FACULDADE DE INFORMÁTICA
PUCRS - Brazil
http://www.inf.pucrs.br/

Modeling Exponential Reliable Production Lines using Kronecker Descriptors

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

Technical Report Series

Number 066
August, 2011

P. Fernandes and A. Sales

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Avenida Ipiranga, 6681 – Porto Alegre, RS – Brazil
Phone: +55 51 3320 3611. Fax: +55 51 3320 3621.
Emails: paulo.fernandes@pucrs.br, afonso.sales@pucrs.br

M.E.J. O'Kelly

Waterford Institute of Technology
Cork Road, Waterford, Ireland
Phone: +353 51 302 000.
Fax: +353 51 378 292.
Email: mejokelly@gmail.com

C.T. Papadopoulos

Aristotle University of Thessaloniki
Thessaloniki, GR-541 24 Greece.
Phone: +30 2310 99 74 66.
Fax: +30 2310 99 64 52.
Email: hpap@econ.auth.gr

Published by PPGCC – FACIN – PUCRS
Avenida Ipiranga, 6681
90619-900 – Porto Alegre, RS – Brasil

Abstract

This paper presents a solution procedure for production lines, based on a Markovian formulation with a Kronecker structured representation (sum of tensor products). Production lines with phase-type distributions for the processing and machine repair times and Erlang distribution times to failures of multiple machines in parallel at each workstation are traditionally modeled as Markovian continuous time discrete state processes. In this paper structured Markovian formalisms are used to manage the well-known state explosion problem associated with other methods of solution. Such formalisms combined with the Kronecker representation deliver impressive memory efficiency in storing very large models, *i.e.*, models with more than 10^9 states. The exact transient and stationary solutions of these models, stored in this fashion, may be obtained using very efficient existing software packages. In this paper, a three-station exponential production line with perfectly reliable machines is considered in order to demonstrate how the equivalent Stochastic Automata Network (SAN) model is derived from the corresponding Finite Capacity Queueing Network model. The throughput of this production line and of other single perfectly reliable machine production lines, with many more states, are derived using Kronecker based software packages (PEPS2007 and GTAexpress) and comparisons with known results are made.

Keywords: Markov chains; Structured Markovian models; Stochastic Automata Networks; Tensor (Kronecker) algebra; Production lines; Modeling

1 Introduction and Literature Review

Modeling formalisms are usually applied to describe real (and large) systems, capturing their behavior and computing performance indices. A well-known modeling formalism used for this purpose is *Markov Chains* [21]. Markov chain (MC) models are present in many domains such as chemistry, bioinformatics, economics, social sciences, to cite a few. MC models use simple primitives (*states* and *transitions*) to exemplify system's evolution and operational semantics. However, this model mapping in a non-structured manner leads to the state-space explosion problem, where the absence of a sophisticated (and efficient) solution in general becomes prohibitive the use of MC for many large systems.

Over recent years a number of structured modeling 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 [12] are certainly the 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 of this formalism were proposed, providing solution approximations and even some product-form solutions for the QN models. However, most real life systems do not satisfy the necessary conditions for product-type solutions, such as infinite capacity queues.

Other examples of structured modeling formalisms are PEPA - *Performance Evaluation Process Algebra* [11], (SG)SPN - *(Superposed Generalized) Stochastic Petri Nets* [9], SAN - *Stochastic Automata Networks* [17], among several other approaches.

The idea behind these approaches is to exploit the structure of the system to solve the underlying CTMC. Two approaches are given in the literature: (i) the Stochastic Automata Networks (SAN) approach originated by [17] and her colleagues, *e.g.*, [18] and [10]; and (ii) the hierarchical model approach with asynchronously interacting sub-models originated by [5, 6]. The SAN approach is used in this work and it is applied, with a brief explanation in Section 3, to production lines with a single perfectly reliable machine at each station and inter-station finite buffers. [7] demonstrated that there is a correspondence between these two approaches.

Tensor algebra is an effective and efficient algebra for manipulating products and sums of matrices of the type that are embedded in generator matrices of CTMCs. Early researchers, who observed this were [1], *a reliability model*, [14] and [12], *queueing models*, [2], *a reliability model*, and [17], *in modeling SAN*. [10] introduced the concept of the generalized tensor product to numerically solve SAN with functional transitions in contrast to constant transitions. The latter was achieved using sophisticated software tools, *e.g.*, PEPS2007 [3] and GTAexpress [8].

