Universitá degli studi di Torino
Department of Computer Science

HANDLING LARGE STATE SPACES IN TRANSIENT ANALYSIS OF MARKOVIAN PROCESSES

Ph.D Thesis

Candidate: Alessio Angius

Advisor: András Horváth
Co-advisor: Gianfranco Balbo

Submitted on March 2013

To my family.

Thanks for making this possible.

Acknowledgements

As first, I want to thank András Horváth, I have been lucky to work under his supervision since from the Master thesis. Besides, I consider him a dear friend.

Secondly, I would like to thank the entire research group for their personal and professional support, they created the best environment to learn and improve as a researcher.

During the Ph.D., I have had the possibility to be guest of Verena Wolf and Holger Hermanns in Saarbrucken. It has been an amazing experience for which I will be always grateful to them.

Of course, I have also to thank my parents (Giampiero and Lina), my brother Fabio, my grandparents (Giovanna Rosa, Piera, Paolo, Stefano), the uncles, the aunts, and the little cousin Michele.

Last but not least, I want to dedicate a special thanks to Rosaria for her love and patience during these years.

Contents

1
Abstract

This dissertation is placed in the context of *systems performance evaluation*. In particular, we deal with models based on stochastic processes; thus, given a mathematical abstraction of a system, our goal is to provide an exhaustive quantitative analysis of the system properties in form of probability measures.

Stochastic processes can be treated from two points of view:

- transient analysis: which aims to observe the evolution of the process during a finite time interval,
- steady state analysis: which considers the behaviour of the process once it gets stable.

In this thesis, we focus on a special class of processes, namely, *Markov chains*. Markov chains have been used for a long time to describe systems having random behaviours, such as distributed computer networks, telecommunication systems, and manufacturing or logistics infrastructures.

The main reason of their success is that, in principle, they are computationally friendly. However, standard analysis techniques developed for Markov chains are precluded in case of huge state spaces.

This problem occurs more often than not because the number of states considered by Markov chains grows exponentially with the number of modelled objects; this phenomenon is known as *state space explosion*.

1.1 Brief summary of the state of the art

In order to make doable the analysis of large state spaces, a lot of effort has been dedicated to the development of both analytical and numerical methods. In this direction, several efficient analysis techniques have been proposed to compute the stationary measures of specific models.

The most notable example is placed in the context of *queuing networks* where a family of algorithms exploits the fact that, under particular assumptions, the steady state probabilities enjoy the so-called *product form*. Roughly speaking, this means that the global distribution of the system can be computed as a product of measures computed by considering its components as isolated from the rest of the net. [56, 41, 13, 35, 36, 38, 37, 39, 28, 44, 11, 79, 26].

Moving to a more general context, another idea to manage complexity and largeness of state spaces is the exploitation of model or system structure in the description and analysis of the underlying Markov chain. A similar idea was proposed by Plateau for the first time in [75, 76] where the generator matrix of a Markov chain resulting from a network of *stochastic automata* was represented in a highly compact form by exploiting the features of its structure. The approach has been also adopted for *Generalized Stochastic Petri Nets* (GSPNs) [58, 31, 32] and *Stochastic Process Algebras* (SPAs) [19].

In contrast to stationary analysis, transient analysis is more complicated because it enjoys properties such as product forms or flow equivalences in very few situations. Exact computations are possible only for moderate size models or for very particular situations, like networks of infinite server queues [46, 67, 18, 17]. In these networks clients are independent of each other and this leads to the fact that the number of clients at a station follows a Poisson distribution whose mean can be calculated by a set of *ordinary differential equations*.

By putting aside the explicit description of the state space, system of ordinary differential equations has been massively used to analyze large scale systems from the point of view of their deterministic approximation. The outcomes of these analyses, called *mean-field* or *fluid* approximations, are often trajectories that can be seen as the approximate average behaviour of the model [61, 62, 83, 54, 48]. Higher order *moment closure* techniques exist as well by providing an approximation not only for the mean but also for higher order moments and joint moments [34, 80, 68, 24].

Methods based on aggregation can also be developed, see, for example, [15]. There are few techniques that maintain the original state space of the model and, as a consequence, allow to calculate distributions and not only moments. In [47] an iterative method is suggested to solve the time-dependent Kolmogorov equations of the model but this approach suffers from the state space explosion problem. A memory efficient approach is proposed instead in [91, 21] where the number of ODEs that describe the transient behaviour is decreased by assuming a limited dependency structure among the queues of the network. As last, in [77] Buchholz and Sanders describe an approach that combines the randomization approach for transient analysis of Markov chains with a representation of probability vectors as Kronecker products of small component vectors.

1.2 Objective of the thesis

This dissertation deals with the problem of transient analysis in case of large state space. The goal of the thesis is dual; in particular, we aim to:

1. show that there are cases in which an accurate analysis of the structure of the model can avoid the use of approximate techniques,

2. propose a new approximate technique for the computation of transient probabilities.

In reference to the first objective, we point out that although the use of approximations is appealing because of their lower computational cost, their use should be the last attempt to perform the analysis of a model. In fact, an accurate qualitative analysis of the model structure often suggests an alternative use of standard techniques and, consequently, the preservation of the exactitude of the results. In the first part of the thesis, we provide an example of such situations by showing a case where we were able to build an exact framework to analyze large systems by applying relatively small changes to standard techniques. Specifically, we propose the results obtained in [5, 3] where we considered manufacturing production lines and biochemical systems as *reward models*. This expedient allowed us to study models whose analysis is precluded by using common techniques. This part of the thesis is based on the work of Telek and Rácz described in [84]. However, we extended the original technique to multi-reward models in such a way that joint moments are also computed.

On the contrary, the second part of this dissertation, based on the works described in [4, 6], considers those situations in which the only possible solution is to renounce the exactitude of the results. In this context, the scarcity of techniques able to provide an approximation of transient probabilities motivated us to investigate a possible new strategy to perform such analysis. Moreover, most moment based approximations such as mean-field and fluid analysis are, more often than not, based on the fact that the number of considered objects is large. Indeed, this is not the common situation in case of shared resources where, typically, several users compete for a limited number of items, or in case of blocking mechanisms that are often considered as binary conditions.

For the reasons above, we developed a novel technique which takes inspiration from the concept of *transient product form*. In particular, we assume that the transient probabilities of the model can be decomposed into a *quasi product form*. This assumption simplifies the dependency structure of the model and leads to a relatively small set of ordinary differential equations that can be used to compute an approximation of the transient probabilities. The approximation allowed us to analyze huge state spaces (beyond 10^{16}) with a low computational cost. Moreover, we were able to reproduce accurately features that, by definition, cannot be represented through approximation based only on moments; such as bi-modal distributions and extremely rare events.

1.3 Structure of the thesis

Metaphorically speaking, the structure of this dissertation is a path that goes from standard techniques to approximate methods. Chapter 2 introduces basic concepts about stochastic processes and then focuses on Markov chains in order to explain the notions required for the reading of this thesis.

Subsequently, the dissertation splits in two parts. The first part is composed of two chapters: Chapter 3 which explains the notion of Markov reward models, provides the formal definition of *accumulated reward* and *completion time*, and shows how their higher order moments can be recursively computed; Chapter 4 which illustrates the use of the framework on real case studies.

The second part is composed of three chapters: Chapter 5 gives a formal definition of transient product form and describes a first attempt to use product form as approximation of the transient probabilities of Markov chains; Chapter 6 provides an extension of the previous approach, namely, quasi product form, which can be used to approximate a larger family of models; Chapter 7 provides several numerical illustrations and shows the accuracy of the method. Finally, in Chapter 8 we summarize our conclusions and list possible future extensions of the works described.

2
Introduction to stochastic processes

2.1 Stochastic processes

In this section we will briefly explain what stochastic processes are and how they are used in the context of system performance evaluation.

We will illustrate basic definitions about stochastic processes by providing also some criteria that allow to coarsely classify them. Then we will conclude the section by listing the measures that constitute the elementary units of any analysis performed through stochastic processes.

2.1.1 What is a stochastic process?

Stochastic processes are probabilistic models of systems that evolve in a random way. More formally, a stochastic process (s.p.) is a collection of random variables $\{X(\alpha), \alpha \in \mathcal{T}\}$, indexed by a parameter α taking values in the parameter set $\mathcal{T}$ which usually represents time. Each random variable $X(\alpha)$ takes values in a set $\mathcal{S}$ known as *state space* of the stochastic process.

Stochastic processes can be classified through several sophisticated criteria, here we propose the three most common.

As first, we consider the time domain $\mathcal{T}$: if we assume to observe a system at discrete time points $\alpha = 0, 1, 2 \ldots$ then the set $\mathcal{T}$ is defined in $\mathbb{N}$ and the stochastic process $\{\overline{X}(n), n \in \mathbb{N}\}$ is called *discrete-time* s.p.[1]. Else-wise if $\mathcal{T}$ coincides with $\mathbb{R}$, we will refer at the collection $\{X(t), t \in \mathbb{R}\}$ as a *continuous-time* stochastic process.

A similar reasoning can be done about the domain of the state space; hence, if $X(\alpha)$ can assume only discrete (continuous) values then we call the process *discrete-space* (*continuous-space*). The distinction above leads to the second criteria.

The third and last criteria deals with the *transitions* of a process. Commonly, the transitions of a process are described by mean of functions that given a possible evolution of the s.p up to a given time instant n (called *trajectory* from now-on) provide, in a probabilistic way, those states in which the process can evolve at time $n + 1$[2].

It is clear that the complexity of a stochastic process, defined as the computational effort required to compute the probability of the trajectories that the process can follow, grows with the complexity of its transition function.

In this work we will deal with a particular family of stochastic processes which is characterized by the simplicity of its transitions: the *Markovian* processes. A process is classified as *"Markovian"* if the following property holds: *"Given a trajectory $\{\overline{X}(0) = x_0, \overline{X}(1) = x_1, \ldots, \overline{X}(n) = x_n\}$ the probability that we will find the system in a particular state y at time $n + 1$ depends only on the state $\overline{X}(n)$"*.

This property leads to an intuitive interpretation that can be informally summarized by the sentence: *"The history of the process does not count, the future depends only on the present"*.

As we will see in the next chapters, even if the Markovian property is restrictive, Markovian processes are massively used in the context of performance evaluation. This is because non-Markovian processes are much harder to deal with computationally.

2.1.2 Stochastic process analysis

Stochastic processes are commonly used to predict system measures of interest that cannot be computed directly because of physical constraints, time or financial costs, etc.

[1]From now-on, we will use a line on the top of random variables indexed by integers.

[2]In sake of simplicity we assumed a discrete-time stochastic process.

Thus, the investigations that a system analyst can perform on a stochastic model are heterogeneous and more or less complicated according to the purpose of the study. Some simple examples are:

- *the probability to find more than* 10 *customers in a queue;*
- *the mean waiting time of an internet service user;*
- *the variance of the time required to finish a job given that at time n it was only half complete.*

Furthermore, disciplines such as *model checking* increase drastically the expressiveness of the properties that a system analyst can formulate on stochastic models (see for example [9, 20, 52, 33]).

Nevertheless, any analysis passes through at least one common step that is independent from the measures of interest: the choice between *transient period* and *steady-state* (known also as *transient* and *limiting* behaviour, respectively).

The study of a given measure Γ during the transient period corresponds to give an answer to the question *"How does Γ behave at time t?"*, whereas the study of the steady state of Γ considers the case of t approaching infinity.

Roughly speaking, the transient period is the phase of the process history affected by the initial state whereas the steady-state (if exists and the process is ergodic) is the stable state where the impact of the initial state on Γ is completely disappeared.

The choice of one instead of the other does not affect only the meaning of the results, but it determines also the choice of the numerical method and consequently the complexity of the problem. As we will see, transient behaviours are computationally difficult and analytically intractable in a large number of cases due to the fact that the transient period lacks of properties that in steady-state hold.

As last, it is worth to introduce the measures that, in the major part of the cases, compose the performance evaluation targets:

Distribution of the stochastic process

Have knowledge of the process distribution means being able to know the probability of finding the process in a given state $s \in \mathcal{S}$ at a certain time instant. In many cases, if the computation of the process distribution is a feasible task, then almost all the measures of interest can be derived (more or less) directly from it without much additional effort. The computation of the distribution is often not cheap from a computational point of view; hence, when it is possible, this computation is bypassed.

State-based functions

A state-based function is a function $f(s)$, $s \in \mathcal{S}$, which associates a numerical value to each state of the s.p.. Typically, these functions represent quantities such as throughputs of servers, number of customers in a queue, boolean properties. These quantities can be analyzed by calculating their distribution or through their *moments*.

Definition 1 *Given the probability distribution of a random variable X taking values in a state space $\mathcal{S}$ and a function $f : \mathcal{S} \to \mathbb{R}$, the nth moment of $f(X)$ is defined as*[1]*:*

$$E[f(X)^n] = \sum_{s\in\mathcal{S}} f(s)^n Pr(s) \quad n \geq 1$$

where $Pr(s)$ denotes the probability that X equals s.

It is frequently preferred to normalize higher moments ($n \geq 1$) according to the first because in this way the quantities relate only to the spread and shape of the distribution, rather than its location.

Such measures take the name of *central moments* and, by definition, have the form

$$E[(f(X) - E[f(X)])^n] = \sum_{s\in\mathcal{S}} (f(s) - E[f(X)])^n Pr(s) \quad n \geq 1.$$

Despite their definitions, the computation of these measures does not always require the explicit computation of the distribution, see for instance, ([61, 62] or [85, 86] where some approximations are presented.

Typical moments of interest are the *expectation* (first moment) and the *variance* which corresponds to the second central moment.

Nevertheless in the rest of the dissertation we will provide reasons which explain why the knowledge of higher moments can be important.

[1]Assuming that the state space is countable, otherwise the summation has to be substituted with an integral.

First Passage Time

In many applications it is important to know when the system reaches a particular set of states. The measure that provides this information takes the name of *first passage time* and is defined as:

$$T = min\{n \geq 0 : X(n) \in B\}$$

where B is the subset of the state space in which we are interested. The random variable T typically represents the time until the system fails, or the time until it stops or the time until it acquires other specific attributes. Several studies can be performed on the r.v. T; typical examples are: the probability that it is finite, its moments, its distribution, etc.

Rewards

Rewards can be seen as the dynamic counterpart of state-based functions since they capture how the stochastic process behaves instead of where the process is.

Stochastic processes model the evolution of systems in time; thus, it is natural that a system can incur costs or generate rewards depending on states it visits and how long it sojourns there or how it changes states. Rewards are the tool that allows to perform this kind of analysis. A reward can be accumulated, decreased or lost according to specific system policies.

As we will see in the following chapters, rewards can be analyzed from the point of view of their distribution as well as from the one of their moments.

The majority of the problems about stochastic processes can be reduced to the measures listed above [60], hence, the handling of these tools allows to deal with more complex measures that are compositions of the simpler problems listed above.

2.2 Markov chains

The aim of this section is to introduce the fundamental concepts about Markov chains by providing the basic definitions that will be used in the rest of the dissertation.

We will also exploit this section to introduce the notation, the graphical representation, and some examples about Markov chains.

Moreover, we will give a first mention of numerical methods for Markov chains whose description will be deepened in the following chapters.

2.2.1 Discrete-time Markov chains

Definition 2 *A stochastic process is said to be a* discrete-time Markov chain *(DTMC) if the following property holds:*

$$Pr\{\overline{X}(n+1) = x_{n+1}|\overline{X}(n) = x_n, \overline{X}(n-1) = x_{n-1}, \ldots, \overline{X}(0) = x_0\} = Pr\{\overline{X}(n+1) = x_{n+1}|\overline{X}(n) = x_n\}, \forall x_i \in \mathcal{S} \tag{2.1}$$

where $\mathcal{S}$ is a countable state space.

The property above, known as discrete-time Markovian property, allows to define one step probabilities of making a transition from state i to state j at time step n. Let $p_{i,j}(n)$ denote them, their definition is given by the conditional probabilities:

$$p_{i,j}(n-1) = Pr\{\overline{X}(n) = j|\overline{X}(n-1) = i\}, \forall i, j \in \mathcal{S} \tag{2.2}$$

with the condition that $\sum_{j \in \mathcal{S}} p_{i,j}(n-1) = 1$ for every i belonging to $\mathcal{S}$.

Transition probabilities are conveniently represented through a matrix $\mathbf{P}(n) = [p_{i,j}(n)]_{|\mathcal{S}|\times|\mathcal{S}|}$, $n \in \mathbb{N}$, called *transition probability matrix.*

The transition probability matrix governs the evolution on the time of a DTMC according to the following equation:

$$\pi(n) = \pi(n-1)\mathbf{P}(n-1) \tag{2.3}$$

where $\pi(n)$ is a vector representing the probability distribution at time n in such a way that its ith entry $\pi_i(n) = Pr\{\overline{X}(n) = i\}$, $1 \le i \le |\mathcal{S}|$.

It is straightforward to construct a matrix $\mathbf{P}(m, n)$ which represents the probability to be in state j at time n given that at time m the process was in state i, $0 \le m \le n$. The following equalities derive from basic matrix properties

$$\begin{aligned} \pi(n) &= \pi(0)\mathbf{P}(0, n-1)\mathbf{P}(n-1) \\ &= \pi(0)\mathbf{P}(0)\mathbf{P}(1, n) \end{aligned} \tag{2.4}$$

If the probability matrix does not change on the time, i.e. $\mathbf{P} = \mathbf{P}(n)$ for every n, then the process is said to be *time-homogeneous* and the following recursive equalities hold:

$$\begin{aligned} \pi(n) =& \pi(0)\mathbf{P}^n \\ =& \pi(n-1)\mathbf{P} \end{aligned} \tag{2.5}$$

The steady-state distribution π of a DTMC is defined as

$$\pi = \lim_{n\to\infty} \pi(n) \tag{2.6}$$

In the case of time-homogeneous chains, if such limit exists, π corresponds to the solution of the following system of linear equations:

$$\begin{cases} \pi\mathbf{P} = \pi \\ \sum_{\forall i \in \mathcal{S}} \pi_i = 1 \end{cases} \tag{2.7}$$

2.2.2 Continuous-time Markov chains

Given a countable state space $\mathcal{S}$, consider a continuous-time stochastic process $\{X(t),\ t \in \mathbb{R}\}$ behaving in such a way that when the process arrives to a state i at time t, it stays there for a random amount of time and then jumps to a new state $j : j \neq i$ with a certain probability. Such a process has a bi-variate discrete-time counterpart that is composed of a set of random variable pairs $\{(\overline{X}(n), \overline{S}(n)) : n \geq 0\}$ where:

1. $\overline{X}(n) \in \mathcal{S}$: represents the nth state reached by the process,
2. $\overline{S}(n) \in \mathbb{R}$: is the time when the nth state is reached by the process,
3. $(\overline{X}(0), \overline{S}(0))$ is the initial state and $\overline{S}(0)$ is equal to zero.

Figure 2.1 shows a possible trajectory of $\{X(t),\ t \in \mathbb{R}\}$ which is piecewise constant, providing also a quick glimpse of how the continuous process is connected to the discrete one. More formally, the relation among the processes is defined as follows

$$X(t) = \overline{X}(n), \quad t \in [\overline{S}(n), \overline{S}(n+1)).$$

The time that the process spends in the ith state takes the name of *sojourn time* and is defined as $\overline{Y}(i+1) = \overline{S}(i+1) - \overline{S}(i)$. In sake of simplicity, in the following we focus on the time homogeneous case.

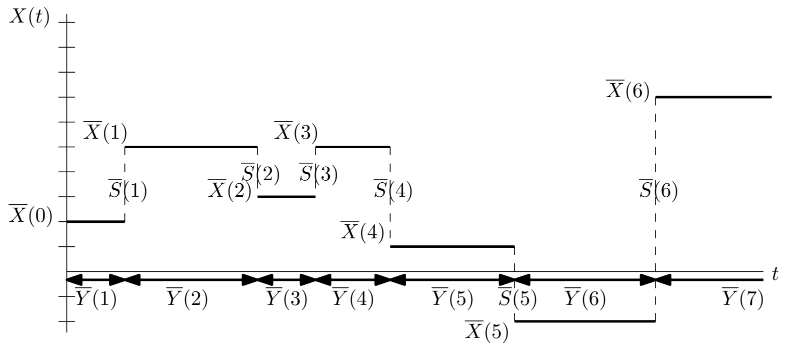


Figure 2.1: Representation of a continuous-time process

Definition 3 *We refer to the process* $\{X(t),\, t \in \mathbb{R}\}$ *as a* time homogeneous continuous-time Markov chain *(CTMC) if its discrete counterpart satisfies the following properties:*

- *the sojourn time and the probability of the next state are independent of each other;*
- *given any time instant t, the probability to move from i to j is governed by a discrete-time Markov chain (called embedded DTMC);*
- *the sojourn time is not affected by the history of the system prior to the time t, hence each r.v. $\overline{Y}(n)$ is exponentially distributed.*[1]

These independence assumptions can be described mathematically as

$$\begin{aligned} Pr\{\overline{X}(n+1) = x_{n+1}, \overline{Y}(n+1) > y | \overline{X}(n) = x_n, \overline{Y}(n) = y_n, \ldots, \overline{X}(0) = x_0\} \\ = Pr\{\overline{X}(n+1) = x_{n+1}, \overline{Y}(n+1) > y | \overline{X}(n) = x_n\} \\ = p_{x_n, x_{n+1}} e^{-q_{x_n} y} \qquad (2.8) \end{aligned}$$

where q_{x_n} is the parameter of the exponential distribution of the sojourn of the state x_n. Another consequence of Definition 3 is the so-called *continuous-time Markov property* which is defined as

$$Pr\{X(t+s) = j | X(s) = i, X(u) : 0 \leq u \leq s\} = Pr\{X(t+s) = j | X(s) = i\} \qquad (2.9)$$

[1]We refer the reader to [81] where the properties of the exponential distribution are deepened.

As in the discrete case, the transition probabilities can be represented through a stochastic matrix $\mathbf{H}(t, t+s) = [p_{i,j}(t, t+s)]_{|\mathbb{S}|\times|\mathbb{S}|}$ where $p_{i,j}(t, t+s)$ denotes the probability to find the process in state j at $t+s$ given that it was in the state i at time t.

Definition 4 *The infinitesimal generator matrix $\mathbf{Q} = [q_{i,j}]_{|\mathbb{S}|\times|\mathbb{S}|}$ defines the rates with which a CTMC can move from state i to state j. In formula, it corresponds to*

$$\mathbf{Q} = \lim_{\Delta\to 0} \frac{\mathbf{H}(t, t+\Delta) - \mathbf{I}}{\Delta} \tag{2.10}$$

It is possible to prove that the entries of the infinitesimal generator are structured in such a way that the elements off the diagonal are greater or equal to zero whereas those on the diagonal are specified as

$$q_{i,i} = -\sum_{\forall j\neq i} q_{i,j}.$$

As a consequence, each row of $\mathbf{Q}$ sums to zero.

The terms of equation (2.8) are related with those of the infinitesimal generator as follows

1. the sojourn time in state i is exponentially distributed according to the parameter $q_i = -q_{i,i}$,
2. the transition probabilities $p_{i,j}$ of the embedded DTMC arise from the ratio $\frac{q_{i,j}}{q_i}$.

Also in the continuous-time setting the transition probability matrices are decomposable into matrix products, leading to the following properties:

Property 1 *Given three time instants s, u, t such that $s \leq u \leq t$, the following property holds:*

$$\mathbf{H}(s, t) = \mathbf{H}(s, u)\,\mathbf{H}(u, t) \tag{2.11}$$

Property 2 *For every s, t, $0 \leq s \leq t$ the transition matrix $\mathbf{H}(s, t)$ is differentiable both on t and s according to the* Chapman-Kolmogorov *equations:*

$$\frac{\partial \mathbf{H}(s, t)}{\partial t} = \mathbf{H}(s, t)\,\mathbf{Q} \tag{2.12}$$

$$\frac{\partial \mathbf{H}(s, t)}{\partial t} = \mathbf{Q}\mathbf{H}(s, t). \tag{2.13}$$

More precisely, equation (2.12) is called *"forward"* whereas (2.13) takes the name of *"backward"*. The introduction of an initial distribution $\pi(0)$ into the Chapman-Kolmogorov equations completes the characterization on time of a CTMC,

$$\frac{\partial \pi(0)\mathbf{H}(0,t)}{\partial t} = \frac{d\pi(t)}{dt} = \pi(0)\mathbf{H}(0,t)\,\mathbf{Q} = \pi(t)\mathbf{Q}, \tag{2.14}$$

leading to an *ordinary differential equation* system (ODE system) composed of $|\mathcal{S}|$ equations. Each equation has the form

$$\frac{d\pi_i(t)}{dt} = \sum_j q_{j,i}\pi_j(t). \tag{2.15}$$

Time-homogeneity has two important consequences:

- the transition matrix $\mathbf{H}(t, t+s)$ is equivalent to a *matrix-exponentiation*[1]. In formula:
$$\mathbf{H}(t,t+s) = e^{\mathbf{Q}s} = \sum_{k=0}^{\infty} \frac{(\mathbf{Q}s)^k}{k!}. \tag{2.16}$$
- if the CTMC is *irreducible* and *recurrent* then its equilibrium distribution corresponds to the null-solution of the ODE system described by equation (2.15) (we remind the reader interested in the definitions of irreducibility and recurrence to [60]). Thus, casting the probabilities to sum to one, the following linear equation system arises:
$$\begin{cases} \pi\mathbf{Q} = \mathbf{0} \\ \sum_{i\in\mathcal{S}} \pi_i = 1. \end{cases} \tag{2.17}$$

2.2.3 Numerical computation of transient distributions

In this section we will briefly introduce some methods that are commonly used to analyze the transient behaviour of both discrete and continuous-time Markov processes.

Since the comprehensive discussion of numerical methods for Markov chains is beyond the scope of this dissertation, we will limit ourselves to describe those techniques that are the most important for the analysis presented in this dissertation.

For the readers interested to deepen the topic, we suggest [81, 60].

[1]The relation holds when the infinitesimal generator is uniformizable. Such property will be introduced in the next section.

2.2.3.1 Discrete-time

Equation (2.3) shows that the solution of DTMC consists of repeatedly multiplying the probability distribution vector obtained at step $n-1$ with the corresponding transition matrix to obtain the probability distribution at step n.

The application of the formula in general does not pose problems but its complexity grows quadratically in regard of the number of states of the process and cannot be reduced without additional assumptions on the model.

However, when n is large and the number of states is relatively small (limited to few thousands) some savings in computational time can be obtained by exploiting properties of linear algebra.

A trivial example is the iterative squaring of the time-homogeneous transition matrix $\mathbf{P}$ in order to construct the matrix $\mathbf{P}\left(0, 2^j\right)$ by computing only j matrix product. Note that even if the matrix $\mathbf{P}$ is sparse at the beginning, it tends to loose its sparsity during the computation of $\mathbf{P}^n$, as a consequence this approach is not appropriate for large sparse Markov chains.

Another common technique to analyze the transient period of a DTMC consists in the study of the *z-transform* (also called *generating function* of the process)

$$\mathbf{P}^*(z) = \sum_{i=0}^{\infty} z^i \mathbf{P}^i, \tag{2.18}$$

where z is a complex number.

In many case $\mathbf{P}^*(z)$ cannot be easily computed but if the number of states is low or the chain is highly structured, it can be inverted to obtain the explicit expression for $\pi(n)$. The series in equation (2.18) converges whenever $|z| < 1$. Then,

$$\begin{aligned}\mathbf{P}^*(z) &= \mathbf{I} + \sum_{i=1}^{\infty} z^i \mathbf{P}^i \\ &= \mathbf{I} + z\mathbf{P}\mathbf{P}^*(z). \qquad (2.19)\end{aligned}$$

Hence,

$$\mathbf{P}^*(z) = [\mathbf{I} - z\mathbf{P}]^{-1}.$$

Once computed the matrix $\mathbf{P}^*(z) = [p^*_{i,j}(z)]$ we have that $p^*_{i,j}(z)$ is a rational function of z that can be expanded in such a way that can be inverted by the method of partial fractions.

Generating functions have the advantage of providing a compact way to represent the transient distribution of stochastic processes but their computation is often complicated and become quickly unfeasible at the growing of the state space. Moreover, the inversion of the function is often critical from a numerical point of view.

However, due to its properties, that can be exploited to perform more complex investigations, the theoretical study of the z-transform is important independently of the numerical method that will be used in practice to analyze the process. This last sentence will be clearer in the next chapter where we will show that higher order moments can be derived directly from the generating function of the process.

2.2.3.2 Continuous-time

From a computational point of view transient analysis of CTMC carries more problems than its discrete counterpart. The main reasons are that:

- as shown in equation (2.16), the transient period of CTMCs is connected to a matrix exponentiation that, at least in principle, involves the evaluation of infinite series;
- by construction of infinitesimal generators, the Chapman-Kolmogorov equations are not numerically stable because the computation involves values of different magnitudes that can be also negative.

Hence, transient analysis has to be performed carefully.

A first intuitive way to solve the problem is the integration of the Chapman-Kolmogorov ODE system (2.15) by using common ODE solution techniques such as *Euler method* and *Runge-Kutta* methods (see [45]). These techniques work in such a way that, given a first-order differential equation $y' = f(t, y(t))$ and an initial condition $y(t_0) = y_0$, the solution is a differentiable function $y(t)$ such that

$$y(t_0) = y_0, \quad \frac{d}{dt}y(t) = f(t, y(t)).$$

In our context, the solution $y(t)$ corresponds to the distribution vector $\pi(t)$ and the function $f(t, y(t))$ is simply $\pi(t)\mathbf{Q}$.

Roughly speaking, all the procedure to solve ODE systems split the time interval $[0, t]$ in k parts to integrate the curve on time. The literature about ODE solvers is so vast that is not possible to list the pros and cons of every technique but, in general, they do not offer sound solution against numerical instabilities and are particularly sensitive to ill-conditioned infinitesimal generators.

On the other hand, the main pro of ODE numerical integration is that it is one of the few techniques able to analyze time inhomogeneous CTMCs which is the case where $f(t, y(t)) = \pi(t)\mathbf{Q}(t)$.

Another technique to perform the transient analysis of CTMCs is through their generating functions, as it happens for the discrete case. In fact, similar results holds for continuous-time r.v. as well.

In continuous-time setting the generating function, called *Laplace transform*, has the form

$$\mathbf{H}^*(0, s) = \int_0^\infty e^{-st}\mathbf{H}(0, t)dt$$

and, by following a path similar to that used for the discrete case, it is possible to prove that the Laplace transform of a CTMC satisfies the equation

$$\mathbf{H}^*(0, s) = [s\mathbf{I} - \mathbf{Q}]^{-1}.$$

Unfortunately, also this expression has the same limitations of the DTMC case, hence, it is not appropriate for large systems.

Note that none of the above two techniques deal explicitly with the matrix exponentiation that is, at least in principle, all what we need to compute the transient period of any CTMC.

This is because the computation of matrix exponentials is in itself a general and massively investigated research area and several methods have been developed over the years to compute it (see [69] where nineteen different methods are described).

From a theoretical point of view, any of them is able to solve our task; however, most of them suffer from numerical instability.

The method that fits perfectly in the context of CTMCs is the so-called *uniformisation* method introduced in [57].

Uniformisation (called also randomisation) is based on a truncated Taylor series expansion of the matrix exponential, has a quantifiable error, involves only vector-matrix operations, and requires only two vectors and a sparse matrix of size $|\mathcal{S}| \times |\mathcal{S}|$ to perform the computation.

Definition 5 *A CTMC with rate matrix* $\mathbf{Q}$ *is uniformizable if the* $max_i\{-q_{i,i}\} < \infty$.

A uniformizable continuous-time Markov chain can be decomposed into a DTMC and a Poisson process (PP). The DTMC takes into account the transitions of the CTMC while the PP provides the distribution of the number of transitions occurring in a given a time interval.

The transition matrix of the DTMC arises from $\overline{\mathbf{P}} = \frac{1}{\alpha}\mathbf{Q} + \mathbf{I}$ where $\mathbf{I}$ is the identity matrix and $\alpha \geq max_i|q_{i,i}|$. Thus, we have that

$$\begin{aligned} &Pr\{X(t) = j|k \text{ transitions in } [0,t]\} = \\ &\qquad = \sum_{i\in\mathcal{S}} \pi_i(0)p_{i,j}(0,k) \\ &\qquad = \sum_{i\in\mathcal{S}} Pr\left\{X(t) = i|k-1 \text{ transitions in } [0,t]\right\} p_{i,j} \qquad (2.20) \end{aligned}$$

By construction, $\overline{\mathbf{P}}$ is a stochastic matrix, hence it guarantees numerical stability.

Based on this decomposition the distribution at time t, can be written as

$$\begin{aligned} Pr\left\{X(t) = x\right\} = &\sum_{k=0}^{\infty} Pr\left\{X(t) = x|k \text{ transitions in } [0,t]\right\} \cdot \\ &Pr\left\{k \text{ transitions in } [0,t]\right\} \end{aligned}$$

where

$$Pr\{k \text{ transitions in } [0,t]\} = \frac{(\alpha t)^k}{k!}e^{-\alpha t}. \qquad (2.21)$$

The infinite summation can be truncated after k iterations with a controlled error γ in such a way that

$$\gamma \leq \sum_{i=k+1}^{\infty} \frac{(\alpha t)^i}{i!}e^{-(\alpha t)}$$

Hence, the computation of the distribution at time t by using uniformisation has the same complexity of a discrete transient analysis of $\overline{\mathbf{P}}$ up to time k, where k is the number of steps required to bound the error below γ. However, the complexity of the analyses remains dramatically affected by the cardinality of the state space justifying the aim of this thesis.

In sake of completeness, let us list some considerations about uniformization. A CTMC is always uniformizable if its state space is finite. This is a trivial consequence of the fact that the exit rates has to be finite for each state. On the other hand, in case of infinite state spaces and level dependent rates, the property does not hold. In this situation, a common escamotage is the truncation of the state space by considering only those states that have a not negligible probability mass to be reached. The truncation can be done in two possible ways:

- *a priori*: by exploiting the qualitative properties of the model,
- *dynamically*: by updating step by step the set of states having the major part of the probability mass.

Usually, the dynamic strategy is based on the concept of *sliding windows* [50] and allows to save time during the computation. As last, some techniques able to handle time inhomogeneous chains have been developed in [2, 88, 87].

2.2.4 Graphical representation

The probability matrix (infinitesimal generator) of a discrete-time (continuous-time) Markov chain can be represented graphically as a directed graph having one node for each state in $\mathcal{S}$ and a directed arc from node i to j every time $p_{i,j} > 0$ ($q_{i,j} > 0$). For instance:

Example 1 *Suppose to model the accumulated score of a game based on repeated die rolls. The game starts from a null score and stops when the score reaches Z points. The rules are such that a player can:*

- *gain a point every time the result of the roll is equal to* 5 *or* 6*,*
- *loose a point if the launch gives a* 1 *or a* 2*,*
- *stay at the same score with a* 3 *or a* 4 *.*

As additional assumptions, the score cannot be negative and the number of rolls is unlimited.

It should be obvious that this process can be represented through a time-homogeneous discrete-time Markov chain having the following transition ma-

trix

$$\mathbf{P} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \cdots & 0 & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \ddots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}.$$

where the states are ordered in such a way that they goes from 0 *(on the top of the matrix) to the target Z (last row of the matrix). The transition diagram representing such a matrix is depicted in Figure 2.2*

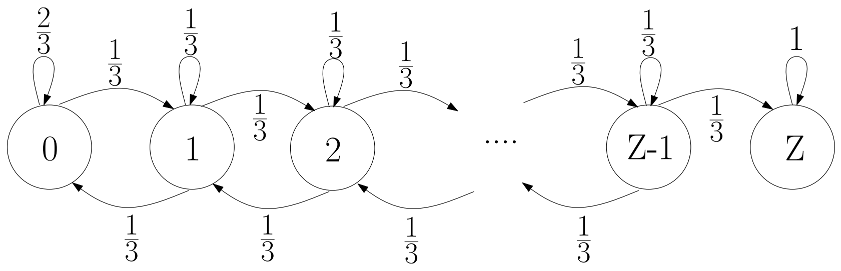


Figure 2.2: Example of DTMC - transition diagram

Example 2 *Assume to model the production of a manufacturing machine taking into account that the production can be interrupted because of machine failures and re-established through maintenance.*

Furthermore, let λ_1, λ_2 and λ_3 denote the mean time required to repair the machine once it gets broken, the mean failure time, and the mean time with which a new item is produced, respectively. Moreover, let us assume that all times are exponential. As a consequence, we have that the described machine behaves as a CTMC whose transition diagram is infinite and corresponds to that depicted in Figure 2.3.

2.3 Formalisms for stochastic processes

Formalisms, such as queuing networks, Petri nets and process algebras, provide intuitive tools to describe stochastic processes in a compact way.

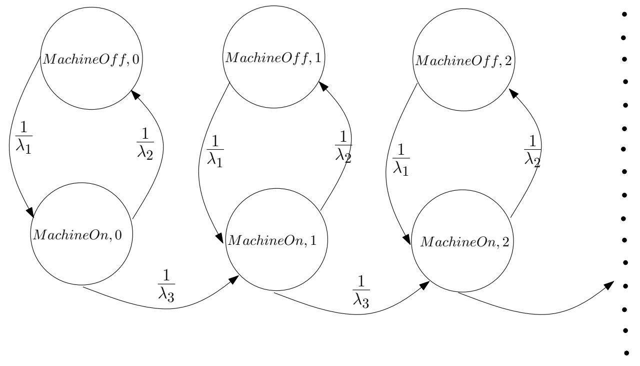


Figure 2.3: Manufacturing machine - CTMC

Several formalisms have been developed and extended during the years due to the fact that their introduction has been one of the key points of the success of stochastic modeling in the context of performance evaluation.

In this dissertation we will use two of them: queuing networks and stochastic Petri nets.

Queuing networks are convenient because some properties of Markov chains match with particular classes of these networks. However, the description of complex blocking mechanisms becomes clumsy with them. Hence, in those cases we will use Petri nets.

2.3.1 Queuing networks

Queuing networks are one of the first formalisms designed to represent stochastic processes. They are based on the concepts of *service stations* and *customers*.

A service station can have one or more independent servers working in parallel. When a customer arrives to a station, it waits if all the servers are busy; otherwise it is served and leaves the station after the service.

Commonly, a service station is defined through a standard notation

$$A/B/X/Y/Z,$$

proposed for the first time by Kendall in [59], where:

- A describes the interarrival distribution, such as exponential (commonly denoted with M), Erlang (Erl), Phase-type distributions (PH), General (G).

- B describes the distribution of the service times.
- X is the number of independent servers.
- Y is the maximum capacity of the waiting room and can assume values from zero to infinity. When the number of customers in queue reaches the threshold Y no more customers are accepted.
- Z is the queue discipline, it determines how the customers in the queue are selected for the service. Typical disciplines are First Come First Served (FCFS), Last Come First Served (LCFS), Processor Sharing, Random Order.

Stations are connected through *routing probabilities* determining how the customers move inside the system after a service in a station.

If exponential distributions (or distribution decomposable into exponential phases) govern both the arrival processes and the service times then the queuing system can be represented through a CTMC in such a way that, given a network composed of a set of M stations $\{s_1, s_2, \ldots, s_M\}$, the state of the CTMC is defined by the number of customers in each queue at time t. Therefore the r.v. representing the state of the CTMC is a vector of M integers $X(t) = |X_1(t), X_2(t), \ldots, X_M(t)|$ where the ith entry is associated with the station s_i. Let:

- $\lambda_i(x)$ be the rate of the arrivals to the ith station when the system is in the state x,
- $\mu_i(x)$ be the service rate of the ith station when the system is in the state x,
- $r_{i,j}(x)$ be the probability with which a customer moves to the jth station after being served at station s_i when the system is in the state x,
- $r_{i,0}(x)$ be the probability that a customer leaves the system after a service at station s_i when the system is in the state x.

