 & 0 & (1,1) \\ 0 & 0 & \mu_2 & 0 & (1,2) \end{array} \qquad Q_{r^+_{2,3}}^{(M_3)} = \begin{array}{|ccc|c} (0,0) & (0,1) & (0,2) & \\ 0 & 1 & 0 & (0,0) \\ 0 & 0 & 0 & (0,1) \\ 0 & 0 & 0 & (0,2) \end{array}$$

$$Q_{r^-_{2,3}}^{(M_2)} = \begin{array}{|cccc|c} (0,0) & (0,1) & (1,1) & (1,2) & \\ 0 & 0 & 0 & 0 & (0,0) \\ 0 & -\mu_2 & 0 & 0 & (0,1) \\ 0 & 0 & -\mu_2 & 0 & (1,1) \\ 0 & 0 & 0 & -\mu_2 & (1,2) \end{array} \qquad Q_{r^-_{2,3}}^{(M_3)} = \begin{array}{|ccc|c} (0,0) & (0,1) & (0,2) & \\ 1 & 0 & 0 & (0,0) \\ 0 & 0 & 0 & (0,1) \\ 0 & 0 & 0 & (0,2) \end{array}$$

$$Q_{r^{(u)+}_{3,x}}^{(M_2)} = \begin{array}{|cccc|c} (0,0) & (0,1) & (1,1) & (1,2) & \\ 0 & 0 & 0 & 0 & (0,0) \\ \mu_3 & 0 & 0 & 0 & (0,1) \\ 0 & \mu_3 & 0 & 0 & (1,1) \\ 0 & 0 & \mu_3 & 0 & (1,2) \end{array} \qquad Q_{r^{(u)+}_{3,x}}^{(M_3)} = \begin{array}{|ccc|c} 0 & 0 & 0 & (0,0) \\ 0 & 0 & 0 & (0,1) \\ 0 & 1 & 0 & (0,2) \end{array}$$

$$Q^{(M_2)}_{r^{(u)-}_{3,x}} = \begin{array}{|cccc|c} (0,0) & (0,1) & (1,1) & (1,2) & \\ 0 & 0 & 0 & 0 & (0,0) \\ 0 & -\mu_3 & 0 & 0 & (0,1) \\ 0 & 0 & -\mu_3 & 0 & (1,1) \\ 0 & 0 & 0 & -\mu_3 & (1,2) \end{array} \qquad Q^{(M_3)}_{r^{(u)-}_{3,x}} = \begin{array}{|ccc|c} (0,0) & (0,1) & (0,2) & \\ 0 & 0 & 0 & (0,0) \\ 0 & 0 & 0 & (0,1) \\ 0 & 0 & 1 & (0,2) \end{array}$$

and finally we have matrix Q as follows:

$$\begin{array}{|cccccccccccc|c}
(0,0;0,0) & (0,0;0,1) & (0,0;0,2) & (0,1;0,0) & (0,1;0,1) & (0,1;0,2) & (1,1;0,0) & (1,1;0,1) & (1,1;0,2) & (1,2;0,0) & (1,2;0,1) & (1,2;0,2) & \\
-\mu_1 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0,0;0,0) \\
\mu_3 & -(\mu_1+\mu_3) & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0,0;0,1) \\
0 & 0 & -\mu_1 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & 0 & 0 & (0,0;0,2) \\
0 & \mu_2 & 0 & -(\mu_1+\mu_2) & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & 0 & (0,1;0,0) \\
0 & 0 & 0 & \mu_3 & -\sum_{i=1}^{3}\mu_i & \mu_2 & 0 & \mu_1 & 0 & 0 & 0 & 0 & (0,1;0,1) \\
0 & \mu_3 & 0 & 0 & 0 & -(\mu_1+\mu_3) & 0 & 0 & \mu_1 & 0 & 0 & 0 & (0,1;0,2) \\
0 & 0 & 0 & 0 & \mu_2 & 0 & -(\mu_1+\mu_2) & 0 & 0 & \mu_1 & 0 & 0 & (1,1;0,0) \\
0 & 0 & 0 & 0 & 0 & 0 & \mu_3 & -\sum_{i=1}^{3}\mu_i & \mu_2 & 0 & \mu_1 & 0 & (1,1;0,1) \\
0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & -(\mu_1+\mu_3) & 0 & \mu_1 & (1,1;0,2) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_2 & 0 & -\mu_2 & 0 & 0 & (1,2;0,0) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_3 & -(\mu_2+\mu_3) & \mu_2 & (1,2;0,1) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & -\mu_3 & (1,2;0,2)
\end{array}$$

This resulting matrix is the stochastic equivalent[1] of the one generated by the queueing network model present in Section 2.1. However, it has two differences that do not affect the stochastic equivalence.

The first difference is the existence of an unreachable state (0,0; 0,2). The row and column corresponding to this state are indicated in grey in the above matrix. Giving any initial state, the model will never reach state (0,0; 0,2). This state must be included because of the 12 states tensor structure, which is the Cartesian product of the automata local states (four local states for automaton M_2, and three local states for automaton M_3).

The second difference between the MARKOV algorithm produced matrix (Section 2.1) and the SAN produced matrix is the order of states. Such a difference is stochastic irrelevant, since one of the matrices may be transformed into another by linear transformations for rows and columns. Once again, the reason for adopting the order in the tensor generated matrix is the Cartesian product of local states of the automata.

3.2. *Larger example*

As the number of stations increase, the equivalent SAN model becomes more complex. For instance, a SAN model for a product line with five stations ($K = 5$) and buffer capacities $B_i = i - 1$ ($i = 2, \ldots, 5$) is presented in Figure 3. The main addition to such a model is the need of more sophisticated synchronisations regarding the unblocking events. For example, there is the possibility that a service completion in the last station would unblock all the previous stations.