Regarding the numerical solution of structured analysis techniques, [7] summarized various iterative methods such as the *Power* method, the *Jacobi* method, projection methods such as the *Generalized Minimal Residual* (GMRES) algorithm and the *Arnoldi* method using the Krylov subspace, the *Gauss-Seidel* or *SOR*, the block Gauss-Seidel and the block SOR methods and approaches to accelerate convergence such as aggregation and disaggregation steps and preconditioning for both SANs and hierarchical models (see [21], [22], [19] and [20], among others, for a description of these numerical methods).

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. [13] examined a simplified model of a manufacturing system with kanban control and [7] modeled this system as a (SG)SPN.

The SAN formalism has an underlying CTMC represented by a set of tensor products (*i.e.*, by a *Kronecker* descriptor), instead of a single matrix used in the QN 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.

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, synchronizing events in SAN, or cooperations in PEPA).

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 analyzed using the SAN formalism based on a tensor (Kronecker) representation. We focus our attention in the usage of the SAN formalism, however, any other structured formalism could be employed without loss of generality. In Section 2, to illustrate the solution procedure, the queueing network (QN) model of a three station line is briefly described. In Section 3, we present the basic modeling primitives of the SAN formalism and the SAN equivalent model of this QN model, presenting also the algorithm for the equivalent SAN model generation. In Section 4, the throughput of this production line is given, as well as the throughputs of other production lines of the class being considered 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 behavior 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 3^{rd} station (last) in finished form. All jobs at each station are processed according to FIFO queueing discipline.

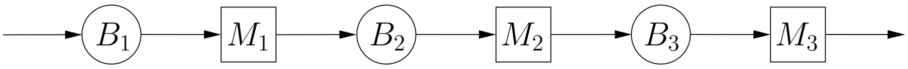


Figure 1: A three-station production line.

The assumptions of this Markovian model are: (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 don't fail; (v) the general rule that deliberate idleness at a station is not allowed applies; and (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 characterizes the *saturated* line of the saturation model. The fact that the last station is never blocked relates to the storage capacity for final products.

The system under consideration is modeled as a two-dimensional stochastic process $N(\mathrm{t}) = [N_1(\mathrm{t}),\ N_2(\mathrm{t})]$. See [16] and [15] 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 idle, if s_i=1, station i is busy and if s_i=2, station i is busy and blocking preceding station. 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.

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

(i) derivation of the states of the system. Two formulas were given in Papadopoulos, Heavey and O'Kelly (1989) for obtaining the number of states for equal and unequal buffer capacities, respectively;

(ii) ordering of the states. This process facilitates the determination of the structure of the transition matrix;

(iii) generation of the transition matrix (or the conservation matrix). The generating steps of a recursive algorithm are given in [16];

(iv) 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 Cathal Heavey and with appropriate instructions is available at the website: http://purl.oclc.org/NET/prodline associated with the text by [15] with the abbreviated name MARKOV.

Applying the above recursive algorithm 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 A of order 11, equal to the number of states, is obtained (see [16] and [15] for the details of the derivation):

$$A = \begin{vmatrix}
-\mu_1 & \mu_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\mu_1-\mu_3 & \mu_3 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\mu_1-\mu_3 & 0 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mu_1 & 0 & 0 & -\mu_1-\mu_2 & \mu_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mu_1 & 0 & 0 & -\sum_{i=1}^{3}\mu_i & \mu_3 & \mu_2 & 0 & 0 & 0 & 0 \\
0 & 0 & \mu_1 & 0 & 0 & -\mu_1-\mu_3 & 0 & \mu_2 & 0 & 0 & 0 \\
0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_1-\mu_2 & \mu_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\sum_{i=1}^{3}\mu_i & \mu_3 & \mu_2 & 0 \\
0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_3 & 0 & \mu_2 \\
0 & 0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_2 & \mu_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & -\mu_2-\mu_3
\end{vmatrix}$$

For the balanced case (with identical mean service rates, μ_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 equal to 0.61333.