The construction of the infinitesimal generator of the CTMC is straightforward. Given two states of the system x and y, the rate with which the chain moves from the first to the second is described by the following relation

$$q_{x,y} = \begin{cases} \lambda_i(x) & y = |x_1, \ldots, x_i + 1, \ldots, x_M| \\ \mu_i(x) r_{i,j} & y = |x_1, \ldots, x_i - 1, \ldots x_j + 1, x_M| \\ \mu_i(x) r_{i,0} & y = |x_1, \ldots, x_i - 1, \ldots, x_M|. \end{cases} \tag{2.22}$$

A network is called *closed* if the number of customers inside the system is constant, meaning that both departures from or arrivals to the system are not possible. Typically, in the case of closed networks the maximum number of customers inside a queue is equal to the number of customers inside the system. In these cases the state space $\mathcal{S} = \left\{ x = |x_1, ..., x_M| \,\middle|\, \sum_{i=1}^{M} x_i = N \right\}$ is composed of $\binom{N+M-1}{M-1}$ states, where N is the number of customers inside the system.

On the contrary, we will refer to a network as *open* if every customer that enters the system will leave it eventually. In this case the system state space grows over a number of dimensions equal to the number of stations.

Example 3 *Let us consider the simple network depicted in Figure 2.4 representing a server connected to two devices. Jobs arrive to the server with a constant rate λ_1 and compete for the resources of the system.*

After each service at the first station, the job can leave the system with probability $r_{1,0}$ or use one of the two devices with probabilities $r_{1,2}$ and $r_{1,3}$, respectively. Jobs leaving the two devices go back to the server. Furthermore, assume that the first station is a $M/M/\infty$ whereas the other two have only one server and finite waiting room capacities Y_2 and Y_3.

Then the network can be represented through the CTMC corresponding to the transition diagram depicted in Figure 2.5.

Note that the graph has been truncated after the third arrival at the first station; from now-on, when we will need to truncate an infinite transition diagram, we will summarize arcs pointing to states that have not been depicted (and those that came from them) by using one unlabeled arc.

2.3.2 Stochastic Petri nets

Petri nets were proposed in 1962 by Carl Adam Petri [74] and are a formalism for the description of concurrency and synchronization.

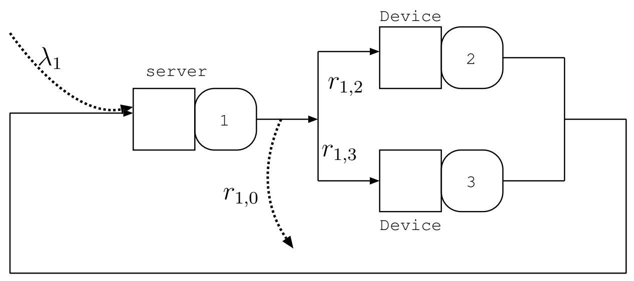


Figure 2.4: Open central server - queuing network

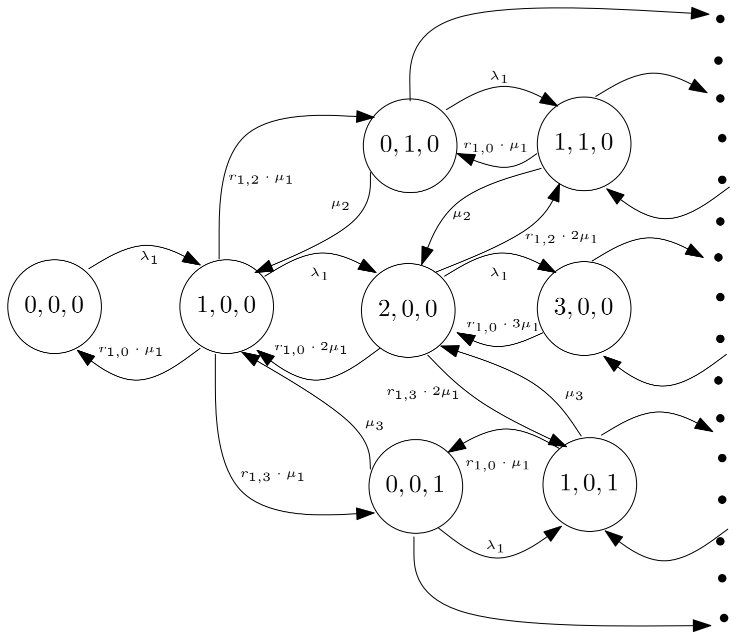


Figure 2.5: Open central server - CTMC graph

Since first described by Petri, they have became popular due to their well-founded analysis techniques and intuitive graphical representation. A Petri-net (PN) comprises the following components:

- *places*: drawn by circles. Places model conditions or objects, e.g., a program variable.
- *transitions*: drawn by rectangles. Transitions model activities which

change the values of conditions and objects.

- *arcs*: specifying the interconnection of places and transitions thus indicating which objects are changed by a certain activity. Each arc is associated with a multiplicity that describes the conditions under which a certain activity can occur or the outcome of an activity.
- *tokens*: drawn by black dots. Tokens represent the specific value of the condition or object, e.g., the value of a program variable.

Petri nets are bipartite graphs, hence we may only connect a place to a transition or vice versa.

The state of a PN, called *marking*, corresponds to the amount of tokens in each place, so it can be represented by a vector of integers. Tokens move among places through the *firing* of transitions, in such a way that each arc that goes from a place s to a transition t defines the number of tokens that will be removed from s after the firing of t and, in a similar way, the outgoing arcs of a transition t define the number of tokens that will be received by places after the firing of t. More formally:

Definition 6 *A Petri net is defined as* 5-tuple, $PN = \{P, T, I^-, I^+, \mathcal{M}_0\}$ *where:*

- $P = \{s_1, s_2, \ldots, s_M\}$ *is a finite and non-empty set of places,*
- $T = \{t_1, t_2, \ldots, t_R\}$ *is a finite and non-empty set of transitions,*
- I^-, $I^+ : P \times T \to \mathbb{N}_0$ *are the backward and forward incidence functions of transitions, respectively,*
- $\mathcal{M}_0$ *is the initial marking.*

The functions I^-, I^+ are conveniently described by introducing the vectors $i_n^- = |i_{n,1}^-, ..., i_{n,M}^-|$ and $i_n^+ = |i_{n,1}^+, ..., i_{n,M}^+|$, where every entry $i_{n,m}^-$ ($i_{n,m}^+$) represents the number of tokens removed from (inserted to) the mth place by the firing of transition t_n.

The sets composed of the transitions having a non-negative entry in i_n^- and i_n^+ are called *Pre* and *Post* set of transition t_n and denoted by ${}^\bullet t_n = \{s_m | i_{n,m}^- > 0\}$ and $t_n^\bullet = \{s_m | i_{n,m}^+ > 0\}$, $1 \leq n \leq R$, respectively.

Then, the overall effect of transition n corresponds to a vector $e_n = i_n^+ - i_n^-$ and, hence, given a marking $x = |x_1, x_2, \ldots, x_M|$, the transition t_n can occur if $x \geq i_n^-$ and its occurrence produces a new marking $x' = x + e_n$.

The set of reachable markings, possibly after the firing of several transitions, from a given initial marking $\{x : \mathcal{M}_0 \rightsquigarrow x\}$ takes the name of *reachability set*; similarly, the graph having as nodes the elements of the reachability set and as arcs $x \to y$ those transitions able to generate the marking y from the marking x is called *reachability graph*.

Example 4 *As an example consider once more the manufacturing machine described in Example 2. Such a system can be easily represented as a Petri net having three places and three transitions. As depicted in Figure 2.6, the modeling of the failure/repair cycle is described by two transitions, $Fail$ and $Repair$, that move the unique token, representing the state of the machine, from place $MachineOff$ to $MachineOn$ and vice-versa.*

When the machine is in the operative state ($MachineOn$), it is enabled to produce through the transition $Product$ which adds a token into the place $AmountOfProd$ every time it fires.

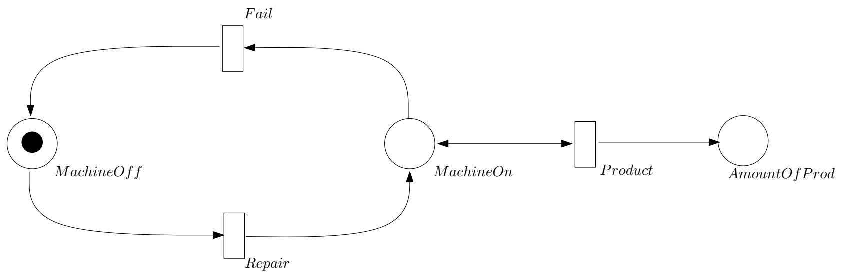


Figure 2.6: Manufacturing Machine - Petri net

As three places are involved, the marking of the PN is a triple $x = |x_1, x_2, x_3|$ describing the number of tokens present in places $MachineOff$, $MachineOn$ and $AmountOfProd$, respectively.

Then the effects of the transitions over the places are:

$$\begin{aligned} e_1 &= |1,0,0| - |0,1,0| = |1,-1,0| \\ e_2 &= |0,1,0| - |1,0,0| = |-1,1,0| \\ e_3 &= |0,1,1| - |0,1,0| = |0,0,1|. \end{aligned}$$

As defined by Petri, PNs provide a sound and effective framework to analyze qualitative properties of concurrent systems (see [10, 14]). However, they are unable to provide any quantitative information. For this reason Molloy and Natkin [70, 71] defined stochastic Petri nets (SPN) by adding a set $\Lambda = \{\lambda_1(x), \ldots, \lambda_R(x)\}$ to the original definition. All the firing times are exponentially distributed according to the parameters defined by the functions $\lambda_i(x)$, $1 \leq i \leq R$. As a consequence, the reachability graph is isomorphic to a CTMC in such a way that the entries of its infinitesimal generator have the form:

$$q_{x,y} = \begin{cases} \sum_{\substack{n\,:\,y = x + e_n \wedge \\ x \geq i_n^-}} \lambda_n(x) & x \neq y \\ \sum_{y \neq x} q_{x,y} & \textit{otherwise.} \end{cases} \tag{2.23}$$

Example 5 *Let us consider again the net in Figure 2.6. The association of the rates λ_1, λ_2 and λ_3 with the corresponding transitions $Fail$, Repair and Product generates a SPN whose reachability graph is isomorphic to the transition diagram depicted in Figure 2.3.*

As last consideration, it is worth to say that SPN has been extended in several ways by introducing inhibitor arcs, immediate transitions (which lead to the definition of Generalized SPNs provided in [10]), colored tokens, queuing places etc.

All these extensions increase the expressiveness of the formalism, but, at the same time, make heavier the notation; hence, we have chosen not to include them in this work, even if their use is possible as well.

We suggest [10, 14] to the readers that are interested in these facets.

Part I

Markov reward models

3
Markov reward models

The analysis of Markov chains is still an open problem due to the fact that the state space blows up exponentially with the number of modelled objects and, as mentioned earlier, several approximate techniques have been proposed to tackle the problem.

However, we argue that the use of approximations has to be the ultimate attempt to analyze a Markov chain.

The reason is that even if traditional techniques may fail, there are situations where less general methods are able to handle Markovian processes having cumbersome state spaces due to particular properties that the models hold. The identification of these cases is not always immediate and requires a preliminary analysis of the models. Such analysis, called *characterization of the process* by Kulkarni [60], aims to find properties that help to simplify the problem. Some trivial examples of these situations are redundancies in the structure of the chain or the subset of parameters playing the main role in the model dynamics.

One of the goals of this chapter is to provide a *"proof of concept"* about the importance of this phase by showing a simple case where the structure of the model suggests a way to reduce considerably the state space of the chain; namely, the modeling of productions whose accumulation does not affect the dynamics of the rest of the model.

As second aim, this chapter allows to deepen the discussion about the numerical methods introduced in Section 2.2.3 by showing how they can be used in practice and, more important, how they are not to be intended as monolithic methods that cannot be manipulated to better fit the analysis.

In order to point out what we want deal with, consider once again the system introduced in Figure 2.6 and its underlying CTMC (Figure 2.3). The diagram consists of an infinite duplication of the cycle $MachineOn/MachineOff$ representing the accumulated production of the machine. It is clear that the complexity of the problem can be drastically reduced if the production is not described explicitly in the state space because, in that case, the chain would be limited to two states only.

Thus, our target is to consider the stochastic process which describes only the state of the machine and build another stochastic process which depends on the first and describes the amount of the production. This goal can be achieved by considering the accumulations as rewards that the process can gain according to the states that it passes through or according to the transitions that occur.

Stochastic reward models allow this kind of analysis; they were introduced in the '80s for the performance analysis of communication systems and a large number of numerical methods were developed to make effective use of them.

They can be classified according to four criteria: (1) the stochastic behaviour of the underlying process, (2) the type of the reward accumulation, (3) the possibility and the type of the loss that the reward can suffer (no loss, partial loss, complete loss), (4) and the evaluated measure (for a more detailed description, see [72, 73, 55]).

In this dissertation we consider the cases in which (1) the underlying process is Markovian, (2) the reward accumulation is through instantaneous impulses of pre-determined "gains", (3) no reward loss is possible and (4) the measure of reference corresponds to the moments of the accumulated reward and completion time. Accordingly, the context is very akin to the one considered in [84, 22]. It is evident that the Markovian reward model (MRM) that we present has a limited sphere of applicability, however it can be applied to a high number of models.

In the following sections we deal with the definitions and provide recursive formulas needed to compute the moments of accumulated reward and the completion time. The two derivations follow a common path but their description in a unique section would be overloaded; hence, we decided to maintain them separated and to focus on different facets among the two sections in order to not to be redundant.

We will start with the discrete-time setting by focusing on gains that occur deterministically during a sojourn in a state. This means that during a sojourn in a given state i the accumulated reward is always increased by $r_i \geq 0$.

Moreover, we will provide the formulas to compute the moments of the completion time, defined as the minimum time required to complete a prefixed amount of accumulation. This corresponds to the computations of moments of first passage times.

Instead, in the section dedicated to the continuous-time setting, we will extend the model by allowing more than one reward variable in such a way that the computation of correlations is also possible.

Furthermore, we will describe the possibility to build such reward models starting from a common SPN.

As last, it is worth to say that the reading of the first part is not strictly related to the second; thus, the reader that is not interested in the argument can skip it without affecting the understanding of the following chapters.

3.1 Rewards governed by DTMCs

Consider a stochastic process having a discrete-time Markov chain (DTMC) that modulates the discrete accumulation of a reward gained in each time slot and let $r_i, 0 \leq r_i \leq c$, identify the integer reward rate associated with state i where c is the maximum reward per step.

The reward grows in such a way that a sojourn of length $l \in \mathbb{N}$ in state i provides lr_i reward. Denoting by $\overline{X}(n)$ the state of the chain after the nth transition and by $\overline{Z}(n)$ the reward accumulated in the first n steps, we have $\overline{Z}(n) = \sum_{i=0}^{n-1} r_{\overline{X}(i)}$ for $n \in \mathbb{N}$ and we assume that $\overline{Z}(0) = 0$. Furthermore, let us split the transition probability matrix into $c+1$ matrices $\mathbf{P}^{(i)}, 0 \leq i \leq c$, having the following entries

$$p_{k,l}^{(i)} = \begin{cases} p_{k,l} & \text{if } k \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

where $S_i, 0 \leq i \leq c$, denotes the set of states with reward rate i. Our aim is to characterize the joint probability of the accumulated reward (AR) and the background state defined for $n, k \in \mathbb{N}$ as

$$F_{ij}(n,k) = Pr\{\overline{Z}(n) = k, \overline{X}(n) = j | \overline{X}(0) = i\}.$$

It is easy to see that the following recursion holds for $n = 1, 2, 3, ...$ and $k = 0, 1, 2, ...$

$$F_{ij}(n,k) = \sum_{m=0}^{\min(k,c)} \sum_{l \in S_m} F_{il}(n-1, k-m) p_{l,j}$$

which, introducing the matrix notation $\mathbf{F}(n,k) = [F_{ij}(n,k)]$, becomes

$$\mathbf{F}(n,k) = \sum_{m=0}^{\min(k,c)} \mathbf{F}(n-1, k-m)\mathbf{P}^{(m)} \tag{3.1}$$

Another measure of interest is the minimum time required to complete the production of a pre-determined quantity k. This measure takes the name of *completion time* (CT) and corresponds to the characterization of the first-passage time $\overline{C}(k) = min\left[n : \overline{Z}(n) \geq k\right]$.

Due to the no-loss property the reward accumulation is monotone on the time which allows to represent completion time as the dual problem of $F_{i,j}(n,k)$. In fact, it is easy to see that the following relation holds for $n, k \in \mathbb{N}$

$$Pr(\overline{Z}(n) < k) = Pr(\overline{C}(k) > n).$$

The joint distribution of the completion time and the background state is defined for $n, k \in \mathbb{N}$ as

$$G_{ij}(n,k) = Pr\{\overline{Z}(n) \geq k, \overline{Z}(n-1) < k, \overline{X}(n) = j | \overline{X}(0) = i\}$$

and satisfies the relation

$$G_{ij}(n,k) = \sum_{m=1}^{\min(k,c)} \sum_{h=m}^{c} \sum_{l \in S_h} F_{il}(n-1, k-m) p_{lj} \tag{3.2}$$

where $n, k \in \mathbb{N}$.
By introducing the matrix notation $\mathbf{G}(n,k) = [G_{ij}(n,k)]$, equation (3.2) becomes

$$\mathbf{G}(n,k) = \sum_{m=1}^{\min(k,c)} \sum_{h=m}^{c} \mathbf{F}(n-1, k-m)\mathbf{P}^{(h)} . \tag{3.3}$$

The accumulated reward and the completion time can be analyzed in z-transform domain.

Theorem 1 *The double z-transform of* $\mathbf{F}(n,k)$ *is given by*

$$\mathbf{F}^{**}(z_1,z_2) = \left(\mathbf{I} - z_1 \sum_{m=0}^{c} z_2^m \mathbf{P}^{(m)}\right)^{-1} \tag{3.4}$$

Proof. Multiplying the left hand side of (3.1) by $z_1^n z_2^k$ and summing for $n = 1, 2, \ldots$ and $k = 0, 1, \ldots$ gives

$$\begin{aligned}\sum_{n=1}^{\infty}\sum_{k=0}^{\infty} z_1^n z_2^k \mathbf{F}(n,k) =& \mathbf{F}^{**}(z_1,z_2) - \sum_{k=0}^{\infty} z_2^k \mathbf{F}(0,k) = \\ & \mathbf{F}^{**}(z_1,z_2) - \mathbf{I}\end{aligned} \tag{3.5}$$

because $\mathbf{F}(0,0) = \mathbf{I}$ and $\mathbf{F}(0,k) = \mathbf{0}$ for $k = 1,2,3,....$ By performing the same operation on the right hand side of (3.1) we have

$$\begin{aligned}&\sum_{n=1}^{\infty}\sum_{k=0}^{\infty} z_1^n z_2^k \sum_{m=0}^{\min(k,c)} \mathbf{F}(n-1,k-m)\mathbf{P}^{(m)} = \\ &\sum_{m=0}^{c}\sum_{n=1}^{\infty}\sum_{k=m}^{\infty} z_1^n z_2^k \mathbf{F}(n-1,k-m)\mathbf{P}^{(m)} = \\ &z_1 \sum_{m=0}^{c} z_2^m \sum_{n=1}^{\infty}\sum_{k=m}^{\infty} z_1^{n-1} z_2^{k-m} \mathbf{F}(n-1,k-m)\mathbf{P}^{(m)} = \\ &z_1 \sum_{m=0}^{c} z_2^m \sum_{n=0}^{\infty}\sum_{k=0}^{\infty} z_1^n z_2^k \mathbf{F}(n,k)\mathbf{P}^{(m)} = \\ &z_1 \sum_{m=0}^{c} z_2^m \mathbf{F}^{**}(z_1,z_2)\mathbf{P}^{(m)}\ .\end{aligned} \tag{3.6}$$

From (3.5) and (3.6) we have

$$\mathbf{F}^{**}(z_1,z_2) - \mathbf{I} = z_1 \sum_{m=0}^{c} z_2^m \mathbf{F}^{**}(z_1,z_2)\mathbf{P}^{(m)}$$

which by simple rearrangements proves the theorem. □

Theorem 2 *The double z-transform of* $\mathbf{G}(n,k)$ *is*

$$\begin{aligned}&\mathbf{G}^{**}(z_1,z_2) = \\ &\mathbf{I} + \sum_{m=1}^{c}\sum_{h=m}^{c} z_1 z_2^m \left(\mathbf{I} - z_1 \sum_{m=0}^{c} z_2^m \mathbf{P}^{(m)}\right)^{-1} \mathbf{P}^{(h)}\end{aligned} \tag{3.7}$$

Proof. Multiplying the left hand side of (3.3) by $z_1^n z_2^k$ and summing for $n = 1, 2, \ldots$ and $k = 0, 1, \ldots$ gives

$$\sum_{n=1}^{\infty}\sum_{k=0}^{\infty} z_1^n z_2^k \mathbf{G}(n,k) = \mathbf{G}^{**}(z_1,z_2) - \sum_{k=0}^{\infty}\mathbf{G}(0,k) = \mathbf{G}^{**}(z_1,z_2) - \mathbf{I}$$

because $\mathbf{G}(0,0) = \mathbf{I}$ and $\mathbf{G}(0,k) = \mathbf{0}$ for $k = 1, 2, 3, \ldots.$ By performing the same operation on the right hand side of (3.3) we have

$$\sum_{n=1}^{\infty}\sum_{k=0}^{\infty} z_1^n z_2^k \sum_{m=1}^{\min(k,c)}\sum_{h=m}^{c}\mathbf{F}(n-1,k-m)\mathbf{P}^{(h)} =$$
$$\sum_{m=1}^{c}\sum_{h=m}^{c} z_1 z_2^m \sum_{n=1}^{\infty}\sum_{k=m}^{\infty} z_1^{n-1} z_2^{k-m}\mathbf{F}(n-1,k-m)\mathbf{P}^{(h)} =$$
$$\sum_{m=1}^{c}\sum_{h=m}^{c} z_1 z_2^m \mathbf{F}^{**}(z_1,z_2)\mathbf{P}^{(h)}$$

from which by applying (3.4) the theorem follows. □

Analysis by double inverse z-transform are viable for small or highly structured DTMCs. For this reason we provide recursive methods to compute the moments of both the accumulated reward and completion time.

3.1.1 Moments of accumulated reward

The following theorem provides an efficient approach for the calculation of the moments of the accumulated reward defined as

$$\mathbf{K}^{(l)}(n) = \sum_{k=0}^{\infty} k^l \mathbf{F}(n,k), n = 0, 1, 2, \ldots$$

Theorem 3 *The moments of the accumulated reward,* $\mathbf{K}^{(l)}(n), n = 0, 1, 2, \ldots,$ *satisfy the following recursive relation*

$$\mathbf{K}^{(l)}(n) = \sum_{i=0}^{l}\binom{l}{i}\mathbf{K}^{(l-i)}(n-1)\sum_{m=0}^{c} m^i \mathbf{P}^{(m)}. \tag{3.8}$$

Proof. The moments $\mathbf{K}^{(l)}(n)$ can be derived starting from (3.1) by multiplying both sides by k^l and summing up for $k = 0, 1, 2, ...$ which leads to

$$\mathbf{K}^{(l)}(n) = \sum_{k=0}^{\infty} k^l \sum_{m=0}^{\min(k,c)} \mathbf{F}(n-1, k-m)\mathbf{P}^{(m)}$$

from which by changing the order of the summation and applying

$$k^l = \sum_{i=0}^{l} \binom{l}{i} (k-m)^{l-i} m^i$$

we get

$$\begin{aligned} \mathbf{K}^{(l)}(n) = & \sum_{m=0}^{c} \sum_{k=m}^{\infty} \sum_{i=0}^{l} \binom{l}{i} (k-m)^{l-i} m^i \mathbf{F}(n-1, k-m)\mathbf{P}^{(m)} = \\ & \sum_{m=0}^{c} \sum_{i=0}^{l} \binom{l}{i} \sum_{k=m}^{\infty} (k-m)^{l-i} \mathbf{F}(n-1, k-m) m^i \mathbf{P}^{(m)} = \\ & \sum_{m=0}^{c} \sum_{i=0}^{l} \binom{l}{i} \mathbf{K}^{(l-i)}(n-1) m^i \mathbf{P}^{(m)} \end{aligned}$$

which yields the expression given in (3.8). □

The base cases of the recursion are given by the matrices

$$\mathbf{K}^{(l)}(0) = \begin{cases} \mathbf{I} & \text{if } l = 0 \\ \mathbf{Z} & \text{otherwise} \end{cases}$$

where $\mathbf{Z}$ denotes a matrix of zeros.

The recursion given in (3.1) for the distribution of the accumulated reward is based on the so-called *forward* scheme. It is easy to verify that the following *backward* counterpart also holds

$$\mathbf{F}(n, k) = \sum_{m=0}^{\min(k,c)} \mathbf{P}^{(m)} \mathbf{F}(n-1, k-m) \tag{3.9}$$

and it leads to the following theorem which is the backward counterpart of Theorem 3 and thus provides a backward scheme for the computation of the moments of the accumulated reward.

Theorem 4 *The moments of the accumulated reward,* $\mathbf{K}^{(l)}(n), n = 0, 1, 2, ...,$ *satisfy the following backward recursive relation*

$$\mathbf{K}^{(l)}(n) = \sum_{i=0}^{l} \binom{l}{i} \sum_{m=0}^{c} m^i \mathbf{P}^{(m)} \mathbf{K}^{(l-i)}(n-1). \tag{3.10}$$

Proof. Follows the same path used for Theorem 3. □

The expressions given in Theorems 3 and 4 directly provide a procedure for the calculation of the moments. The time complexity of calculating $\mathbf{K}^{(l)}(n)$ for $l = 1, ..., N$ is roughly $\sum_{i=0}^{N}(i+1)$ times the complexity of the transient analysis of the underlying DTMC. The space complexity is instead $N+1$ higher than for the transient analysis of the DTMC.

3.1.2 Moments of completion time

In the following we derive an efficient, recursive scheme for the computation of the moments of the completion time. For this purpose, let us use the notation

$$G_{ij}(n, k, e) = Pr\{\overline{Z}(n) = k + e, \overline{Z}(n-1) < k, \overline{X}(n) = j | \overline{X}(0) = i\}$$

which characterizes not only the time instance of reaching a given amount of reward but also the excess amount upon the moment of completion. In matrix notation we write $\mathbf{G}(n, k, e) = [G_{ij}(n, k, e)]$. The mth moment of the completion time with a given amount of excess will be denoted by

$$L_{ij}^{(m)}(k, e) = \sum_{n=0}^{\infty} n^m G_{ij}(n, k, e) \quad e \in [0, c-1], k = 0, 1, 2...$$

with associated matrix notation $\mathbf{L}^{(m)}(k, e) = [L_{ij}^{(m)}(k, e)]$. It follows that, by summing up for all possible values of the amount of excess, the moments of the completion time, $\mathbf{L}^{(m)}(k)$, can be obtained as

$$\mathbf{L}^{(m)}(k) = \sum_{e=0}^{c-1} \mathbf{L}^{(m)}(k, e)$$

The following theorem provides a recursive relation for the quantities $\mathbf{L}^{(m)}(k, e)$.

Theorem 5 *The moments of the completion time with a given amount of excess, $e, 0 \leq e \leq c-1$, satisfy the forward recursion*

$$\mathbf{L}^{(m)}(k,e) = \sum_{i=0}^{m}\sum_{f=1}^{c-e}\binom{m}{i}\mathbf{L}^{(i)}(k-f,0)\mathbf{L}^{(m-i)}(1,f-1+e)$$

Proof. The recursion is based on the equation

$$\mathbf{L}^{(m)}(k,e) = \sum_{n=0}^{\infty} n^m G_{ij}(n,k,e) =$$

$$\sum_{x_1=0,1,..}\sum_{x_2=0,1,..}(x_1+x_2)^m \sum_{f=1}^{c-e}\sum_{l\in S} G_{il}(x_1,k-f,0)G_{lj}(x_2,1,f-1+e) \quad (3.11)$$

where we exploited the fact that all the possible ways of reaching an amount k of reward with a level e of excess is composed of

- reaching a level $k-f$ with zero excess where $1 \leq f \leq c-e$,
- reaching level k with excess e (i.e., arriving to level $k+e$) without "touching" any level between $k-f$ and $k+e$.

The theorem follows from (3.11) by applying matrix notation and the formula $(x_1+x_2)^m = \sum_{i=0}^{m}\binom{m}{i}x_1^i x_2^{m-i}$. □

The following theorem is the "backward" counterpart of Theorem 5.

Theorem 6 *The moments of the completion time with a given amount of excess, $e, 0 \leq e \leq c-1$, satisfy the backward recursion*

$$\mathbf{L}^{(m)}(k,e) = \sum_{i=0}^{m}\sum_{f=0}^{c-1}\binom{m}{i}\mathbf{L}^{(m-i)}(1,f)\mathbf{L}^{(i)}(k-f-1,e)$$

Proof. The proof follows the line of the proof of Theorem 5. □

In order to carry out the computations we need the matrices that provide the base cases of the recursion. Some of these matrices are trivial to compute and for what concerns these we have:

$$\mathbf{L}^{(0)}(k,e) = \begin{cases} \mathbf{I} & \text{if } k \leq 0, k = -e \\ \mathbf{Z} & \text{if } k \leq 0, k \neq -e \end{cases}$$

$$\mathbf{L}^{(i)}(k,e) = \mathbf{Z} \quad \text{if } i \geq 1, k \leq 0$$

where negative values of k are considered for computational convenience of the recursion. Apart of the matrices listed above we need to compute $\mathbf{L}^{(i)}(1,e)$ with $i \geq 0$ and $0 \leq e \leq c-1$. For this purpose, it is useful to introduce the matrices $\mathbf{P}^{(0,i)}$ with $0 \leq i \leq c$ with entries according to

$$\mathbf{P}^{(0,i)}_{kl} = \begin{cases} \mathbf{P}_{kl} & \text{if } k \in S_0 \text{ and } l \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

With these matrices, we have

$$\mathbf{L}^{(i)}(1,e) = \sum_{k=1}^{\infty}(k+1)^i \left(\mathbf{P}^{(0,0)}\right)^{k-1} \mathbf{P}^{(0,e+1)}\mathbf{P}^{(e+1)} + \mathbf{P}^{(e+1)} \tag{3.12}$$

where the first term of the right hand side considers the cases in which the process starts in a state with zero reward, it stays in the subset of states S_0 for $k-1$ steps, then it jumps to a state in S_{e+1} and reaches level 1 with excess e in the subsequent step. The second term considers instead the cases in which the process starts in a state belonging to S_{e+1} and it reaches level 1 with excess e in a single step. In order to compute $\mathbf{L}^{(i)}(1,e)$ let us denote the summation in (3.12) by

$$\mathbf{R}^{(i)} = \sum_{k=1}^{\infty}(k+1)^i \left(\mathbf{P}^{(0,0)}\right)^{k-1}$$

By noting that

$$\begin{aligned}
\mathbf{R}^{(i)} &= \sum_{l=0}^{i} \binom{i}{l} \sum_{k=1}^{\infty} k^{i-l} \left(\left(\mathbf{P}^{(0,0)}\right)^{k-1}\right) \\
&= \sum_{l=0}^{i} \binom{i}{l} \sum_{k=2}^{\infty} k^{i-l} \left(\left(\mathbf{P}^{(0,0)}\right)^{k-2} \mathbf{P}^{(0,0)} + \mathbf{I}\right) \\
&= \sum_{l=0}^{i} \binom{i}{l} \left(\mathbf{R}^{(i-l)}\mathbf{P}^{(0,0)} + \mathbf{I}\right)
\end{aligned}$$

a recursive formula for $\mathbf{R}^{(i)}$ follows

$$\mathbf{R}^{(i)} = \left[\sum_{l=1}^{i} \binom{i}{l} \left(\mathbf{R}^{(i-l)}\mathbf{P}^{(0,0)} + \mathbf{I}\right) + \mathbf{I}\right] \left(\mathbf{I} - \mathbf{P}^{(0,0)}\right)^{-1}$$

The above expression points out that the presence of many states with zero reward increases the computational complexity. Indeed, in order to compute the $\mathbf{R}^{(i)}$ matrices, the inversion of a matrix with as many rows as many zero states there are is required. We briefly mention that, if a vector based implementation is applied than the matrix inversion can be avoided but the calculations would still require the solution of linear systems during the computations.

The expressions given in Theorems 6 and 5 directly provide a procedure for the calculation of the moments of the completion time. If the number of zero states is low then analyzing the moments of the completion time is as complex as analyzing the moments of the aggregated reward.

3.2 Rewards governed by CTMCs

In this section we move to the continuous-time setting but we remain in a Markovian context; hence, the process that governs the accumulation is a CTMC denoted with $\{X(t), t \geq 0\}$. Within this context the reward accumulation is performed in a slightly different manner. As first difference we have that the accumulation is not given determistically by the sojourn in a state, but dependent on a number of state-dependent activities whose occurrence can result in gain of reward and, possibly, in a state transition of the CTMC. Another difference is that we extend the previous model in such a way that W different types of rewards are considered.

The activities are identified by a vector of W integers describing the gain they provide and the total intensity of the activities providing g amount of reward and moving the CTMC from state i to state j is denoted by $r_{i,j}^{(g)}$ with $i, j \in \mathcal{S}$ and $g \in \mathbb{Z}^W$. (We denote by $\mathbb{Z}$ and $\mathbb{C}$ the set of non-negative integers and the set of complex numbers, respectively; the corresponding set of vectors of length W are denoted by $\mathbb{Z}^W$ and $\mathbb{C}^W$). The intensities $r_{i,j}^{(g)}$ are organized into matrices as $\mathbf{R}^{(g)} = \left[r_{i,j}^{(g)}\right]_{i,j \in \mathcal{S}}$.

As defined, AR is a discrete random vector, denoted with $Z(t) \in \mathbb{Z}^W$, which represents the quantity of reward which was gained up to time t. As an example, assume $W = 3$ and $Z(t) = |3, 1, 4|$. If an activity with reward vector $g = |1, 0, 2|$ occurs in the infinitesimal interval $[t, t + \Delta]$ then the random vector will take the value $Z(t + \Delta) = |3, 1, 4| + |1, 0, 2| = |4, 1, 6|$.

Since we do not consider activities with negative gain, the quantity of the produced reward can only grow or remain stable preserving the duality with the completion time.

3.2.1 Derivation of MRM from SPNs

The MRM as defined above can be constructed starting from a SPN; the first step is to identify the set of places where the accumulation of tokens is monotone and have no impact on the intensity of the transitions. Place i belongs to this set if the following property holds[1]:

$$\forall n, 1 \leq n \leq R : i^-_{n,i} = 0 \text{ and } \exists n, 1 \leq n \leq R : i^+_{n,i} > 0. \tag{3.13}$$

If there are places satisfying the above property, then an MRM with a smaller state space than that of the original CTMC can be built. The construction of the MRM can be done in automatic manner as follows.

The places satisfying the property in (3.13) will be called *monotonic* and their set will be denoted by $\mathcal{M}$ while the rest of the places will be called *non-monotonic* and the corresponding set will be denoted by $\overline{\mathcal{M}}$.

The cardinality of set $\mathcal{M}$, denoted by W, is the number of types of rewards of the MRM. In the MRM, the underlying CTMC models the non-monotonic places while the rewards take into account the growth of the monotonic ones.

Next, the effect of each transition has to be split into two parts. The first part gives the effect of the transition on the places belonging to $\overline{\mathcal{M}}$ and gives rise to a transition in the underlying CTMC of the MRM.

The effect of these transitions will be described by the vectors $e'_i, 1 \leq i \leq R$ which are obtained simply from the vectors $e_i, 1 \leq i \leq R$, introduced in Section 2.3.2, by taking those entries which refer to places belonging to $\overline{\mathcal{M}}$.

The second part takes into account the growth of the monotonic places. The reward produced by a transition $i, 1 \leq i \leq R$, can be simply described by a vector g_i collecting those entries of e_i which corresponds to monotonic places. To sum up, the effect of transition $i, 1 \leq i \leq R$, described by e_i in the original CTMC is decomposed into two vectors e'_i and g_i referring to the non-monotonic and the monotonic places, respectively. Based on $e'_i, g_i, 1 \leq i \leq R$, and the corresponding intensities provided by the set Λ, the construction of the matrices $\mathbf{R}^{(g)}$ describing the MRM is straightforward. The entries are

[1]In sake of simplicity, we are assuming that the function $\lambda_n(x)$ describing the rates of the nth transition depends only on the places belonging to the set ${}^\bullet t_n$, $1 \leq n \leq R$.

obtained as

$$r_{i,j}^{(g)} = \sum_{\forall n: j=i+e'_n \wedge g_n = g} \lambda_n(i) \tag{3.14}$$

Note that the underlying CTMC of the MRM can have finite state space even if the state space of the original CTMC is infinite. This is the case for the manufacturing machine model introduced in Figure 2.6.

Example 6 *Considering the manufacturing machine model, we have two non-monotonic places, $MachineOn$ and $MachineOff$, and one monotonic, $Production$. A state of the underlying CTMC of the MRM is given by a vector of two entries $x = |x_1, x_2|$ since it represents the two possible states of the machine. The effect of the transitions on the underlying CTMC is*

$$e'_1 = |-1, 1|, \; e'_2 = |1, -1|, \; e'_3 = |0, 0|.$$

We have a single type of reward corresponding to the amount of production. Accordingly, $W = 1$ and the vectors describing the effect of the transitions on the reward variable are composed of a single entry:

$$g_1 = |0|, \; g_2 = |0|, \; g_3 = |1|.$$

Then the matrices representing the MRM are simply

$$\mathbf{R}^{((0))} = \begin{vmatrix} 0 & \mu_1 \\ \mu_2 & 0 \end{vmatrix}, \; \mathbf{R}^{((1))} = \begin{vmatrix} \mu_3 & 0 \\ 0 & 0 \end{vmatrix}.$$

3.2.2 Accumulated reward, characterization in time and transform domain

Given the matrices $\mathbf{R}^{(g)}$ for every possible gain vector g, the transient accumulated reward is characterized by the matrix $\mathbf{B}(t, w)$ with entries

$$B_{i,j}(t, w) = Pr\{Z(t) = w, X(t) = j | X(0) = i, Z(0) = 0\}$$

giving the probability that, having started in state i with 0 reward[1], at time t the reward is w and the underlying chain is in state j. Note that w is a vector: $w \in \mathbb{Z}^W$.

[1]Our framework can be easily extended to starting the system from non-zero reward levels or with reward levels distributed according to some distribution but we avoid to handle these cases for sake of simplicity.

Theorem 7 *The transient behaviour of the AR fulfills the following differential equation*

$$\frac{d\mathbf{B}(t,w)}{dt} = -\mathbf{B}(t,w)\mathbf{S} + \sum_{\forall g: w-g\geq 0} \mathbf{B}(t,w-g)\mathbf{R}^{(g)} \tag{3.15}$$

where $\mathbf{S}$ *is a diagonal matrix with entries*

$$S_{i,j} = \begin{cases} \sum_{k\in\mathcal{S}} \sum_{\forall g} r_{i,k}^{(g)} & \textit{if } i=j, \\ 0 & \textit{otherwise.} \end{cases} \tag{3.16}$$

Proof. In order to derive equation (3.15) let us consider first the change of $B_{i,j}(t,w)$ in an infinitesimal time interval. We have

$$B_{i,j}(t+\Delta,w) = B_{i,j}(t,w)\left(1 - \sum_{k\in\mathcal{S}}\sum_{\forall g} r_{j,k}^{(g)}\Delta\right) + \\ \sum_{k\in\mathcal{S}}\sum_{\forall g: w-g\geq 0}\left(B_{i,k}(t,w-g)r_{k,j}^{(g)}\Delta\right) + o(\Delta)$$

where the first term is the probability that state j and w amount of reward has been reached by time t and no activity occurs in $[t,t+\Delta]$, the second term is the probability to get exactly the necessary amount to reach the target w and moving the process to the state j, meanwhile $o(\Delta)$ represents the fact that the probability of two events in $[t,t+\Delta]$ negligible.