The SAN model of Figure 3 is general enough to present and explain the models for any larger production lines. In this model we use the event identifiers according to the description given in Table 1.

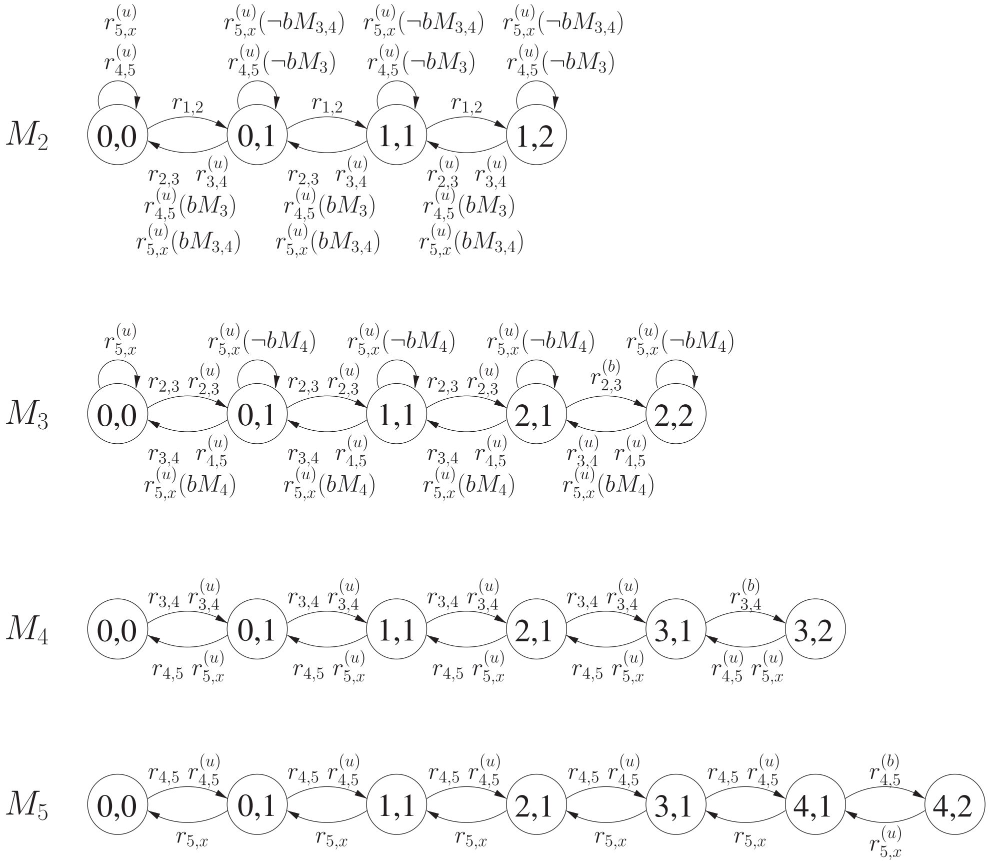
The probabilities used in the model of this larger example with five stations are described in Table 2.

The main difference between this five-station (larger) example (Figure 3), and the simple example with three stations (Figure 2) is the particular treatment needed for the unblocking behaviour. Such behaviour represents the possibility of a chained blocking and unblocking effect.

For instance, when station 4 completes its service on a job, it backward propagates the unblocking to station 3 and it may also unblock station 2. Such propagation is represented by transitions with event $r^{(u)}_{4,5}$, that, in automaton M_2, may alternatively decrease the number of jobs (with probability bM_3 if the precedent stations were blocked) or just stay in the same local state with *loop transitions* (with probability $\neg bM_3$, if the precedent station were not blocked).

Analogously to station 4 unblocking, station 5 service is represented by event $r_{5,x}$ and the backward propagation of the unblocking always unblocks station 4 and may or may not unblock stations 2 and 3. This choice to backward propagate the unblocking is made with alternate probabilities $bM_{3,4}$ and $\neg bM_{3,4}$ for automaton M_2, and with bM_4 and $\neg bM_4$ for automaton M_3.

It is important to recall that, in the SAN formalism, if the same event can fire two or more transitions from a state, it is necessary to indicate a probability indicating the occurrence of this event for each possible transition. For instance, the event $r^{(u)}_{4,5}$ occurs with rate μ_4 and probability bM_3 (which changes the state of automaton M_2 from state $1,2$ to $1,1$)