3 The equivalent SAN model

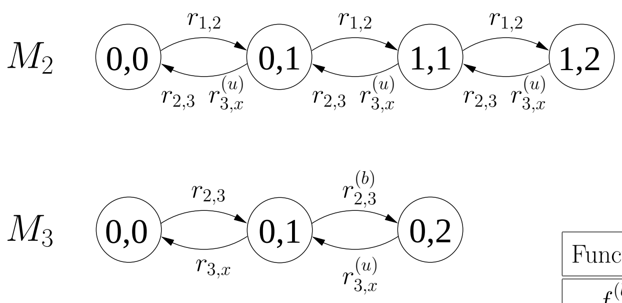
The SAN formalism basic idea is to represent a whole system by a set of subsystems where each subsystem may behave independently of the others, with occasional interdependent behavior between subsystems. Each subsystem is described as a *stochastic automaton*, composed by a set of local states and transitions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

Transitions of a SAN model are fired by *events* that may affect one single automaton (local event) or many automata (synchronizing events). Events have rates describing their frequency of occurrence and those rates can be *constant*, if the event always happens with the same frequency, or *functional* if the event frequency changes according to other automata current local states. The functional rates can be viewed as the evaluation of a expression of a non-typed C language, *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 (see [4] and [10]).

Based on the assumptions presented in Section 2 (K=3 stations in series 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 modeled 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. 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 arrival from the station M_1 to M_2. Local event $r_{3,x}$ (with rate μ_3) stands for the departure from station M_3 to exterior of the production line. *Synchronizing event* $r_{2,3}$ means the passage of jobs from M_2 to M_3 and it has rate μ_2.

Moreover, we additionally model two synchronizing events $r_{2,3}^{(b)}$ and $r_{3,x}^{(u)}$ that concern the status of link of station M_2 to M_3. Basically, these events indicate that M_3 became busy and blocking station M_2 ($r_{2,3}^{(b)}$) or station M_3 complete a service and unblocked M_2 ($r_{3,x}^{(u)}$). Note that both events are described in the same transition on automaton M_2. In this case, the transition in this automaton can be fired by the occurrence of one of those events. Additionally, event $r_{2,3}^{(b)}$ has a functional rate $f_{2,3}^{(b)}$ which is evaluated at each



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	Defition
$f_{2,3}^{(b)}$	$(state\ M_2\ !=\ 0,0) * \mu_2$

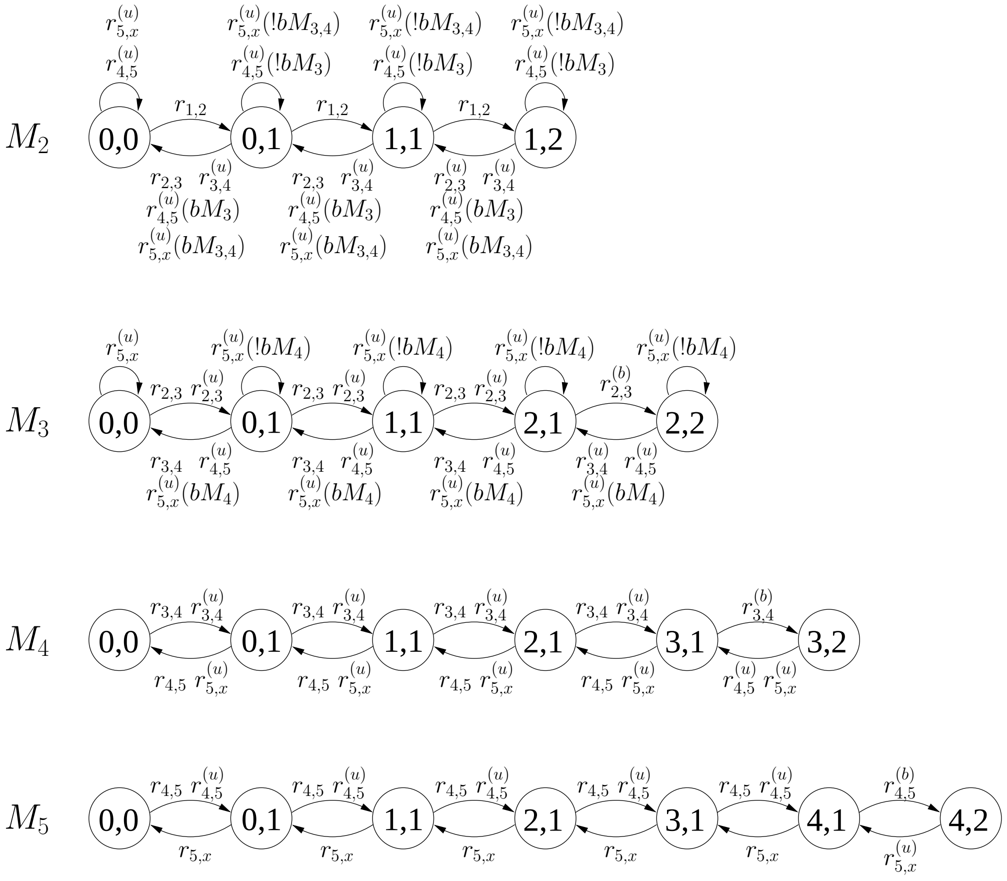
Figure 2: Equivalent SAN model to QN model with B_2=1 and B_3=0.

global state. The function $f_{2,3}^{(b)}$ delivers a μ_2 rate for global states where M_2 station is busy, and a zero rate, *i.e.*, event $r_{2,3}^{(b)}$ does not happen, for global states where server M_2 is empty (*i.e.*, M_2 is in state $0,0$). 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 model definition.