Dividing by Δ, taking the limit $\Delta \to 0$ and rearranging leads to

$$\frac{dB_{i,j}(t,w)}{dt} = -B_{i,j}(t,w)\sum_{k\in\mathcal{S}}\sum_{\forall g} r_{j,k}^{(g)} + \\ \sum_{k\in\mathcal{S}}\sum_{\forall g: w-g\geq 0}\left(B_{i,k}(t,w-g)r_{k,j}^{(g)}\right)$$

from which introducing matrix notation the theorem follows. □

We mention here that matrix $\mathbf{S}$ represents the sojourn times whereas the matrices $\mathbf{R}^{(g)}$ represent the transition rates between states. Thus the infinitesimal generator of the underlying CTMC is

$$\mathbf{Q} = -\mathbf{S} + \sum_{\forall g} \mathbf{R}^{(g)}.$$

A compact solution of (3.15) can be provided only in transform domain. In the sequel the following transforms of $\mathbf{B}(t,w)$ will be applied: the Laplace transform according to the time variable given by

$$\mathbf{B}^*(s,w) = \int_0^\infty e^{-st}\mathbf{B}(t,w)dt$$

and the double continuous/discrete transform, i.e., Laplace transform according to the time variable and z-transform according to all the reward variables, defined by

$$\mathbf{B}^{**}(s,z) = \mathbf{B}^{**}(s,(z_1,\ldots,z_W)) = \\ \sum_{i\in\mathbb{Z}^W}\prod_{j=1}^{W} z_j^{i_j}\int_0^\infty e^{-st}\mathbf{B}(t,i)dt$$

with $z \in \mathbb{C}^W$. In the following, in order to abbreviate, having $z \in \mathbb{C}^W$ and $i \in \mathbb{Z}^W$ we will write $\prod_{j=1}^{W} z_j^{i_j}$ simply as z^i.

Theorem 8 *The continuous/discrete transform of the accumulated reward is given by:*

$$\mathbf{B}^{**}(s,z) = \left[s\mathbf{I} + \mathbf{S} - \sum_{\forall g} z^g \mathbf{R}^{(g)}\right]^{-1} \tag{3.17}$$

where $\mathbf{I}$ *is the identity matrix.*

Proof. The continuous/discrete transform of the left hand side of (3.15) with $z \in \mathbb{C}^W$ is given by

$$\begin{aligned}\sum_{i\in\mathbb{Z}^W} z^i \int_0^\infty e^{-st}\frac{d\mathbf{B}(t,i)}{dt} &= \sum_{i\in\mathbb{Z}^W} z^i\left(s\mathbf{B}^*(s,i) - \mathbf{B}(0,i)\right) \\ &= s\mathbf{B}^{**}(s,z) - \mathbf{I} \qquad (3.18)\end{aligned}$$

because $\mathbf{B}(0,0) = \mathbf{I}$ and $\mathbf{B}(0,i) = \mathbf{0}$ when $i \neq 0$. The same operation on the right hand side of (3.15) leads to

$$\begin{aligned}
&\sum_{\forall i \in \mathbb{Z}^W} z^i \int_0^\infty e^{-st} \left(-\mathbf{B}(t,i)\mathbf{S} + \sum_{\forall g: i-g \geq 0} \mathbf{B}(t, i-g)\mathbf{R}^{(g)} \right) dt \\
&= \sum_{\forall i \in \mathbb{Z}^W} z^i \left(-\mathbf{B}^*(s,i)\mathbf{S} + \sum_{\forall g: i-g \geq 0} \mathbf{B}^*(s, i-g)\mathbf{R}^{(g)} \right) \\
&= -\mathbf{B}^{**}(s,z)\mathbf{S} + \sum_{\forall i \in \mathbb{Z}^W} z^i \left(\sum_{\forall g: i-g \geq 0} \mathbf{B}^*(s, i-g)\mathbf{R}^{(g)} \right) \\
&= -\mathbf{B}^{**}(s,z)\mathbf{S} + \sum_{\forall g} z^g \left(\sum_{\forall i \in \mathbb{Z}^W : i \geq g} z^{i-g}\mathbf{B}^*(s, i-g)\mathbf{R}^{(g)} \right) \\
&= -\mathbf{B}^{**}(s,z)\mathbf{S} + \sum_{\forall g} z^g \left(\sum_{\forall i \in \mathbb{Z}^W} z^{i}\mathbf{B}^*(s, i)\mathbf{R}^{(g)} \right) \\
&= -\mathbf{B}^{**}(s,z)\mathbf{S} + \sum_{\forall g} z^g \mathbf{B}^{**}(s,z)\mathbf{R}^{(g)} \qquad (3.19)
\end{aligned}$$

From (3.18) and (3.19) we have

$$s\mathbf{B}^{**}(s,z) - \mathbf{I} = -\mathbf{B}^{**}(s,z)\mathbf{S} + \sum_{\forall g} z^g \mathbf{B}^{**}(s,z)\mathbf{R}^{(g)}$$

which by applying trivial algebra provides the theorem. □

3.2.3 Moments of the accumulated reward

In this section we first define a method for the analysis of the factorial moments of the AR starting from the double transform expression given in (3.17). Since the numerical calculations of this method can easily be unstable, we make then modifications in order to define a numerically stable algorithm. Finally, we show that the truncation error is controllable and provide some indications for what concerns the computational complexity. Throughout the section we follow an approach similar to that of [84] extending it to the multiple reward variable case.

3.2.3.1 Recursion for the moments of the accumulated reward

Let us denote by $Z_{i,j}(t)$ the amount of the jth type of reward at time t given that the initial state is i. Then, having a vector $n \in \mathbb{Z}^W$, the associated joint factorial moment, assuming state i as initial state, is given by the expected value

$$f_i^{(n)}(t) = E\left[\prod_{j=1}^{W}\prod_{k=0}^{n_j-1}(Z_{i,j}(t)-k)\right]. \tag{3.20}$$

Let us illustrate the use of the factorial moments assuming $W = 2$. Using $n = |1, 0|$ the expected value in (3.20) provides simply the mean quantity of the 1st type of reward. Using $n = |2, 0|$ we obtain its second factorial moment which can be used to compute the second "normal" moment for the 1st type of reward as

$$E\left[Z_{i,1}(t)^2\right] = f_i^{(|2,0|)}(t) + f_i^{(|1,0|)}(t).$$

For a general description of the relation between factorial moments and "normal" moments, see [49]. The formula in (3.20) allows for expressing joint measures of the various reward types as well. For example, the covariance of the 1st and the 2nd type of rewards can be obtained as

$$\begin{aligned}E\left[(Z_{i,1}(t) - E\left[Z_{i,1}(t)\right])(Z_{i,2}(t) - E\left[Z_{i,2}(t)\right])\right] &= \\ f_i^{(|1,1|)}(t) - f_i^{(|1,0|)}(t) f_i^{(|0,1|)}(t)&.\end{aligned}$$

The column vector formed by $f_i^{(n)}(t)$ will be denoted by $f^{(n)}(t) = \left(f_i^{(n)}(t)\right)$. Based on basic properties of the z-transform and denoting $\sum_{i=1}^{W} n_i$ by $\#n$ we can write

$$f^{(n)}(t) = \left.\frac{\partial^{\#n}\mathbf{B}^*(t,z)}{\prod_{i=1}^{W}\partial z_i^{n_i}}\right|_{z=1} \cdot \mathbf{1} \tag{3.21}$$

where $\mathbf{1}$ is the row vector of 1s. In the sequel, in order to abbreviate, the partial derivative in (3.21) will be written in "vector" notation leading to

$$f^{(n)}(t) = \left.\frac{\partial^{n}\mathbf{B}^*(t,z)}{(\partial z)^{n}}\right|_{z=1} \cdot \mathbf{1}. \tag{3.22}$$

From (3.17), based on basic properties of the Laplace-transform, we have

$$\mathbf{B}^*(t,z) = \exp\left(\left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right) t\right) \tag{3.23}$$

where exp(●) denotes the matrix exponential function[1]. Applying (3.23) in (3.22), using the definition of the matrix exponential function and changing the order of the derivative and the summation leads to

$$f^{(n)}(t) = \sum_{i=0}^{\infty} \frac{t^i}{i!} \frac{\partial^n}{(\partial z)^n} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right)^i \Bigg|_{z=1} \cdot \mathbf{1}$$

By using the notation

$$\mathbf{N}^{(n)}(i) = \frac{\partial^n}{(\partial z)^n} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right)^i \Bigg|_{z=1} \tag{3.24}$$

we clearly have

$$f^{(n)}(t) = \sum_{i=0}^{\infty} \frac{t^i}{i!} \mathbf{N}^{(n)}(i) \cdot \mathbf{1} \tag{3.25}$$

and in the following we show that $\mathbf{N}^{(n)}(i)$ can be computed in a recursive manner. We can write $\mathbf{N}^{(n)}(i) =$

$$\frac{\partial^n}{(\partial z)^n} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right) \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right)^{i-1} \Bigg|_{z=1} =$$

$$-\frac{\partial^n}{(\partial z)^n} \mathbf{S} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right)^{i-1} \Bigg|_{z=1} + \tag{3.26}$$

$$\frac{\partial^n}{(\partial z)^n} \sum_{\forall g} z^g \mathbf{R}^{(g)} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)}\right)^{i-1} \Bigg|_{z=1} \tag{3.27}$$

[1]Note that the relation holds if all the matrices uniformizable.

It is easy to see that the term in (3.26) equals $-\mathbf{S}\mathbf{N}^{(n)}(i-1)$. The term in (3.27), by changing the order of the summation and the derivative and by applying properties of derivatives of products, can be written as

$$\sum_{\forall g} \frac{\partial^n}{(\partial z)^n} z^g \mathbf{R}^{(g)} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)} \right)^{i-1} \Bigg|_{z=1} =$$

$$\sum_{\forall g} \sum_{k \in \mathbb{Z}^W : k \leq n} \Bigg[\prod_{j=1}^{W} \binom{n_j}{k_j} \cdot \tag{3.28}$$

$$\left(\frac{\partial^k}{(\partial z)^k} z^g \mathbf{R}^{(g)} \cdot \frac{\partial^{n-k}}{(\partial z)^{n-k}} \left(-\mathbf{S} + \sum_{\forall g} z^g \mathbf{R}^{(g)} \right)^{i-1} \right) \Bigg] \Bigg|_{z=1}$$

whose product of binomial coefficients will be abbreviated in the sequel as

$$\binom{n}{k} = \prod_{j=1}^{W} \binom{n_j}{k_j}.$$

The first derivative of the right hand side of (3.28) evaluated at $z = 1$ will be denoted by $c_{g,k}$ and results to be

$$c_{g,k} = \frac{\partial^k}{(\partial z)^k} z^g \Bigg|_{z=1} = \frac{\partial^{\#k} z_1^{g_1} z_2^{g_2} \dots z_W^{g_W}}{\partial z_1^{k_1} \partial z_2^{k_2} \dots \partial z_W^{k_W}} =$$

$$\prod_{i=1}^{W} \prod_{j=0}^{k_i - 1} (g_i - j). \tag{3.29}$$

Based on (3.26), (3.27), (3.28) and (3.29) we have

$$\mathbf{N}^{(n)}(i) = -\mathbf{S}\mathbf{N}^{(n)}(i-1) +$$

$$\sum_{k \in \mathbb{Z}^W : k \leq n} \binom{n}{k} \sum_{\forall g} c_{g,k} \mathbf{R}^{(g)} \mathbf{N}^{(n-k)}(i-1) \tag{3.30}$$

which provides the required recursion. The initial condition for the recursion is $\mathbf{N}^{(0)}(0) = \mathbf{I}$ and $\mathbf{N}^{(n)}(0) = \mathbf{0}$ for any $n \neq 0$. Note that the recursion results in $\mathbf{N}^{(0)}(i) = \mathbf{Q}^i$ which corresponds to the fact that the factorial moment associated with the 0 vector provides the transient behaviour of the underlying CTMC.

3.2.3.2 Numerically stable recursion for the moments of the accumulated reward

The solution provided by (3.25) and (3.30) is not numerically stable but if the matrix $\mathbf{S}$ and all the matrices $\mathbf{R}^{(g)}$ are uniformizable then the problem can be solved by applying the idea of uniformisation introduced in the previous chapter.

Algebraically uniformisation is represented by introducing the quantity $\alpha \geq max_{i\in\mathcal{S}}(\mathbf{S}_{i,i})$ and considering the matrices

$$\mathbf{S}' = -\frac{\mathbf{S}}{\alpha} + \mathbf{I} \text{ and } \mathbf{R}'^{(g)} = \frac{\mathbf{R}^{(g)}}{\alpha} \tag{3.31}$$

whose entries are between zero and one and their sum

$$\mathbf{Q}' = \mathbf{S}' + \sum_{\forall g} \mathbf{R}'^{(g)}$$

is the transition probability matrix of a DTMC. By using the quantities defined in (3.31), equation (3.23) can be written as

$$\begin{aligned} \mathbf{B}^*(t,z) = & \exp\left(\left(\alpha(\mathbf{S}'-\mathbf{I}) + \sum_{\forall g} z^g q \mathbf{R}'^{(g)}\right) t\right) = \\ & \exp\left(\left(\mathbf{S}' + \sum_{\forall g} z^g \mathbf{R}'^{(g)}\right) \alpha t - \alpha t \mathbf{I}\right) = \\ & \exp(-\alpha t) \exp\left(\left(\mathbf{S}' + \sum_{\forall g} z^g \mathbf{R}'^{(g)}\right) \alpha t\right) \end{aligned} \tag{3.32}$$

from which, by introducing

$$\mathbf{N}'^{(n)}(i) = \frac{\partial^n}{(\partial z)^n} \left(\mathbf{S}' + \sum_{\forall g} z^g \mathbf{R}'^{(g)}\right)^i \Bigg|_{z=1}$$

we have

$$f^{(n)}(t) = \sum_{i=0}^{\infty} \frac{(\alpha t)^i}{i!} e^{-\alpha t} \mathbf{N}'^{(n)}(i) \cdot \mathbf{1}. \tag{3.33}$$

Following the same steps, given in (3.26-3.29), that lead to a recursion for $\mathbf{N}^{(n)}(i)$, the following recursive relation can be obtained for $\mathbf{N}'^{(n)}(i)$

$$\mathbf{N}'^{(n)}(i) = \begin{cases} \mathbf{I} & i = 0, n = 0 \\ \mathbf{0} & i = 0, n \neq 0 \\ \mathbf{S}'\mathbf{N}'^{(n)}(i-1) + & \\ \displaystyle\sum_{k \in \mathbb{Z}^W : k \leq n} \binom{n}{k} \sum_{\forall g} c_{g,k} \mathbf{R}'^{(g)} \mathbf{N}'^{(n-k)}(i-1) & \\ & \text{otherwise} \end{cases} \tag{3.34}$$

3.2.3.3 Error control

As in case of uniformisation based transient analysis of CTMCs, the infinite sum in (3.33) can be approximated by a finite sum. However, the error control is not as straightforward as in case of the transient analysis. For the transient probabilities, given by $f^{(0)}(t)$ in our framework, it is known that the sum of the entries is one. No such information is available in advance for the quantities $f^{(n)}(t)$ with $n \neq 0$.

The fact that the entries of $f^{(n)}(t)$ can be computed with arbitrary precision with a finite sum is ensured by the following trivial relation that holds entry-wise for the matrices $\mathbf{N}'^{(n)}(i)$

$$\mathbf{N}'^{(n)}(i) \leq (\mathbf{Q}')^i \prod_{j=1}^{W} \prod_{k=0}^{n_j - 1} (i g_{max} - k)$$

where g_{max} denotes the maximal entries of the vectors representing the gains, i.e., $g_{max} = \max_{\forall g, \forall i: 1 \leq i \leq W} g_i$. For any $\epsilon > 0$ there exists such K that

$$\left|\left| \sum_{i=K}^{\infty} \frac{(\alpha t)^i}{i!} e^{-\alpha t} (\mathbf{Q}')^i \prod_{j=1}^{W} \prod_{k=0}^{n_j - 1} (i g_{max} - k) \right|\right| \leq \epsilon \tag{3.35}$$

where $||v||$ is the sum of the entries of vector v. With such K we ensure

$$||f^{(n)}(t)|| - ||f_K^{(n)}(t)|| \leq \epsilon.$$

where $f_K^{(n)}(t)$ is the finite approximation given by

$$f_K^{(n)}(t) = \sum_{i=0}^{K} \frac{(\alpha t)^i}{i!} e^{-\alpha t} \mathbf{N}'^{(n)}(i) \cdot \mathbf{1}.$$

However, since not all transitions lead to maximal reward gain, the truncation point calculated based on (3.35) gives a pessimistic estimate for the number of terms that have to be considered for a given accuracy. A better approach in practice is to keep under control both the Poisson probabilities and the relative increment of the vectors $f^{(n)}(t)$. Accordingly, we truncate the infinite sum at K when

$$1-\sum_{i=0}^{K}\frac{(\alpha t)^i}{i!}e^{-\alpha t}\leq\epsilon_1 \text{ and } \frac{||f_{K+1}^{(n)}(t)||-||f_K^{(n)}(t)||}{||f_K^{(n)}(t)||}\leq\epsilon_2.$$

3.2.3.4 Computational complexity

The computational complexity in time of the proposed method can be related to that of randomisation based transient analysis of CTMCs (which is not easy to characterize; the interested reader is referred to [30] for further information).

As already mentioned, $f^{(0)}(t)$ provides the transient behaviour of the underlying CTMC and its computation requires as much effort as randomisation does. The calculation of a given factorial moment, $f^{(n)}(t)$, characterised by the vector n, necessitates the calculations of all factorial moments, $f^{(k)}(t)$, for which $k \in \mathbb{Z}^W : k \leq n$.

Moreover, the larger n the more terms in (3.33) are needed to obtain the predefined complexity. This effect, however, depends heavily on the reward structure of the model and is hard to capture formally.

Let us report here our experimental findings for a few cases. The first l factorial moments of a single reward variable requires about $l+1$ times more calculations than randomisation. Having W reward variables, computing the first l factorial moments for all of them requires approximately $1+Wl$ times more time than randomisation. For what concerns joint moments, to compute the covariance for every pair of reward variables is about $1+W+\binom{W}{2}$ times heavier than randomisation.

For what concerns space requirements, two situations have to be distinguished. If we are interested in the behaviour for every possible initial state then the computation requires the storage of square matrices whose size corresponds to the size of the state space. For a given factorial moments, $f^{(n)}(t)$, we need a matrix for every vector $k \in \mathbb{Z}^W : k \leq n$. If we are interested in the behaviour with a given initial situation (deterministic initial state or a particular initial distribution) then the calculations can be carried out on vectors.

Once again, to compute $f^{(n)}(t)$ we need a vector for every $k \in \mathbb{Z}^W : k \leq n$. This means that the space requirement is as many times larger than that of randomisation as many vectors $k \in \mathbb{Z}^W : k \leq n$ we have to consider.

4 Case studies

In this section, we provide some numerical results obtained by using MRMs in two different contexts: manufacturing production lines where discrete-time models are used and biochemical systems for which the continuous-time counterpart is exploited.

The two bulks of tests are different also from the point of view of the outcome that we wanted to achieve through the tests. For the first situation, we have a sort of real case studies which come from a collaboration with the *Mechanical Engineering department of Politecnico di Milano*, thus, the tests were driven by the will to show that the mathematical model behave as the real evidence. Secondly, the hope was to discover and maybe explain unexpected behaviours.

On the contrary, for the biochemical cases we aimed to promote the use of MRMs as a smarter solution for the analysis of CTMCs. This idea was motivated by the fact that in the context of systems biology these techniques are almost unknown although a large number of biological models has monotonic productions.

In the following we compute several measures of interest as expectation, variance, correlation, standard deviation, skewness, covariance. Furthermore, we show also that the estimation of the distribution is also possible. The solver for the MRM has been implemented in *JAVA* prototype[1] tool whereas the distribution estimation has been computed by using MRM Solve 2.0 [55, 82]. All the experiments have been performed on a common notebook powered by a *Intel Centrino Dual Core* with 4Gb of RAM.

4.1 Manufacturing production lines - rewards governed by DTMCs

The production lines that we analyze are concatenations of manufacturing stations separated by a finite dimension buffer that connects them and has the task to carry the production of the first station to the second. Graphically, they corresponds to the block depicted in Figure 4.1 where B_n denote the nth buffer capacity.

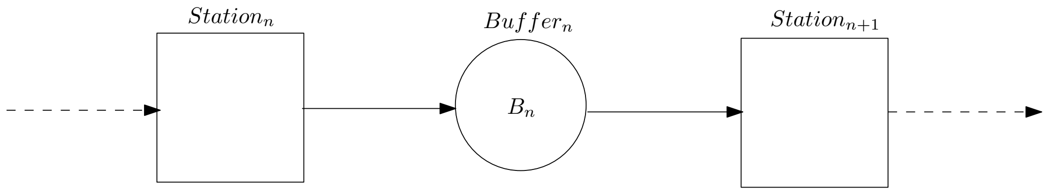


Figure 4.1: Machine-Buffer-Machine block

Each manufacturing station (machine from now-on) is described trough a DTMC having operational states associated to, possibly different, non-null rewards and failure states having null reward.

We assume that in each time slot the first machine puts parts in the buffer (if it is in an operational state) and then the second machine takes away parts from the buffer (if it is in an operational state) producing the final outcome of the machine-buffer-machine block. When the buffer is full the first machine cannot produce and the same happens to the second machine when it does not find parts in the buffer. As last, in order to fully characterize the behaviour of a machine inside the system, we need to define two features. The first is the policy with which a machine produces:

[1]A tool based on MRM for the modeling and analysis of flexible manufacturing systems is available at the url http://www.di.unito.it/~angius

- *Full batching:* the machine produces only if it is allowed to produce the maximum production of its current state. This implies that if the buffer does not have space to hosts all the produced items the machine does not produce. The same happens when the buffer is not able to supply the number of items required for the subsequent machine.

- *Partial batching:* if the machine is in an operational state, it produces the maximal quantity possible.

The second is the type of failures that we are assuming:

- *Time failures:* the machine reaches non-operational states even if its production is blocked.

- *Failure by use:* the machine can fail only after a production.

Once these properties are set, the definition of the matrices $\mathbf{P}^{(m)}$ describing the MRM model is straightforward by considering the production of the machine at the end of the line.

4.1.1 Skewness and higher order moments

As first example, we illustrate the use of higher order moments in the analysis of a machine-buffer-machine block. We assume that both machines have geometric failure times with mean time to failure equal to 100. Also for what concerns times to repair we assume that the two machines are identical but we consider three different distributions with mean equal to 10: the geometric distribution, the order two hyper-geometric distribution with coefficient of variation (CV) equal to 2, and the order two hypo-geometric distribution with CV equal to 0.5. Both machines produce one part per time slot in the up state. The capacity of the buffer is 5.

The idea behind these tests is to refute the hypothesis that the normal distribution is a good estimator of the distribution of the number of items produced at a given time unit(from now-on service level). This has been done by estimating the symmetry of the distribution through the computation of the skewness[1]. If the service level of the lines was following a normal

[1]The skewness corresponds to the standardized third moment and is equal to $\frac{E\left[(X-\mu)^3\right]}{E[(X-\mu)^2]^{3/2}}$

distribution then the skewness of both the accumulated reward and the completion time would be constant to zero. Figure 4.2 confirms our belief, i.e., in a short time horizon, the hypothesis above is a biased estimator. This is showed on the r.h.s. where all the three curves of the skewness start quite far from zero and get reasonably close to zero after time 2000 only.

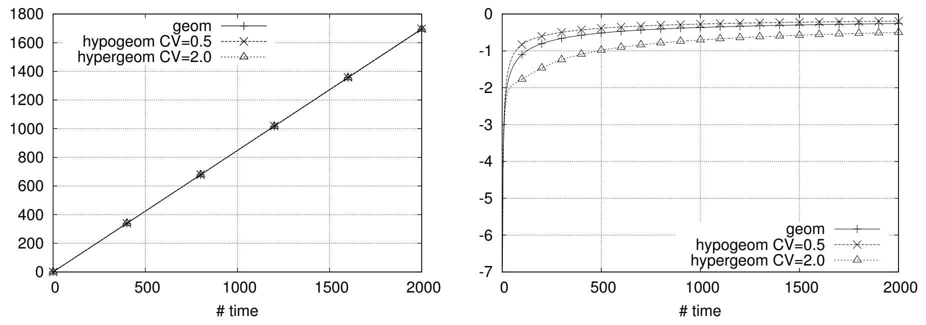


Figure 4.2: Number of produced parts as function of time for the machine-buffer-machine blocks with different coefficients of variation: mean (left) and skewness (right)

Figure 4.3 gives a quantification of the error in regards to the size of the lot. In particular, we can observe that the error can be consistent for small and medium lot sizes. Thus, the normal distribution assumption overestimates the service level leading to the risk of generating optimistic system configurations.

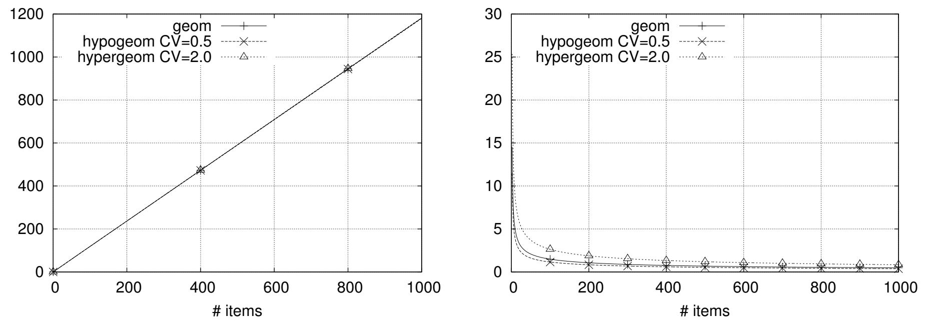


Figure 4.3: Time required to complete a task as function of the number of items for the machine-buffer-machine block with different coefficients of variation : mean (left) and skewness (right)

On the left side of the figures it is possible to observe that as far as the mean sojourn times are the same, the expected service level is independent from the CVs.

4.1.2 Systems with machines producing in batches

The second test deals with the impact of batch productions in case of full batching policy. We consider a block of two machines having failure and repair times distributed according a geometric distribution with mean as in the previous case but the size of the buffer is only 20. We tested different batch sizes and assumed two situations: in the first, the machine at the beginning of the line produces only in batch whereas the second is able to produce only one item per unit time; in the second scenario, we reverse the assumption. Figures 4.4 and 4.5 depict both the mean and the index of dispersion for the first scenario. It is possible to observe that the uncertainty of the service level decreases by increasing the batch size of the first machine although the expected production remains almost equal.

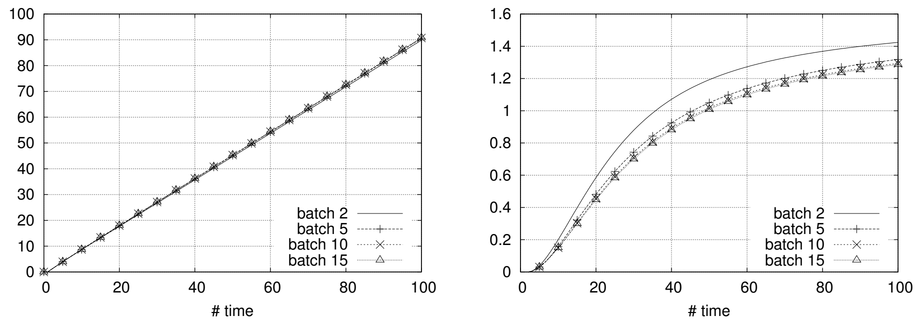


Figure 4.4: Number of produced parts as function of time for the machine-buffer-machine block with different batch sizes on the first machine : mean (left) and index of dispersion (right)

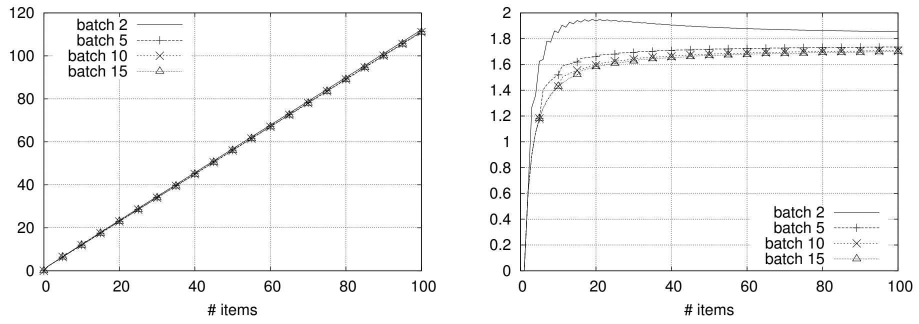


Figure 4.5: Time required to complete a task as function of the number of items for the machine-buffer-machine block with different batch sizes on the first machine : mean (left) and index of dispersion (right)

The results for the second scenario are depicted in Figures 4.6 and 4.7. The figures show that, the expected service level is characterized by step curves. In particular, they show the presence of cold points (minimal result with maximum effort) and hot points (maximum result with the minimal effort) both for short time intervals and for small lot sizes. As an example, assuming a batch size equal to 15, a lot of 16 items requires the same time as a lot composed of 30. Thus, this kind of analysis helps the decision about the dimensions of lots in such a way that they can be optimized by maximizing the number of items and minimizing the time.

The right sides of the figures show the uncertainty of the service level, it is possible to observe that the larger the batch and the more random is the production. In particular, much larger is the dimension of the batch and much larger are the spikes that characterize the accumulate reward index of dispersion.

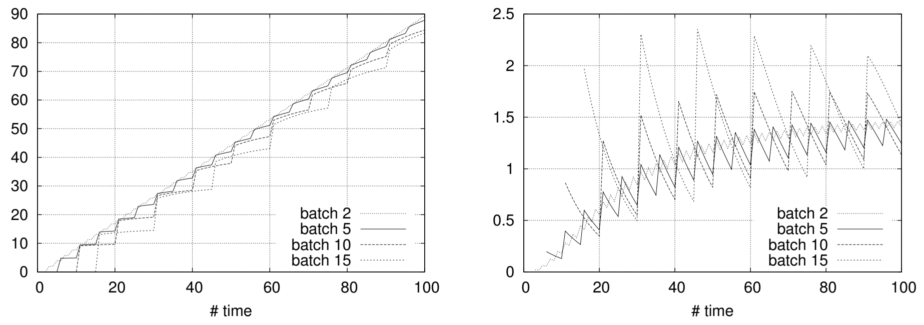


Figure 4.6: Number of produced parts as function of time for the machine-buffer-machine block with different batch sizes on the second machine : mean (left) and index of dispersion (right)

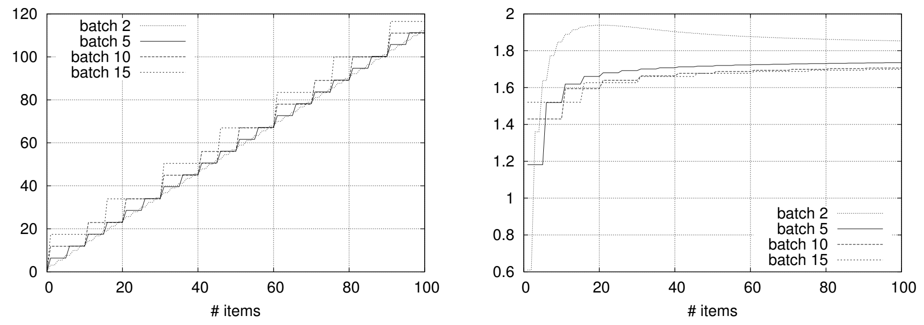


Figure 4.7: Time required to complete a task as function of the number of items for the machine-buffer-machine block with different batch sizes on the second machine : mean (left) and index of dispersion (right)

4.1.3 Two-machine line with degrading machines

In the next case, we consider a station-buffer-station block that operates according to a degradation/reparation scheme. The underlying DTMC of a station has the following transition probability matrix:

$$\mathbf{P} = \left|\begin{array}{cccccccc}
1-p & p & & & & & & \\
 & 1-p & p & & & & & \\
 & & \ddots & & & & & \\
 & & & 1-p & p & & & \\
 & & & & 1-q & q & & \\
 & & & & & 1-q & q & \\
 & & & & & & \ddots & \\
q & & & & & & & 1-q
\end{array}\right|$$

with N_u up-states and N_d down-states. We assume that the quantity produced in the up-states is $N_u, N_u - 1, N_u - 2, ..., 2, 1$, i.e., the productivity of the machines is degrading as it is passing through the up-states. Naturally, there is no production in the down-states. The machines operate in partial batching mode.

We calculated, having empty buffer and fully operational machines as initial state, the mean and the index of dispersion of the number of parts produced by the second machine for $N_u = 10, N_d = 5, p = 0.1, q = 0.5$ varying the size of the buffer B. The results are depicted in Figure 4.8 and 4.9. As expected, the mean amount of production is increasing as the buffer is enlarged. The variability of the production instead is low for low buffer size ($B = 5$), it has a sharp increase for medium buffer size ($B = 10$) and then it decreases as the buffer is enlarged ($B = 50, 100$). The non-monotone behaviour of the variability is due to the fact that both the time to complete failure and the time to repair of the machines are of low variance. The size of the state space is $(N_u + N_d)^2(B + 1)$ which is 22725 in our case for $B = 100$.

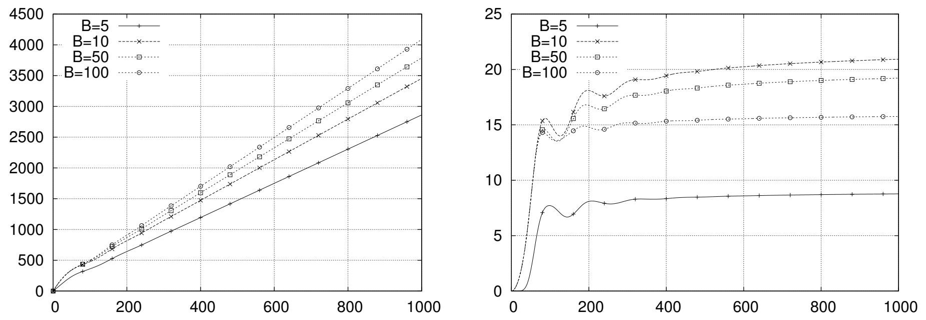


Figure 4.8: Number of produced parts as function of time for the machine-buffer-machine block with different buffer sizes: mean (left) and index of dispersion (right)

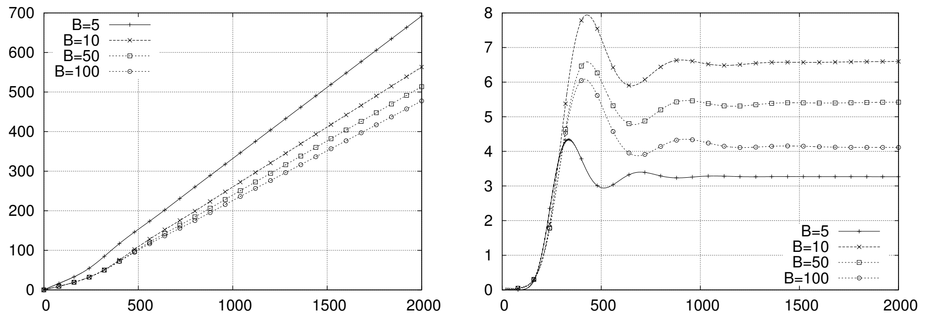


Figure 4.9: Time required to complete a task as function of the number of items for the machine-buffer-machine block with different buffer sizes : mean (left) and index of dispersion (right)

4.1.4 Larger production lines

4.1.4.1 Four machines - reversibility property for second moments

The fourth test deals with four machines with geometric failure and repair times with mean equal to 100 and 10, respectively. We test different buffer sizes.

The idea behind the test is to show that the behaviour of this kind of production lines is symmetric according to the size of the buffers. As an example. this means that a production line having three buffers with capacity 5, 5, 2 provides the same service level of a line whose buffers have been inverted, i.e. 2, 5, 5. This property, called *reversibility*, is always true when the time approaches infinity whereas in a finite time horizon it is true only if the machines start from the same state. Figures 4.10 and 4.11 illustrate the property.

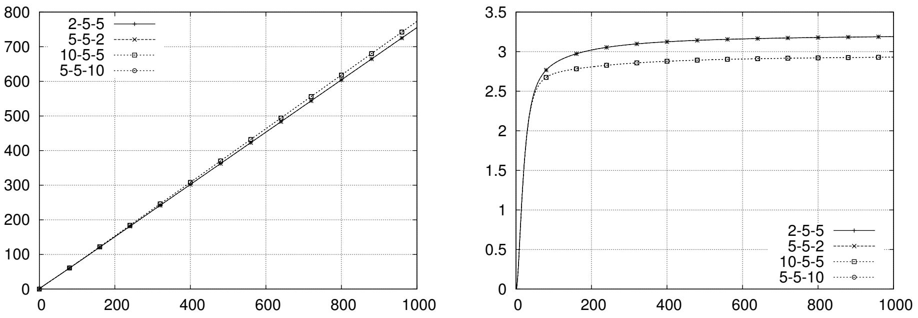


Figure 4.10: Number of produced parts as function of time for a production line composed of four machines and three buffers having sizes : mean (left) and index of dispersion (right)

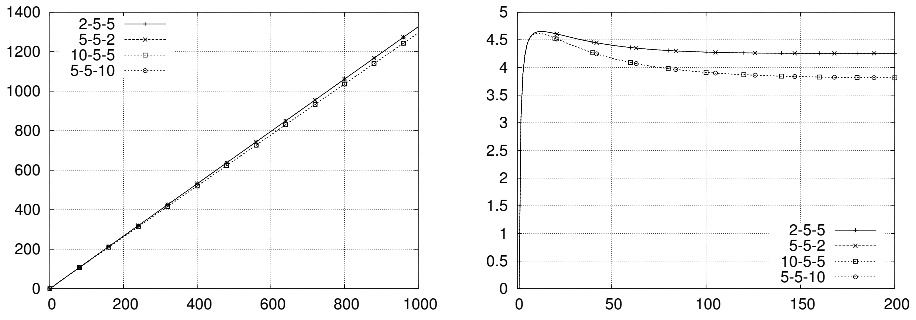


Figure 4.11: Time required to complete a task as function of the number of items for a production line composed of four machines and three buffers having sizes : mean (left) and index of dispersion (right)

4.1.4.2 Six machines - estimation completion time

In case of this last test the dimension of the model that we consider does not allow the computation of completion time. The reason is that the number of states with zero production is high and the inversion of the matrix $\mathbf{P}^{(0,0)}$ generates a dense matrix whose handling is not possible. However, we are still able to estimate the distribution of this measure by using the method described by Telek and Tari in [55, 82]. Given a r.v. X, the method takes in input a finite number of moments of X and returns an upper and lower bound for the probability $\Pr\{X = x\}$. The method works in such a way that the more x is far from the expectation and the more accurate the bounds are. Of course, the accuracy of the method (tighter intervals) improves if the number of moments increase.

We considers a model composed of six machines and all the buffers with a capacity of 5 items; hence the whole state space is composed of $2^6 \times 6^5 = 497664$ states. We computed the first 20 moments of the accumulated reward of the system. Then we used the results to compute an estimation of the accumulated reward distribution. We provide an illustration of the result in Figure 4.13, in particular:

- on left : we have the estimation of the accumulated reward c.d.f. at time 2000,
- on right : we have an estimation of the completion time c.d.f. for a lot of 2000 (the estimation has been computed by exploiting the fact that the accumulated reward is the dual problem of the completion time).

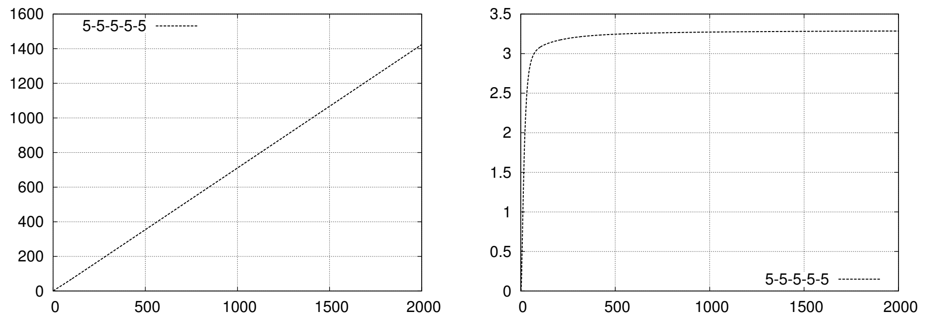


Figure 4.12: Number of produced parts as function of time for a production line composed of six machines and three buffers having sizes : mean (left) and index of dispersion (right)

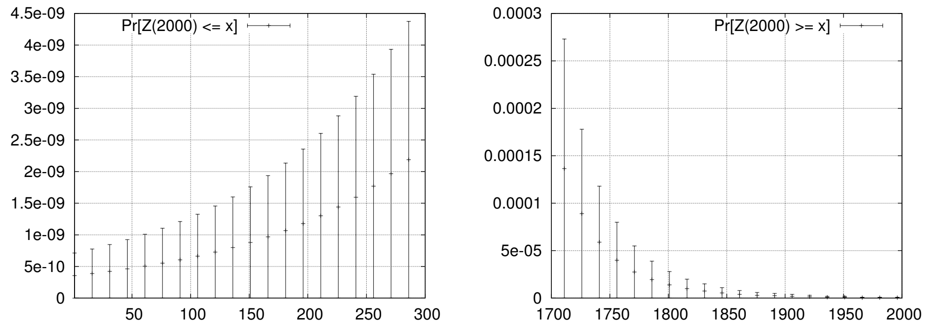


Figure 4.13: C.d.f. estimation of the accumulated reward and completion time for the case of six machines geometrically distributed with buffers capacity of 5

The computation of the moments required half an hour. Note that the computation of the moments of the accumulated reward is feasible for bigger state spaces.