M_2
$r^{(u)}_{5,x}$ $r^{(u)}_{4,5}$
$r^{(u)}_{5,x}(\neg bM_{3,4})$ $r^{(u)}_{4,5}(\neg bM_3)$
$r^{(u)}_{5,x}(\neg bM_{3,4})$ $r^{(u)}_{4,5}(\neg bM_3)$
$r^{(u)}_{5,x}(\neg bM_{3,4})$ $r^{(u)}_{4,5}(\neg bM_3)$
$r_{1,2}$ $r_{1,2}$ $r_{1,2}$
0,0 0,1 1,1 1,2
$r_{2,3}$ $r^{(u)}_{3,4}$ $r^{(u)}_{4,5}(bM_3)$ $r^{(u)}_{5,x}(bM_{3,4})$
$r_{2,3}$ $r^{(u)}_{3,4}$ $r^{(u)}_{4,5}(bM_3)$ $r^{(u)}_{5,x}(bM_{3,4})$
$r^{(u)}_{2,3}$ $r^{(u)}_{3,4}$ $r^{(u)}_{4,5}(bM_3)$ $r^{(u)}_{5,x}(bM_{3,4})$
M_3
$r^{(u)}_{5,x}$
$r^{(u)}_{5,x}(\neg bM_4)$ $r^{(u)}_{5,x}(\neg bM_4)$ $r^{(u)}_{5,x}(\neg bM_4)$ $r^{(u)}_{5,x}(\neg bM_4)$
$r_{2,3}$ $r^{(u)}_{2,3}$ $r_{2,3}$ $r^{(u)}_{2,3}$ $r_{2,3}$ $r^{(u)}_{2,3}$ $r^{(b)}_{2,3}$
0,0 0,1 1,1 2,1 2,2
$r_{3,4}$ $r^{(u)}_{4,5}$ $r^{(u)}_{5,x}(bM_4)$
$r_{3,4}$ $r^{(u)}_{4,5}$ $r^{(u)}_{5,x}(bM_4)$
$r_{3,4}$ $r^{(u)}_{4,5}$ $r^{(u)}_{5,x}(bM_4)$
$r^{(u)}_{3,4}$ $r^{(u)}_{4,5}$ $r^{(u)}_{5,x}(bM_4)$
M_4
$r_{3,4}$ $r^{(u)}_{3,4}$ $r_{3,4}$ $r^{(u)}_{3,4}$ $r_{3,4}$ $r^{(u)}_{3,4}$ $r_{3,4}$ $r^{(u)}_{3,4}$ $r^{(b)}_{3,4}$
0,0 0,1 1,1 2,1 3,1 3,2
$r_{4,5}$ $r^{(u)}_{5,x}$ $r_{4,5}$ $r^{(u)}_{5,x}$ $r_{4,5}$ $r^{(u)}_{5,x}$ $r_{4,5}$ $r^{(u)}_{5,x}$ $r^{(u)}_{4,5}$ $r^{(u)}_{5,x}$
M_5
$r_{4,5}$ $r^{(u)}_{4,5}$ $r_{4,5}$ $r^{(u)}_{4,5}$ $r_{4,5}$ $r^{(u)}_{4,5}$ $r_{4,5}$ $r^{(u)}_{4,5}$ $r_{4,5}$ $r^{(u)}_{4,5}$ $r^{(b)}_{4,5}$
0,0 0,1 1,1 2,1 3,1 4,1 4,2
$r_{5,x}$ $r_{5,x}$ $r_{5,x}$ $r_{5,x}$ $r_{5,x}$ $r^{(u)}_{5,x}$

Function	Definition
$f^{(b)}_{2,3}$	$(state\ M_2\ \neq\ 0,0) * \mu_2$
$f^{(b)}_{3,4}$	$(state\ M_3\ \neq\ 0,0) * \mu_3$
$f^{(b)}_{4,5}$	$(state\ M_4\ \neq\ 0,0) * \mu_4$
bM_3	$(state\ M_3\ =\ 2,2)$
bM_4	$(state\ M_4\ =\ 3,2)$
$bM_{3,4}$	(bM_3) && (bM_4)

Type	Event	Rate
loc	$r_{1,2}$	μ_1
syn	$r_{2,3}$	μ_2
loc	$r^{(b)}_{2,3}$	$f_{2,3}$
syn	$r^{(u)}_{2,3}$	μ_2

Type	Event	Rate
syn	$r_{3,4}$	μ_3
loc	$r^{(b)}_{3,4}$	$f_{3,4}$
syn	$r^{(u)}_{3,4}$	μ_3

Type	Event	Rate
syn	$r_{4,5}$	μ_4
loc	$r^{(b)}_{4,5}$	$f_{4,5}$
syn	$r^{(u)}_{4,5}$	μ_4
loc	$r_{5,x}$	μ_5
syn	$r^{(u)}_{5,x}$	μ_5

Figure 3. SAN model of the production line with $K = 5$ stations and $B_i = i - 1$ (where $i = 2, 3, 4, 5$). Definitions of events and probabilities are given in Tables 1 and 2, respectively.

or probability $\neg$ $bM3$ (remaining at the same state 1, 2, i.e. the loop transition). This modelling strategy of developing complementary functional probabilities in the SAN formalism facilitates the description of the behaviour of events that can be fired or not depending on the current states of other automata. Note also that the functional probability $bM_{3,4}$ is the logical product of probabilities bM_3 *and* bM_4.

3.3. *Algorithm for model generation*

Algorithm 1 (Figure 4) presents the algorithm for the generation of the equivalent SAN model for a production line of K stations with buffers located in front of each station. This algorithm can be explained in general terms at three blocks: the creation of automata and their respective states (lines 1–3); the creation of events of the model (lines 4–11); and assignment of the transitions between states of the automata (lines 12–23).

The creation of automata M_i (where $i = 2, \ldots, K$) with $B_i + 3$ states, where B_i represents the buffer capacity of the buffer located in front of station i, is described between lines 1 and 3 of Algorithm 1. The creation of the events used in the SAN model are defined between lines 4 and 11. Finally, the assignment of the transitions between states are defined between lines 12 and 23 of Algorithm 1.

Table 1. Identifiers for events of the SAN model for the five-station example (Figure 3).