3.1 Algorithm for model generation

In order to illustrate the algorithm for SAN model generation, in Figure 3 we present an equivalent SAN model to the production line with five ($K = 5$) stations and buffer capacities $B_i = i - 1$, where $i = 2..K$. This model is general enough to present and explain the steps needed to the generation process of an equivalent SAN model to any production line.

Note that all states of automata M_2 and M_3 of the SAN model (Figure 3) have *loop transitions*. In the SAN formalism, if a same event can fire two or more transitions from a state, it is necessary to indicate a *probability* (or *proportion*) in relation to the occurrence of this event for each possible transition. In this case, for instance, the occurrence of $r_{4,5}^{(u)}$ happens with rate μ_4 and probability bM_3 (which changes the state of automaton M_2 from state $1,2$ to $1,1$) or probability $!bM_3$ (remaining at the same state $1,2$, *i.e.*, the loop transition). Moreover, in this case, the probability described for the event is not a constant probability, but a *functional probability.* Then bM_3 is described by a function defined by *state* $M_3 == 2,2$ (*i.e.*, the probability is equal to value 1 if the evaluation of the function is satisfied). On the other words, the probability bM_3 allows the occurrence of the event if automaton M_3 is in state $2,2$. On the other hand, $!bM_3$ indicates the *negation* (or *complement*) of the probability bM_3 and hence it is evaluated to 0 (avoiding the


M_2
$r^{(u)}_{5,x}$ $r^{(u)}_{4,5}$
$r^{(u)}_{5,x}(!bM_{3,4})$ $r^{(u)}_{4,5}(!bM_3)$
$r^{(u)}_{5,x}(!bM_{3,4})$ $r^{(u)}_{4,5}(!bM_3)$
$r^{(u)}_{5,x}(!bM_{3,4})$ $r^{(u)}_{4,5}(!bM_3)$
$r_{1,2}$
$r_{1,2}$
$r_{1,2}$
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
$r^{(u)}_{5,x}$
$r^{(u)}_{5,x}(!bM_4)$
$r^{(u)}_{5,x}(!bM_4)$
$r^{(u)}_{5,x}(!bM_4)$
$r^{(u)}_{5,x}(!bM_4)$
$r_{2,3}$ $r^{(u)}_{2,3}$
$r_{2,3}$ $r^{(u)}_{2,3}$
$r_{2,3}$ $r^{(u)}_{2,3}$
$r^{(b)}_{2,3}$
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
$f^{(b)}_{2,3}$	$(state\ M_2 \ ! =\ 0,0) * \mu_2$
$f^{(b)}_{3,4}$	$(state\ M_3 \ ! =\ 0,0) * \mu_3$
$f^{(b)}_{4,5}$	$(state\ M_4 \ ! =\ 0,0) * \mu_4$
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: Equivalent SAN model to the production line with $K = 5$ stations and B_i=$i - 1$ (where $i = 2, 3, 4, 5$).

occurrence of the event for the transition where $!bM_3$ is defined) when bM_3 is evaluated to 1. This modeling strategy of usage of complementary functional probabilities in the SAN formalism allows to easily describe the behavior of events that can be fired or not depending the current states of other automata. Note also that the functional probability $bM_{3,4}$ is a composition of probabilities bM_3 *and* bM_4. For this example, $bM_{3,4}$ is evaluated to 1 if both automata M_3 and M_4 are in the *blocking* states, *i.e.*, if automaton M_3 is in state $2, 2$ *and* automaton M_4 is in state $3, 2$.

In Algorithm 1, we present the algorithm for the equivalent SAN model to a production line of K stations with buffers located in front of the stations. This algorithm can be explained in general terms at three blocks: (i) the creation of automata and their respective states; (ii) the creation of events of the model; and (iii) assignment of the transitions between states of the automata.