4.2 Biochemical systems - rewards governed by CTMCs

In this section we provide two examples of the use of multi-reward in the context of systems biology.

The first model has been designed ad-hoc in order to show the soundness of the method. On the contrary, the second case describes a real biological phenomenon. For both the models, starting from the factorial moments we compute higher order "common" moments of all the *monotonic* places and the first joint moment $E\left[\mathcal{B}_{i,a}(t) \cdot \mathcal{B}_{i,b}(t)\right]$ for each couple of distinct *monotonic* places a, b. The firing times of both the models are defined according to the following function $\lambda_i(x) = k_i \prod_{j \in \bullet t_i} x_j$ where k_i, $1 \leq i \leq R$ are the *kinetic constants.*

4.2.1 Case 1 - Ad-hoc model

As anticipated the first model have been built ad-hoc to test the method. Despite this, to be realistic we designed the model as a composition of common biological structures. In particular, we take inspiration from enzymatic reactions whose structure makes them feasible to be analyzed by MRMs. Roughly speaking, this kind of systems describes the productions of one or more biochemical species whose growth is regulated by the concentrations of substrates and enzymes present in the system. Often blocks of reactions describing the behaviour of enzymes are combined to model a biochemical phenomenon. For the interested reader, we refer to [78] in which a high number of enzymatic reaction systems are described.

Our system, depicted in Figure 4.14, contains two enzymes, E and E_1, competing for the substrate S with which both the enzymes are able to bind. The first enzyme, E, binds with S in order to produce P. The second enzyme E_1 is able instead to generate three different species, Q, R and P, after having bound with either substrate S or B. When E_1 is bound with S, it can produce two units of Q and one unit of P and when it is bound with B a single unit of R plus one unit of P can be generated. We assume that there is such high amount of B present in the system that its quantity can be considered constant. Note that this way it is not possible to use up B and the production of R and P is potentially infinite, giving rise to an infinite state space.

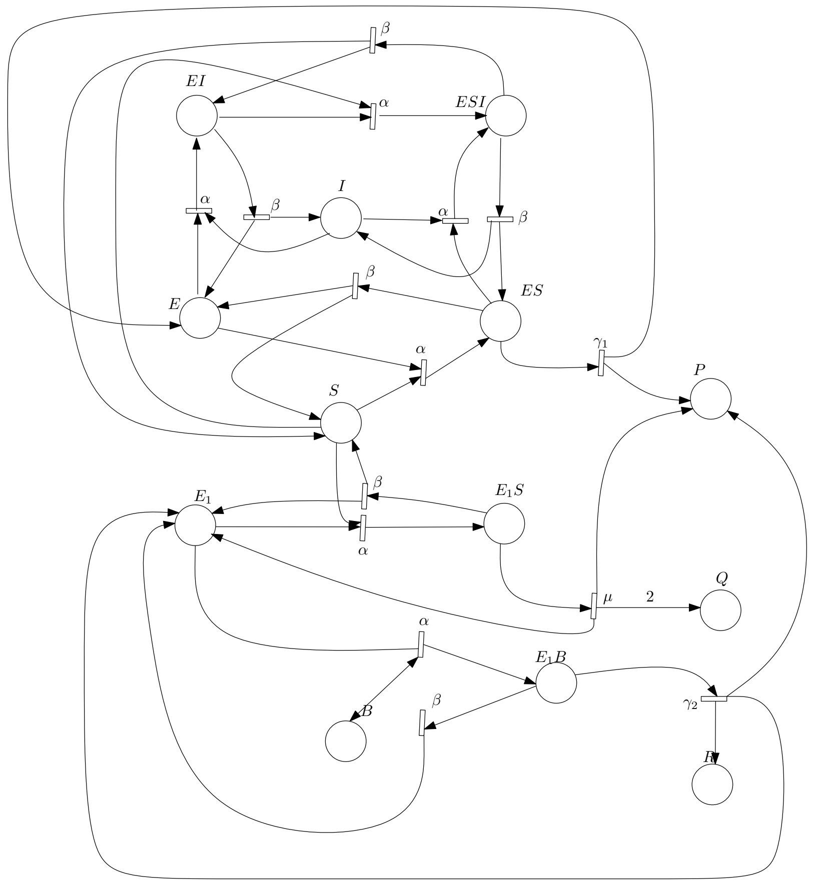


Figure 4.14: Petri net describing the ad-hoc model

A further assumption is that enzyme E can be inhibited by I in such way that the effectiveness of E gets lower in proportion of the quantity of I. The entire system is composed of 15 reactions. The intensities of the binding and unbinding reactions are $\alpha = 1$ and $\beta = 0.1$, respectively. Whereas the production reactions are with intensity $\gamma_1 = \gamma_2 = 5$ and μ whose value will be 1, 5 or 10. The initial state is $E = E_1 = 5, I = S = 50$ and 0 for the other species.

It is clear that the *monotonic* species are Q, P and R and the "productive" activities are given by the reactions with intensities γ_1, γ_2, μ. Cutting off the *monotonic* species from the state descriptor the state space becomes finite with 55076 states. We computed 20 moments for Q, P, and R and the

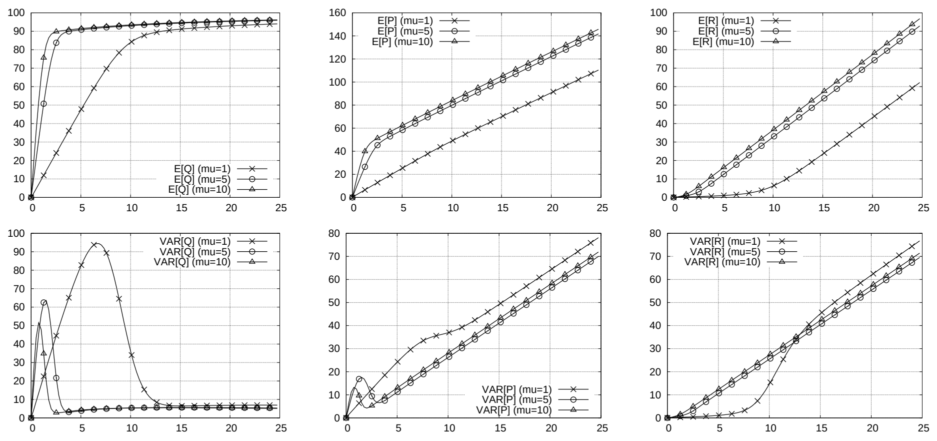


Figure 4.15: Ad-hoc model: expected values (upper part) and variances (lower part) of Q, R and P as function of time for the cases $\mu = 1, 5, 10$.

first joint moment for each couple of distinct variables in the time interval $[0, 25]$. We splitted the computation in 100 time intervals, each of them was computed in 3 minutes.

In Figure 4.15 we depicted the expectations and the variances as function of time. All the three species, Q, P and R, show faster growth for larger values of μ. This is not surprising for P and Q which are directly produced by a reaction whose intensity is μ. For what concerns R, the reason for which it grows faster with larger values of μ is the fact that with larger μ enzyme E_1 is released faster. For what concerns the variance patterns, one can observe that while there is substrate S in the system, the production of Q and R shows peaks of variability. Whereas, the variability of the production of P follows a more uniform pattern. The variance of Q returns then to lower level because its production is bounded.

The correlation coefficient (CC) is a particularly interesting measure for this kind of systems because it illustrates the degree and the kind of dependency between two species. It is computed as the ratio of the covariance of the random variables and the product of their standard deviations. In Figure 4.16 we show the CC as function of time for each couple of monotonic species. (Note that at time 0 the CC is undefined as its value is 0/0; in the figures we set to 0 the CC at time 0 and it results in some discontinuities in the curves.) One can see that the consumption of substrate S has a strong impact on the

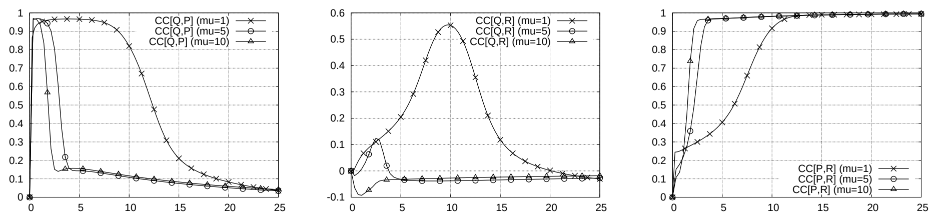


Figure 4.16: Ad-hoc model: correlation between each couple of *monotonic* species as function of time for the cases $\mu = 1,\ 5,\ 10$.

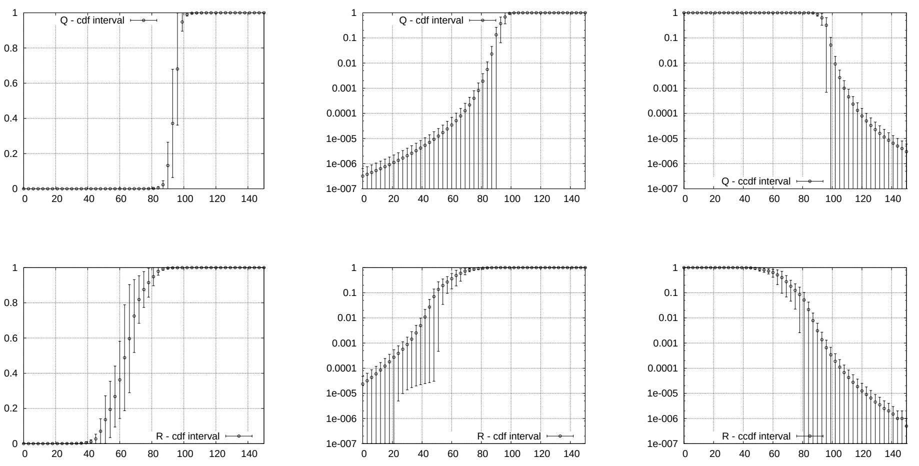


Figure 4.17: Ad-hoc model: bounds of the cdf of the quantity of Q and R at time 25; left: cdf with linear scale, middle: cdf with log scale, right: complementary cdf with log scale.

correlation between the three species of interest. In fact, while S is not used up, Q and P are strongly related and, on the contrary, the correlation between P and R is low. The situation gets inverted when the quantity of S is close to zero due to the fact that the only possible production that can occur is through reaction γ_2 which produces P and R simultaneously. For what concerns the correlation of Q and R, the only case in which significant values occur is $\mu = 1$. This correlation is due to the fact that the more Q has been produced, the more E_1 has been unbound to produce R. This effect is not visible for larger values of μ because the initial phase where there is a significant amount of S is short.

As last measure we provide the bounds of the c.d.f. The results are depicted in Figure 4.17 for Q and R for the situation after 25 time units with $\mu = 1$ (the results for P are similar to those for R). For both species we depicted the bounds of the cdf in linear scale (giving an overall view of the distribution), the bounds of the cdf with logarithmic y-axis (giving detailed view for values smaller than the mean), and the bounds of the complementary cdf with logarithmic y-axis (giving information on the tail of the distribution). Consider Q at $c = 40$, the lower bound for the cdf at this point is 0 while the upper bound is about 10^{-5}. This implies that the probability of having less than 40 units of Q at time 25 is less than 10^{-5}. As for a value larger than the mean, we can read that the probability of having more than 100 units of R at time 25 is less than 0.001.

Let us mention here that obtaining bounds similar to those depicted in Figure 4.17 by simulation has high computational cost. Consider, for example, that having more than 150 units of Q at time 25 is less than 10^{-5}. This means that we need about 10^8 simulation runs to have a reasonable estimate of having more than 150 units of Q at time 25. This requires about 10 times more time than our approach.

4.2.2 Case 2 - DFG degradation

The second model that we propose can be found in the database available on www.sbml.org, it models a real biological phenomenon with 14 biochemical species (places) interacting through 16 reactions (transitions). The system is reported in Figure 4.18. The system models the N-(deoxy-D-fructos-1-y1)-glycine (DFG) degradation pathway, and typically its transient analysis starts from the state with $DFG = n$, $n > 0$, and 0 for the other species (the numerical parameters can be found in [65]).

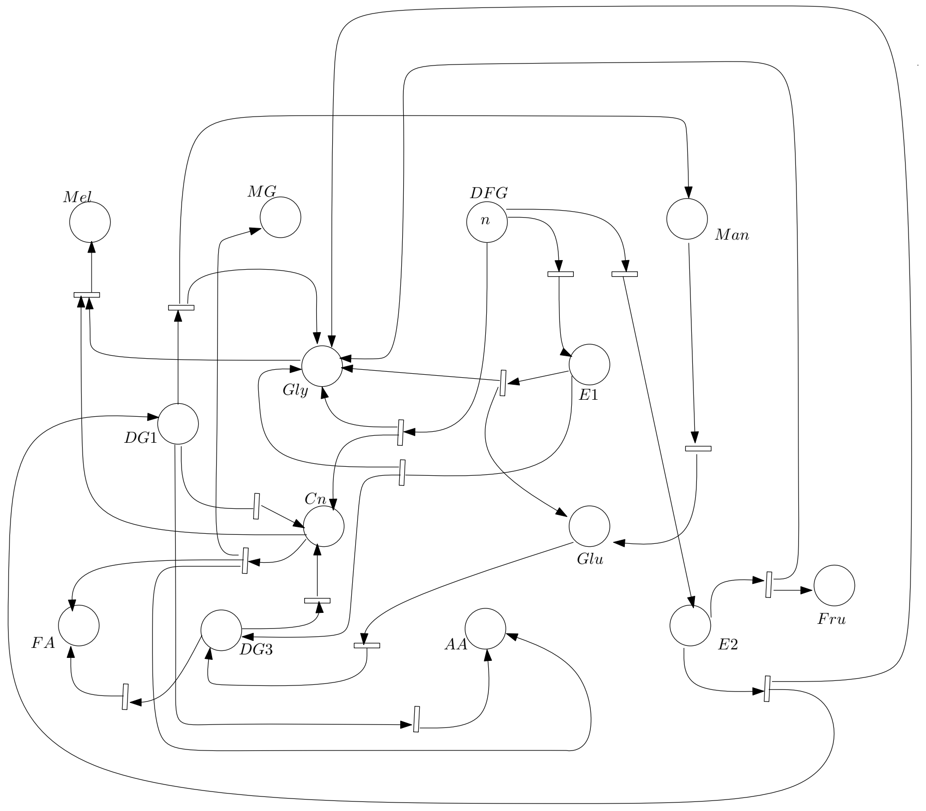


Figure 4.18: Petri net describing the DFG degradation model

In this case, the state space of the original CTMC is bounded but even for low values of n its analysis is unfeasible because of the huge number of states. By using the proposed approach, we can analyze the system using a MRM whose state space is much smaller than the state space of the original CTMC. The monotonic species are AA, FA, MG, Mel and Fru and not considering them explicitly significantly reduces the state space. It is obvious that for larger values of n even our approach can become unfeasible but in several cases our method can make the difference between unfeasible and feasible.

We used the model with $n = 13$ in which case the original state space is composed of 5.200.300 states and, by using our method, it can be reduced to 497.420. In this case, we computed the first 2 moments and all the first joint moments in 40 equidistant points in the interval $[0, 100]$. Each time point needs about 5 minutes of time. The results are depicted in Figures 4.19 and

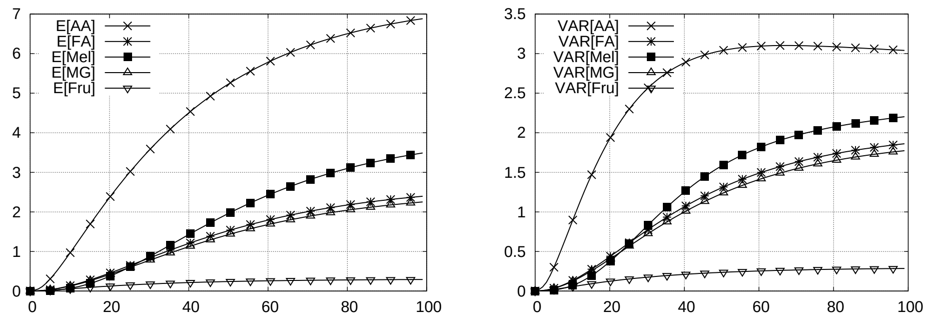


Figure 4.19: DFG Degradation: expected values (left) and variances (right) of AA, Mel, MG, FA, Fru, starting from the state $DFG = 13$.

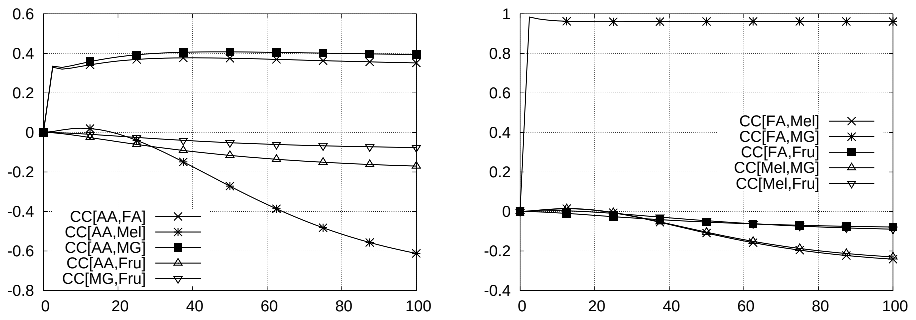


Figure 4.20: DFG Degradation: correlations between couple of *monotonic* species, starting from the state $DFG = 13$.

4.20. Based on Figure 4.20 one can figure out that species FA and MG have strong positive correlation, i.e., large (small) amount of FA implies large (small) amount of MG. Whereas, there is negative correlation between AA and Mel because there is competition between their production.

Since the original CTMC of this case study is finite, the results presented so far can be computed based on the original CTMC as well. In order to illustrate how much we gain using the proposed approach, in Table 4.1 we provide a comparison between the size of the state space of the original CTMC and that of the derived MRM.

n	CTMC	MRM
1	13	10
5	6.188	2.002
10	646.646	92.378
15	17.383.860	1.307.504

Table 4.1: Size of state space of the original CTMC and the derived MRM

Part II

Product form approximation

5 Transient product form

From basic probabilistic rules, we know that: given two random variables Y_1 and Y_2 the probability $Pr\{Y_1 = \alpha \wedge Y_2 = \beta\}$ corresponds to the product between $Pr\{Y_1 = \alpha\}$ and $\Pr\{Y_2 = \beta\}$ if and only if the two random variables are independent among each other. This notion can be easily generalized to a set of M random variables. In particular, by assuming to collect the random variables in a vector X, the following equation is straightforward

$$\Pr\{X = x\} = \prod_{i=1}^{M} \Pr\{X_i = x_i\}. \tag{5.1}$$

Indeed, this form seems particularly suitable because allows to express the distribution of a $M-$dimensional state space by considering the distribution of its components in isolation. On the other hand, it seems also quite unrealistic that such form can be useful for non-trivial purposes.

As an example, the random variables describing the number of customers present at the stations of a queuing network are collected through vectors similar to the one proposed above; however, they do not enjoy the property to be independent among each other, due to the fact that the distribution of each queue is affected by the customers that arrive from the other stations and by the overall number of customers present within the system.

Despite this, significant results have been achieved in this field by exploiting the fact that, under particular assumptions, the queue length distributions of a network behave as they were independent from each other. This property leads to the so-called *product form* which, in general, is defined as follows:

$$\Pr\{X = x\} = \frac{1}{G}\prod_{i=1}^{M} f_i(x_i). \tag{5.2}$$

where G is a *normalization constant* and the functions $f_i(x_i)$, $1 \leq i \leq M$, provides non-normalized measures describing the number of customers in each queue.

It is evident that equation (5.2) is a sort of adjustment of equation (5.1) required to manage the additional assumptions under which the stations behave as independent from each other. However, find a way to define the terms of equation (5.2) for a general queuing network is neither straightforward nor always possible.

Despite this, product form has been massively investigated in the context of queuing networks because they allow the analysis of models with large state space.

As a consequence, the literature has become richer and richer of classes of networks that enjoy this property. The most classical works in this direction include: Jackson networks [56] which are open networks with Poisson arrivals and infinite capacity, exponential servers; Gordon-Newell networks [41] which are closed networks with exponential servers; and BCMP networks [13] which can be either open, closed or mixed and can contain multiple classes of customers.

More recent works have shown that more sophisticated mechanisms preserve the product-form property as well. Some examples are: negative customers which are able to delete a customer from a queue [35, 36]; triggers which redirect other customers among the queues, catastrophes which flush all the customers out of a queue [38, 37]; resets [39] and synchronized arrivals in a set of queues [28].

Moreover, this idea has been extended also to other formalisms, such as: SPNs (see, as examples, [44, 11]) and process algebras ([79, 26]). As last, it has been shown that queuing networks can enjoy a sort of flow-equivalence property [23]. However, the works cited above, as almost all the rest of the literature, consider steady state probabilities only.

At the best of our knowledge, in the context of transient analysis the only notable exceptions are:

- [46] where Harrison and Lemoine prove that a sufficient condition to express the transient behaviour of both open and closed networks in product form is to have arrivals from outside according to Poisson processes (in case of open networks) and an unlimited number of servers in each stations independently of the service times distribution;
- [67] where Massey and Whitt extend the previous result by proving that as long as every station has an infinite number of servers, the transient product form holds also for more general models, as the *Poisson arrival location model* (PALM);
- [18, 17] where Boucherie and Taylor prove that the infinite server policy is not only the sufficient but also the necessary condition to have a transient product form in case of completely Markovian queuing networks having unitary arrivals and departures.

Since the purpose of this part of the dissertation is to use product form in context of the transient analysis of Markovian models, the works above are of great interest for our purpose. In particular, they show that the time dependent version of equation (5.2), called *transient product form* from now-on, is a simple and elegant distribution that, except for pathological cases, has not to be expected in networks where customers wait for service or can be blocked as a consequence of other events. Moreover, it seems unlikely that more general routing policies, such as congestion dependent routing, can lead to more general product forms.

As a consequence of this limited sphere of application, we decided to renounce the exactitude of the results in favor of a wider application area. Thus, our purpose is to investigate the applicability of transient product form as approximation of the transient probabilities of CTMCs.

The chapter is structured as follows. In Section 5.1, we introduce the formal definition of transient product form and the results described in [46, 67, 18, 17]. In Section 5.2, as first, we derive an ODEs system, based on the formalism of SPNs, that allows the use of transient product form whether or not the model enjoy the property; then, we list the main features of the proposed approximation. As last, in Section 5.3 we apply the approach on simple models to show that, even intended as an approximation, the applicability of the approximation is limited.

5.1 Transient product form

In this section, we show that the transient product form is strictly related to:

- networks composed of infinite servers. We focus on $M(t)/G(t)/\infty$ which are stations where customers arrive according to a, possibly time dependent, Poisson distribution and are served immediately after their arrival according to a, possibly time dependent, general distribution.
- time inhomogeneous Poisson processes.

The stochastic process associated to a $M(t)/G(t)/\infty$ station is a birth and death process with a strictly positive birth rate and a death rate which increases linearly with the number of customers inside the station. Hence, customers do not wait in queue and the station is always stable independently of the distributions according to which customers arrive and depart from the station. Because of this property, an $M(t)/G(t)/\infty$ can be interpreted as a source of Poissonian events.

As a consequence, the motion of each customer within a network composed only of infinite servers is independent from the other customers.

In the following, we exploit this fact to prove that networks composed of infinite server stations can be described as the superposition of all the Poisson processes generated by the nodes of the system and by the arrivals from the outside.

As first, we take into consideration closed networks without any assumption about the distributions of the departures times at the service stations. Then, we deal with open networks by assuming that customers arrive from the outside according to a time inhomogeneous Poisson process. The proofs follow the path suggested by Lemoine and Harrison in [46]. However, for the open networks case, we propose also the transient product form theorem provided by Massey and Whitt in [67] which gives a better picture of the measures involved in the product form. Finally, we summarize the results obtained by Boucherie and Taylor about the necessary conditions under which a model enjoys of the transient product form, [18, 17].

5.1.1 Sufficient conditions

5.1.1.1 Closed networks

In this section we follow the path described by Lemoine and Harrison in [46] to prove that closed queuing networks composed of infinite server stations enjoy the transient product form.

We consider a closed system composed of infinite server stations having Y customers which, for sake of simplicity, we assume to be at station 1 at time 0.

This is equivalent to consider a collection $\{C_1(t), \ldots, C_Y(t)\}$ of independent and identically distributed random variables such that $C_i(t) \in \{1, \ldots, M\}$ for all $t > 0$ and $C_i(0) = 1$, $1 \le i \le Y$ where each $C_i(t)$ is associated to a customer and indicates its position in the net at time t.

The independence is a consequence of the fact that customers do not interact among each other; hence, each customer behaves as it was alone within the network. On the other hand, they are identically distributed because we are assuming that all of customers are at the first station at the beginning.

The relation between the random variable describing the queue length of the jth station and the random variables representing the customers is given by the summation

$$X_j(t) = \sum_{i=1}^{Y} \delta(j, C_i(t)) \quad 1 \le j \le M,$$

where δ is equal to one if $C_i(t) = j$ and zero otherwise. The state of the system corresponds to the vector $X(t) = |X_1(t), \ldots, X_M(t)|$ and since all the elements of $C(t)$ are i.d.d., it is immediate that $X(t)$ has the multinomial distribution

$$Pr\{X(t) = x | X_1(0) = Y\} = B(x) \prod_{j=1}^{M} (Pr\{C_y(t) = j | C_y(0) = 1\})^{x_j} \qquad (5.3)$$

where y can assume any value in $1, \ldots, Y$ and $B(x)$ represent the normalization constant for the state x

$$B(x) = \frac{Y!}{x_1! x_2! \ldots x_M!}.$$

5.1.1.2 Open networks

In this section we consider the case of open systems. In particular, we assume that customers arrive according to a non-homogeneous Poisson process $A = \{A(t), t \geq 0\}$ with intensity function $\{\lambda^+(t)\}$ in such a way that the arrival process has independent increments and for $m \in 0, 1, 2 \ldots$ it satisfies the following relation

$$Pr\{A(t) = m\} = e^{-\Lambda^+(t)} \frac{[\Lambda^+(t)]^m}{m!}, \quad t \geq 0$$

with $\Lambda^+(t) = \int_0^t \lambda^+(y)dy$.

All customers are represented by a process $C = \{C(t), 0 \leq t \leq T\}$ where T is a r.v. denoting the total amount of time spent by a customer within the system. Once arrived, the initial position of a customer is assumed random, the routings independent and $E[T]$ finite (eventually customers leave the system). In order to derive the product form, let us denote the probability to find a customer at the jth station at time t with

$$\chi_j(t) = Pr\{T > t, C(t) = j\} \tag{5.4}$$

in such a way that if a customer arrives within the network at time y, then his location at time $t > y$ has the (possibly defective) distribution $\chi(t - y) = |\chi_1(t - y), \chi_2(t - y), \ldots, \chi_M(t - y)|$. The size of the defect of the distribution corresponds to the probability with which the customer has left the network. As a consequence, assuming the system to be empty initially, it is easy to show that the following measure is the expected number of customers occupying station j at time t

$$\rho_j(t) = \int_0^t \lambda^+(y)\chi_j(t - y)dy \tag{5.5}$$

and as a consequence the expected number of customers within the network is

$$\rho(t) = \sum_{j=0}^{M} \rho_j(t) = \int_0^t \lambda^+(y)Pr\{T > t - y\}dy. \tag{5.6}$$

By conditioning on $A(t) = m > 0$, the arrival times of the first m customers are distributed as the order statistics of m independent samples from the distribution on $[0, t]$ having the density function $\frac{\lambda^+(t)}{\Lambda^+(t)}$, $0 \leq y \leq t$, [27].

Thus, at time t, the location of a customer selected at random from among the first m arrivals has the defective distribution

$$\rho'_j(t) = \int_0^t \frac{\lambda^+(y)\chi_j(t-y)}{\Lambda^+(t)} dy \tag{5.7}$$

In general, if we condition on $A(t) = m$ and randomly permute the indices of the first m customers, the locations at time t of these customers are independent and identically distributed according to $\rho'(t)$.

Also $\rho'(t)$ is defective and the size of the defect is equal to the conditional probability that the customer has left the system by time t. Using these observations, one can easily write down a multinomial expression for $Pr\{X(t) = x|A(t) = m\}$ and multiply it by $\Pr\{A(t) = m\}$ where $m = x_1 + \cdots + x_M + b$. The term b represents the number of customers that have left the systems.

Finally, we arrive to the final form by summing up over all b values. In formula

$$\begin{aligned} Pr\{X(t) = x\} &= \sum_{b=0}^{\infty} Pr\{X(t) = x, A(t) = x_1 + \cdots + x_M + b\} \\ &= e^{-\rho(t)} \prod_{j=1}^{M} \frac{\rho_j(t)^{c_j}}{c_j!}. \end{aligned} \tag{5.8}$$

Thus, the number of customers occupying station i at time t has a Poisson distribution with mean $\rho_i(t)$, and the lengths of the queues are independent random variables.

In [67], Massey and Whitt arrive to the same conclusions and provide additional details. In particular the authors:

- specify equation (5.5) in a form that considers explicitly the arrivals from the other stations and prove that the overall equations system has a unique integrable solution,
- show that not only the queue lengths follow a Poisson distribution but also the external departure process.

Since the measures described in the following theorem are considered immediately after a departure from a station, we need to introduce the definition of *stationary excess*:

Definition 7 *The distribution of a r.v. X_e is the stationary excess of the r.v. X if and only if $Pr\{X_e \leq t\} = \frac{Pr\{X \geq t\}}{E[X]}$; For example, if the intervals between successive bus arrivals at a bus stop are independent and identically distributed according to X, then in the long run the time that a person arriving at the bus stop (independent of the arrival process) must wait for the next bus is distributed according to X_e [66].*

Theorem 9 *Given a network composed of $M(t)/G/\infty$ stations only, with stationary Markovian routings, and a non-negative deterministic external arrival rate function $\alpha(t)$ governing a non-homogeneous Poisson process. The queue lengths $X_i(t)$, $1 \leq i \leq M$ are independent Poisson random variables with finite means*

$$E\left[X_i(t)\right] = E\left[\int_{t-S_i}^{t} \lambda_i^+(u)du\right] = E\left[\lambda_i^+(t - S_{i,e})\right] E[S_i] \tag{5.9}$$

where S_i represents the r.v. of the service time at queue i and $S_{i,e}$ its stationary excess. The term $\lambda_i^+(t)$ is the aggregate arrival rate function to queue i, defined as the unique integrable solution to the system of input equations

$$\lambda_i^+(t) = \alpha(t)r_{0,i} + \sum_{j=1}^{M} E\left[\lambda_j^+(t - S_j)\right] r_{j,i} \quad, 1 \leq i \leq M. \tag{5.10}$$

where the constants $r_{i,j}$ denotes the routing probabilities.

In addition, the vector $X(t)$ is independent of the external departure processes before time t that are in themselves Poisson processes with integrable time dependent rate functions

$$\delta_i(t) = E\left[\lambda_i^+(t - S_i)\right] r_{i,0} \quad, 1 \leq i \leq M. \tag{5.11}$$

Moreover, if the routings are acyclic then also the aggregated arrival (departure) processes from (to) queue i, i.e. counting arrivals from (departures to) other stations as well as from (to) the outside, are Poisson processes.

In the same work, on the basis of the theorem above, the authors show that more complex networks based on infinite server stations enjoy the transient product form. An example is the *Poisson arrival location model* (PALM), where:

- a customer's location is specified by a continuous-time stochastic process with values in a general state space $\mathcal{S}$ where the queues are associated with subsets of $\mathcal{S}$; hence, the model can have an infinite collection of queues.

- Time dependent arrivals, and departures are stochastically dependent both on the routing and the time.

Then, the authors show that by applying more constraints to the model and going towards common discrete queuing networks a larger number of measures can be computed easily.

We want to point out the strength of the formulas (5.3) and (5.8). These results allow the computation of the whole distribution of a network by computing only the expectation of the number of customers within every station according to a linear differential equation system. This fact implies that the overall computational complexity of the analysis grows linearly with the number of stations that compose the system. As a consequence the computation of the product form is extremely cheap.

Furthermore, it is worth to say that:

- the model is more general than it might appear because the concept of "queue" has to be intended as classes which include other customer attributes as well as location.

- the proofs assume particular initial states but these conditions can be circumvented by considering different conditions separately. For instance, for the open network case, we can consider the customer in the system separately from the new arrivals.

As last consideration, we want to point out that, in the context of CTMCs, a network composed only of infinite servers belongs to the *density dependent family* of processes defined by Kurtz in [61]. For this reason, the equations describing the expectations of the number of customers within the stations corresponds to the well-known limit provided by Kurtz's theorem.

5.1.2 Necessary conditions - Boucherie's results

In this section we summarize the results obtained by Boucherie and Taylor about the necessary conditions to enjoy a product form in the context of queuing networks.

The work described by Boucherie and Taylor in [18] considers Markovian queuing networks only. Based on the formulas provided in the previous section, the authors derive a *canonical form* to express the transient probabilities of a system distribution as a product. This form allows to prove the following theorem

Theorem 10 *The only "real" queuing networks whose transient distribution can be expressed in product form are infinite server networks.*

Note that systems of queues that are not "really" networked can have product form transient distribution. An example is a network of disconnected $M/M/1$ stations whose marginal distributions are independent and then can be expressed as a product; despite this, it does not enjoy a transient distribution in form as those proposed in Section 5.1.1.

Another pathological case is the one of closed networks of $M/M/K$ stations with a number of customers $N \leq K$. Indeed, in this case we have a transient product form distribution but only because the system behaves as a $M/M/\infty$ network. As last, the theorem takes into account stable queues only (arrival rate that eventually are smaller than the service rate). In fact, it is long time known that stations whose queue length tends to infinity behave as a source of Poissonian events. Thus, in this limit situation, also finite servers stations can enjoy transient product form.

In [17], Boucherie provides also an extension of the theorem above by showing that:

Theorem 11 *A transient product form cannot exists for queuing networks where customers are blocked or lost.*

This last theorem leaves few chances to extend Theorem 10 to a general context and motivated us to investigate product form as approximation.

We briefly mention that in [17] it has been shown that a specific case of the Engset loss model, which involves customers blocking, can be solved as a sum of product forms. This is an interesting result that motivates further research for specific closed forms but does not increase much the possibility to have a general product form that holds for a larger set of models.

5.2 Transient product form approximation

5.2.1 Definition of the approximation

In this section, we derive an approximation, called *product form* approximation (PF), based on the assumption that the transient probabilities of the nodes of the system are in product form.

The first step to derive the PF approximation is the characterization of the terms of the product. Since we derive the equations according to the formalism of SPNs, each term corresponds to the probability that at time t there are x_m tokens in place m, $1 \leq m \leq M$. We derive these terms from the proper summation of the Chapman-Kolmogorov equations. In fact, the probability to find x_m tokens in place m at time t satisfies

$$\frac{dPr\{X_m(t) = x_m\}}{dt} = \frac{d\sum_{\substack{|y_1,\ldots,y_M|:\\ y_m = x_m}} \pi_y(t)}{dt} = \sum_{\substack{|y_1,\ldots,y_M|:\\ y_m = x_m}} \frac{d\pi_y(t)}{dt},\ 1 \leq i \leq M. \tag{5.12}$$

From the equation above, the substitution of the rates reported in equation (2.23) leads to

$$\begin{aligned}\frac{dPr\{X_m(t) = x_m\}}{dt} = &- \sum_{\substack{|y_1,\ldots,y_M|:\\ y_m = x_m}} \pi_y(t) \sum_{n:y \geq i_n^-} \lambda_n(y) \\ &+ \sum_n \sum_{\substack{|y_1,\ldots,y_M|:\\ y - e_n \geq i_n^- \wedge\\ y_m = x_m}} \pi_{y-e_n}(t)\lambda_n(y - e_n).\end{aligned} \tag{5.13}$$

Since we are assuming that token distributions are in product form among each other, every term $\pi_y(t)$ can be decomposed. As a consequence, if the product form assumption holds then we have

$$\frac{dPr\{X_m(t)=x_m\}}{dt} =$$
$$- Pr\{X_m(t)=x_m\}\left(\sum_n \sum_{\substack{|y_1,\dots,y_M|:\\ y\geq i_n^-\wedge\\ y_m=x_m}} \prod_{i\neq m} Pr\{X_i(t)=y_i\}\lambda_n(y)\right)$$
$$+\sum_n Pr\{X_m(t)=x_m-e_{n,m}\} \sum_{\substack{|y_1,\dots,y_M|:\\ y-e_n\geq i_n^-\wedge\\ y_m=x_m}} \prod_{i\neq m} Pr\{X_i(t)=y_i-e_{n,i}\}\lambda_n(y-e_n) \tag{5.14}$$

where we isolated the terms that are not influenced by the summation on y.

Then, we can observe that each summation on y corresponds to the expected intensity of a transition. As a consequence, since every transition n depends only on the places composing the set ${}^\bullet t_n$ (the set of places which determine its enabling), the following relation holds

$$\sum_{\substack{|y_1,\dots,y_M|:\\ y\geq i_n^-\wedge\\ y_m=x_m}} \prod_{i\neq m} Pr\{X_i(t)=y_i\}\lambda_n(y) = \sum_{\substack{|y_1,\dots,y_M|:\\ y\geq i_n^-\wedge\\ y_m=x_m}} \prod_{i\in {}^\bullet t_n\setminus m} Pr\{X_i(t)=y_i\}\lambda_n(y) \tag{5.15}$$

because if $i\notin {}^\bullet t_n$ then the terms $Pr\{X_i(t)=y_i\}$ sum to one.

By denoting with $\Lambda^m(x)$ the set of transitions that depends at the most on place m and with $\Lambda^{\overline{m}}(x)$ its complement, we have

$$\frac{dPr\{X_m(t)=x_m\}}{dt} = -Pr\{X_m(t)=x_m\} \sum_{\substack{x_m \geq i^-_{n,m} \wedge \\ n \in \Lambda^m}} \lambda_n(x)$$

$$- Pr\{X_m(t)=x_m\} \sum_{n \in \Lambda^{\overline{m}}} \left(\sum_{\substack{|y_1,\dots,y_M| : \\ y \geq i^-_n \wedge \\ y_m = x_m}} \prod_{i \in {}^\bullet t_n \setminus m} Pr\{X_i(t)=y_i\} \lambda_n(y) \right)$$

$$+ \sum_{\substack{n \in \Lambda^m \wedge \\ x_m - e_{n,m} \geq i^-_{n,m}}} Pr\{X_m(t) = x_m - e_{n,m}\} \lambda_n(x - e_n)$$

$$+ \sum_{n \in \Lambda^{\overline{m}}} Pr\{X_m(t) = x_m - e_{n,m}\} \Bigg($$

$$\sum_{\substack{|y_1,\dots,y_M| : \\ y - e_n \geq i^-_n \wedge \\ y_m = x_m}} \prod_{i \in {}^\bullet t_n \setminus m} Pr\{X_i(t) = y_i - e_{n,i}\} \lambda_n(y - e_n) \Bigg). \tag{5.16}$$

Equation (5.16) gives a clear interpretation of how the approximation works; i.e. the distribution of a place depends on the events generated by the transitions triggered by itself in an exact way (first and third term), whereas the intensities of the transitions depending also on the other places are considered from the point of view of the expected contribution that each place gives to the transition at time t (second and fourth term).