Event	Meaning
$r_{1,2}$	The passage (routing) of a job from station 1 to station 2, when the intermediate buffer between stations 1 and 2 is not full
$r_{2,3}$	The passage (routing) of a job from station 2 to station 3, when the intermediate buffer between stations 2 and 3 is not full
$r_{2,3}^{(b)}$	The end of service of a job in station 2, indicating at station 3 that station 2 is blocked because the intermediate buffer between stations 2 and 3 is full
$r_{2,3}^{(u)}$	The passage (routing) of a job from station 2 to station 3, which unblocks station 2 because the intermediate buffer between stations 2 and 3 is no longer full
$r_{3,4}$	The passage (routing) of a job from station 3 to station 4, when the intermediate buffer between stations 3 and 4 is not full
$r_{3,4}^{(b)}$	The end of service of a job in station 3, indicating at station 4 that station 3 is blocked because the intermediate buffer between stations 3 and 4 is full
$r_{3,4}^{(u)}$	The passage (routing) of a job from station 3 to station 4, which unblocks at least station 3 because the intermediate buffer between stations 3 and 4 is no longer full; it may also unblock station 2, if station 2 was blocked, i.e. the intermediate buffer between stations 2 and 3 was full
$r_{4,5}$	The passage (routing) of a job from station 4 to station 5 when the intermediate buffer between stations 4 and 5 is not full
$r_{4,5}^{(b)}$	The end of service of a job in station 4, indicating at station 5 that station 4 is blocked because the intermediate buffer between stations 4 and 5 is full
$r_{4,5}^{(u)}$	The passage (routing) of a job from station 4 to station 5, which unblocks at least station 4 because the intermediate buffer between stations 4 and 5 is no longer full; it may also unblock stations 3 and even station 2, if these stations were blocked, i. e. the intermediate buffers between stations 3 and 4, as well as between stations 2 and 3, were full
$r_{5,x}$	The passage (routing) of a job from station 5 to outside the production line
$r_{5,x}^{(u)}$	The passage (routing) of a job from station 5 to outside the production line, which may unblock station 4, station 3 and even station 2, if these stations were blocked, i.e. the intermediate buffers between stations 4 and 5, between stations 3 and 4, as well as between stations 2 and 3, were full

Table 2. Identifiers for probabilities of the SAN model for the five-station example (Figure 3).

Probabilities	Meaning
bM_3	Station 3 is blocked because the intermediate buffer between stations 3 and 4 is full
$\neg bM_3$	Station 3 is not blocked because the intermediate buffer between stations 3 and 4 is not full
bM_4	Station 4 is blocked because the intermediate buffer between stations 4 and 5 is full
$\neg bM_4$	Station 4 is not blocked because the intermediate buffer between stations 4 and 5 is not full
$bM_{3,4}$	Stations 3 and 4 are blocked because the intermediate buffers between stations 3 and 4, as well as between stations 4 and 5, are full
$\neg bM_{3,4}$	Stations 3 and 4 are not blocked because either the intermediate buffer between stations 3 and 4, or the intermediate buffer between stations 4 and 5 is not full

Solving the SAN model produced by this algorithm, it is possible to compute the throughput of the production line represented by the model. To obtain this throughput, it is necessary to observe the probability vector that describes the steady-state solution of the model and to integrate (i.e. to sum) the probabilities of the states where the last station is not empty. This accumulated probability is then multiplied by the service rate μ_K of the last station. Numerically speaking, the throughput is computed by the integration function described in the SAN model defined by $(\text{state } M_K \neq 0, 0) * \mu_K$.

4. Software tools and results

The solution of SAN models is not simple, but it can be undertaken with specialised software tools specifically designed to deal with structured Markovian formalisms. With respect to this paper, two software tools were employed: PEPS2007 (Brenner et al. 2007) and GTAExpress (Czekster, Fernandes, and Webber 2009).

PEPS2007 is one of the software tool packages developed to solve SAN models and represents the underlying CTMC in a compact tensor algebra format. PEPS2007 computes the model's reachable states, given any initial state, and it determines exact steady-state and transient solutions. The numerical solution of the tensor representations developed by PEPS2007 consists of multiplying a vector by a non-trivial structure in a process known as VDP – Vector-Descriptor Product (Fernandes, Plateau, and Stewart 1998).

The other software tool used was GTAExpress, a tool that uses a sophisticated approach to compute the transition probabilities of the CTMC and store them in a sparse format, i.e. a sparse matrix stored in a Harwell–Boeing Format

Algorithm 1 *SAN model generation*

```
for $i = 2$ to $K$ do
    create automaton $M_i$ with $B_i + 3$ states, where $B_i$ is the buffer capacity
end for
create local event $r_{1,2}$ with rate $\mu_1$
for $i = 2$ to $K - 1$ do
    create synchronising event $r_{i,i+1}$ with rate $\mu_i$
    create local event $r^{(b)}_{i,i+1}$ with functional rate $f^{(b)}_{i,i+1}$ where
    $f^{(b)}_{i,i+1} = (state\ M_i \neq 0, 0) * \mu_i$
    create synchronising event $r^{(u)}_{i,i+1}$ with rate $\mu_i$
end for
create local event $r_{K,x}$ with rate $\mu_K$
create synchronising event $r^{(u)}_{K,x}$ with rate $\mu_K$
for $i = 2$ to $K$ do
    assign event $r_{i,i+1}$ in all departures of $M_i$ but last
    assign event $r_{i,i+1}$ in all arrivals of $M_{i+1}$ but last
    assign event $r^{(u)}_{i,i+1}$ in all arrivals of $M_{i+1}$ but last
    assign event $r^{(u)}_{i,i+1}$ in all departures of $M_i$
    assign event $r^{(u)}_{i,i+1}$ in all departures of $M_{i-1}$
    for $i = i - 2$ to $2$ do
        assign event $r^{(u)}_{i,i+1}$ in all departures of $M_j$ with functional probability
        $bM_i$ ($\forall M_i..M_{j+1}$ blocked), where $bM_i = (state\ M_i\ =\ B_i, 2)$
        assign event $r^{(u)}_{i,i+1}$ in all loops of $M_j$ with functional
        probability $!bM_i$ ($\exists M_i..M_{j+1}$ not blocked)
    end for
    assign event $r^{(b)}_{i,i+1}$ in the last arrival of $M_{i+1}$
end for
```

Figure 4. Algorithm 1: SAN model generation.