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 synchronizing 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
    $f^{(b)}_{i,i+1} = (state\ M_i\ !=\ 0, 0) * \mu_i$
    create synchronizing event $r^{(u)}_{i,i+1}$ with rate $\mu_i$
end for
create local event $r_{K,x}$ with rate $\mu_K$
create synchronizing 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
```

The creation of automata M_i (where $i = 2..K$) with $B_i + 3$ states, where B_i represents the buffer capacity of the buffer located in front of station i, are described between lines 1 and 3 in Algorithm 1. The creation of the events used in the SAN model are defined between lines 5 and 11. And finally the assignment of the transitions between states are defined between lines 12 and 23 in Algorithm 1.

Using this algorithm, it is possible to compute the throughput of the production line described by the SAN model. In order to obtain this throughput, it is necessary to analyze the probability vector that describes the stationary solution of the model and to integrate the probabilities of the states whose the last station is not empty, multiplying by the service rate μ_K of this station, *i.e.*, the throughput is computed by the integration function described in the SAN model defined by $(state\ M_K\ != \ 0, 0) * \mu_K$.

4 Software tools and results

PEPS2007 [3] is one of software tool packages developed for modeling and solving SAN models, which represents the underlying CTMC in a compact format and uses tensor algebra to compute the reachable states of a SAN model, given an initial state, and to deliver stationary and transient exact solutions. The numerical solution of Kronecker representations applied by PEPS2007 consists on multiplying a vector by a non-trivial structure in a process known as Vector-Descriptor Product (VDP).

On the other hand, GTAexpress [8] is a software tool that uses a sophisticated approach to compute the underlying CTMC and store it on a sparse format, *i.e.*, a sparse matrix stored in a Harwell-Boeing Format (HBF), [21]. Hence, GTAexpress spent more memory than the PEPS2007 approach to store the underlying CTMC, however it allows to employ 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.

PEPS2007 and GTAexpress were used to solve the SAN model shown in Figure 2 that result in a 0.61333 throughput. This result is the same as the exact throughput obtained using the MARKOV software package associated with the text by [15]. In addition, these software tool packages were 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 expected value of 0.61333 was obtained found for the solution of matrix A - Section 2, page 7).

4.1 Pilot experiments

In Table 1 the throughputs, computed using the PEPS2007 and GTAEXPRESS software tools, of SAN models equivalent to the listed production line models are given together with the respective product state space (PSS), the reachable state space (RSS), memory size to store the model in KBytes and the time in seconds required to compute stationary solution. All the production lines in Table 1 have a single perfectly reliable machine at each station with identical mean service rates μ_i=1 (i=1, ..., K) and buffer sizes as stated. The experiments were performed on a machine with two Intel Xeon E5520 (Nehalem) Quad-core processors, where each processor runs at 2.27 GHz and 8 MB L3 shared by all cores. Besides the memory needed to store the model indicated in Table 1, the stationary solution computation requires the storage of a probability vector with the RSS size. Actually, the storage of such probability vector is the limit to perform the solution of very large production lines. However, this probability vector describing the stationary solution allow us to compute additional information about the model as, for instance, buffers average occupation and server response times.

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 well documented state space explosion problem is coped, but in addition SAN has associated software tools (PEPS2007 and GTAEXPRESS) which provide transient and stationary (steady state) exact solutions of very large models.

Actually, we may model any production line in order to compose really large SAN models, since each station (server and buffer) will be represented by an automaton, and each possible passage from a station to another will be described by a set of events with functional rates for blocked behaviors. Presently, we were able to solve models around 130 million states, which corresponds for instance to a production line of 18 stations with no intermediate buffer.

The paper developed, in detail, the equivalent SAN model to a three station production line with a single perfectly reliable machine at each station and inter-station buffers of finite size (1 and 0). The computed throughput of this line and of other lines of similar type but with larger associated state

Table 1: Throughput for different production line configurations.

Model		*PSS*	*RSS*	*Throughput*	*Method*	*Memory*	*Time to*
K	B_i					*used*	*solve*