The ODEs system composed of the equations (5.16) can be interpreted as M interacting, time inhomogeneous Markov chains in which each chain corresponds to the number of tokens present in the place. The number of tokens can be increased or decreased by two type of signals: the first, time independent, that represent the firing of transitions that depend at the most on the place in itself; whereas the second, time dependent, that are generated with the contribution of the others places. As a consequence, the signals generated by places different from m, $1 \leq m \leq M$, are perceived by the distribution m as generated from an inhomogeneous Poisson process.

Denoting with Y_i, $1 \leq i \leq M$, the maximum number of tokens reachable by the ith place, it follows that the total number of equations that describe the approximation is $\sum_{m=1}^{M}(Y_m + 1)$.

Numerical solution techniques developed for time inhomogeneous Markov chains, like the one proposed in [7], can be applied to calculate the transient probabilities.

The approach is able to deal with huge state spaces due to the fact that the overall ODE system grows linearly with the number of places.

5.2.2 Consistency with transient product form networks

In this section, we prove that PF approximation is consistent with the definition of transient product form when the network enjoys the property. In particular, we consider the open network case.

In this context, it is evident that: given the mth station, its set Λ^m is composed of the departures from it and the arrivals from the outside of the network, whereas the set $\Lambda^{\overline{m}}$ contains all the arrivals from the other queues. Thus, by assuming infinite server policy for all the queues we can re-write equation (5.16) as follows

$$\begin{aligned}
\frac{dPr\{X_m(t) = x_m\}}{dt} &= -Pr\{X_m(t) = x_m\}\Bigg(\lambda_m + (1 - r_{m,m})\, x_m \mu_m \\
&+ \sum_{i=1, i\neq m}^{M} \sum_{\substack{|y_1, \ldots, y_M| : \\ y_m = x_m}} Pr\{X_i(t) = y_i\} y_i \mu_i r_{i,m}\Bigg) \\
&+ Pr\{X_m(t) = x_m + 1\}\, (x_m + 1)\, \mu_m (1 - r_{m,m}) \\
&+ Pr\{X_m(t) = x_m - 1\} \Bigg(\lambda_m + \sum_{i=1, i\neq m}^{M} \sum_{\substack{|y_1, \ldots, y_M| : \\ y_m = x_m - 1}} Pr\{X_i(t) = y_i\} y_i \mu_i r_{i,m}\Bigg)
\end{aligned} \tag{5.17}$$

where the states having a negative number of customers in queue have null probability.

To prove that in this case PF approximation is equal to the transient product form, we have to show that the mean value of each marginal distribution corresponds to the parameters of the inhomogeneous Poisson processes generated by the stations. Since we know that the parameters corresponds to the expected values, we apply the following summation.

$$\sum_{k=0}^{\infty} k \frac{dPr\{X_m(t)=k\}}{dt} = \sum_{k=0}^{\infty} k \Bigg(-Pr\{X_m(t)=k\} \Bigg(\lambda_m + (1-r_{m,m})\, k\mu_m$$
$$+ \sum_{i=1, i\neq m}^{M} \sum_{\substack{|y_1,\ldots,y_M|:\\ y_m = k}} Pr\{X_i(t)=y_i\} y_i \mu_i r_{i,m} \Bigg)$$
$$+ Pr\{X_m(t)=k+1\}(k+1)\mu_m(1-r_{m,m})$$
$$+Pr\{X_m(t)=k-1\} \Bigg(\lambda_m + \sum_{i=1, i\neq m}^{M} \sum_{\substack{|y_1,\ldots,y_M|:\\ y_m = k-1}} Pr\{X_i(t)=y_i\} y_i \mu_i r_{i,m} \Bigg)\Bigg) \tag{5.18}$$

It is straightforward that equation (5.18) corresponds to the derivative of the mean number of customers present at the mth station. Additionally, by using methods similar to those applied in Chapter 3, it is possible to prove that the following equation is satisfied

$$\frac{dE\left[X_m(t)\right]}{dt} = \lambda_m - E\left[X_m(t)\right]\mu_m(1-r_{m,m}) + \sum_{i=1, i\neq m}^{M} E\left[X_i(t)\right]\mu_i r_{i,m}. \tag{5.19}$$

Thus, by applying Laplace transform, we obtain:

$$\begin{aligned} &sE^*\left[X_m(s)\right] - E\left[X_m(0)\right] = \\ &\lambda_m - E^*\left[X_m(s)\right]\mu_m(1-r_{m,m}) + \sum_{i=1, i\neq m}^{M} E^*\left[X_i(s)\right]\mu_i r_{i,m}. \end{aligned} \tag{5.20}$$

By rearranging the terms of equation (5.20) and applying basic Laplace transform properties, the following relation arises

$$E\left[X_m(t)\right] = \int_0^t \left(E\left[X_m(0)\right] + \lambda_m + \sum_{i=1, i\neq m}^{M} E\left[X_i(y)\right]\mu_i \right) e^{\mu_m(1-r_{m,m})(t-y)} dy \tag{5.21}$$

that, by assuming the term $E\left[X_m(0)\right]$ equal to zero, is exactly the measure $\rho_m(t)$ as defined in equation (5.5). Thus, if the network enjoys transient product form the PF is exact (but of course its use would be more expensive that the direct application of the formula provided in the previous section).

5.2.3 Limiting behaviour

In principle, the steady state determined by the PF assumption can be calculated by setting the left hand side of (5.16) to zero and solving the resulting set of equations. In practice, this is not feasible because the set of equations is not linear and the number of equations can be large. Exact steady state is determined by the Chapman-Kolmogorov equations by setting the left hand side to zero.

One can observe that the right hand side of (5.16) contains summations of the right hand side of Chapman-Kolmogorov equation for given set of states. This implies that the limiting behaviour of the PF approximation is such that it satisfies sums of those equations that determine the exact steady state. Moreover, if the exact steady state is uniquely determined by these sums of equations then the limiting behaviour of the approximation is exact.

5.2.4 From queuing networks to SPN

In this section, we deal with the features provided by the expressive power of SPNs. In particular, we investigate the effect of *synchronizations*. Synchronizations are those transitions that, having two or more places in input, generate events that depend on the local situation of more places.

It is worth to spend few lines about this discussion because theorems about transient product form deal only with queuing networks where events are triggered at the most by one component of the net.

As described in equation (5.16) if the transition i is a synchronization we have to take in consideration the overall probability with which all the places belonging to $^\bullet t_i$ contain a sufficient number of tokens to enable the transition.

Although, this is not a problem as long as we can express the probability to find transition i enabled as a product of marginal probabilities, the fact that an event depends on two or more places (and potentially changes their states) is in clear contrast with the assumption that they are independent among each other.

Let us explain the concept by introducing the following simple example:

Example 7 *Consider a Petri net composed of two places and three transitions where t_1 and t_2 consume a token from P_1 with rate μ_1 and one from P_2 with rate μ_2, respectively. Transition t_3 instead is a synchronization and removes a token from both the places with a single firing that occurs with rate μ_3 (Figure 5.1).*

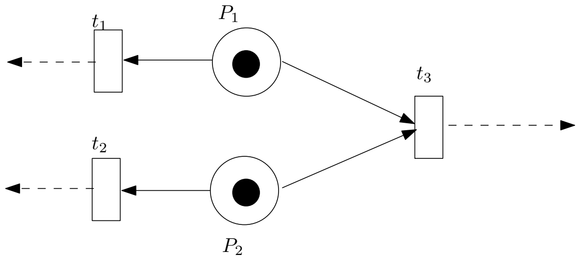


Figure 5.1: Example of synchronization.

We further assume that no arrivals are possible and only a token is present in P_1 and P_2 in order to keep small the number of terms with which we will deal with. This implies that the state of the system $X(t)$, $t \geq 0$, is composed of two random variables and can assume only four configurations, i.e. $|1,1|$, $|1,0|$, $|0,1|$ and $|0,0|$.

In spite of its simplicity the time dependent distribution of the net cannot be expressed in product form due to the presence of t_3.

The proof arises from the fact that the distribution of P_1 is independent of P_2 if and only if

$$Pr\{X_2(t) = x_2\} = \frac{Pr\{X_1(t) = x_1, X_2(t) = x_2\}}{Pr\{X_1(t) = x_1\}}$$

but the probability to find a token in one of the two places, let's say P_1, is equal to

$$\begin{aligned} Pr\{X_1(t)=1\} &= Pr\{X_1(t)=1, X_2(t)=1\} + Pr\{X_1(t)=1, X_2(t)=0\} \\ &= e^{-(\mu_1+\mu_2+\mu_3)t} + e^{-\mu_1 t}\left(1-e^{-\mu_2 t}\right), \quad \forall t \geq 0 \end{aligned}$$

which implies

$$e^{-(\mu_1+\mu_2+\mu_3)t} + e^{-\mu_2 t}\left(1-e^{-\mu_1 t}\right) = \frac{e^{-(\mu_1+\mu_2+\mu_3)t}}{e^{-(\mu_1+\mu_2+\mu_3)t} + e^{-\mu_1 t}\left(1-e^{-\mu_2 t}\right)}$$

that is satisfied only for the trivial cases $t = 0$ and $\mu_3 = 0$.

We proved that the net depicted in Figure 5.1 does not enjoy a time dependent product form as long as the transition that synchronizes the two places is part of the model.

Indeed this is a particular case, but the same reasoning can be applied to fit any Petri net having synchronizations. For this reason, if synchronizations are involved we exclude to achieve an exact solution by applying the product form.

Despite this, in the following tests, we apply PF approximation to models involving this type of transitions in order to evaluate its applicability as approximation.

Other features provided by SPNs and not considered explicitly by transient product form theorems are:

- transitions triggered by arcs having a multiplicity greater than one: at least one entry of vectors i_n^-, $1 \leq n \leq R$, is greater than one. They are a particular case of synchronizations where a place synchronize with itself. In particular, if, for instance, a transition requires two tokens to be enabled, then the first token that arrives to the place is blocked until a second token arrives.

- bulk arrivals: at least one entry of vectors i_n^+, $1 \leq n \leq R$, is greater than one.

- forks: more than one entry of vectors i_n^+, $1 \leq n \leq R$, is greater than zero.

Also in these cases, we do not expect a transient product form even if the transitions are defined as infinite server. However, by observe that:

- the mentioned PALM model described by Massey and Whitt goes far beyond the concept of single arrivals to the stations and still preserves transient product form due to the fact that it can be considered as a complex superposition of *generalized Poisson processes*;
- the unicity of the solution of equation (5.10) is not precluded by using bulk arrivals and forks because the intensities of the arrivals from other stations are scaled of constant factors;
- the motion of the customers (or tokens) are still independent among each others as long as only infinite server transitions are involved.

On the basis of these observations, we conjecture that if all the transitions are infinite server and they are not synchronizations then a stochastic Petri net could enjoy a (maybe complex) time dependent product form.

5.3 Application of PF approximation

In the following, we test PF as approximation of the transient behaviour of CTMCs by moving step by step away from the conditions that satisfy the Boucherie-Taylor's theorems.

We aim to show those situations that, above all, characterize our approximation until we arrive to a case in which the results are rather poor. This case will constitute the motivation for developing a more flexible product form approximation whose final form will be described in the next chapter.

The following investigation is based on three models for which we provide, in numerical illustrations, the comparison between the results obtained by solving the original CTMCs and those approximated through PF.

5.3.1 Stations with finite number of servers

The slightest violation of Boucherie-Taylor's theorem is represented by a network having one or more stations with a finite number of servers.

For this reason, as first model we consider a network, depicted in Figure 5.2, composed of six $M/M/1$ stations. The net represents an on demand production system where customer requests arrive with rate λ at station 1 where they are processed and sent to the production stations with service rate μ_1. In the optimal case, each customer request needs only two production

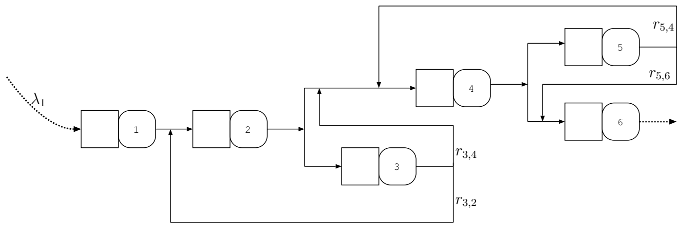


Figure 5.2: On demand production system.

steps that are performed at station 2 and 4 with rate μ_2 and μ_4 , respectively. The model considers also a backlogging scenario in which, after a service at station 2 (station 4), each semi-finished product is subject to a quality check which is not successful with probability $r_{2,3}$ ($r_{4,5}$). When the quality check at station 2 (4) is not satisfied the semi-finished product is sent to station 3 (5) where it is monitored with rate μ_3 (μ_5).

If the problem is a false positive or fixable, then the product is reintegrated at the next station of the production line with probability $r_{3,4}$ ($r_{5,6}$). Otherwise the production request is sent back to station 2 (4). At the last station, the final product is delivered to the customer with rate μ_6.

We assumed $r_{2,3} = r_{4,5} = 0.3$, $r_{3,2} = r_{5,4} = 0.8$, $\mu_1 = \mu_2 = \mu_4 = \mu_6 = 1$, and $\mu_3 = \mu_5 = 1.5$ and tested two values of λ, i.e. 0.6 and 1. In order to provide a comparison we considered the original CTMC as well but, since its state space is huge, we were not able to solve this model numerically; thus, we used Monte Carlo simulations instead.

In Figure 5.3 we compare the results obtained by PF approximation (empty dots) and results obtained from the simulations of the CTMC (filled dots) by starting from the state in which all the queues are empty. We generated 2×10^6 simulation runs which required about 5 minutes and provide estimates for the variance with visible uncertainty (see, for example, the curve corresponding to station 4 in Figure 5.3).

For the computation of the product form approximation we truncated the queues in such a way that the maximal number of clients at a station is 50. Accordingly, the number of ODEs is $6 \times 51 = 306$. The computations took a second.

It is evident from Figure 5.3 that in both cases the product form leads to an accurate estimation of the expectations of the queue length whereas, especially for the case with $\lambda = 1$, the approximation of the variance is

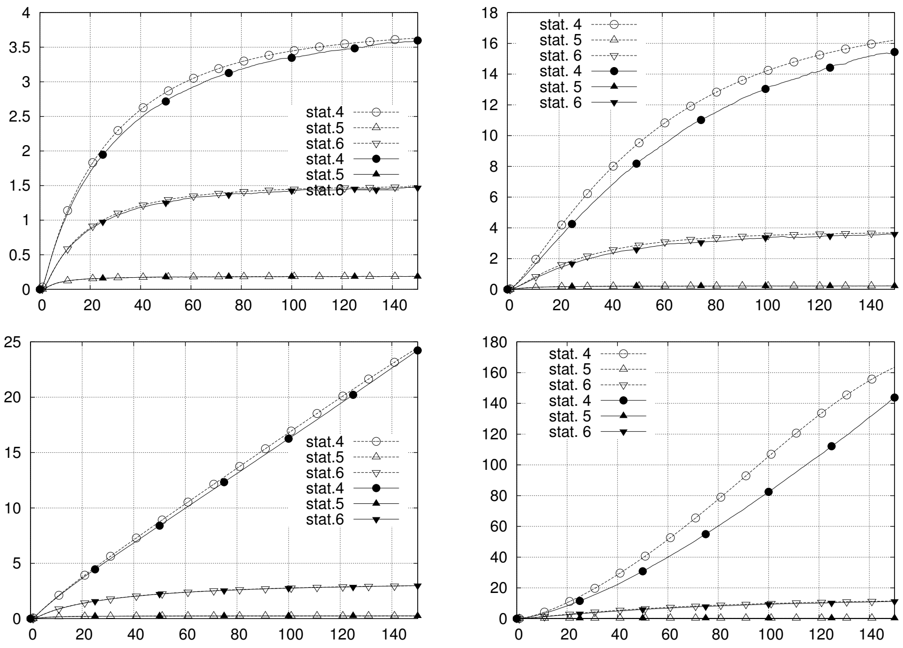


Figure 5.3: On-demand production system: expectations (left) and variances (right) of the number jobs at stations 4, 5, and 6 with $\lambda = 0.6$ (top) and $\lambda = 1$ (bottom) parameters.

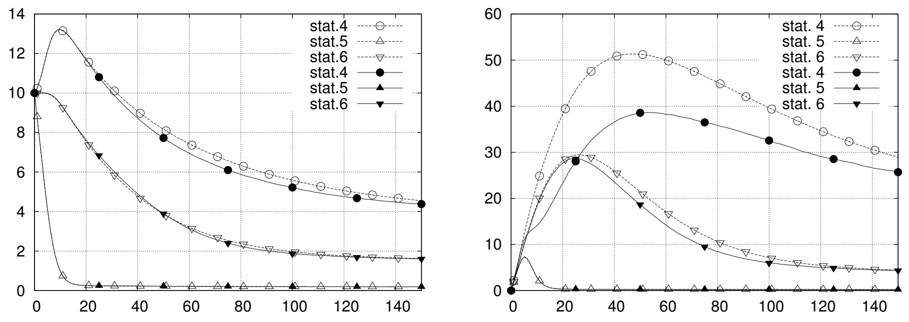


Figure 5.4: On-demand production system: expectations (left) and variances (right) of the number jobs at stations 4, 5, and 6 with $\lambda = 0.6$ and starting from the state $[0, 0, 0, 10, 10, 10]$.

slightly worse although it still follows the original curve.

The results we did not depict (i.e., expectations and variances for stations 1, 2 and 3) are more precise than those depicted in Figure 5.3.

In order to put under stress our method, we tested a third scenario in which the last three stations do not starts empty but with 10 customers in their queue (and $\lambda = 0.6$).

Thus, at the beginning, we should have an adverse situation for our method since a high number of customers wait in queue.

Figure 5.4 shows that the approximation is still able to represent the behaviour of the original curves although the distance between the approximated and the simulated variances appears more pronounced.

Looking at the results presented above, the use of the product form seems an appealing strategy to approximate the transient probabilities of Jackson networks.

However, this class of networks is able to generate processes whose events depend mostly on the local situation of each queue. Indeed, this context facilitates our approximation since stations cannot be strongly dependent among each other. For this reason, in the next section we start to work on more complex models where events depend on more general conditions.

5.3.2 Synchronizations and blocking mechanisms

In the following we provide tests made on two models involving a synchronization. The first in which the only synchronization can always fire in the future whereas the second in which, moving forward in time, the probability to find the synchronization enabled tends to zero causing a livelock of the system.

Consider the Petri net depicted in Figure 5.5 and assume that every transition i has firing times exponentially distributed with state dependent parameters given by the function $\lambda_i(x) = k_i \prod_{j \in \bullet t_i} x_j$ where k_i, $1 \leq i \leq R$, are strictly positive constants[1]. By construction, the token distribution of

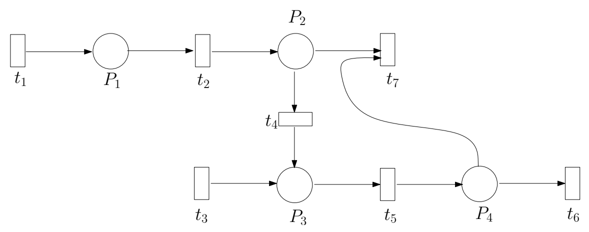


Figure 5.5: Petri net having a synchronization .

places P_1 and P_3 behaves as a $M/M/\infty$ station. On the other hand, P_2 and P_4 have a slightly different behaviour due to transition t_7 which allows that two tokens, one taken from P_2 and the other from P_4, leave the net at the same time. Of course, as long as t_7 is disabled, tokens depart from P_2 and P_4 as they would do in a $M/M/\infty$ station. As an example of the equations that compose our approximation, consider the probability of having n tokens in

[1]Note that this interpretation of infinite server do not corresponds to the canonical definition of infinite server transition of Stochastic Petri nets whereas it is widely used in the context of biochemical systems as defined by [40].

P_2. Then, by applying equation (5.16) we have

$$\frac{dPr\{P_2(t)=n\}}{dt} = -Pr\{P_2(t)=n\}\left(nk_4 + \left(k_2 \cdot \sum_{j=1}^{\infty} j \cdot Pr\{P_1(t)=j\}\right)\right.$$
$$\left. + \left(k_7 \cdot n \cdot \sum_{j=1}^{\infty} j \cdot Pr\{P_4(t)=j\}\right)\right)$$
$$+ Pr\{P_2(t)=n-1\}\left(k_2 \cdot \sum_{j=1}^{\infty} j \cdot Pr\{P_1(t)=j\}\right)$$
$$+ Pr\{P_2(t)=n+1\}\left((n+1)\cdot k_4 + \left(k_7 \cdot (n+1) \cdot \sum_{j=1}^{\infty} j \cdot Pr\{P_4(t)=j\}\right)\right).$$

It is evident that the summations on j are the expectations of the number of tokens present in places P_1 and P_4 and, in particular, that the flow from P_1 to P_2 is also exact according to the definition of transient product form. Similar reasoning can be made for the distributions of the other places. In order to evaluate our approximation, we assumed that the initial state is $|0,0,0,0|$, we considered the parameters

$$k_1 = 20, k_2 = 0.5, k_3 = 2, k_4 = k_5 = k_6 = 1$$

as fixed and we tested different values of k_7 since it is the cause for which the system does not exhibit a transient product form. The calculations required less than two seconds.

In Figure 5.6 we show the mean and the variance of the quantity of P_2 and P_4. The mean is approximated well for both places while the variance is captured well for P_4 and is underestimated for P_2. In Figure 5.6 it is also possible to observe that the approximation gets poor at the increasing of k_7.

As last, we propose a model in which the application of our approximation provides results that are inaccurate.

This last case is the well-known Lotka-Volterra model whose SPN is depicted in Figure 5.7. This model has been proposed independently by Lotka [64] and Volterra [89] and describes the evolution of two populations in competition.

In Figure 5.8, where the states of the corresponding Markov chain are depicted, it is possible to see that:

1. the state space is potentially infinite and grows over two dimensions;

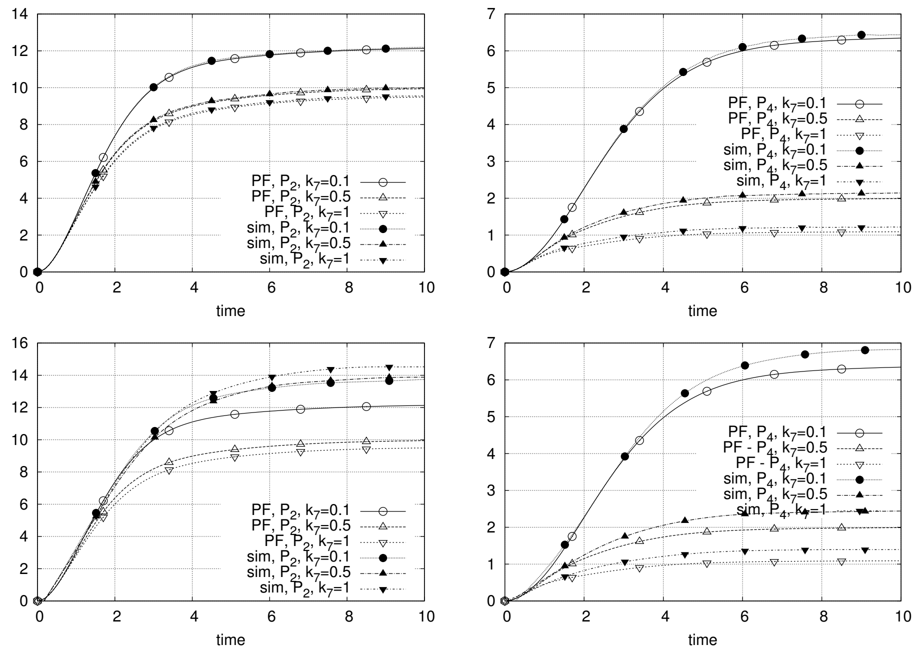


Figure 5.6: Network of infinite queues with a synchronization: mean (top) and variance (bottom) of the number of tokens in P_2 and P_4.

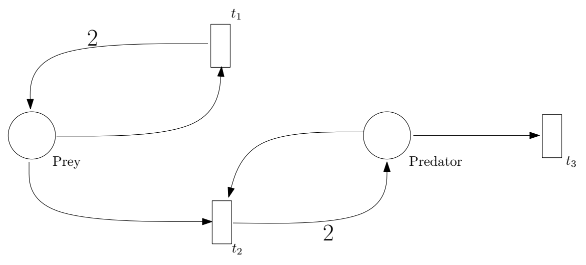


Figure 5.7: Petri net of the Lotka-Volterra model.

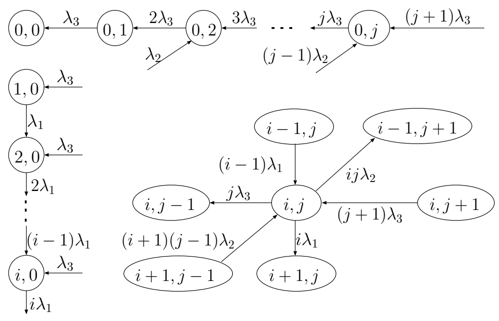


Figure 5.8: Boundaries and a generic state of the Markov chain of the Lotka-Volterra model (each state is labelled by the number of preys and predators).

2. the CTMC has an absorbing state in $|0, 0|$;

3. in case of predators extinction the model degenerates in a pure birth process where preys grow monotonically.

The third point has a dramatic effect on the results provided by PF approximation because it is not able to catch the effect generated by the disabling of synchronization t_2.

The numerical analysis of this model is very often unfeasible by using common techniques also from the point of view of the steady state (see [29]). For this reason, it is usually approximated by using *mean-field* techniques [16].

Such approximation is used to compute the expected number of tokens in each place by solving the system of ODEs

$$\frac{E\left[Prey(t)\right]}{dt} = k_1 E\left[Prey(t)\right] - k_2 E\left[Prey(t)\right] E\left[Predator(t)\right],$$
$$\frac{E\left[Predator(t)\right]}{dt} = k_2 E\left[Prey(t)\right] E\left[Prey(t)\right] - k_3 E\left[Predator(t)\right]. \quad (5.22)$$

With proper parameters, the system in equation (5.22) leads to oscillation along closed curves except if the system is started in equilibrium state.

We chose to introduce this topic because it happens that the curves provided by this approximation coincide with the expectations of the distribution computed by using the product form as an approximation.

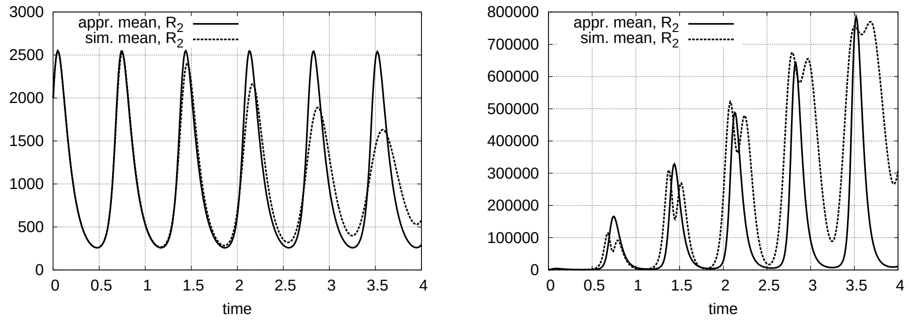


Figure 5.9: The Lotka-Volterra model with $\lambda_1 = 10, \lambda_2 = 0.01, \lambda_3 = 10$ and initial state $(2000, 2000)$: mean value (left) and variance (right) of the number of predators by simulation and by product form approximation.

It is not surprising, since the interactions among the two inhomogeneous chains, according to the definition of PF approximation, occurs by mean of the average quantities.

As first test, we start the model in state $(2000, 2000)$ and use reaction rates $k_1 = 10, k_2 = 0.01, k_3 = 10$. The mean and the variance of the number of predators obtained by simulation and by the product form approximation are depicted in Figure 5.9.

Even if the population levels are high, the mean deviates away soon from the stable oscillation pattern because the probability of predators extinction gets high on the time.

The product form approximation predicts instead stable oscillation of the mean and provides in expectation the same results of the mean-field approach (which is not depicted in Figure 5.9 because it cannot be distinguished from the mean provided by the product form approach). However, the product form approximation provides more information such as the variance that gives instead a more precise picture of what happens in the original model.

In Figure 5.10 we depicted the same quantities as in Figure 5.9 but starting the model from state $(200, 200)$ and with rates $k_1 = 10, k_2 = 0.1, k_3 = 10$. As the number of preys and predators are lower, the model deviates from stable oscillation faster. The product form approximation is not able to capture this behaviour and provides imprecise estimate of both the mean and the variance.

Figure 5.11 shows the probability of extinction of predators as function

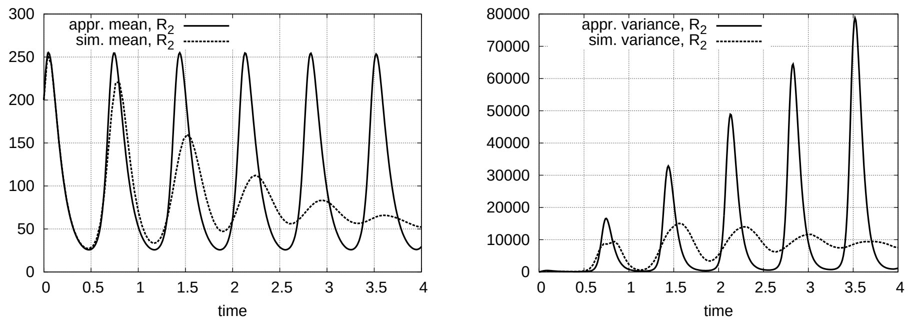


Figure 5.10: The Lotka-Volterra model with $\lambda_1 = 10, \lambda_2 = 0.1, \lambda_3 = 10$ and initial state $(200, 200)$: mean value (left) and variance (right) of the number of predators by simulation and by product form approximation.

of time obtained by simulation and the proposed approximation.

The reason that the approximation fails to give precise estimates is that the behaviour of the system depends strongly on the correlation of the population levels which is not captured by the product form probabilities. The high correlation is evidently caused by transition t_2.

This last test makes evident that a complete product form is not able to catch the disabling of the synchronization. Surely there are situations in which the error caused by PF is acceptable but it is also clear that the analysis of general models requires a more flexible product form. For this reason, in the following chapter we relax the product form assumption in such a way that the approximation is able to approximate a wider range of models.

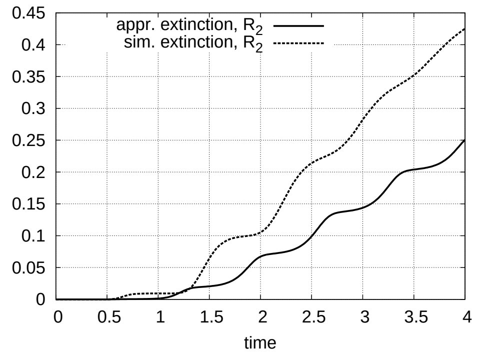


Figure 5.11: The Lotka-Volterra model with $\lambda_1 = 10, \lambda_2 = 0.1, \lambda_3 = 10$ and initial state $(200, 200)$: extinction of predators by simulation and by product form approximation.

6 Quasi-product form approximation

In the previous chapter we introduced the transient product form and investigated its use to approximate those models whose distributions cannot be decomposed exactly in such a form. In particular, we proposed an approximation technique which operates on the original state space of the model and is based on the assumption that the transient probabilities of the model can be written in product form. This assumption leads to a highly compact description of the transient probabilities because the space complexity of the computations grows only linearly with the number of modelled components.

It turned out that PF approximation can lead to highly imprecise results when it is used on models where synchronizations are involved.

Since synchronizations are an important modeling feature, in this chapter, we advance the technique proposed earlier by relaxing the assumption that all the transient probabilities are in product form. The new assumption, which we call *quasi product form* (QPF) assumption, leads to a computational method

- whose space complexity is still lower than that of the original problem,
- that results in a good approximation for a wider range of models.

For what concerns the SPN formalism, the idea behind the approach is that a place is allowed to preserve some dependencies with some other places of the net. In other words, subsets of places are in product form among each other and inside the subsets places remain correlated. A similar idea has been originally proposed by Whitt in [91] where he conjectured that good approximations can arise by splitting the original distribution of a queuing network in disjuncts subsets of stations. However, an extensive numerical analysis of this approach, called *partial product form* (PPF), has never been done. Moreover, the approach that we propose is more general due to the fact that the subsets have not to be disjoint among each other.

Our method is most akin to the one proposed by Casale in [21] where first passage times of closed queuing networks are approximated by Bayesian decomposition that in practice is equivalent to the one proposed in the following.

In the previous chapter we used the Lotka-Volterra model to show the problems that arise by using the product form approximation on models in which synchronizations lead to blocking. In these situations, the approximation quantifies the blockings in an imprecise way. Clearly this is a non-negligible alarm bell. In fact, in a large number of cases the purpose of synchronizations is to generate blocking; for example, in case of modeling shared resources.

To illustrate the concept we present a model in which this problem occurs. The model comes from biology but a similar structure is massively used also in the context of artificial systems.

Example 8 *We consider a gene regulatory network, called exclusive switch [63].*

Rougly speaking, the model consists of a switch that regulates the productions of two genes. Each of the two gene products, P_1 and P_2, inhibits the expression of the other if a molecule is bound to the promoter region of the DNA (called simply Dna in the following). In other words, if the Dna is bound to a molecule of P_1 (P_2) only molecules of type P_1 (P_2) can be produced, and if the Dna is free both types of proteins are produced. An illustration of the Petri net corresponding to the exclusive switch is depicted in Figure 6.1.

As shown, the model involves five places, namely Dna, Dna.P_1, Dna.P_2, P_1, P_2 where the "dot" means that the Dna is bound to P_1 (P_2). Thus, a state x is a vector of five non-negative integers, $|x_1, x_2, x_3, x_4, x_5|$, with the places ordered as above. The initial state is $|1, 0, 0, 0, 0|$.

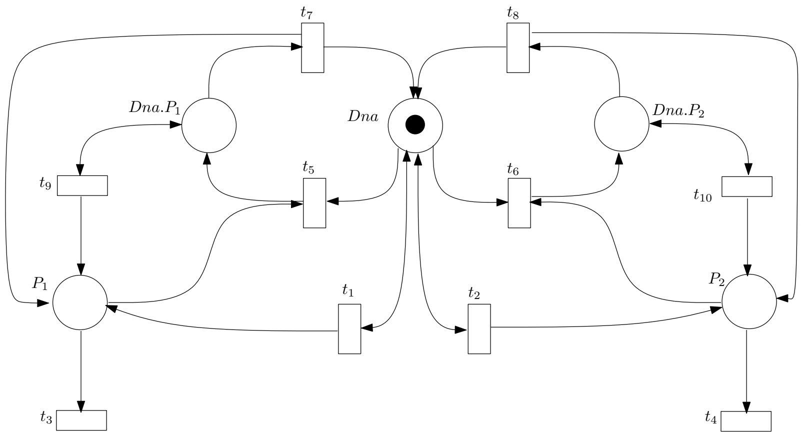


Figure 6.1: Petri net model corresponding to the exclusive switch.

The exclusive switch dynamics are characterized through ten transitions with infinite server semantics:

- t_1 *models production of* P_1 *in case of* $x_1 = 1$ *(free promoter region),*
- t_2 *models production of* P_2 *in case of* $x_1 = 1$ *(free promoter region),*
- t_3 *describes the degradation of* P_1*,*
- t_4 *describes the degradation of* P_2*,*
- t_5 *represents the binding of* P_1*,*
- t_6 *represents the binding of* P_2*,*
- t_7 *corresponds to the unbinding of* P_1*,*
- t_8 *corresponds to the unbinding of* P_2*,*
- t_9 *models the production of* P_1 *when* $x_2 = 1$ *(the promoter is occupied by a molecule of* P_1*),*
- t_{10} *models the production of* P_2 *when* $x_3 = 1$ *(the promoter is occupied by a molecule of* P_2*).*

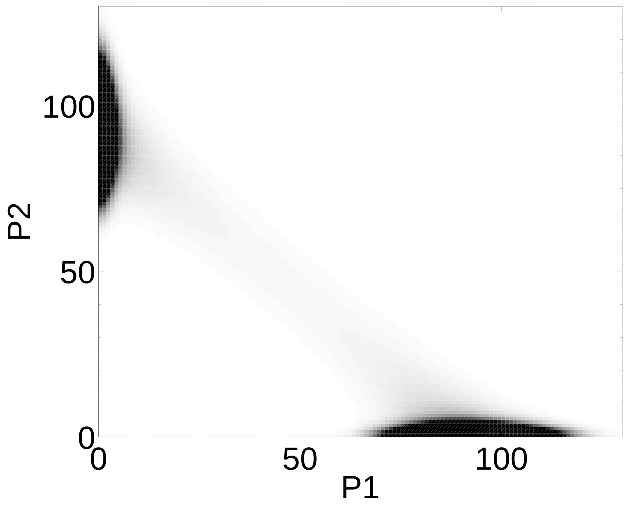


Figure 6.2: Example of bistable joint distribution of P_1 and P_2 (the darker is the region and the more likely is the state).

The initial state of the system leads to the P-invariant[1]

$$Dna + Dna.P_1 + Dna.P_2 = 1$$

and therefore the possible values for these three places, i.e. $|x_1, x_2, x_3|$, *are* $|1, 0, 0|$, $|0, 1, 0|$ *and* $|0, 0, 1|$.

An interesting feature of this model is that if the binding to the promoter is likely and the unbinding is rare then the distribution of P_1 and P_2 can become bistable leading to joint distributions similar to the one depicted in Figure 6.2. This means that a large amount of P_1 corresponds to small quantities of P_2 and vice-versa. This happens in this setting because each gene can "monopolize" the promoter region increasing its population while molecules of the other population can only degrade.

It is evident that the promoter region is modelled to cause blocking, in this situation the application of PF approximation is not reasonable and leads to completely erroneous results such as those reported in Figure 6.3 where the mean and the variance of the tokens present in P_1 (P_2 has the same behaviour

[1]Roughly speaking, a P-invariant is a set of places in which the sum of tokens remains constant in each state of the reachability set.

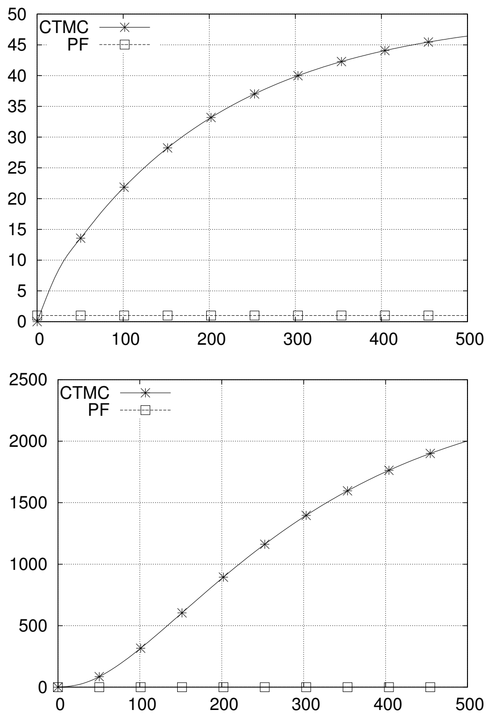


Figure 6.3: Exclusive switch model: mean value (left) and variance (right) of the number of tokens present in P_1 (P_2) by simulation and by product form approximation.

as we used symmetric parameters) are reported. Figure 6.3 shows how the networks is almost completely turned off according to the PF approximation while it is producing proteins according to the original model. The reason is that we used a of parameters with which the number of molecules (tokens) in places P_1 and P_2 can grow only if their gene monopolize the promoter region; otherwise the rate with which the two genes are produced is not enough to win against degradation (the parameters are given as the set 1 in Table 7.1). Since the PF approximation is not able to represent the monopolization of the promoter region, the bi-stable behaviour is not captured.

It is evident that the network presented above models a shared resource and the two genes can be considered as customers that compete for it.

The aim of the chapter is to provide a memory efficient, approximate technique that is able to provide a good characterization of the distributions for such models.

Although we motivated the QPF approximation by an example with a rigid blocking mechanisms scheme, it can outperform the PF approximation for models without synchronizations or heavy blocking, e.g. networks of finite server queues.