(HBF) (Stewart 1994). In fact, GTAexpress uses more memory than the PEPS2007 approach in storing the details of the underlying CTMC, but it allows the employment of a '*lite*' approach to obtain the numerical solution of a SAN model, i.e. the numerical solution is computed by a vector–matrix iterative multiplication.

The complexity[2] of the solution of SAN models in both PEPS2007 and GTAexpress is related to the model size, i.e. the number of states of the model. While in PEPS2007 the model size is expressed as the total number of states (called product state space – PSS), in GTAexpress the model size is expressed by the number of reachable state states (called reachable state space – RSS). It is important to recall that the use of SAN formalisms may include some unreachable states due to the Cartesian product of each of the automata states. A case in point is state (0,0; 0,2) in the three-station model considered above. For models with a large number of unreachable states (when PSS is much larger than RSS) the use of GTAexpress is likely to be much more efficient.

Nevertheless, the time spent to solve a SAN model is just partially estimated by the complexity, since both PEPS2007 and GTAexpress uses iterative methods to achieve the solution. These iterative methods perform a number of VDP multiplications which is dependent on the model's numerical parameters. In fact, the complexity of the VDP operation is linear to the model size and the interested reader will find further information in Czekster et al. (2011a). It is reasonable to assume that the complexity of the solution of all production line models presented in this paper can be estimated by the model size, expressed by the PSS for the PEPS2007 solution, and by the RSS for the GTAexpress solution.

Both PEPS2007 and GTAExpress were used to solve the SAN model shown in Figure 2 and determined the throughput to be 0.61333 jobs/unit time. This result is the same as the exact throughput obtained using the MARKOV software package associated with the text by Papadopoulos et al. (2009). These software tool packages were also used to compute the throughput of the symmetrical SAN model of Figure 2, which corresponds to a three-station balanced production line (mean service rate 1) with a perfectly reliable machine at each station but with $B_2 = 0$ and $B_3 = 1$ and the mean value of the throughput was found to be 0.61333 jobs/unit time, as expected. The same procedure was followed for the SAN model shown in Figure 3.

4.1. *Pilot experiments – Numerical results*

The throughputs of SAN models equivalent to the listed production line models are presented in Table 3. These results were computed using the PEPS2007 and GTAexpress software tools. In this table, the respective product state space (PSS), the reachable state space (RSS), memory size to store the model and the time required to compute the steady-state solution are given. All the production lines in Table 3 have a single perfectly reliable machine at each station with identical mean service rates $\mu_i = 1$ $(i = 1, \ldots, K)$ and buffer sizes as stated. The experiments were performed on Intel

Table 3. Throughput for different production line configurations.

Model							
K	B_i	PSS	RSS	Throughput	Method	Memory used	Time to solve
3†	B_2 =1	12	11	0.61333*	GTAExpress	4 KB	≈0 sec.
	B_3 =0				PEPS2007	3 KB	≈0 sec.
5‡	$i-1$	840	751	0.68289*	GTAExpress	76 KB	0.02 sec.
					PEPS2007	17 KB	0.10 sec.
5	6	6561	6319	0.81500*	GTAExpress	578 KB	0.24 sec.
					PEPS2007	69 KB	1.46 sec.
6	4	16,807	15,465	0.75515*	GTAExpress	1.5 MB	0.81 sec.
					PEPS2007	156 KB	5.36 sec.
11	0	59,049	17,711	0.42383*	GTAExpress	9.16 MB	68.80 sec.
					PEPS2007	588 KB	54.34 sec.
12	0	177,147	46,368	0.41956*	GTAExpress	34.14 MB	557.11 sec.
					PEPS2007	1.59 MB	235.12 sec.
13	0	531,441	121,393	0.41595*	GTAExpress	132.20 MB	76.02 min.
					PEPS2007	4.61 MB	17.37 min.
10	1	262,144	151,316	0.56002*	GTAExpress	19.68 MB	195.88 sec.
					PEPS2007	2.21 MB	291.56 sec.
6	8	161,051	155,760	0.83939*	GTAExpress	15.83 MB	12.96 sec.
					PEPS2007	1.30 MB	80.20 sec.
8	3	279,936	235,416	0.70258*	GTAExpress	26.82 MB	44.56 sec.
					PEPS2007	2.27 MB	204.05 sec.
7	5	262,144	242,047	0.77651*	GTAExpress	26.39 MB	27.49 sec.
					PEPS2007	2.11 MB	168.78 sec.
8	$i-1$	604,800	517,412	0.68166	GTAExpress	59.59 MB	106.49 sec.
					PEPS2007	4.92 MB	521.61 sec.
5	28	923,521	920,639	0.94383	GTAexpress	87.79 MB	136.95 sec.
					PEPS2007	7.39 MB	903.19 sec.
5	35	2,085,136	2,080,805	0.95401	GTAExpress	199.85 MB	6.86 min.
					PEPS2007	16.58 MB	43.58 min.
10	3	10,077,696	7,997,214	0.69319	GTAExpress	1.05 GB	2.99 hours
					PEPS2007	80.49 MB	4.94 hours
8	8	19,487,171	18,534,131	0.82958	GTAExpress	2.30 GB	1.22 hour
					PEPS2007	153.78 MB	6.46 hours
6	30	39,135,393	38,991,744	0.94430	GTAExpress	–	–
					PEPS2007	308.12 MB	16.24 hours
11	3	60,466,176	46,611,179	0.68973	GTAExpress	–	–
					PEPS2007	482.66 MB	408 hours

*The numerical results for these throughputs match the respective results obtained using the MARKOV software package associated with the text by Papadopoulos et al. (2009).