3	0	9	8	0.56410	GTAexpress	4 KB	≈ 0 sec.
					PEPS2007	3 KB	≈ 0 sec.
4	$i-1$	120	110	0.68674	GTAexpress	16 KB	≈ 0 sec.
					PEPS2007	7 KB	≈ 0 sec.
5	6	6,561	6,319	0.81500	GTAexpress	578 KB	0.24 sec.
					PEPS2007	69 KB	1.46 sec.
6	4	16,807	15,456	0.75515	GTAexpress	1.5 MB	0.81 sec.
					PEPS2007	156 KB	5.36 sec.
7	2	15,625	12,649	0.65702	GTAexpress	1.31 MB	0.97 sec.
					PEPS2007	152 KB	6.10 sec.
7	$i-1$	60,480	52,331	0.68169	GTAexpress	5.44 MB	5.05 sec.
					PEPS2007	527 KB	30.79 sec.
8	3	279,936	235,416	0.70258	GTAexpress	26.82 MB	44.56 sec.
					PEPS2007	2.27 MB	204.05 sec.
8	$i-1$	604,800	517,412	0.68166	GTAexpress	59.59 MB	106.49 sec.
					PEPS2007	4.92 MB	521.61 sec.
12	0	177,147	46,368	0.41956	GTAexpress	34.14 MB	557.11 sec.
					PEPS2007	1.59 MB	235.12 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	411.84 sec.
					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	1.70 day
18	0	129,140,163	14,930,352	0.402752	GTAexpress	—	—
					PEPS2007	0.99 GB	33 days

spaces using PEPS2007 and GTAexpress are the same as the exact solutions obtained by traditional methods. In addition the authors propose to expand the work to the determination of the exact throughput of more complex production lines such as multiple machine station lines, unreliable machine lines and special purpose sections of production lines such as merge/split, fork/join, closed loop, and assembly/disassembly elements.

We have also presented in this paper an algorithm that automatically generates the equivalent SAN model for any K-station production line with a single perfectly reliable machine at each station and inter-station finite buffers.

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.

References

[1] V. Amoia, G. D. Micheli, and M. Santomauro. Computer-Oriented Formulation of Transition-Rate Matrices via Kronecker Algebra. *IEEE Transactions on Reliability*, R-30(2):123–132, 1981.

[2] C. Beounes. Stochastic Petri Net Modeling for Dependability Evaluation of Complex Computer Systems. In *Proceedings of the International Workshop on Timed Petri Nets*, pages 191–198, Washington, DC, USA, 1985. IEEE Computer Society.

[3] L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. PEPS2007 - Stochastic Automata Networks Software Tool. In *Proceedings of the 4th International Conference on Quantitative Evaluation of SysTems (QEST 2007)*, pages 163–164, Edinburgh, UK, September 2007. IEEE Computer Society.

[4] L. Brenner, P. Fernandes, and A. Sales. 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, February 2005.

[5] P. Buchholz. A hierarchical view of GCSPNs and its impact on qualitative and quantitative analysis. *Journal of Parallel and Distributed Computing*, 15(3):207–224, July 1992.

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

[7] P. Buchholz. Structured Analysis Approaches for Large Markov Chains. *Applied Numerical Mathematics*, 31(4):375–404, 1999.

[8] R. Czekster, P. Fernandes, and T. Webber. GTAexpress: a Software Package to Handle Kronecker Descriptors. In *Proceedings of the 6th International Conference on Quantitative Evaluation of Systems (QEST 2009)*, pages 281–282, Budapest, Hungary, September 2009. IEEE Computer Society.

[9] S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In R. Valette, editor, *Proceedings of the* 15^{th} *International Conference on Applications and Theory of Petri Nets*, pages 258–277. Springer-Verlag Heidelberg, 1994.

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

[11] J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996.

[12] L. Kaufman. Matrix Methods for Queuing Problems. *SIAM Journal on Scientific and Statistical Computing*, 4(3):525–552, 1983.

[13] D. Mitra and I. Mitrani. Analysis of a Kanban Discipline for Cell Coordination in Production Lines, II: Stochastic Demands. *Operations Research*, 39(5):807–823, 1991.

[14] M. F. Neuts. *Matrix geometric solutions in stochastic models, an algorithmic approach*. John Hopkins University Press, Baltimore, 1981.

[15] C. Papadopoulos, M. O'Kelly, M. Vidalis, and D. Spinellis. *Analysis and Design of Discrete Part Production Lines*. Springer, New York, NY, USA, 2009.

[16] H. Papadopoulos, C. Heavey, and M. O'Kelly. Throughput rate of multistation reliable production lines with inter station buffers: (I) Exponential Case. *Computers in Industry*, 13(3):229–244, December 1989.

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

[18] B. Plateau, J. Fourneau, and K. Lee. PEPS: a package for solving complex markov models of parallel systems. In *Proceedings of the 4th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, pages 341–360, Mallorca, Spain, September 1988. Elsevier Ed.

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Second Edition, Society for Industrial and Applied Mathematics, 2003.

[20] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal RESidual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[21] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, Princeton, NJ, USA, 1994.

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