The chapter is organized as follows. In Section 6.1 we introduce the QPF approximation. In section 6.2 we summarize the guidelines to generate the directed acyclic graph required to describe our decomposition. An algorithm to implement the procedure based on the quasi product form assumption is discussed in Section 6.3. In Section 6.4 a preliminary error validation approach is discussed.

6.1 Definition of the approximation

In this section we introduce a more relaxed product form in order to expand the applicability of the approximation to a wider range of models. In particular, we will assume that there exist sets of places whose conditional probabilities depend only on a set of other places and not on all the rest of the places. For example, if we assume that the conditional probabilities of place 1 and 2 depend only on places 3, 4 and 5 then we can write

$$\begin{aligned}&Pr\{X_1 = x_1, X_2 = x_2 \mid X_3 = x_3, X_4 = x_4, ..., X_M = x_M\} =\\&Pr\{X_1 = x_1, X_2 = x_2 \mid X_3 = x_3, X_4 = x_4, X_5 = x_5\}\end{aligned}$$

A set of assumptions like the one above allows us to decompose the probability $Pr\{X_1 = x_1, X_2 = x_2, ..., X_M = x_M\}$ into a product. As this product is not in the classical product form given in (5.16), we will refer to it as quasi product form and in the following we provide its formal description.

The quasi product form decomposition of the transient probabilities is conveniently described by a *directed acyclic graph* (DAG), denoted by $\mathcal{G}$.

The set of the nodes of the graph is denoted by $\mathcal{V}$ and a given node, $v \in \mathcal{V}$, represents a subset of the places. The index set of the places represented by node v is denoted by $I(v)$. The set $\mathcal{V}$ must be such that it provides a partitioning of the set of places, i.e., $\cup_{v\in\mathcal{V}} I(v) = \{1, 2, ..., M\}$ and $\forall v_1, v_2 \in \mathcal{V}, v_1 \neq v_2 : I(v_1) \cap I(v_2) = \emptyset$.

The set of edges of the DAG, denoted by $\mathcal{E}$, provides the assumed dependency structure of the transient probabilities. Specifically, if $e = (u, v) \in \mathcal{E}$

then the conditional probability of the places in v *depends* on those places that are present in u.

The set of places present in the predecessors of v will be denoted by $P(v)$, i.e., $P(v) = \cup_{u:(u,v)\in\mathcal{E}} I(u)$. The conditional probability of the places in $I(v)$ is independent of those that are not present in $P(v)$, i.e.,

$$Pr\{\wedge_{i\in I(v)}(X_i = x_i) \mid \wedge_{j\in\{1,2,...,M\}\setminus I(v)}(X_j = x_j)\} = \\ Pr\{\wedge_{i\in I(v)}(X_i = x_i) \mid \wedge_{j\in P(v)}(X_j = x_j)\}$$

where $\wedge$ denotes conjunction. By considering every node of the DAG, the probability of a given state of the system, $|x_1, ..., x_M|$, can be written as

$$\pi_x(t) = \prod_{v\in\mathcal{V}} Pr\{\wedge_{i\in I(v)}(X_i = x_i) \mid \wedge_{j\in P(v)}(X_j = x_j)\} = \\ \prod_{v\in\mathcal{V}} \frac{Pr\{\wedge_{i\in Q(v)}(X_i = x_i)\}}{Pr\{\wedge_{j\in P(v)}(X_j = x_j)\}} \tag{6.1}$$

where we applied the notation $Q(v) = I(v) \cup P(v)$.

From (6.1), it is straightforward that also the following relation holds

$$Pr\{\wedge_{i\in\{1,2,...,M\}\setminus Q(v)}(X_i = x_i) | \wedge_{i\in Q(v)} (X_i = x_i)\} = \\ \frac{\prod_{v'\neq v\in\mathcal{V}} Pr\{\wedge_{i\in I(v')}(X_i = x_i) \mid \wedge_{j\in P(v')}(X_j = x_j)\}}{Pr\{\wedge_{j\in P(v)}(X_j = x_j)\}} \tag{6.2}$$

In the following we give three examples for the DAG $\mathcal{G}$.

Example 9 *For the exclusive switch the assumption of complete product form would be expressed by a graph with M nodes, $v_1, ..., v_M$, such that $I(v_i) = \{i\}$, and an empty set of arcs, $\mathcal{E} = \emptyset$. With this graph the probabilities are in the form given by the birth death processes described in the previous chapter.*

A possible PPF decomposition can be expressed by considering three disconnected nodes, v_1, v_2 and v_3, such that node v_1 is associated with the places $Dna, Dna.P_1$ and $Dna.P_2$, node v_2 is associated with P_1 and node v_3 with P_2. This leads to the following decomposition of the transient probabilities

$$Pr\{Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3, P_1 = x_4, P_2 = x_5\} = \\ Pr\{Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3\}Pr\{P_1 = x_4\}Pr\{P_2 = x_5\}$$

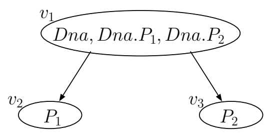


Figure 6.4: The DAG representing the a possible QPF decomposition for the exclusive switch

Finally, by maintaining the same nodes a possible DAG representing QPF, is depicted in Figure 6.4. This leads to the following decomposition of the transient probabilities

$$\begin{aligned} Pr\{Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3, P_1 = x_4, P_2 = x_5\} = \\ Pr\{Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3\}\times \\ Pr\{P_1 = x_4 \mid Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3\}\times \\ Pr\{P_2 = x_5 \mid Dna = x_1, Dna.P_1 = x_2, Dna.P_2 = x_3\} \end{aligned}$$

From now-on, in order to shorten the notation, given a set $\mathcal{W}$ the term $Pr\{\wedge_{i\in\mathcal{W}}(X_i(t) = x_i)\}$ will be denoted by $\pi_x^{\mathcal{W}}(t)$ whereas for any $\mathcal{W}$ and $\mathcal{W}'$ $\pi_x^{\mathcal{W}|\mathcal{W}'}(t)$ will corresponds to $Pr\{\wedge_{i\in\mathcal{W}}(X_i(t) = x_i)| \wedge_{i\in\mathcal{W}'} (X_i(t) = x_i)\}$. As last, we will refer to $\{1, 2, \ldots, M\}\backslash\mathcal{W}$ with $\overline{\mathcal{W}}$.

In order to compute the transient probabilities based on the quasi product form assumption expressed by the DAG $\mathcal{G}$, we need the quantities appearing in (6.1). Since $P(v) \subseteq Q(v)$, the quantities in the denominator can be computed simply by appropriate summing of the quantities in the numerator. The quantities in the numerator can instead be computed by exploiting the same reasoning made for the complete product form in the previous chapter, i.e.,

$$\frac{d\pi_x^{Q(v)}(t)}{dt} = \frac{d}{dt} \sum_{\substack{|y_1,\ldots,y_M| : \\ k \in Q(v), y_k = x_k}} \pi_y(t). \tag{6.3}$$

From which we have that

$$\frac{\pi_x^{Q(v)}(t)}{dt} = -\pi_x^{Q(v)}(t) \sum_{\substack{|y_1,\ldots,y_M|:\\ k\in Q(v), y_k = x_k}} \pi_y^{\overline{Q}(v)|Q(v)}(t) \sum_{n:y\geq i_n^-} \lambda_n(y)$$
$$+\sum_n \pi_{x-e_n}^{Q(v)}(t) \sum_{\substack{|y_1,\ldots,y_M|:\\ k\in Q(v), y_k = x_k\\ y-e_n\geq i_n^-}} \pi_{y-e_n}^{\overline{Q}(v)|Q(v)}(t)\lambda_n(y-e_n) \tag{6.4}$$

where we decomposed the product according to equation (6.2).

If the transient probabilities satisfy the quasi product form decomposition expressed by the graph $\mathcal{G}$, then for any transition i, $1 \leq i \leq R$, we have two possible scenarios:

1. $^\bullet t_i \backslash Q(v) = \emptyset$: the transition i is completely triggered by the places belonging to the set $Q(v)$ (or no place at all). In this case the summation over y does not affect the transition and then can be simplified because it sums to one,

2. $^\bullet t_i \backslash Q(v) \neq \emptyset$: the transition i is partially or completely triggered by other places. In this case the intensity of the transition is seen by the marginal $Q(v)$ as the expected flow generated by the other places. These flows can depend on the places in $Q(v)$.

Note that the possible irregularities of the state space, such as the case where the network is closed, are considered implicitly by the summation over y which takes only the states composing the state space.

Consider also that the terms $\pi_x^{\overline{Q}(v)|Q(v)}(t)$ bring much more information than required; in fact, the computation of the incoming flows requires only the quantities $\pi_x^{^\bullet t_n\backslash Q(v)|Q(v)}(t)$ which arises from proper summations of the marginals.

Denoting with $\Lambda^{Q(v)}$ the set of transitions which depends only on the places in the set $Q(v)$ and with $\Lambda^{\overline{Q}(v)}$ its complement, we have that the quasi product form approximation satisfies the equation

$$
\begin{aligned}
\frac{\pi_x^{Q(v)}(t)}{dt} &= -\pi_x^{Q(v)}(t) \sum_{n \in \Lambda^{Q(v)}: x \geq i_n^-} \lambda_n(x) \\
&- \pi_x^{Q(v)}(t) \sum_{\substack{|y_1,\dots,y_M|: \\ k \in Q(v), y_k = x_k}} \pi_y^{\bullet t_n \setminus Q(v)|Q(v)}(t) \sum_{n \in \Lambda^{\overline{Q}(v)}: y \geq i_n^-} \lambda_n(y) \\
&+ \sum_{n \in \Lambda^{Q(v)} n : x - e_n \geq i_n^-} \pi_{x-e_n}^{Q(v)}(t) \lambda_n(x - e_n) \\
&+ \sum_{n \in \Lambda^{\overline{Q}(t)}} \pi_{x-e_n}^{Q(v)}(t) \sum_{\substack{|y_1,\dots,y_M|: \\ k \in Q(v), y_k = x_k \\ y_n - e_n \geq i_n^-}} \pi_{y-e_n}^{\bullet t_n \setminus Q(v)|Q(v)}(t) \lambda_n(y - e_n). \qquad (6.5)
\end{aligned}
$$

We arrived to an equation similar to (5.16) where a complete product form is considered. The computation of the time-dependent incoming flows of each marginal is the crucial point of our approximation. The faster we can compute these measures and the less complex is the problem.

However, this time the computation is not simple as the previous case; in fact, if the computation of the value of incoming flows at time t depends on more marginals it can happen that they are interleaved among each others. Then, we have to normalize them according to their dependencies. Since the computation of these dependencies requires further summations, it becomes heavy from a computational point of view if several marginals are involved. As a consequence, a good idea is to choose the DAG in such a way that the dependencies of the quantities $\pi_y^{\bullet t_n \setminus Q(v)|Q(v)}(t)$ are minimized, $\forall v \in \mathcal{V}$.

In order to provide an example of how the approximation works, in the following we derive equation (6.5) for all the nodes of the DAG introduced in Example 9 and depicted in Figure 6.4. This way we provide the necessary differential equations for the exclusive switch.

Example 10 *According to the partitioning given in Example 9, the places indicated by $Q(v_1)$ are Dna, Dna.P_1 and Dna.P_2 and the set of places given by $P(v_1)$ is the empty set. The possible values, $|x_1, x_2, x_3|$, for these three places are $|1, 0, 0|$, $|0, 1, 0|$ and $|0, 0, 1|$. Let us first consider the case when the Dna is free, i.e., $|x_1, x_2, x_3| = |1, 0, 0|$. By following (6.3) and applying* *the Chapman-Kolmogorov equations we have*

$$\frac{dPr\{Dna=1, Dna.P_1=0, Dna.P_2=0\}}{dt} = \sum_{x_4,x_5}\Big($$
$$-Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4, P_2=x_5\}(k_5x_4+k_6x_5)+$$
$$k_7\cdot Pr\{Dna=0, Dna.P_1=1, Dna.P_2=0, P_1=x_4-1, P_2=x_5\}+$$
$$k_8\cdot Pr\{Dna=0, Dna.P_1=0, Dna.P_2=1, P_1=x_4, P_2=x_5-1\}\Big) \quad (6.6)$$

where the first term in the summation of the right-hand side corresponds to binding of the Dna to P_1 (with speed k_5) or P_2 (with speed k_6) and the second and third term describes the unbinding of P_1 (k_7) and P_2 (k_8). By applying the quasi product form assumption given in Example 9, the right-hand side of (6.6) becomes

$$-\Big(k_5\cdot\sum_{x_4} x_4 Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}+$$
$$k_6\cdot\sum_{x_5} x_5 Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_2=x_5\}\Big)+$$
$$k_7\cdot Pr\{Dna=0, Dna.P_1=1, Dna.P_2=0\}+$$
$$k_8\cdot Pr\{Dna=0, Dna.P_1=0, Dna.P_2=1\}$$

Note that the first (second) term of the above quantity is proportional to the expected amount of P_1 (P_2) given that the system is in a state with $Dna=1, Dna.P_1=0, Dna.P_2=0$. By similar reasoning, for $|x_1,x_2,x_3|=|0,1,0|$, i.e, when the Dna is bound to P_1, we get

$$\frac{dPr\{Dna=0, Dna.P_1=1, Dna.P_2=0\}}{dt} =$$
$$-k_7\cdot Pr\{Dna=0, Dna.P_1=1, Dna.P_2=0\}+$$
$$k_5\cdot\sum_{x_4} x_4 Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}$$

For the case when the Dna is bound to P_2, i.e., for $(x_1,x_2,x_3)=(0,0,1)$, we *get the counterpart of the above expression as*

$$\frac{dPr\{Dna=0, Dna.P_1=0, Dna.P_2=1\}}{dt} =$$
$$-\,k_8 \cdot Pr\{Dna=0, Dna.P_1=0, Dna.P_2=1\}+$$
$$k_6 \cdot \sum_{x_5} x_5 Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_2=x_5\}$$

Now we turn our attention to node v_2 (Figure 6.4). The places indicated by $Q(v_2)$ are Dna, Dna.P_1, Dna.P_2 and P_1 while the species given by $P(v_2)$ are Dna, Dna.P_1 and Dna.P_2. We have to consider all possible values for all four places given by $Q(v_2)$. We first consider the case when we have free Dna (consequently, no Dna.P_1 and Dna.P_2), i.e., $|x_1,x_2,x_3|=|1,0,0|$, and a generic amount, x_4, of P_1. By following equation (6.4) we get

$$\begin{aligned}
&\frac{dPr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}}{dt} = \qquad (6.7)\\
&-\,k_1 \cdot Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}+\\
&k_1 \cdot Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4-1\}-\\
&k_3 \cdot x_4 \cdot Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}+\\
&k_3 \cdot (x_4+1) \cdot Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4+1\}-\\
&k_5 \cdot x_4 \cdot Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}-\\
&k_6 \cdot \sum_{x_5} \frac{x_5 Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_2=x_5\}}{Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0\}} \times\\
&\qquad\qquad Pr\{Dna=1, Dna.P_1=0, Dna.P_2=0, P_1=x_4\}+\\
&k_7 \cdot Pr\{Dna=0, Dna.P_1=1, Dna.P_2=0, P_1=x_4-1\}+\\
&k_8 \cdot Pr\{Dna=0, Dna.P_1=0, Dna.P_2=1, P_1=x_4\}
\end{aligned}$$

where the terms on the right-hand side correspond to, respectively: outgoing probability by production of P_1 (with free Dna); incoming probability by production of P_1 (with free Dna); outgoing probability by degradation of P_1; incoming probability by degradation of P_1; binding of Dna with P_1; binding of Dna with P_2; unbinding of Dna with P_1; and unbinding of Dna with P_2. It is worth to note that the effect of the quasi product form assumption is that the term corresponding to the binding of Dna with P_2 is determined by the conditional expected value of P_2 given that the Dna is free. Indeed the *summation in that term is equal to*

$$E\{P_2 \mid Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0\}$$

which corresponds to the expected value of P_2 conditioned by the state of the promoter region. Next we consider the situation that the Dna is bound to P_1, i.e., $|x_1, x_2, x_3| = |0, 1, 0|$, and a generic amount, x_4, of P_1. It leads to

$$\begin{aligned}
&\frac{dPr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4\}}{dt} = \\
&- k_3 \cdot x_4 \cdot Pr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4\} + \\
&k_3 \cdot (x_4 + 1) \cdot Pr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4 + 1\} - \\
&k_9 \cdot Pr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4\} + \\
&k_9 \cdot Pr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4 - 1\} - \\
&k_7 \cdot Pr\{Dna = 0, Dna.P_1 = 1, Dna.P_2 = 0, P_1 = x_4\} + \\
&k_5 \cdot (x_4 + 1) \cdot Pr\{Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0, P_1 = x_4 + 1\}
\end{aligned}$$

where the terms on the right-hand side correspond to, respectively: outgoing probability by degradation of P_1; incoming probability by degradation of P_1; outgoing probability by production of P_1 (with bound Dna); incoming probability by production of P_1 (with bound Dna); unbinding of P_1; and binding of P_1.

The last situation to consider for what concerns v_2 is when the Dna is bound to P_2, i.e., $|x_1, x_2, x_3| = |0, 0, 1|$, and a generic amount, x_4, of P_1. We get

$$\begin{aligned}
&\frac{dPr\{Dna = 0, Dna.P_1 = 0, Dna.P_2 = 1, P_1 = x_4\}}{dt} = \\
&- k_3 \cdot x_4 \cdot Pr\{Dna = 0, Dna.P_1 = 0, Dna.P_2 = 1, P_1 = x_4\} + \\
&k_3 \cdot (x_4 + 1) \cdot Pr\{Dna = 0, Dna.P_1 = 0, Dna.P_2 = 1, P_1 = x_4 + 1\} - \\
&k_8 \cdot Pr\{Dna = 0, Dna.P_1 = 0, Dna.P_2 = 1, P_1 = x_4\} + \\
&k_6 \cdot \sum_{x_5} \frac{x_5 Pr\{Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0, P_2 = x_5\}}{Pr\{Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0\}} \times \\
&\qquad\qquad Pr\{Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0, P_1 = x_4\}
\end{aligned}$$

where on the right-hand side the terms are, respectively: outgoing probability by degradation of P_1; incoming probability by degradation of P_1; unbinding *of P_2; and binding with P_2. As before, the effect of the quasi product form assumption is that the speed of the binding of the Dna with P_2 is proportional to the conditional expected amount of P_2.*

The last node of the DAG (Figure 6.4), v_3, leads to the counterpart of the expressions reported above for node v_2.

Assuming that the number of states that the ith place can assume is $\tilde{B}_i$, the number of equations of the original model is at the most equal to $\prod_{i=1}^{M}(\tilde{B}_i + 1)$. The number of equations describing the quasi product form in (6.4) can be determined as follows. Let K denote that subset of the nodes of the DAG that provides a complete and not redundant set of ODEs, i.e., $K = \{v \,\big|\, v \in \mathcal{V}, \nexists u \in \mathcal{V} \text{ such that } Q(v) \subset Q(u)\}$. Having defined K, the necessary number of ODEs for the quasi product form is at the most $\sum_{v\in K} \prod_{i\in Q(v)}(\tilde{B}_i + 1)$. This means that the original M-dimensional state space is reduced to $\max_{v\in K} |Q(v)|$ dimensions.

As last observation, by looking at (6.1), one can check that the ODEs provided by equation (6.4) maintain unity of the total probability, i.e., for every node $v \in \mathcal{V}$ summing $\pi_x^{Q(v)}(t)$ for every possible values of $x_i, i \in Q(v)$, gives one.

If there exists a place whose index, i, is present in both $Q(u)$ and $Q(v)$ with $u, v \in \mathcal{V}, u \neq v$, then the marginal probabilities of place i, i.e., $P\{X_i = x_i\}$, can be derived using the quantities associated with u, i.e., $\pi_x^{Q(u)}(t)$, or using the quantities associated with v, i.e., $\pi_y^{Q(v)}(t)$. By considering the derivative $dP\{X_i = x_i\}/dt$ which can be computed based on both u and v by summation of their associated ODEs given in (6.4), it is easy to show that the different ways of calculating $P\{X_i = x_i\}$ lead to the same result. The above reasoning can be generalized to any marginal distribution of the model.

6.2 Choice of the DAG

Assuming a completely general network the choice of the DAG is a tricky problem. For this reason, we split the discussion in two parts. The first part deals with the case where the exactitude of the results is precluded only by the presence of finite server stations. On the contrary, the second part considers a completely general context.

6.2.1 Finite server networks

In [21], Casale suggested a method based on a conditional entropy metric; indeed, this is an elegant solution but it is based on the equilibrium distribution of the process. Thus, there is no guarantee that the DAG (Bayesian tree in his case) leads to the best possible decomposition.

We do not use this approach mostly because of two reasons: the first is that we are interested also in models whose equilibrium distribution is unknown or does not exist; the second is that the method, apparently, does not exploit the only information about transient product form, i.e. the presence of infinite servers. The dependency on a server whose output flow is similar to an inhomogeneous Poisson process can be neglected without amplifying the approximation error. For this reason dependency on infinite servers are good candidates to be neglected. This fact is implied in the QPF assumption since every place perceives the incoming flows generated by places with which it is in product form as inhomogeneous Poisson processes whether or not they are. We also expect that a station $M/M/k$ is “one step” closer to product form than a $M/M/k-1$ station, $k \geq 2$, because potentially there are more departures that are independent among each other.

These facts suggest that: given the set of stations, there is a sort of hierarchy of candidates to generate product form probabilities that goes from infinite servers to $M/M/1$ stations. The rank must take into account also:

- the number of stations that might receive the output. As an example, the output flow from an $M/M/1$ station cannot generate non product form dependencies if customers leave the system after the service;
- the service time and the load of the station (visits in case of closed networks) because long queues implies high dependencies between subsequent departures;
- the initial state (for the same reason above).

The fear is that considering the equilibrium distribution all these facets might be not considered because in equilibrium all these types of stations are in product form [13].

As last consideration, it is important to point out that the above reasonings do not consider the case of a *saturated station* whose number of customers in queue tends to infinity. In fact, in this particular case a finite server station becomes an optimal candidate to be in product form with the other nodes of the net since it behaves as a generator of Poisson events.

6.2.2 General context

By exploiting the whole expressiveness of SPNs, we move in to a setting where we can have bulk arrivals, forks and synchronizations (which probably are the only feature that leads to results not even comparable to the originals).

The first problem moving out of the setting of queuing networks is that the state space of the system is more irregular. As an example, consider the net on the left side of Figure 6.5. It is composed of a cycle between A and B which decides if eventually the place D will receive one token or two. By using the DAG depicted on the r.h.s. we have that the marginal $\{C, D\}$ requires the computation of an event generated by B. In this case we would desire to compute the rate of t_3 only by summing on the fly the terms of the marginal $\{A, B\}$; however we need to store the fact that, for example, the state $|0, 0, 0, 0|$ does not exists. In general, the handling of this kind of irregularities is extremely heavy from a computational point of view because requires the storage of all the exceptions.

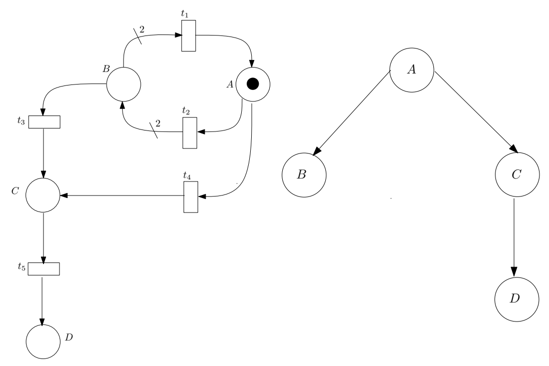


Figure 6.5: Example of network leading to an irregular state space: Petri net (left), A possible DAG (right)

This facet suggests that a qualitative analysis of the reachability set can help during the generation of the DAG. Luckily, the Petri net formalism offers a large number of well-founded mathematical methods for the analysis of the structural properties that can be used to check deadlocks, livelocks, traps, and other peculiarities of the structure of the net. Additionally, advanced techniques such as *Binary decision diagrams* (BDD) allows the storage and qualitative analysis of reachability sets with a low memory consumption (see for example [25, 8] where state space greater than 10^{30} are handled).

Note that knowing the real number of states of each place is fundamental because otherwise it is not possible to quantify the number of ODEs with which we have to deal with.

This is important because the first, and most obvious, criteria to generate the DAG is the computational effort that we are available to spend to analyze the network. Indeed, this choice affects the definition of the nodes of the graph since in the best situation each node is in complete product form with the others; hence, the minimal number of equations with which we have to deal with is equal to the sum of the number of states generated by each node.

Starting from a set of nodes whose state space is acceptable from the point of view of the equations that they generate (and possibly far from the limit that we prefixed), the second criteria deals with the flows that introduce dependencies among the nodes.

We have already discussed about finite server transitions, the other concern is about synchronizations. Every synchronization is potentially troublesome; thus, our first target would be to define a DAG able to generate a set of marginals in which every transition is described completely by at least a marginal.

Unfortunately this is not always possible in case of strongly connected networks, such as the one in Figure 6.6, where our reasoning would lead to the graph depicted on the r.h.s. Figure 6.6 which generates the marginal $\{A, B, C\}$ corresponding to the original problem.

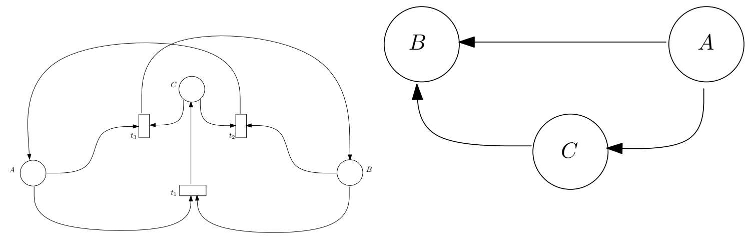


Figure 6.6: Example of a strongly connected network (left). A DAG describing a possible dependency structure (right),

If not all the synchronizations can be described completely through at least one marginal (that is equivalent to say that the synchronization is triggered by places in product form among each other) we suggest to give priority according to the following criteria (written in order of importance):

1. synchronizations leading to blocking,
2. high probability to fire[1].

As last consideration, in general, synchronizations leads to greater errors than finite server transitions; hence it is better to put first the dependencies of synchronizations instead of finite server transitions. Refinements of the DAG can be performed by considering the number of corresponding equations. If the number of the equations is low, it is reasonable to add dependencies or merge nodes. If, on the contrary, the number of equations is more than the threshold, we have three choices: increase the computational effort; return to the step in which nodes are decided and try different strategies; accept a, probably, poorer result.

It is clear that there is no guarantee that the overall propagation of the error will be the optimal.

At the same time, it is also plain that moving outside the context of queuing networks the number of variables having a role in the generation of a the DAG is probably too high to get the approach automated.

As a consequence, the generation of the graph requires some knowledge about the structure of the model and its qualitative properties. We argue that in case of models with a reasonable mathematical abstraction of the problem this preamble would not require much effort.

On the contrary, in case of huge models involving several random variables that interact in complex and interleaved ways a qualitative analysis is not only to be expected but would be required.

In any case we are conscious that the developing of tools that can help these decisions are needed. A possible way is to drive the generation of the DAG according to the results obtained by assuming the model as it would be in complete product form (or in partial product form if synchronizations are involved).

[1]This last criteria will be clearer in the next chapter where we show that there are cases in which the more a synchronization is frequent the more it leads to errors. Note that we are not talking about the speed; a transition can be likely also if it is slow with regards to the rest of the transitions if it competes only with those slower than it.

We leave the decomposition of heavily structured models as future work by focusing the attention on models in which the structure of the net gives a clear hint about their decomposition. As very last observation, in this direction we mention that considering the covariances of infinite servers queuing networks as described by Whitt [67] and the moment closures for biochemical processes as defined by Hespanha in [53], is also an option.

6.3 Algorithm

In this section, we provide a sketch of the implementation of the algorithm that follows from the quasi product form assumption. We focus on the representation of the system of ODEs in such a way that it can be used in common ODE solvers.

As described in the previous section, the computation of the transient probabilities based on the quasi product form assumption requires the quantities involved in (6.1) and, in particular, it needs the probabilities $\pi_x^{Q(v)}(t)$ since they allow the computation of any marginal probability referring to a subset of the places in $Q(v)$. Thus, the collection of all the marginal distributions representing the sets $Q(v), v \in \mathcal{V}$, is enough to carry out the computations. Nevertheless, since it can happen that there exist v_1 and v_2 such that $Q(v_1) \subset Q(v_2)$, considering all nodes in $\mathcal{V}$ can lead to a redundant set of ODEs.

As an example, this happens in case of the exclusive switch by using the DAG depicted in Figure 6.4; the node v_1 has only outgoing arcs and, consequently, $Q(v_1)$ is contained both in $Q(v_2)$ and $Q(v_3)$. The overhead caused by this redundancy can be either negligible or non-negligible depending on the applied quasi product form assumption. In Table 6.1, from line 1 to 10, we propose a simple way to eliminate the redundancy by computing the minimal set of marginal distributions (stored in the variable *Marg*). The algorithm consists of two nested loops which collect (in the variable Q) the places representing the dependencies of a node (including the places in the node itself) and construct a new marginal distribution only if the node has incoming arcs or if the node does not have outgoing arcs at all (in order to guarantee the presence of those places that are completely independent from the others). The object representing the new marginal distribution itself is instantiated in line 9 and added to the set of marginals collected in *Marg*.

In the following we concentrate on the so-called evaluation step, i.e., the

```
0   $Preprocessing()$ begin
    // Makes the marginal distribution set
1     $Marg := \emptyset$;
2     forall $v \in \mathcal{V}$ do
3       $Q := \emptyset$;
4       forall $i$ such that$(i, v) \in \mathcal{E}$ do
5         $Q := Q \cup I(i)$;
6       end
7       if $Q \neq \emptyset \vee (Q == \emptyset \wedge \nexists (v, i) \in \mathcal{E})$ then
8         $Q := Q \cup I(v)$;
9         $Marg := Marg \cup marginal.init(Q)$;
10    end
11  end
```

Table 6.1: Algorithm: preprocessing for the quasi product form approximation

computation of the derivatives that are necessary to perform the numerical integration of the ODEs. This step requires to represent the marginal distributions and the following variables are necessary in order to carry out the computations:

- Q: set containing the indexes of the places composing the marginal distribution,
- *states*: list of all possible values that the quantities of the places present in Q can assume,
- $\Lambda^{Q(v)}$: list of those transitions that depends only on the marginal $Q(v)$,
- $\Lambda^{\overline{Q}(v)}$: list of those transitions that depends on other marginals,
- *conditions*: data structure that, given an index of a transition, returns the indexes of those places in Q that as an effect on it,

A partial implementation of a data structure with the above variables is reported in Table 6.2 where we detailed the function *init* used in line 9 of Table 6.1.

Note that, since several transitions can have a null impact on a marginal distribution, it is worth to store in $\Lambda^{Q(v)}$ and $\Lambda^{\overline{Q}(v)}$ only those transitions that are able to move probability among at least two states of $Q(v)$. The representation of *conditions* is trivial since they can be expressed through simple matrices.

```
0   data struct marginal begin
1     Q; // the indexes of the places composing the marginal
2     states; //states of the marginal distribution
3     $\Lambda^{Q(v)}$; // list of transitions in completely described by the places in Q(v)
4     $\Lambda^{\overline{Q}(v)}$ ; // list of transitions depending also on other marginals
5     conditions; // data structure containing the indexes of the places that condtion the incoming flows
    ......
6     init(Q) begin
7       this.Q := Q
8       $\Lambda^{Q(v)} := \{r | \exists j \in Q, c_{r,j} > 0 \wedge {}^\bullet t_r \backslash Q(v) = \emptyset\}$;
9       $\Lambda^{\overline{Q}(v)} := \{r | \exists j \in Q, c_{r,j} > 0 \wedge {}^\bullet t_r \backslash Q(v) \neq \emptyset\}$;
10    end
11  end
```

Table 6.2: Algorithm: data structure describing a marginal distribution

Finally, in Table 6.3 we provide a naive algorithm for the evaluation step where *s.prob, s.der*, and *s.succ(r)* refer, respectively, to the probability of a state at time t, the derivative of the probability of a state at time t and the state reached by the occurrence of transition r in the state s.

In lines 10-14 where we consider the incoming flows generated by other marginals. In particular, we assumed that the object $IncomingFlows$ stores all the intensities of incoming flows for the marginal m. The values collected into $IncomingFlows$ get updated in line 3 and can be retrieved by the method *get* that takes in input the transition and the current state s. Incoming flows are computed by appropriate summing of the other marginals and can be stored in two possible ways. The first solution is to store the intensities; this means that we store the quantities

$$\sum_{\substack{(y_1,\dots,y_M):\\ k \in Q(v), y_k = x_k}} \pi_y^{{}^\bullet t_n \backslash Q(v)|Q(v)}(t)\lambda_n(y) \quad \forall x, n$$

which corresponds to the expected flow of transition t in the marginal state x. The second solution is to store the marginal distributions $\pi^{{}^\bullet t_n \backslash Q(v)|Q(v)}(t)$, $n \in \Lambda^{\overline{Q}(v)}$. The first case is preferable when several intensities have the same condition and can be re-used for a large number of states. For example, this happens for the exclusive switch with the decomposition presented in Figure 6.4; in this case we need to store only two values: $E\{P_1 \mid Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0\}$ and $E\{P_2 \mid Dna = 1, Dna.P_1 = 0, Dna.P_2 = 0\}$.

The second is preferable when the state space is irregular and the re-

```
Eval() begin
  $condexp := \emptyset$
  forall $m \in Marg$ do
  $IncomingFlows := ComputeIncomingFlows(m);$
    forall $s \in m.states$ do
      forall $t \in m.\Lambda^{Q(v)}$ do
        $rate = ComputeRate(s);$
        $s.der = s.der - rate * s.prob;$
        $s.succ(t).der = s.succ(t).der + rate * s.prob;$
      end
        forall $t \in m.\Lambda^{\overline{Q}(v)}$ do
            $rate = IncomingFlow.get(t, s);$
            $s.der = s.der - rate * s.prob;$
            $s.succ(r).der = s.succ(r).der + rate * s.prob;$
      end
    end
  end
end
```

Table 6.3: Algorithm: procedure describing the evaluation step

use of the intensities is low. In this case, the method *get* must perform an additional summation to compute the overall incoming rate.

The number of values stored in the data structure *IncomingFlows* is strongly related to the model and the applied quasi product form decomposition. Considering that each marginal uses only a subset of the transitions and they probably use a limited set of places not belonging to the set Q, it is likely that the number of values that need to be stored is low and in general negligible with regards to the number of states. There can be however situations in which many values must be computed and the same one is applied many times for a sequence of states. For this reason, we suggest to store these quantities as a cache from which recently calculated entries can be retrieved.

In the future we aim to use advanced data structures like BDDs that can help a more efficient computation/storage of incoming flows. This idea is motivated by the fact that a decomposition in conditional probabilities has already been done for the approximation of steady state by Ciardo et al in [90]. The work points out how these structures provide a natural decomposition of a vector of random variables and allow to rebuild conditional probabilities by moving among the levels of the graph.

The last consideration is about the "cut" of the states having negligible probability mass. This technique is based on a threshold under which the states are not considered during the integration step. Consequently, the overall computational time can be significantly reduced. Furthermore, in case of unbounded state spaces, it allows us not to define the bound "a priori". The use of this technique in combination with the quasi product form approach is feasible and effective but beyond the scope of this thesis; the reader is referred to [51].

6.4 Error evaluation

A bounding of the error like in case of uniformisation is probably an unrealistic investigation and also the thorough error analysis of the approximations based on quasi product forms is still an open problem.

We present, however, a preliminary approach to validate the results obtained by the quasi product form assumption. This approach provides a first quick evaluation of the goodness of the results and can point out where and how much the quasi product form deviates from the original behaviour of the system under study.

Let us assume that the transient probabilities have been computed up to time t under the quasi product form assumption. The probability of a given state, $\pi_x(t)$, can be then calculated by (6.1). We can compute the derivative $d\pi_x(y)/dt$ under the quasi product form assumption by applying the formulas provided in (6.1). Let us denote this quantity by $p'_{\text{QPF}}(t,x)$. The same derivative can be computed considering the behaviour of the original CTMC based on (2.15), i.e., *without* assuming quasi product form. In other words, we use the quasi product form assumption to compute the probabilities up to time t and then calculate how much these probabilities would be moved by the original CTMC in an infinitesimal interval. The resulting derivative will be denoted by $p'_{\text{CTMC}}(t,x)$. The difference of the two derivatives can be used to quantify how much the quasi product form assumption deviates from the original behaviour. In particular, a trustworthy measure would be the quantity

$$\max_x \left| p'_{\text{QPF}}(t,x) - p'_{\text{CTMC}}(t,x) \right| \tag{6.8}$$

i.e., the maximum of the absolute value of the differences, to quantify the error introduced by the quasi product form approximation at time t.

The informations provided by this measure can be useful; for instance, we can reject the approximation if the error deviates from small values or does not stabilize. In general, a positive trait of the measure in (6.8) is that it does not require to calculate the transient behaviour of the original Markov chain, and thus it can be calculated in a memory efficient manner. Moreover, the computation can be limited to small subsets of the state space that are of interest for the purpose of the analysis. A negative trait is that it does not provide a bounding of the error.

7 Numerical illustrations

In this chapter we show numerical results obtained using the quasi product form assumption. We apply the approximation to several models with various settings of the parameters in order to give an hint about the accuracy of the method. Moreover, we aim to show on practical cases how quasi product form works and can be used.

As first, we consider queuing networks in order to deepen the discussion about the approximation of finite server systems that we started in Chapter 5. In this setting, we aim to provide:

- a justification for the extension of partial product form to the quasi product form; this is done by showing that the comparison between the two approximations is in favor of the "quasi" version even if we consider extremely simple network; this happens because partial product form can describe only a very limited set of dependencies,
- a hint on how to choose the dependencies in cases where the routing of the net is not a DAG,
- more illustrations about the accuracy of our method as approximation of transient distribution in queuing networks.

Then, we change context to show results obtained on biochemical systems. In particular, we focus our attention on the, already mentioned, exclusive switch model and a model alike, namely a system of multi-attractors, whose switch mechanism is more complex. These models describe phenomena where different productions are modulated by switches. We propose these tests with a dual purpose, i.e. to show that

- the decomposition in stochastic dependencies fits particularly well for these kind of problems,
- analyses based on the computation of first and second moments only brings informations that could be misleading in regards of the real behaviour of the model.

As last, we propose a more general model where forks, synchronizations and finite server are present together. The aim of this last test is to put under stress our approximation and point out its limits.

For all the cases, we compare the results obtained by the proposed quasi product form approximation with the exact behaviour computed on the original CTMC of the model either by uniformisation or by simulation when the state space is large. In order to provide a visual comparison of the original behaviours and the approximated values, we provide in figures the expectations, the variances and the marginal distributions of those objects that better represent the dynamics of the models. The algorithm based on the quasi product form assumption has been implemented in JAVA using the odeToJava package[1] to solve the system of ODEs. All the experiments have been performed on a Intel Centrino Dual Core with 4Gb of RAM.

7.1 Finite number of servers

7.1.1 Open central server network

As first, we consider a network with a topology identical to the one presented at page 25 in Figure 2.4 with the only difference that here we assume that the first station is an $M/M/3$ station, i.e., three jobs can be under service at a time at the server. We tested the approximation with the parameters

[1]Available at *http://www.netlib.org/ode/* and developed by M. Patterson and R. J. Spiteri.

$r_{1,2} = 0.2$, $r_{1,3} = 0.3$, $\lambda_1 = 0.6$, $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 1$. Note that with single server policy the system would not be stable due to the presence of the loop. The number of jobs at a queue is at most 15, further arrivals are lost.

We analyzed the model by using its original CTMC, by all possible partial product form decompositions, and by the QPF decomposition suggested by the routing of the net. Figure 7.1 depicts all the DAGs, the graph (d) is the only one that cannot be expressed in partial product form.