† This model corresponds to the SAN model presented in Figure 2.

‡ This model corresponds to the SAN model presented in Figure 3.

Xeon E5520 (Nehalem) processors, running at 2.27 GHz with 8 MB L3 cache. In addition, to the memory needed as indicated in Table 3, the steady-state solution requires the storage of a probability vector of size equal to the number of states in the RSS. In fact, the need to store probability vectors is the factor limiting the capability of determining the solution of very large production lines using the methodology previously described.

As can be seen from Table 3, whereas GTAExpress is often faster than PEPS2007 in deriving the exact solution of lines with up to about 20 M states, the analyst is required to use PEPS2007 for the solution of lines with a number of states greater than this. In the experience of the authors, the exact solution of serial production lines with up to 300 K states may be developed using the Markovian approach. The contribution of the use of the Kronecker descriptor methodology is the capacity to solve exactly serial production systems with states up to 18 M states in a reasonable time. For lines with a number of states in excess of 40 M the CPU time required is measured in days using the PEPS2007 S/W. Clearly the analyst must decide whether to use traditional simulation methods to obtain approximate results or to use the Kronecker descriptor methodology to obtain exact results when the time requirements of the latter exceed a reasonable time. It may be noted that lines with zero intermediate buffers have a reachable state space (RSS) considerably less than the product state space (PSS) for all lines.

5. Conclusions and further research

This paper has demonstrated that the SAN formalism has significant potential as an efficient alternative solution methodology to the more traditional approaches to the exact solution of production lines with a large number of states. In representing the system transitions by a Kronecker descriptor the impact of the well-documented state space explosion problem is less severe. In addition, the associated software tools of SAN (PEPS2007 and GTAExpress) provide exact steady-state solutions of quite large models (up to 4×10^7 RSS).

The paper developed, in detail, the equivalent SAN model for a three- and five-station production line with a single perfectly reliable machine at each station and inter-station buffers of finite size. We have presented in this paper an algorithm that automatically generates the equivalent SAN model for any K-station production line of the type analysed. These results led us believe that the integration of the algorithm with the existing SAN solvers would provide a single software tool of interest to researchers and practitioners alike. In fact, a prototype of such a tool was recently presented (Fernandes et al. 2011a). However, much work still has to be undertaken in order to deliver the solution of more complex production lines using the formalisms discussed above.

The main contribution of this paper is to demonstrate that, with respect to the exact steady-state solution of these serial exponential production lines, the use of the Kronecker descriptor methodology can accommodate conservatively systems with more than 100 times the number of reachable states that traditional Markovian methods can analyse exactly. It should be noted that this Kronecker descriptor methodology has been successfully applied to the exact analysis of computer and communication systems. That being said, an interesting future work would be to compare the numerical results of the SAN formalism presented here with other techniques such as that proposed by Tan (2003). These techniques, being quite distinct, and yet very efficient, it would be an interesting area for further research to test the limits of both.

With respect to future work, the Kronecker descriptor methodology solves both the exact transient (with specified initial condition) and the exact steady-state problem simultaneously. Thus, studies could be undertaken with respect to the transient solutions of production lines (Fernandes et al. 2011a). Production lines with different topologies could be investigated as each station including the servers and buffers may be represented by an automaton with transitions from one state to another with either constant or functional rates depending on the global state of the system. In this regard it may be possible to compute the exact transient and the exact steady-state solutions of much more complex production lines than those considered above. For example, lines with unreliable machines, merge/split, fork/join, stations with parallel machines, closed loops and assemble/disassembly sections of production lines may be amenable to exact analysis using the Kronecker descriptor methodology. In work of this nature it is often beneficial to the practitioner to present solutions in a manner that may easily be used. In future work the authors will keep this concept in mind. Hence, the development of an appropriate decision support system (DSS) at a future date might be a valuable area of research. The overall objective of this work is to be in a position to compute the throughput of very complex production lines directly by the use of the SAN methodology or by a combination of the use of the SAN methodology in association with the well-known aggregation or decomposition methods.

Acknowledgements

The order of authors is merely alphabetical. Paulo Fernandes is funded by CNPq Brazil (PQ 307284/2010-7). Afonso Sales receives grants from CAPES Brazil (PNPD 02388/09-0). Chrissoleon Papadopoulos for this research was co-financed by the European Union

(European Social Fund -– ESF) and Greek national funds through the Operational Program 'Education and Lifelong Learning' of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALES. Investing in knowledge society through the European Social Fund.

Notes

1. We consider two matrices stochastic equivalent when they represent the same stochastic process, i.e. they deliver the same steady state and transient solution.
2. The term 'complexity' is employed here in the computational sense, i.e. the amount of computational resources to achieve the solution Cook 1983. Specifically, we are interested in the processing resources, and, therefore, a value which is proportional to the time to obtain the solution. Usually, the time to achieve the solution is estimated as the number of floating point operations needed to compute the solution, and the complexity itself is expressed as a relation (e.g., linear, exponential) with a proportional value and the number of floating point multiplications.

References

Altiok, T. 1997. *Performance analysis of manufacturing systems*. New York: Springer.

Amoia, V., G. De Micheli, and M. Santomauro. 1981. "Computer-oriented formulation of transition-rate matrices via Kronecker algebra." *IEEE Transactions on Reliability* R-30 (2): 123–132.