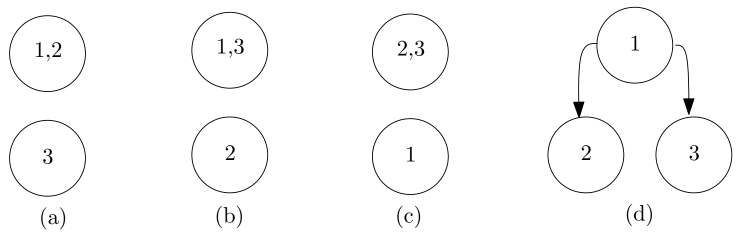


Figure 7.1: DAGs used to analyse the open central server model.

With the proposed QPF decomposition the set of required marginal distributions are the probabilities $P\{X_1 = x_1, X_2 = x_2\}$ and $P\{X_1 = x_1, X_3 = x_3\}$. We considered two situations: starting with empty queues and starting with five jobs at the server. In Fig. 7.2 and 7.3 we depicted the mean and the variance of the number of jobs at the server as function of time. Starting with empty queues, both the QPF and PPF approximations lead to good results with the exceptions of the third PPF decomposition that fails to provide an accurate estimate of the variance. Starting with five jobs at the server, the approximations are worse and only the QPF decomposition captures the mean correctly and approximate the variance well.

The original CTMC is composed of $16^3 = 4096$ states. The PPF approximations lead to $16^2 + 16 = 272$ ODEs while the QPF uses $2 \times 16^2 = 512$ ODEs. The number of equation of the QPF decomposition is about two times more but the dimensionality of the the approaches are the same as both consider the dependencies of at most two stations. The calculations with QPF decomposition required about 1 second of CPU time.

7.1.2 Multi-path network

As second example, we consider a more complex pipeline where each request can be satisfied by following four different production paths. The network is

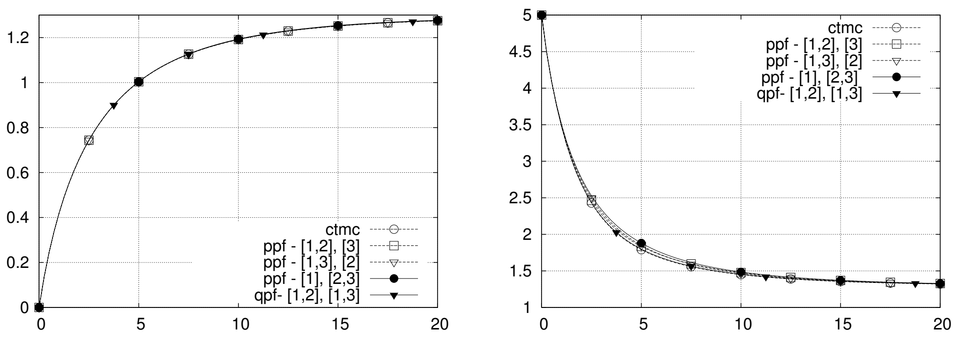


Figure 7.2: Open server network: mean number of jobs at the server; starting with empty queues (left), with five jobs at the server (right).

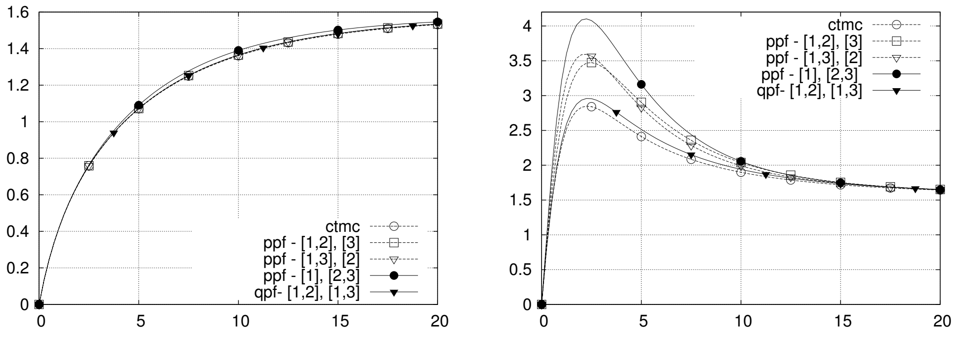


Figure 7.3: Open server network: variance of number of jobs at the server; starting with empty queues (left), with five jobs at the server (right).

depicted in Fig. 7.4. Requests arrive with a fixed rate $\lambda_1 = 0.6$ to the first station that constitutes a pre-processing step. After that, the possible paths for a request are stations 2-4, 2-5, 3-4 and 3-5 followed by a post-processing phase that takes place at station 6. The parameters that we consider are such that the routing probabilities are symmetric with $r_{1,2} = r_{2,4} = r_{3,5} = 0.5$ but the processing intensities are asymmetric with $\mu_2 = \mu_4 = 0.4$ and $\mu_3 = \mu_5 = 2$, i.e., the branch composed of stations 2 and 4 is slow while the one containing stations 3 and 5 is fast. The pre-processing and the post-processing stations are serving the jobs with intensity $\mu_1 = \mu_6 = 1$. The maximum number of clients for each queue is 50.

By considering each station and each arc as part of the dependency graph, the QPF decomposition suggested by the topology is not a DAG. Thus,

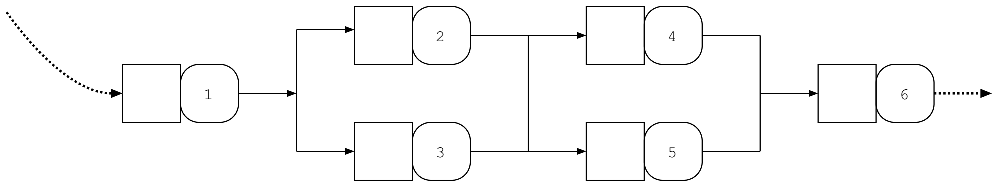


Figure 7.4: Multipath model.

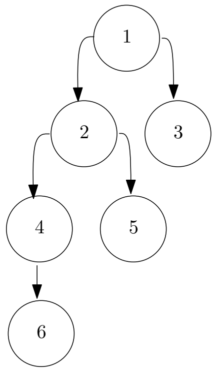


Figure 7.5: Quasi product form decomposition for the multipath model.

we need to discriminate among dependencies in order to generate a DAG that would lead to better accuracy. According to the arguments provided in Section 6.2, we neglect the dependencies on the less loaded queues which are in station 3 and 5. This strategy leads to the DAG depicted in Figure 7.5. The necessary marginals with this DAG are all two dimensional: $P\{X_1 = x_1, X_2 = x_2\}$, $P\{X_1 = x_1, X_3 = x_3\}$, $P\{X_2 = x_2, X_4 = x_4\}$, $P\{X_2 = x_2, X_5 = x_5\}$ and $P\{X_4 = x_4, X_6 = x_6\}$.

We tested the model starting from two different initial states. We first consider the case in which initially all queues are empty. In the second case the system starts with 10 requests in stations 2, 3, 4, and 5. Expectations and variances of the number of jobs at stations 4, 5 and 6 for the two cases are depicted in Fig.s 7.6 and 7.7. By comparing the two figures, one can observe to what extent starting from the second initial state penalizes station 4. After about 10 time units the average number of clients at station 4 is about 14 while the same average never exceeds 3 starting from an empty system. The longer run effect can be seen instead looking at the variances: for station 4 this quantity is increasing up to about 100 time units and for station 5 as well it reaches much higher values than in case of starting from an empty network. All these behaviours are captured well by the QPF approximation. In Fig. 7.8 we depicted the probabilities of having no clients at station 4 and 6. On these curves as well one can observe the effect of the choice of the initial state.

The network is composed of $51^6 = 1.76 \times 10^{10}$ states. The number of ODEs for the QPF approximation is $5 \times 51^2 = 13005$. The presented results were calculated in about one minute.

7.1.3 On-demand production system

As last queuing network, we consider again the network proposed in Figure 5.2. But, to put under stress the QPF approximation, we added two more loops that depart from stations 3 and 5 and lead to station 1 (the new version of the net is depicted in Fig. 7.9).

The QPF decomposition we apply is described by the DAG depicted in Figure 7.10.

The differences between the set of edges of the DAG and the routing of the network are the arcs $(3,1)$, $(3,2)$, $(3,4)$, $(5,1)$, $(5,4)$, and $(5,6)$. Four of these arcs are not present in the DAG because they form cycles. The other two, $(3,4)$ and $(5,6)$, are excluded instead to keep low the number of dimensions of the marginal distributions that are necessary to compute the approximation. With the above described DAG the necessary marginal distributions are $P\{X_1 = x_1, X_2 = x_2\}$, $P\{X_2 = x_2, X_3 = x_3\}$, $P\{X_2 = x_2, X_4 = x_4\}$, $P\{X_4 = x_4, X_5 = x_5\}$ and $P\{X_4 = x_4, X_6 = x_6\}$.

The parameters that we use are $\lambda_1 = 0.4$, $r_{2,3} = r_{4,5} = 0.5$, $r_{3,1} = r_{5,1} =$

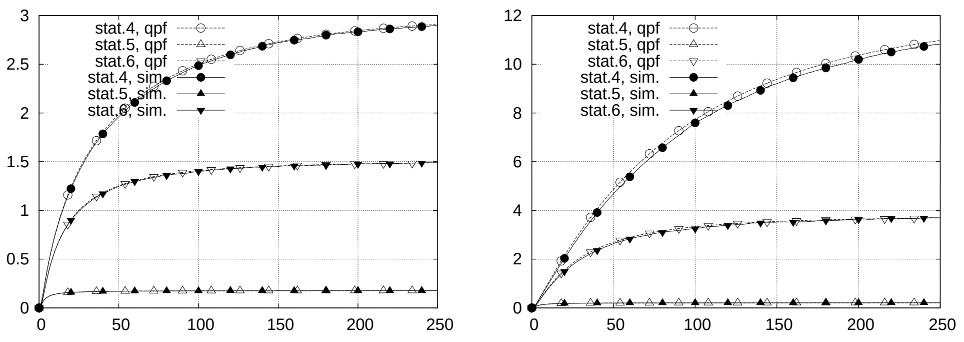


Figure 7.6: Multi-path: expectations (left) and variances (right) of the stations 4, 5 and 6 over time starting from state $\{0,0,0,0,0,0\}$

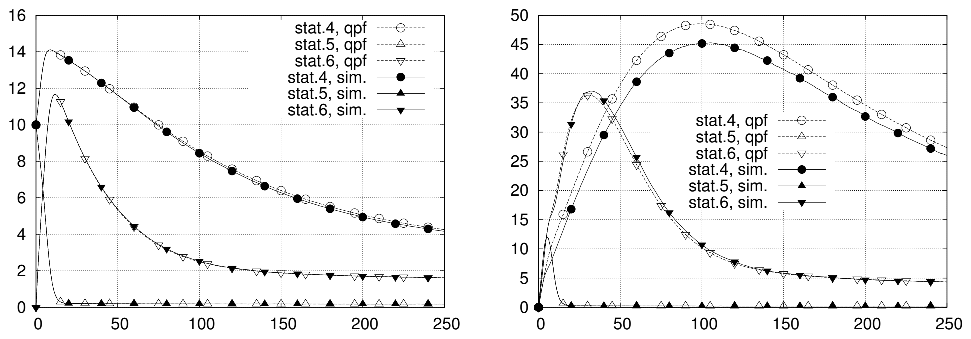


Figure 7.7: Multi-path: expectations (left) and variances (right) of the stations 4, 5, and 6 over time starting from state $\{0, 10, 10, 10, 10, 0\}$

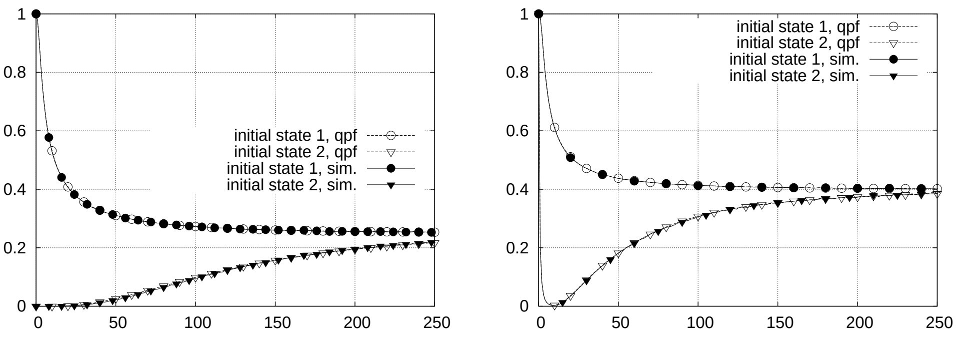


Figure 7.8: Multi-path: Probability of empty queue in station 4 (left) and 6 (right) over time starting from the two different initial states

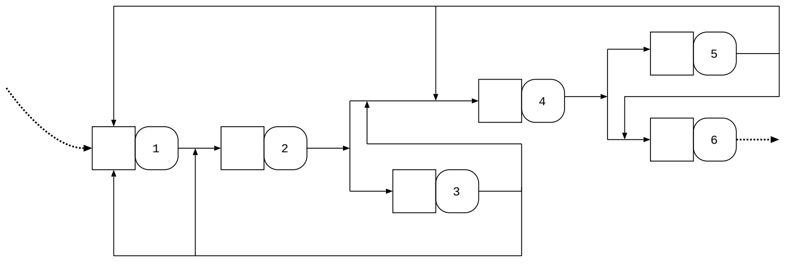


Figure 7.9: On-demand production system with two more loops

0.2, $r_{3,2} = r_{5,4} = 0.7$, $\mu_1 = \mu_2 = \mu_4 = \mu_6 = 1$, and $\mu_3 = \mu_5 = 1.5$. Note that with these parameters clients enter the cycles with high probability. The maximum number of clients for each queue is 50. We assume that the queues

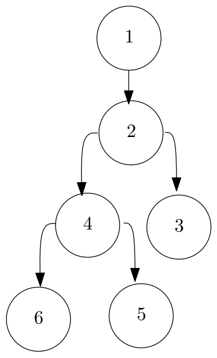


Figure 7.10: DAG describing the quasi product form decomposition for the on-demand production system

are empty initially. The number of states, the number of ODEs representing the QPF and the computation times are the same as in case of the example in Sec. 7.1.2.

In Fig. 7.11 and 7.12 we show the mean and the variance of the number of clients at the stations as function of time. Fig. 7.13 depicts instead the probability of the empty queue. It can be seen that the QPF approximation provides a precise view of these quantities.

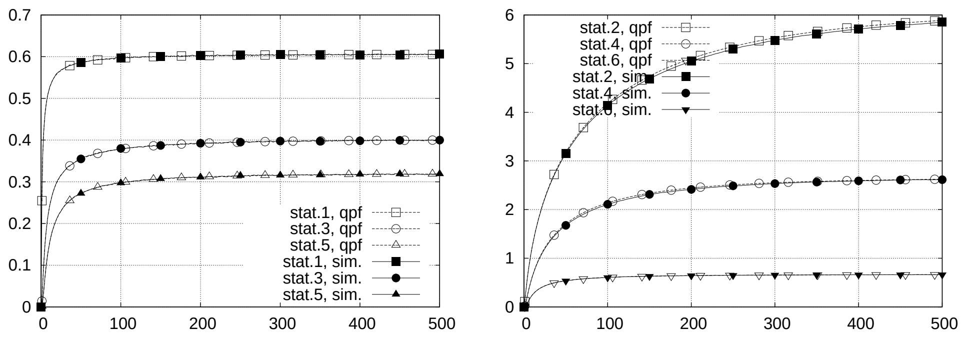


Figure 7.11: On-demand production system: expectation of the number jobs at the stations

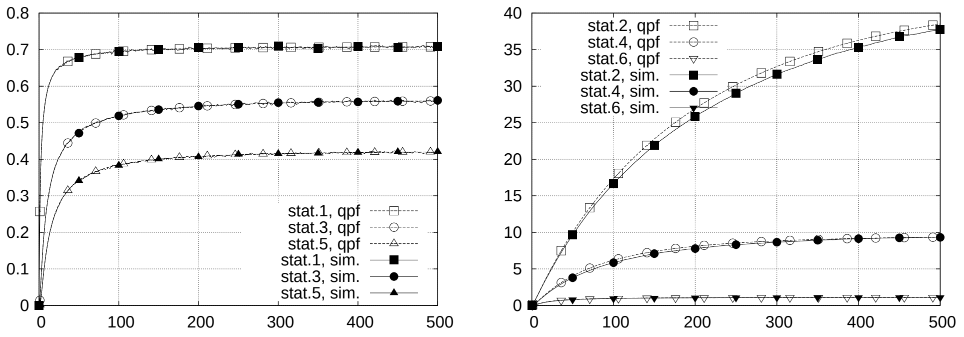


Figure 7.12: On-demand production system: variance of the number jobs at the stations

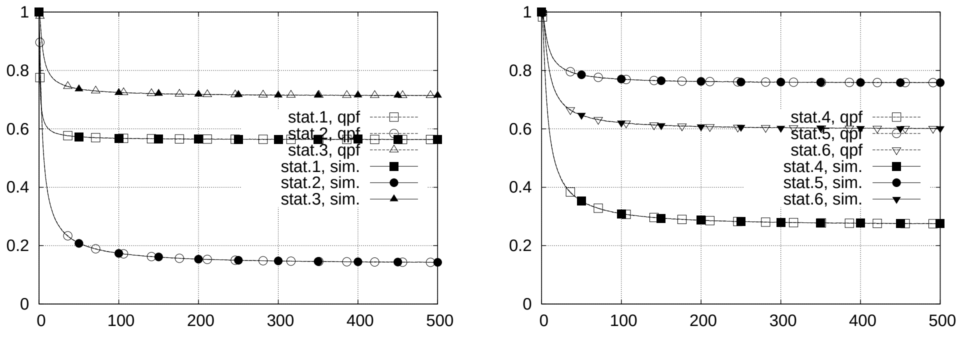


Figure 7.13: On-demand production system: probability of empty queue at the stations

7.2 Modulation through switches

7.2.1 Exclusive switch model

The model is described in Example 8 and as anticipated there, if the unbinding of the promoter is unlikely, the exclusive switch model behaves in a bistable way because either of the two proteins, P_1 and P_2, can monopolize the promoter region of the *Dna* and obstruct consequently the growth of the other. In this situation, the amounts of the two proteins are inversely correlated in such a way that a high number of molecules of P_1 corresponds to low quantities of P_2 and vice-versa. Intuitively, this fact seems to indicate that the quasi product form assumption represented in Figure 6.4 leads to imprecise approximation because it does not consider directly the joint

# Set	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}
1	0.5	0.5	0.005	0.005	0.01	0.01	0.005	0.005	0.5	0.5
2	1.0	2.0	0.1	0.1	0.01	0.01	0.005	0.005	1.0	2.0

Table 7.1: Exclusive switch : The two sets of parameters used to perform the tests

distribution of P_1 and P_2. Nevertheless, as it will be illustrated by the presented numerical results, the negative correlation between the two proteins and the associated bistable marginal distributions can be captured, in an indirect manner, by the state of the promoter whose description is given by the places Dna, $Dna.P1$ and $Dna.P2$.

In order to have an exact comparison between our approximation and the exact solution, we chose two sets of parameters (reported in Table 7.1) in such a way that it is very unlikely that the growth of P_1 and P_2 goes up to 200; this allows us to compute exact solution of the CTMC. The first set is symmetric, i.e., the two proteins have the same probability to monopolize the promoter region. With the second set P_2 has an advantage over P_1.

As mentioned earlier, the initial state is $x = |1, 0, 0, 0, 0|$. In Figure 7.14 the marginal protein distribution is depicted for three time points (because of the symmetric settings the probabilities are identical for P_1 and P_2). One can note that already after 100 time units, the protein distribution gets split in two parts forming a bistable distribution. The quasi product form approach is able to catch precisely the shape of this distribution. As time elapses the bistability gets more marked. At time $t = 250$ the approximation still provides a good picture of the behaviour of the model but the numerical values are not as precise as for smaller values of t. In steady state, which can be observed at $t = 1000$, the quasi product form assumption captures well the bistability but gives a quite inaccurate approximation of the lower probabilities (those less than 10^{-3}) and of the probability of having zero of one of the two proteins.

In Figure 7.15 we show the mean and the variance of the protein quantity. The approximate mean is very accurate for all time points while the variance is underestimated. The fact that the variances are less accurate is not surprising since each marginal distribution perceives in an exact way only one binding. (One such example is the summation in (6.7)). The overall effect of this is that the approximate variance is lower than the exact one.

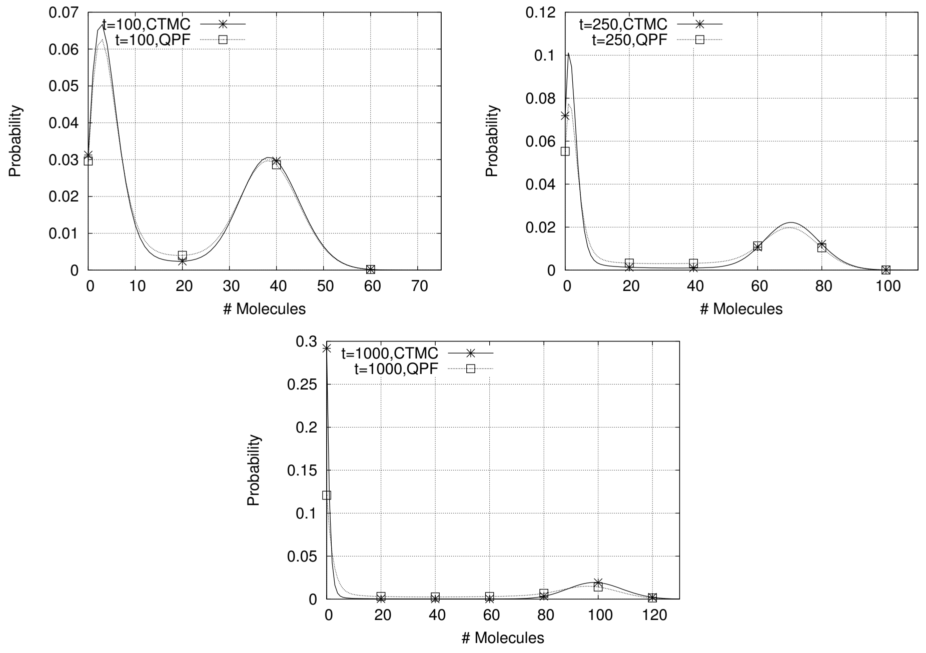
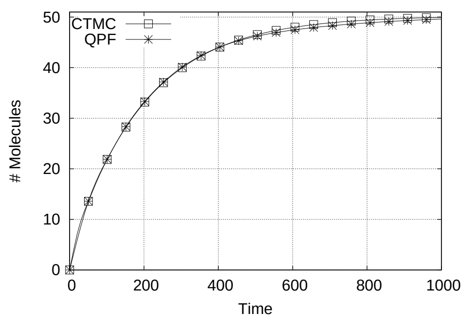


Figure 7.14: Exclusive switch: marginal distributions of P_1 (P_2) at time $t = 100, 250$ and 1000 using the symmetric set of parameters



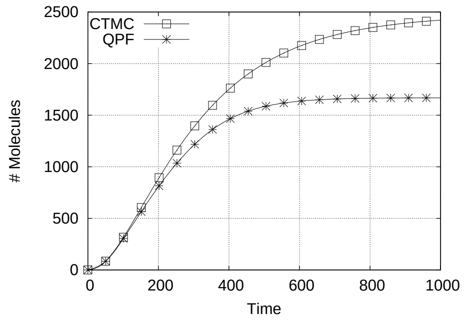


Figure 7.15: Exclusive switch: The expectation and the variance of the quantity of P_1 (P_2) as function of the time with the symmetric set of parameters

Parameter set	Method	free Dna	Dna bound to P_1	Dna bound to P_2
1	**CTMC**	0.004956	0.497521	0.497521
1	**QPF**	0.009597	0.495201	0.495201
2	**CTMC**	0.023392	0.293140	0.657854
2	**QPF**	0.024851	0.284785	0.690362

Table 7.2: Exclusive switch : probability of having the Dna free, bound to P_1 and bound to P_2 after 1000 time units

In Table 7.2 we provide the probabilities of having the Dna promoter region free, bound to P_1 and bound to P_2 after 1000 time units. The approximation captures the fact that the promoter region is free with low probability but the numerical value is almost twice larger than the real value.

In order to show that the quasi product form approximation does not take advantage of the symmetry of the previous setting, we provide now the results for the asymmetric set of parameters. As shown in Table 7.1, in this case the production of P_2 is two times faster than that of P_1. Figure 7.16 depicts the distribution of the two proteins after 50 and 100 time units (with this parameter set steady state is almost reached at $t = 100$). One can see that the quasi product form approximation provides a very precise view of the protein distributions. Consequently, the expectations and the variances (Figure 7.17) and the probabilities of the three promoter regions (Table 7.2) are reproduced accurately as well.

The numerical integration of the ODEs associated with quasi product form assumption required less than 20 seconds whereas the solution of the

CTMC through uniformisation took several minutes. Considering space complexity, if the considered maximum for the protein quantities is p_{max}, then the quasi product form assumption leads to $3 \times 2 \times (1 + p_{max})$ equations while the number of states in the original CTMC is $3 \times (1 + p_{max})^2$.

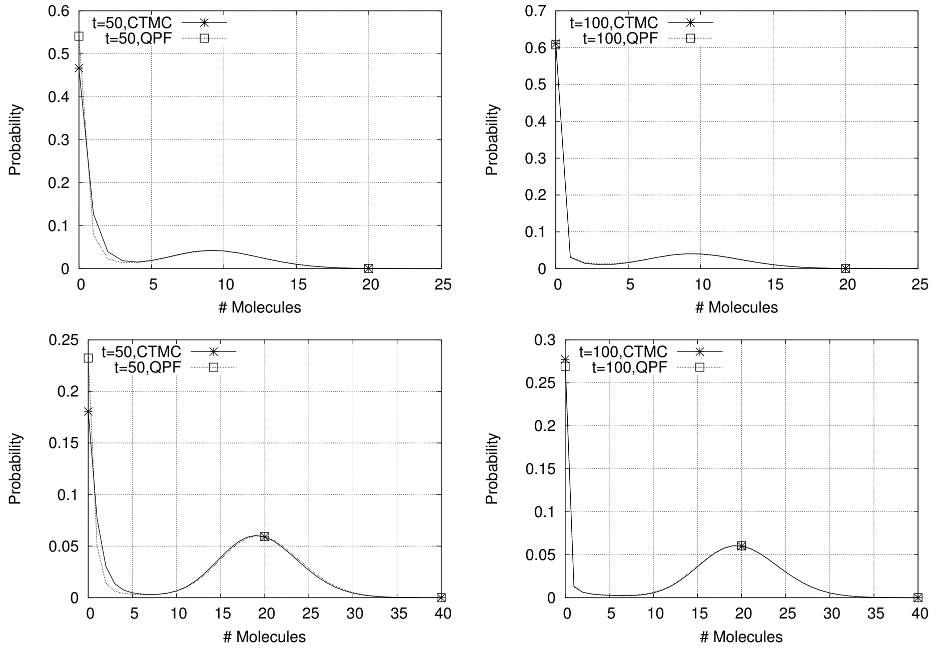


Figure 7.16: Exclusive switch: marginal distributions of P_1 and P_2 at time $t = 50, 100$ using the asymmetric set of parameters

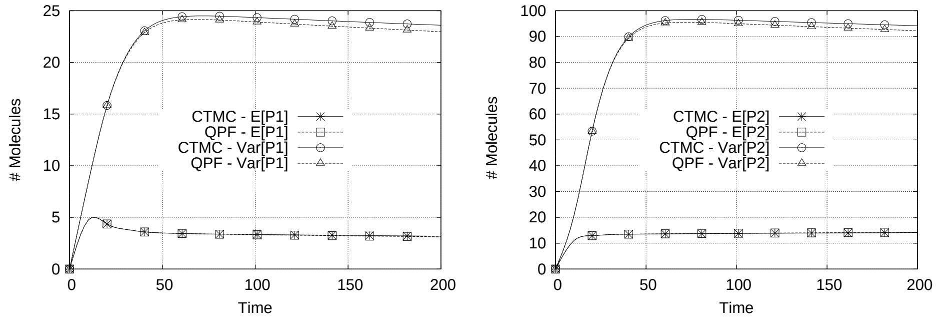


Figure 7.17: Exclusive switch: the expectation and the variance of the quantity of P_1 (left) and P_2 (right) as function of the time with the asymmetric set of parameters

For this model we illustrate the preliminary error evaluation proposed in Section 6.4. Figure 7.18 reports the maximum error on the derivative as a function of the time. For both parameters sets, the error is low all along the calculations and it stabilizes as the process reaches steady state. For the second parameter set, the error is somewhat higher and it reflects the fact that in this case the original probabilities are captured with less precision (see Table 7.2).

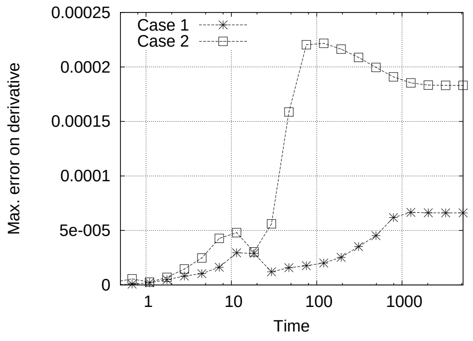


Figure 7.18: Error measure for the exclusive switch for the two different sets of parameters.

7.2.2 Multi-attractor model

As a second example, in order to test the quasi product form approximation with a more complex biological model, we propose a part of the multi-attractor model considered by Zhou et al. [92] describing the interactions among three genes, namely, *Pax*, *Mafa*, and *Delta*. Each gene has a corresponding protein that is able to bind itself to promoter regions on the *Dna*. The SPN representing the model is depicted in Figure 7.19 where the dotted boxes denote the three switches corresponding to the three promoter regions. Considering all possible bindings, the model involves 13 places. The first 10 of these can assume only boolean values and represent all the possible states of the three promoter regions. The last three, instead, describe the number of molecules of proteins present in the system.

The places Pax, $Mafa$ and $Delta$ represent the proteins whereas DD, DM and DP denote the promoter regions. The "dot" has the same meaning as in case of the exclusive switch model. We have four types of transitions:

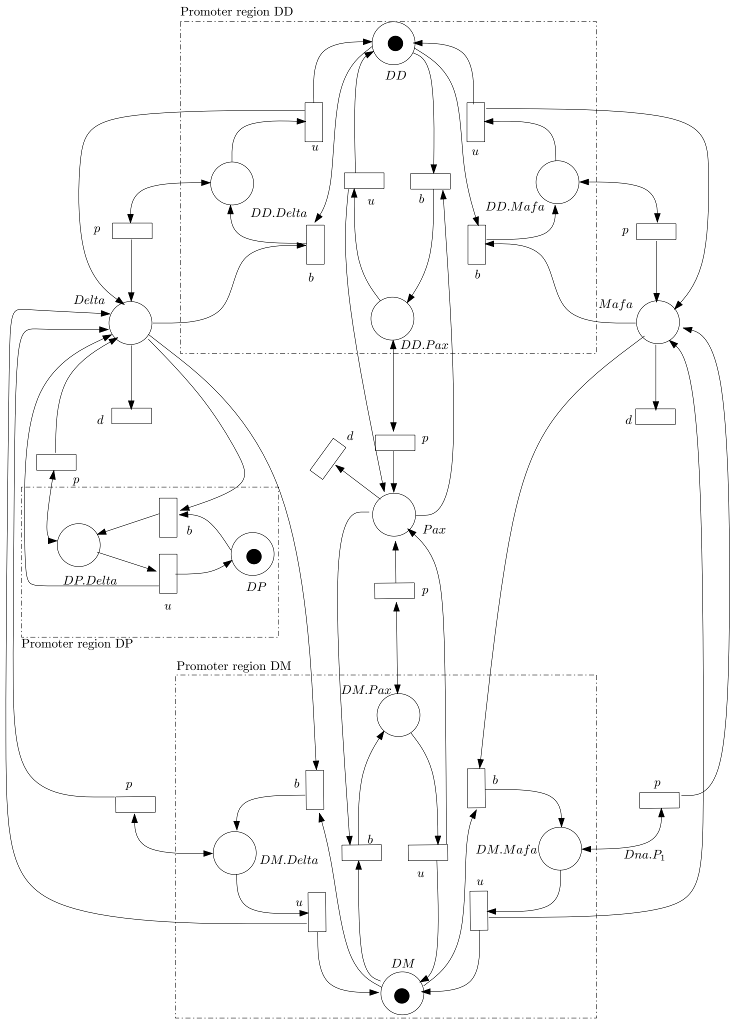


Figure 7.19: The Multi-attractor model: Petri net

- **d** (degradation): represents the proteins degradations and depends on the amount of proteins present (infinite server policy). For example, the transition $Delta \rightarrow \emptyset$ occurs with intensity $k_d \cdot [Delta]$ where k_d is a positive constant and $[Delta]$ is the amount of $Delta$ in the state.
- **b** (binding): by mean of these reactions, a protein occupies a promoter region. For instance, the transition $Delta + DD \rightarrow DD.Delta$ moves a token from DD to $DD.Delta$ and removes a token from $Delta$ with

intensity $k_b \cdot [Delta]$.

- **u** (unbinding): through these reactions a promoter region returns "free". An example is the transition $DD.Delta \rightarrow Delta + DD$ which moves a token from $DD.Delta$ to DD and returns the token used to occupy the region to $Delta$.
- **p** (production): these reactions describe the production of proteins. Examples are the transition $DD \rightarrow DD + Delta$ (unbounded production) or $DD.Delta \rightarrow DD.Delta + Delta$ (bounded production).

As in the case of the exclusive switch, the overlap of the common promoters leads to invariants:

$$\begin{aligned} MD + MD.Pax + MD.Mafa + MD.Delta &= \\ DD + DD.Pax + DD.Mafa + DD.Delta &= \\ PD + PD.Delta &= 1 \end{aligned}$$

Accordingly, the production of the proteins is modulated in $2 \times 4 \times 4 = 32$ different ways corresponding to all the possible combinations of the states in which promoter regions can be.

The state space of the underlying CTMC is infinite and the number of states having a non negligible probability mass blows up over three dimensions. In this situation, if the parameters are not such that the protein quantities remain quite low, any analytical solution of the CTMC is unfeasible by using common techniques whereas the analysis through the quasi product form assumption remains possible.

The quasi product form assumption we propose is similar to the one used in case of the exclusive switch. It is described by a DAG of 4 nodes in such a way that node v_1 is associated with all the species representing the promoter regions, and nodes v_2, v_3 and v_4 correspond to *Pax*, *Mafa*, and

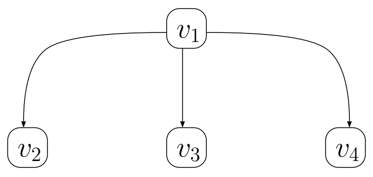


Figure 7.20: The Multi-attractor model: the DAG describing the quasi product form (right)

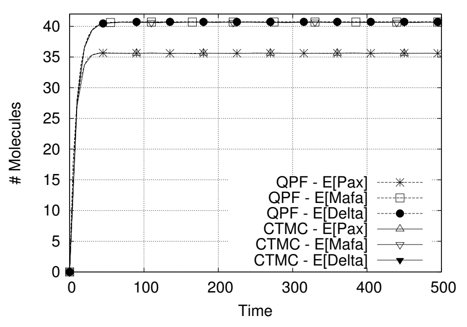
Delta, respectively. As depicted in Figure 7.20, the DAG has three edges indicating that the dependencies among the proteins is taken into account, in an indirect manner, through the state of the promoter regions. This implies that the resulting system of ODEs has one equation for each protein, for every possible protein quantity and every possible state of the promoter region. Consequently, if the considered maximal protein quantity is p_{max} for every protein then the number of equations is $3 \times (p_{max} + 1) \times 32$. This is much less than the number of states in the original CTMC which, considering the same range of protein levels, equals $(p_{max} + 1)^3 \times 32$.

We test the quasi product form approach on this model with three sets of parameters as reported in Table 7.3. Since the state space of the original model is large, we compare the results of the quasi product form approach with statistics obtained through the Monte Carlo simulation of the original CTMC. The initial state of the model is such that all the promoter regions are free and no proteins are present in the system.

Due to the low propensity of the binding reactions compared to the other rates, the first set of parameters represents the most desirable situation to apply the proposed quasi product form assumption. In fact, by using our assumption binding reactions are the only transitions that occur according to incoming flows that depend on other marginals. As an example, the state $Pr\{DP = 1.DM = 1, DD = 1, Pax = 0\}$ tends to switch to the state $Pr\{DP.Delta = 1, DM = 1, DD = 1, Pax = 0\}$ according to a time dependent intensity equal to $k_b \cdot E\left[Delta|DP = 1, DM = 1, DD = 1\right]$. If these reactions are much less frequent than the others then the distribution of a protein is barely influenced by another one and the quasi product form assumption is plausible. Figures 7.21 and 7.22 reflect this situation showing a perfect match between the results obtained through the quasi product form approximation and the simulations of the original CTMC.

Parameter set	k_d	k_b	k_u	k_p
1	0.1	0.01	1.0	5.0
2	0.1	0.01	0.001	5.0
3	0.1	1.0	1.0	5.0

Table 7.3: Multi-attractor model: the three sets of parameters used to perform the tests where k_d refers to degradation reactions, k_b and k_u correspond to binding and unbinding reactions, respectively, and k_p to production reactions.



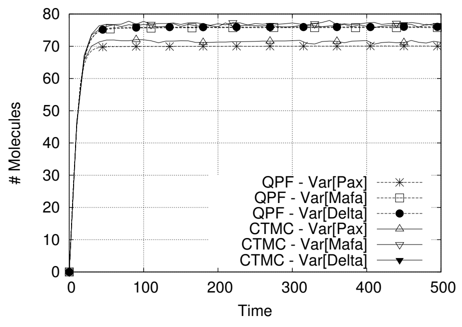
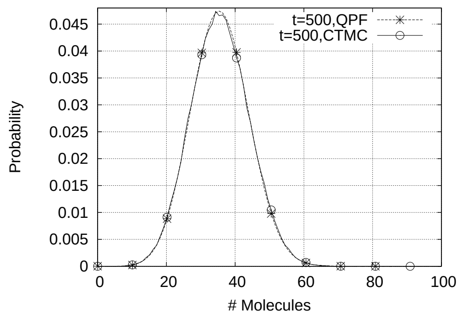
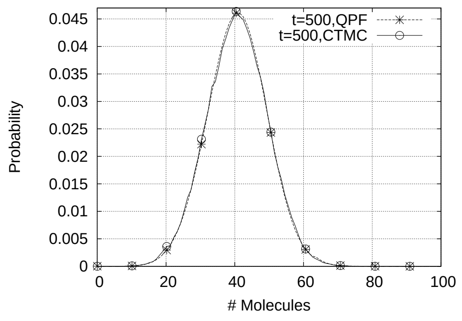


Figure 7.21: Multi-attractor model: expectations (left) and variances (right) of the three proteins with $k_d = 0.1, k_b = 0.01, k_u = 1, k_p = 5$





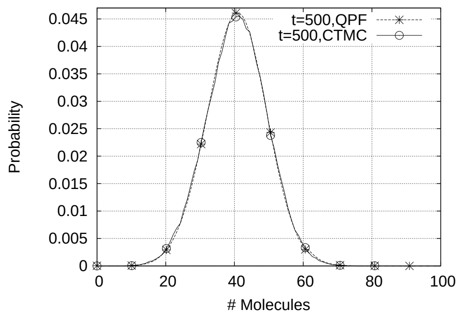


Figure 7.22: Multi-attractor model: marginal probabilities of *Pax* (left), *Mafa* (right) and *Delta* (below) with $k_d = 0.1, k_b = 0.01, k_u = 1, k_p = 5$

The second set of parameters is able to generate strong correlations among the distribution of the proteins (similarly, to those present in the exclusive switch model). This is achieved by setting the unbinding constants to a lower value (see Table 7.3) which implies that, even if they are rare, bindings will

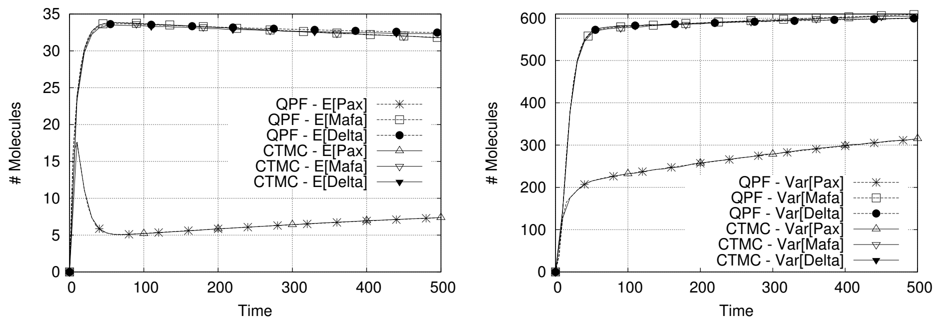


Figure 7.23: Multi-attractor model: expectations (left) and variances (right) of the three proteins with $k_d = 0.1, k_b = 0.01, k_u = 0.001, k_p = 5$

eventually occur and proteins can monopolize the promoter. Despite the fact that this setting is less favorable for the quasi product form assumption, the approximation, as it can be seen in Figure 7.23, catches both the expectations and the variances of the three proteins. Moreover, as shown in Figure 7.24, also the marginal distributions of the proteins are captured precisely. Note that all three proteins have bistable distributions but this is hard to see in case of *Pax* because this protein is at level 0 with high probability. The goodness of the approximation is evident from the curves representing the marginal distributions (Figure 7.24). Both the probability mass at zero and the rarer event around 50 are precisely reconstructed by the proposed approximation. Figure 7.25 provides a better picture of QPF accuracy by focusing on two intervals where the marginal probabilities of *Pax* assume small values, namely $[1, .., 10]$ and $[35, .., 70]$.