Baldo, L., et al. 2005. "Performance models for master/slave parallel programs." *Electronic Notes in Theoretical Computer Science* 128 (4): 101–121.

Bariş, T., and S. Gershwin. 2011. "Modelling and analysis of Markovian continuous flow systems with a finite buffer." *Annals of Operations Research* 182 (1): 5–30.

Bellman, R. 1960. *Introduction to matrix analysis*. New York: McGraw-Hill.

Beounes, C. 1985. "Stochastic Petri net modeling for dependability evaluation of complex computer systems." In *Proceedings of the 1st international workshop on timed Petri nets*, 191–8. Washington, DC: IEEE Computer Society.

Birkhoff, G., and R.E. Lynch. 1984. *Numerical solution of elliptic problems*. Philadelphia, PA: SIAM.

Brenner, L., P. Fernandes, and A. Sales. 2005. "The need for and the advantages of generalized tensor algebra for Kronecker structured representations." *International Journal of Simulation: Systems, Science & Technology (IJSIM)* 6 (3/4): 52–60.

Brenner, L., et al. 2007. "PEPS2007 – Stochastic automata networks software tool." In *Proceedings of the 4th international conference on quantitative evaluation of systems (QEST 2007)*, 163–4. Washington, DC: IEEE Computer Society.

Brenner, L., et al. 2009. "Modelling Grid5000 point availability with SAN." *Electronic Notes in Theoretical Computer Science* 232: 165–178.

Brewer, J.W. 1978. "Kronecker products and matrix calculus in system theory." *IEEE Transactions on Circuits and Systems* 25 (9): 772–780.

Buchholz, P. 1992. "Numerical solution methods based on structured descriptions of Markovian models." In *Computer performance evaluation – Modelling techniques and tools*, edited by G. Balbo and G. Serazzi, 251–67. Amsterdam: Elsevier.

Buchholz, P. 1994. "A class of hierarchical queueing networks and their analysis." *Queueing Systems* 15 (1–4): 59–80.

Buchholz, P. 1999. "Structured analysis approaches for large Markov chains." *Applied Numerical Mathematics* 31 (4): 375–404.

Buchholz, P., and T. Dayar. 2003. "Block SOR for Kronecker structured representations." In *Proceedings of the 2003 international conference on the numerical solution of markov chains*, 121–43. New York: Springer.

Buzacott, J.A. 1972. "The effect of station breakdowns and random processing times on the capacity of flow lines with in-process storage." *AIIE Transactions* 4 (4): 308–313.

Buzacott, J.A., and D. Kostelski. 1987. "Matrix-geometric and recursive algorithm solution of a two-stage unreliable flow line." *IIE Transactions* 19: 429–438.

Chanin, R., et al. 2006. "Analytical modeling for operating system schedulers on NUMA systems." *Electronic Notes in Theoretical Computer Science* 151 (3): 131–149.

Cook, S.A. 1983. "An overview of computational complexity." *Communications of the ACM* 26 (6): 400–408.

Czekster, R. M., P. Fernandes, and T. Webber. 2009. "GTAexpress: A software package to handle Kronecker descriptors." In *Proceedings of the 6th international conference on quantitative evaluation of systems (QEST 2009)*, 281–2. Washington, DC: IEEE Computer Society.

Czekster, R.M., P. Fernandes, and T. Webber. 2011a. "Efficient vector-descriptor product exploiting time–memory trade-offs." *SIGMETRICS Performance Evaluation Review* 39 (3): 2–9.

Czekster, R.M., et al. 2011b. "Stochastic model for QoS assessment in multi-tier web services." *Electronic Notes in Theoretical Computer Science* 275: 53–72.

Dallery, Y., Z. Liu, and D.F. Towsley. 1994. "Equivalence, reversibility, symmetry and concavity properties in fork-join queueing networks with blocking." *Journal of the ACM* 41 (5): 903–942.

Davio, M. 1981. "Kronecker products and shuffle algebra." *IEEE Transactions on Computers* 30 (2): 116–125.

Donatelli, S. 1994. "Superposed generalized stochastic Petri nets: Definition and efficient solution." In *Application and theory of Petri nets*, edited by R. Valette. Berlin: Springer.

Donatelli, S. 1994. "Superposed stochastic automata: A class of stochastic Petri nets amenable to parallel solution." *Performance Evaluation* 18: 21–36.

Dotti, F. L., et al. 2005. "Modular analytical performance models for ad hoc wireless networks." In *Proceedings of the 3rd international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt 2005)*, 164–73. Washington, DC: IEEE Computer Society.

Dotti, F., P. Fernandes, and C. Nunes. 2011. "Structured Markovian models for discrete spatial mobile node distribution." *Journal of the Brazilian Computer Society* 17 (1): 31–52.

Farina, A.G., P. Fernandes, and F. M. Oliveira. 2002. "Representing software usage models with stochastic automata networks." In *Proceedings of the 14th international conference on software engineering and knowledge engineering*, 401–7. New York: ACM.

Fernandes, P., B. Plateau, and W.J. Stewart. 1998. "Efficient descriptor-vector multiplication in Stochastic Automata Networks." *Journal of the ACM* 45 (3): 381–414.

Fernandes, P., et al. 2011a. "Production line analysis tool (PLAT)." In *Proceedings of the 41st international conference on computers & industrial engineering (CIE41)*, 248–53. USC: Los Angeles.

Fernandes, P., et al. 2011b. "Performance evaluation of software development teams: A practical case study." *Electronic Notes in Theoretical Computer Science* 275: 73–92.

Gershwin, S.B., and O. Berman. 1981. "Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers." *AIIE Transactions* 13 (1): 2–11.

Gershwin, S.B., and I.C. Schick. 1981. "Modeling and analysis of three-stage transfer lines with unreliable machines and finite buffers." *Operations Research* 31 (2): 354–380.

Graham, A. 1981. *Kronecker products and matrix calculus with applications*. Ellis Howard.

Heavey, C., H.T. Papadopoulos, and J. Browne. 1993. "The throughput rate of multistation unreliable production lines." *European Journal of Operations Research* 68 (1): 69–89.

Hermanns, H., U. Herzog, and V. Mertsiotakis. 1998. "Stochastic process algebras – Between lotos and Markov chains." *Computer Networks and ISDN Systems* 30 (9/10): 901–924.

Hillier, F.S., and R.W. Boling. 1967. "Finite queues in series with exponential or Erlang service times – A numerical approach." *Operations Research* 15: 286–303.

Hillston, J. 1995. "Compositional Markovian modeling using a process algebra." In *Computations with Markov chains*, edited by W.J. Stewart, 177–96. Dordrecht: Kluwer Academic.

Hillston, J. 1996. *A compositional approach to performance modelling*. New York: Cambridge University Press.

Hillston, J., and L. Kloul. 2007. "Formal techniques for performance analysis: Blending SAN and PEPA." *Formal Aspects of Computing* 19 (1): 3–33.

Hunt, G.C. 1956. "Sequential arrays of waiting lines." *Operations Research* 4: 674–683.

Kaufman, L. 1983. "Matrix methods for queueing problems." *SIAM Journal on Scientific and Statistical Computing* 4 (3): 525–552.

Kemper, P. 1996. "Numerical analysis of superposed GSPNs." *IEEE Transactions on Software Engineering* 22 (9): 615–628.

Kronecker, L. 1865. "Über einige Interpolationsformeln für ganze Funktionen mehrerer Variabeln." In *L*, edited by H. Hensel, 133–41. Kronecker's Werke: Chelsea Publishing Company.

Levantesi, R., A. Matta, and T. Tolio. 2003. "Performance evaluation of continuous production lines with machines having different processing times and multiple failure modes." *Performance evaluation* 51 (2–4): 247–268.

Lynch, R.E., J.R. Rice, and D.H. Thomas. 1964. "Tensor product analysis of partial difference equations." *Bulletin of the American Mathematical Society* 70: 378–384.

Mitra, D., and I. Mitrani. 1991. "Analysis of a kanban discipline for cell coordination in production lines, II: Stochastic demands." *Operations Research* 39 (5): 807–823.

Neuts, M.F. 1981. *Matrix-geometric solutions in stochastic models*. Baltimore, MD: Johns Hopkins University Press.

Papadopoulos, H.T., C. Heavey, and M.E.J. O'Kelly. 1989. "Throughput rate of multistation reliable production lines with inter station buffers: (I) Exponential case." *Computers in Industry* 13 (3): 229–244.

Papadopoulos, H.T., C. Heavey, and M.E.J. O'Kelly. 1990. "Throughput rate of multistation reliable production lines with inter station buffers: (II) Erlang case." *Computers in Industry* 13 (4): 317–335.

Papadopoulos, C.T., et al. 2009. *Analysis and design of discrete part production lines*. New York: Springer.

Papadopoulos, H.T., and M.E.J. O'Kelly. 1989. "A recursive algorithm for generating the transition matrices of multistation series production lines." *Computers in Industry* 12: 227–240.

Plateau, B., 1984. De l'Evaluation du parallelism et de la synchronisation. Thesis (PhD). Orsay, Paris.

Plateau, B. 1985. "On the stochastic structure of parallelism and synchronization models for distributed algorithms." *ACM SIGMETRICS Performance Evaluation Review* 13 (2): 147–154.

Plateau, B., and J.M. Fourneau. 1991. "A methodology for solving Markov models of parallel systems." *Journal of Parallel Distributed Computing* 12: 370–387.

Plateau, B., J.M. Fourneau, and K.H. Lee. 1988. "PEPS: A package for solving complex Markov models of parallel systems." In *Proceedings of the 4th international conference on modelling tools and techniques for computer performance evaluation*, edited by R. Puijanger, 341–60. Amsterdam: Elsevier.

Regalia, P.A., and S.K. Mitra. 1989. "Kronecker products, unitary matrices and signal processing applications." *SIAM Rev.* 31 (4): 586–613.

Ricci, G., and T. Levi-Civita. 1900. "Methodes du calcul differentiel absolu et leurs applications (Methods of absolute differential calculus and their applications)." *Math. Ann.* 54 (1/2): 125–201.

Saad, Y. 2003. *Iterative methods for sparse linear systems*. 2nd ed.. Society for Industrial and Applied Mathematics.

Saad, Y., and M.H. Schultz. 1986. "GMRES: A Generalized Minimal RESidual algorithm for solving nonsymmetric linear systems." *SIAM Journal on Scientific and Statistical Computing* 7 (3): 856–869.

Stewart, W.J. 1994. *Introduction to the numerical solution of Markov chains*. Princeton, NJ: Princeton University Press.

Stewart, W.J. 2009. *Probability, Markov chains, queues, and simulation: The mathematical basis of performance Modeling*. Princeton, NJ: Princeton University Press.

Tan, B. 2003. "State-space modeling and analysis of pull controlled production systems." In *Analysis and modeling of manufacturing systems*, edited by S. Gershwin , et al., 363–98. Berlin: Springer.

Uysal, E., and T. Dayar. 1998. "Iterative methods based on splitting for stochastic automata networks." *European Journal of Operational Research* 110 (1): 166–186.