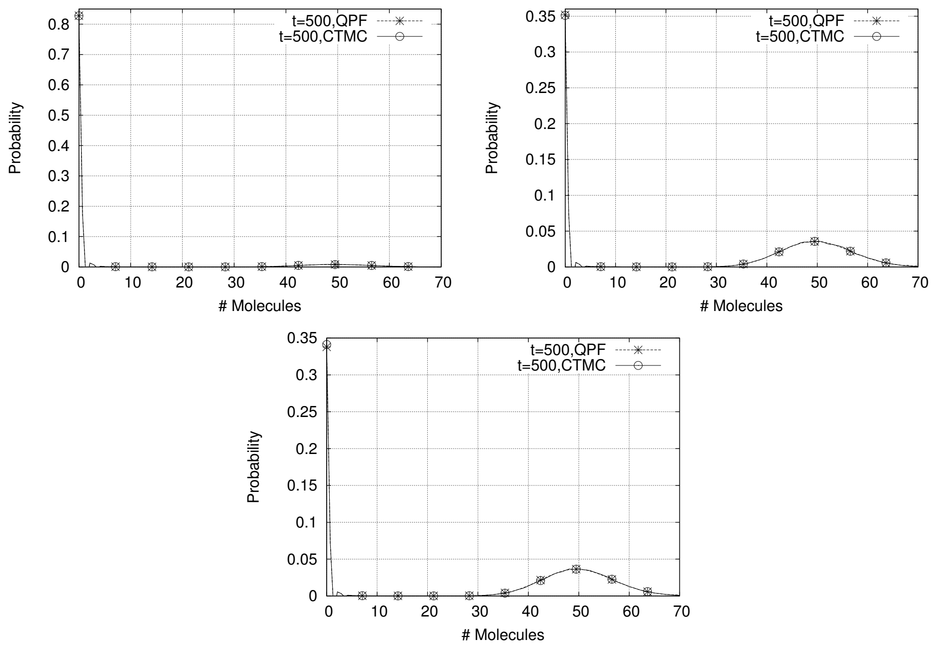


Figure 7.24: Multi-attractor model: marginal probabilities of *Pax* (left), *Mafa* (right) and *Delta* (below) with $k_d = 0.1, k_b = 0.01, k_u = 0.001, k_p = 5$

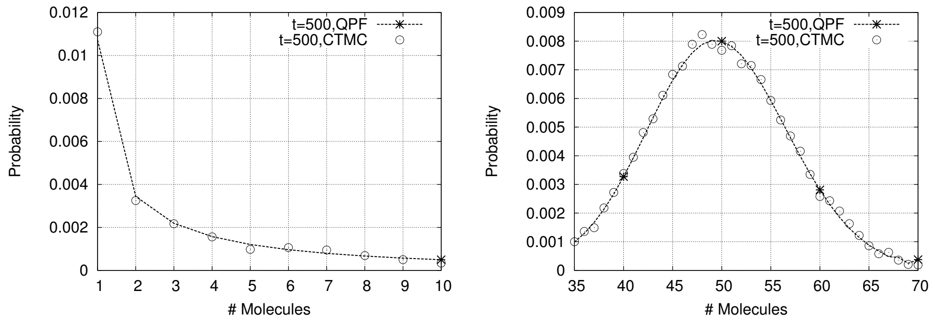


Figure 7.25: Multi-attractor model: rare probabilities of *Pax* marginal distribution, intervals $[1, 10]$ (left) and $[35, 70]$ (right) , with $k_d = 0.1, k_b = 0.01, k_u = 0.001, k_p = 5$

As last example, we provide a case in which the quasi product form approximation is not able to provide a good estimation of the probability distri-

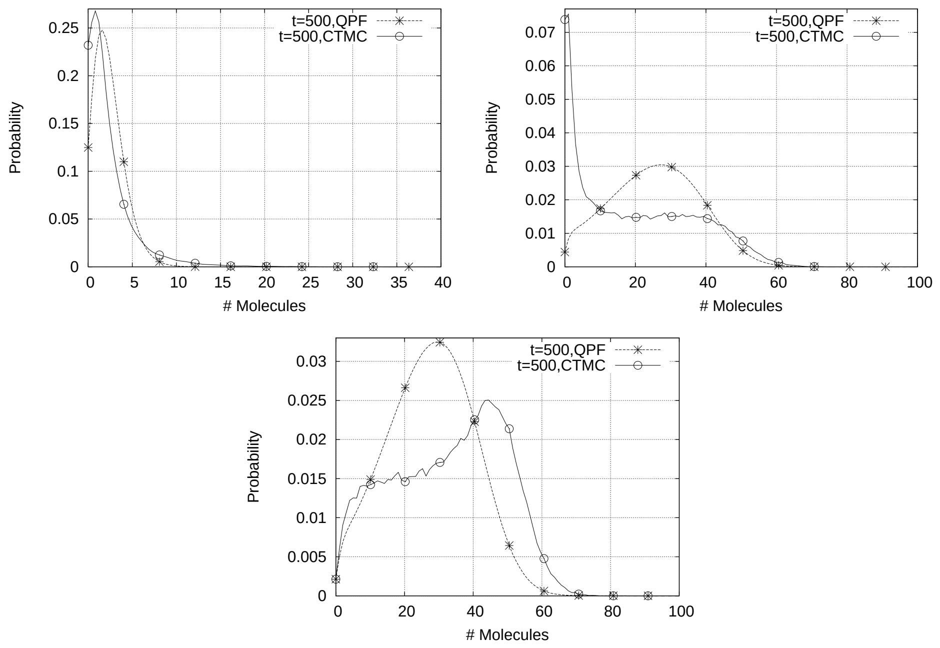


Figure 7.26: Multi-attractor model: marginal probabilities of *Pax* (left), *Mafa* (right) and *Delta* (below) with $k_d = 0.1, k_b = 1, k_u = 1, k_p = 5$

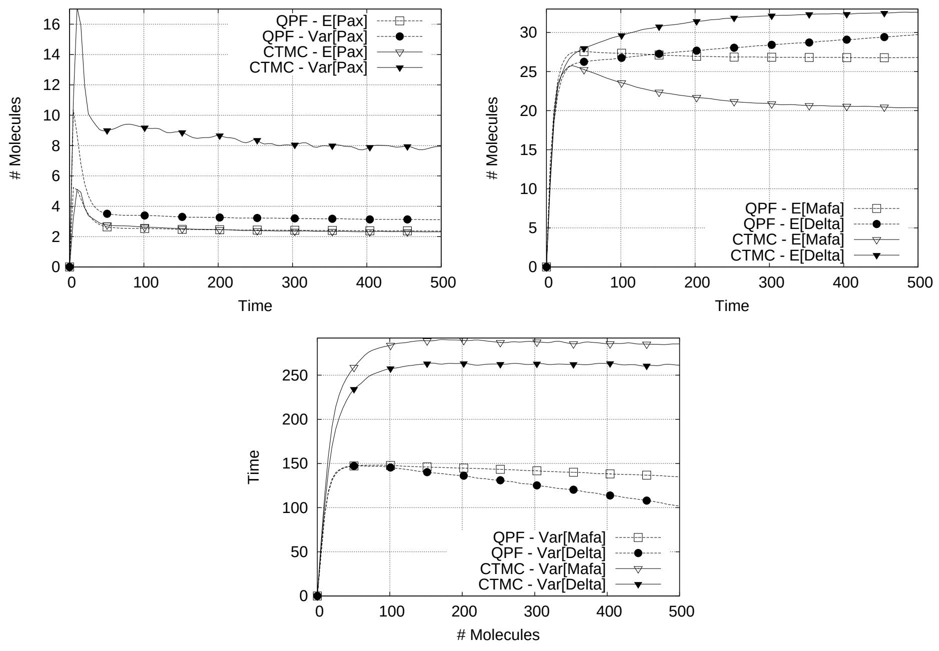


Figure 7.27: Multi-attractor model: expectations and variances of the three proteins with $k_d = 0.1, k_b = 1, k_u = 1, k_p = 5$

butions. In order to challenge the quasi product form assumption, we choose a set of parameters in which the binding and the unbinding reactions are extremely frequent. Since in our approximation the marginal distributions "communicate" only through expected values, we expect that the computations give a result which is similar to the original in average but is not able to catch the effects of the fluctuations given by the frequent bindings and unbindings. Figure 7.26 depicts the marginal distributions of the proteins and their approximations. In case of *Pax* the approximation is reasonable, while for *Mafa* and *Delta* the irregular shapes are not captured well. Nevertheless, we point out that, even if the peculiarities of the distributions are not captured (e.g., the peak near zero for *Mafa*), the approximated distributions provide a good picture of the support of the original distributions. Finally, in Figure 7.27, it is possible to observe the expectations and the variances of the three proteins. The slopes of the original curves are preserved by the approximation and the error over the trajectories is reasonable.

The computation of the quasi product form required about half an hour

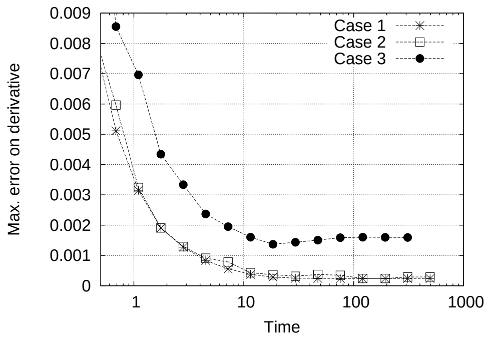


Figure 7.28: Error measure for the multi-attractor model for the different sets of parameters.

for all the test. By setting a limit of 100 molecules for the growth of all the three proteins the number of equations considered is 3.2×10^4. The corresponding original, three-dimensional state space of the proteins contains about 3×10^7 states. Common, exact analysis techniques cannot handle such amount of states using conventional hardware.

As last, Figure 7.28 depicts the maximum error on the derivative as function of the time by using the different parameter sets. The error is higher and the difference between the well-approximated cases (first and second set of parameters) and the poorly approximated case (third set of parameters) is reflected by the error measure.

7.3 On demand multilevel manufacturing system

As last model, we propose the multilevel manufacturing production line whose SPN is depicted in Figure 7.29. The system is composed of five levels which describe the assembling of a product from the instant in which it is ordered to the moment of its delivery. The dotted block at the top of Figure 7.29 represents the phase in which requests for products arrive, are accepted, and immediately sent to the four manufacturing stations placed at the beginning of the line.

This phase is described through: transition t_1 which considers requests arrivals as Poisson events; place L_0 which represents requests that are waiting for acceptance; transition t_2 which models the acceptance of a request and its virtual change of state from "pending" to "in production".

Transition t_2 is a fork because we assumed that the final product is composed of four different parts that have to be built and assembled; thus, after the acceptance the order is submitted to four different manufacturing stations to alert them that a new item has to be produced.

Once submitted, requests arrive at the first level of the line where each manufacturing station has the task to carry on the construction of one of the components required for the product to deliver, namely A, B C and D. Each station serves requests one by one and pushes forward the produced components independently from the other stations; this means that a component can arrive to the next assembling stage even if the other three components required to satisfy the order are still waiting to be constructed.

This phase is described by places A_1, B_1, C_1 and D_1 which represent request for production of component A, B, C and D respectively, and by the transitions t_3, t_4, t_5 and t_6 which model the building of a component and its transportation to the next level.

The block labeled with "Production Level 2" in Figure 7.29 represents the first assembling of the parts produced in the previous level. In particular, a component A is assembled with one of type B and C is put together with D. Transitions t_7 and t_8 represent the task through the synchronization of places A_2 with B_2 and C_2 with D_2.

In the same way the results of the assembling are combined together in the next stage of the production (transition t_9 which synchronizes places AB_3 and CD_3). Finally, in the last level the final product is delivered or backlogged in production because damaged. The backlogging is modelled through transition t_{11} whereas transition t_{10} describes the delivery of the product.

In order to put under stress our approximation we assume that every transition of the model but t_{10} and t_{11} is single server. Furthermore, we assume that each place has a finite capacity of 25 items but we choose the parameters settings in such a way that every place reaches the saturation point with negligible probability. For every production level the transition rates are different, i.e. $\mu_3 = \mu_7 = \mu_9 = 2$, $\mu_4 = \mu_8 = 3$, $\mu_5 = 4$, $\mu_6 = 5$. Requests are accepted with rate 1 and deliveries are performed according to an infinite server policy with rate $t_{10} = 0.5$ and backlogging probability of 0.01.

Observing Figure 7.29, it is evident that all the places at production levels

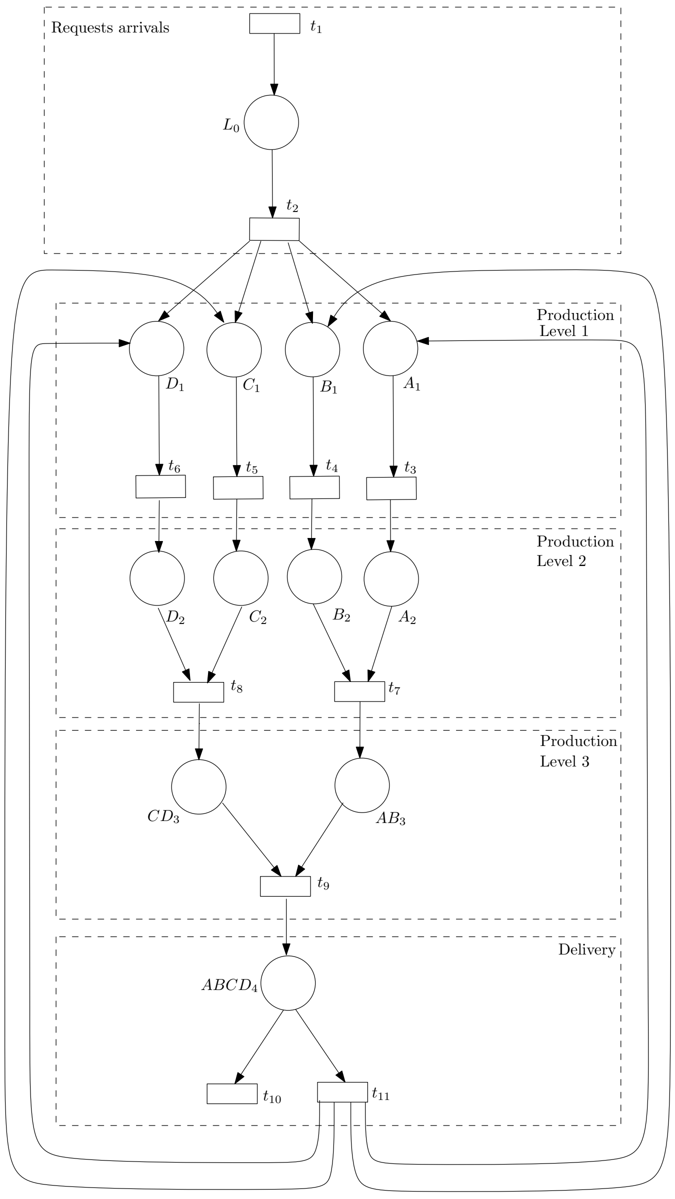


Figure 7.29: On the demand multilevel manufacturing system.

1, 2 and 3 are heavily dependent among each other due to the fact that they are generated by the same event (firing of t_2). Moreover, the state space is characterized by the following two invariant laws:

1. the sum of the tokens present in A_1 and A_2 (C_1 and C_2) is always equal to the sum of the tokens in B_1 and B_2 (D_1 and D_2)

2. the sum of tokens in places AB_3,A_2 and A_1 (or B_2 and B_1) is equal to the summation of the tokens present CD_3, C_2 and C_1 (or D_2 and D_1).

This scenario would suggest that the only way to apply the quasi product form is to maintain the correlation among all the places belonging to the production/assembling stages.

Indeed, this would lead to an accurate approximation, but the computational cost would get quickly unbearable for a common hardware. Thus, we decided to generate the DAG in such a way that each marginal describes at most the distribution of three places. The DAG that we chose, depicted in Figure 7.30, represents a quite strong assumption, i.e. the components A, B, C and D are produced independently from each other. Furthermore, the

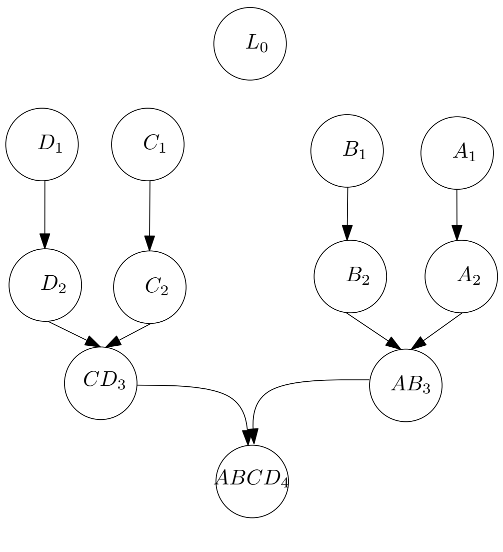


Figure 7.30: DAG representing the quasi product form assumption used to analyze the multilevel manufacturing system.

proposed DAG implies that we have to take care explicitly of the invariants present among levels; as an example, the marginal $\{A_1, A_2\}$ is in complete product form with that $\{B_1, B_2\}$; however, every couple (a_1, a_2) is affected only by those configurations of B_1 and B_2 for which $b_1 + b_2 = a_1 + a_2$.

As first test, we assume that requests arrive with rate $\lambda = 0.2$. hence, the load of the net is quite low. Figure 7.31 depicts the mean and variance of the number of tokens in places A_1, A_2, AB_3 and $ABCD_4$. It is possible to observe that while the approximation of the two measures is quite accurate for place A_1 the curves describing place A_2 diverge from the original behaviour quite soon; in particular, they are overestimated.

This is because, by assuming our decomposition, the marginal $\{A_2, B_2, AB_3\}$ is "delegated" to compute the departures to AB_3 by mean of t_7 but it considers the arrivals from A_1 and B_1 as uncorrelated independent Poisson processes. This is a consequence of the fact that, according to the proposed DAG, we lost the knowledge about the fact that tokens arrive in A_1 and B_1 at the same time.

Thus, the correlation between the firing of t_3 and t_4 is lost and the arrivals in A_2 and B_2 are considered only by mean of their expected incoming flows that have intensities proportional to the parameters μ_3 and μ_4 multiplied by the current probability to have at least one token in A_1 and B_1, respectively. In particular, these values will be small since the load of the net is quite low and consequently the probability to find A_1 and B_1 empty is high.

In other words, the underestimation of the firing of t_7 is a consequence of the fact that the marginal $\{A_2, B_2, AB_3\}$ perceives the arrivals in places A_2 and B_2 properly from the point of view of their expectations (by mean of the marginals $\{A_1, A_2\}$ and $\{B_1, B_2\}$) but cannot catch the fact that after an arrival in A_2 (B_2) the subsequent most likely event is an arrival in B_2 (A_2). Then, according to the QPF decomposition, transition t_7 is blocked with an higher probability than in the original case.

The divergence of the variance reflects the fact that, by assuming independent Poisson arrivals into the two places, the number of configurations enabling t_7 with a non negligible probability increases drastically.

In Figure 7.31 we report the mean and the variance of the token distribution of place $ABCD_4$ (bottom right). These measures should suffer the errors generated in the previous levels. Instead the slope of the original curves is preserved by the QPF and the gap between approximated and original curves is sensibly reduced. We conjecture that, preserving the correlations of the places involved in synchronizations, the decomposition is able to reconstruct part of the strong dependencies lost during the first levels.

As last, Figure 7.31 reports also the mean and the variance of place AB_3, the gap between original and approximated curves is slightly smaller than for place A_2. We skip the curves for the other places composing the first, the second and the third level because they have similar errors.

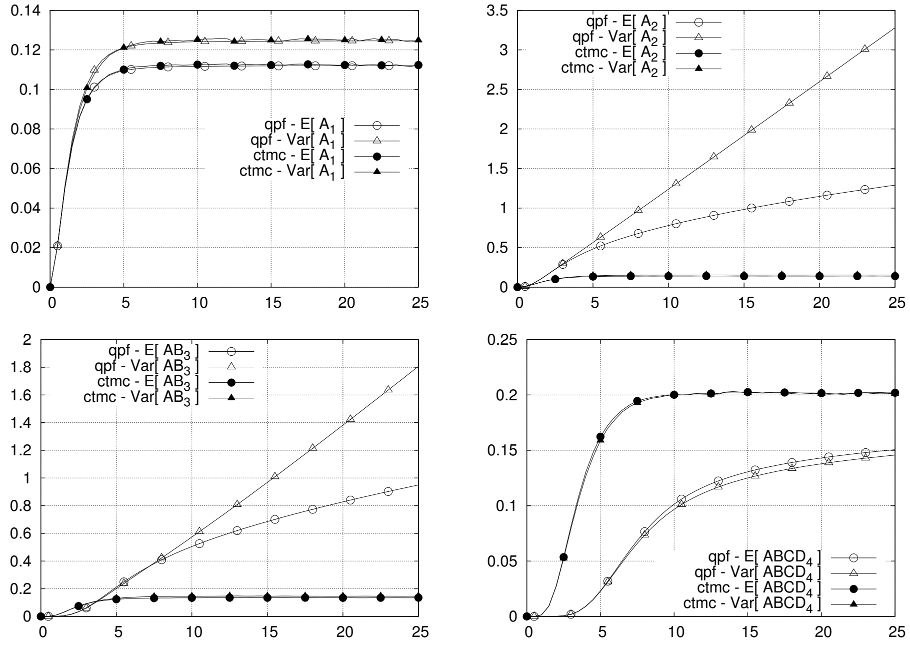


Figure 7.31: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) with requests arrival rate equal to $\lambda = 0.2$

We further investigated the method by increasing the load of the net; in particular, we tested $\lambda = 0.4$ and $\lambda = 0.8$. Figures 7.32 and 7.33 shows that by increasing the arrival rate the approximation of $ABCD_4$ gets more accurate: for the first case at time 25 the relative error is reduced to 0.2% whereas for the second case it is around 0.12% only.

In order to put under stress the QPF approximation, we tested the case where a final product returns to the first level with probability 0.25. Figure 7.34 shows that although the approximation of A_1 gets slightly worse those of A_2, AB_3 $ABCD_4$ maintain the same accuracy.

In the following two tests, we investigate the behaviour of our approximation in settings where we loosened the correlations among the places. In the first scenario, we performed the analysis by starting from the situation where all the places but L_0 and $ABCD_4$ contain five tokens and chose as arrival

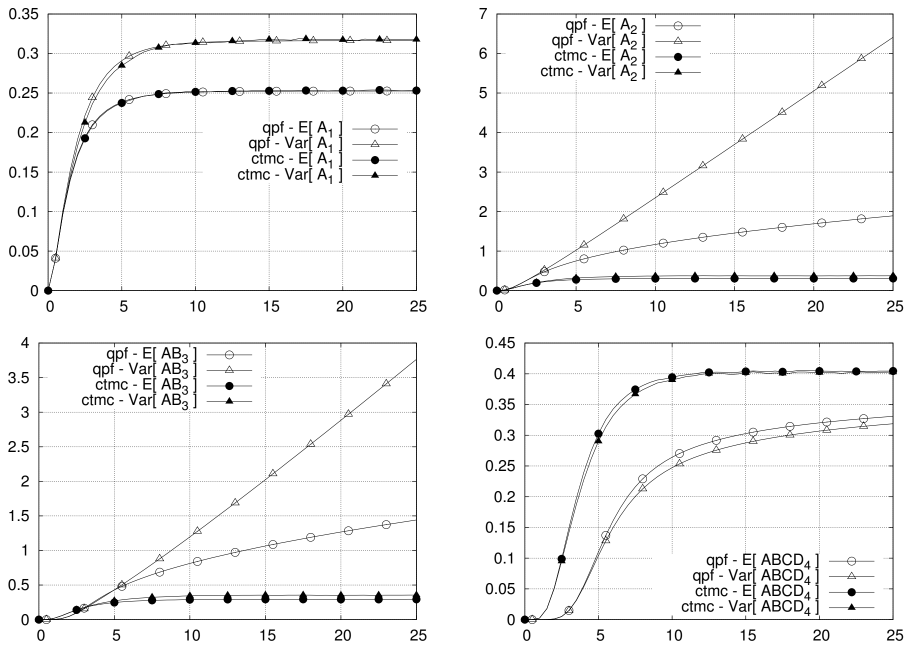


Figure 7.32: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) with requests arrival rate equal to $\lambda = 0.4$

rate $\lambda = 0.2$. Thus, all the synchronizations are enabled at the very beginning of the analysis. For this reason, we expect to see a preamble where also the measures of the nodes placed in the middle of the net are approximated correctly.

Figure 7.35, where mean and variance of places A_2 and AB_3 are depicted, confirms our belief. In fact, observing the plots it is possible to note that until 4 time units both the approximated mean and variance follow precisely the original curves; then, the variances start to diverge as in the previous cases.

Figure 7.36 shows the results provided by a test alike to the previous with the only difference that, this time, we started the analysis from a pseudo-random state, i.e. $|0, 3, 5, 2, 4, 2, 3, 2, 3, 3, 2, 0|$. As a consequence, several places in the middle of the line will contain at least one token all along the time. This situation further reduces the difference between the original and

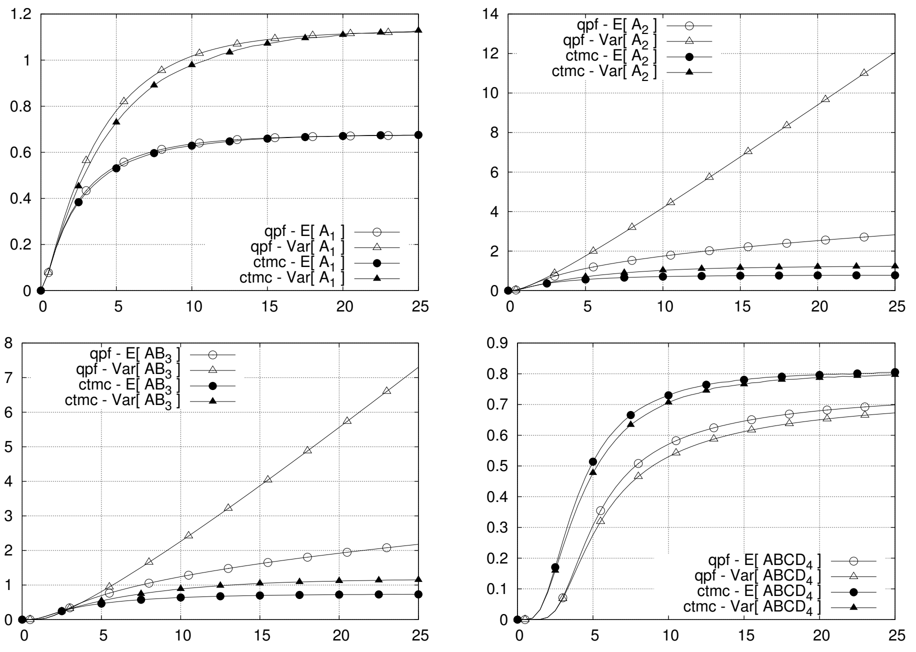


Figure 7.33: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) with requests arrival rate equal to $\lambda = 0.8$

the approximated curves all along the transient period.

Figure 7.37 reports the probability to find empty the place $ABCD_4$. In general, all the approximated curves follow the behaviour of the original model with acceptable accuracy[1].

According to the decomposition, the QPF approximation required the marginals : $\{L_0\}$, $\{A_1, A_2\}$, $\{B_1, B_2\}$, $\{C_1, C_2\}$, $\{D_1, D_2\}$, $\{A_2, B_2, AB_3\}$, $\{C_2, D_2, CD_3\}$ and $\{AB_3, CD_3, ABCD_4\}$. Since every place was bounded up to 25, the total number of equations is $26 + 4 \times 26^2 + 3 \times 26^3 = 55458$ against the 9.5×10^{16} of the original model. All the computations required around 8 minutes.

[1]We do not report the curves of the case with backlogging probability equal to 0.25 because it is very similar to those of the case with $\lambda = 0.8$ and backlogging probability equal to 0.01.

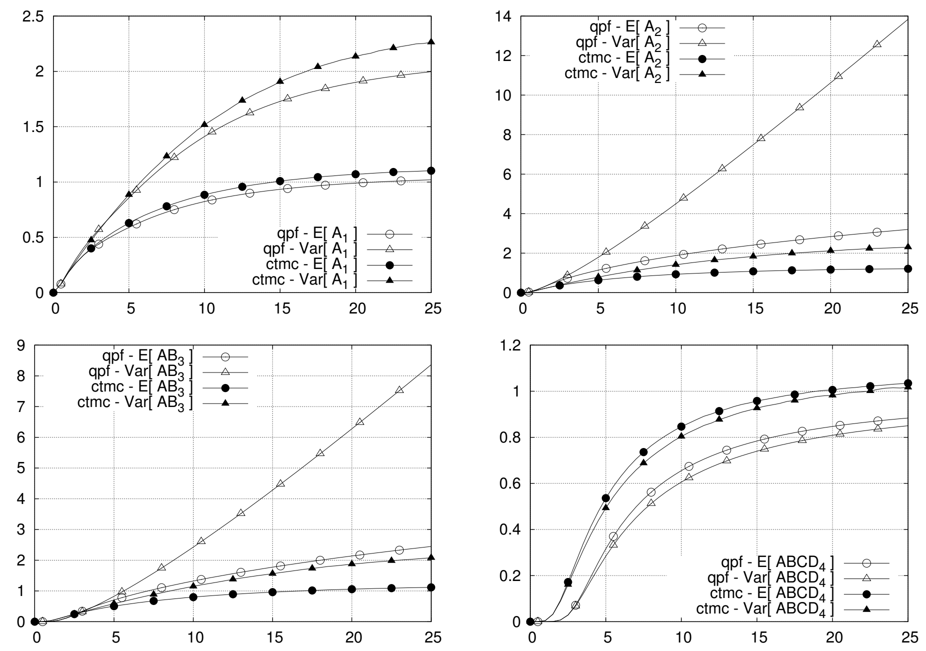


Figure 7.34: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) starting from all empty places with $\lambda = 0.8$ and backlogging probability equal to 0.25.

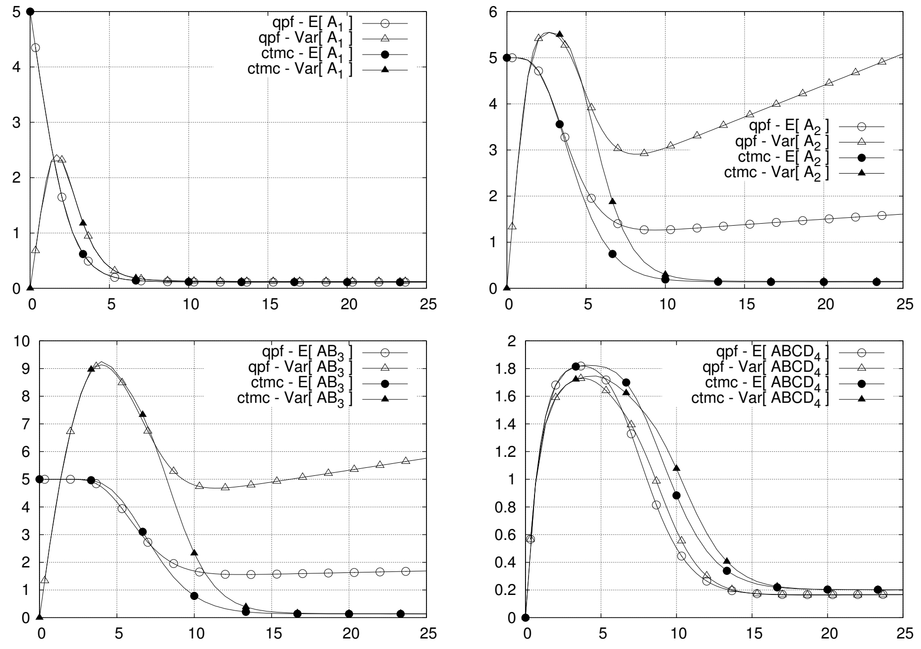


Figure 7.35: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) starting from a state all the places but L_0 and $ABCD_4$ have five tokens with $\lambda = 0.2$.

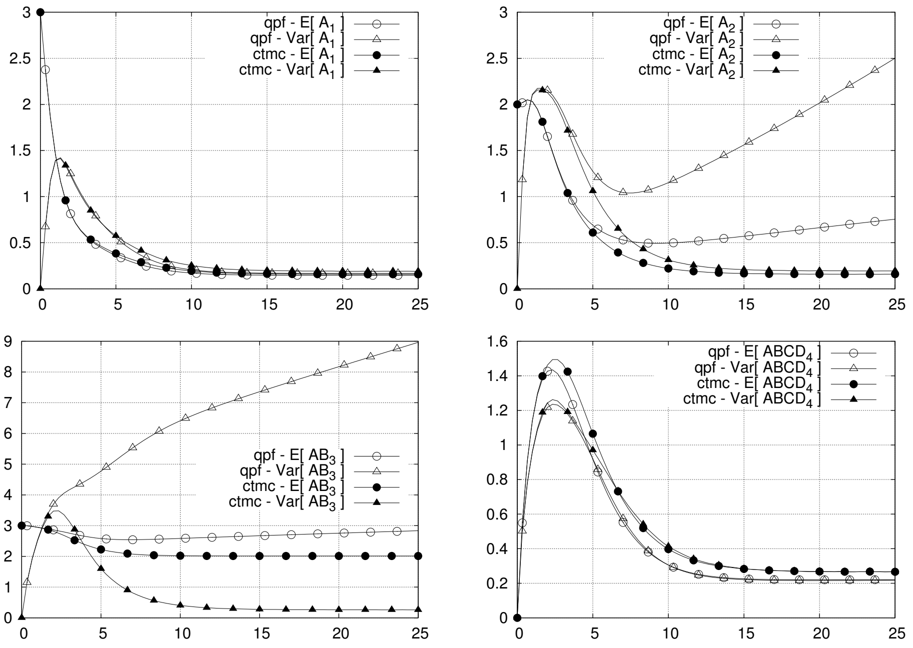


Figure 7.36: On demand multilevel manufacturing system : Expectations and variances of the number of tokens in places A_1 (top left), A_2 (top right), AB_3 (bottom left) and $ABCD_4$ (bottom right) starting from $x = |0, 3, 5, 2, 4, 2, 3, 2, 3, 3, 2, 0|$ with $\lambda = 0.2$.

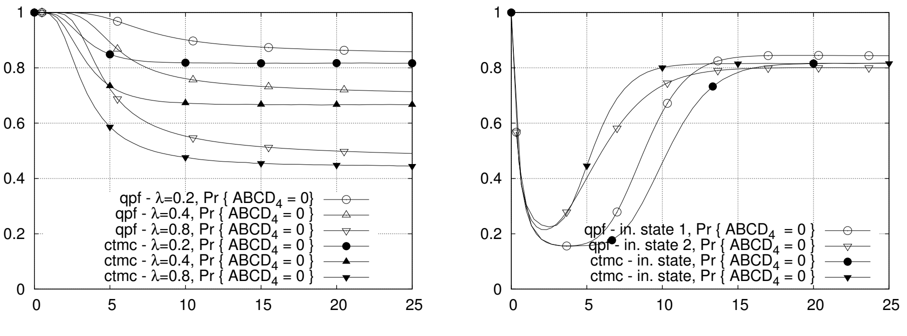


Figure 7.37: On demand multilevel manufacturing system : Probability to find the place $ABCD_4$ empty as function of the time .

8 Conclusions and future works

8.1 Conclusive remarks

8.1.1 First part

In the first part of this thesis we provided a proof of concept of how an accurate qualitative analysis of a Markov chain can preserve the feasibility of standard analysis techniques and, as a consequence, the exactitude of the results. Indeed, the cases that we proposed correspond to particular situations. However, models describing monotonic productions (and, more generally, monotonic measures) are not rare and their analysis through Markov reward models does not seem to be well-known in several communities. We proposed those of systems biology and flexible manufacturing. Additionally, we extended the state of the art by proposing a recursive way to compute joint moments as well.

The results obtained by using the framework in the context of manufacturing production lines led to extremely encouraging results due to the fact that the original problem is directly connected to the measures of accumulated reward and completion time. This allowed the computation of properties that otherwise can be analyzed only through the direct observation of the production line or by Monte Carlo approaches. It is also evident that, until now, we limited the analysis to very small machine blocks but the approach permits the investigation of much bigger systems. From the point of view of biochemical systems, our approach can tackle the problem and avoid the use of approximation in a larger number of situations.

8.1.2 Second part

In the second part of the thesis, we focused on the concept of transient product form. This is in itself an achievement since, according to the authors that we have cited, transient product form does not seem well-known.

On the base of the concept of transient product form, we developed an approximation, called quasi product form, that can be used in a large number of situations and is able to describe accurately the transient probabilities of, possibly time dependent, CTMCs. In particular, we showed that the use of quasi product form is particularly suitable in case of productions modulated by switches. In fact, often such situations lead to distributions that are not compact and can generate multi-modal shapes. In these scenarios approximations based on moments are untrustworthy because they provide the most unlikely among the possible trajectories. Additionally, we have shown that our approximation is an accurate approximation of the transient behaviour of queuing networks composed of stations having a finite number of servers.

8.2 Future works

8.2.1 First part

Markov reward models are a well-known framework that has been massively investigated during the years in the context of performance evaluation. For this reason, many variants of the proposed approach exist and are a sound solution for the analysis of large models. However, there are still open problems. The most important is probably the closure of the moments in case of non-negative accumulated rewards and possible total loss. Another interesting investigation is the developing of an efficient method for the computation of rewards when their amounts affect the underlying process.

From the point of view of the analysis of flexible manufacturing systems, the number of possible extensions is large. Several of them are straightforward, for instance: cyclic production lines, multi-class productions, covariances between the state of the machines and the accumulated production.

On the other side, an interesting extension of MRMs in the context of biochemical systems would be the computation of the measures known as *trends*. These measures address specific behavioural questions, such as the likelihood for a biochemical species to reach a peak/deadlock state, or to exhibit monotonic/oscillatory behaviours [12, 1].

8.2.2 Second part

The investigations made in the second part of the thesis suggest many interesting future works.

First of all, it is evident that the conditions under which a model enjoys the transient product form are extremely strict. This is justified by the fact that the overall transient behaviour of every component of the model is totally described by a single differential equation. Nowadays, common hardware is able to carry on the computation of much larger ODEs systems. Thus, a first possible future investigation is to find a similar approach that:

1. maintains the exactitude of the results,
2. might be more expensive than transient product form but remains computationally friendly,
3. has a larger sphere of application.

Secondly, we want to improve the quasi product form approximation by investigating:

1. more sophisticated patterns to generate the DAG describing the quasi product form decomposition,
2. the developing of more accurate methods to control the error generated by the approximation
3. its integration with advanced data structures such as BDDs.

As last, the accuracy of some results pointed out that there are cases in which Markov chains carry more information than necessary. This fact would suggest the existence of situations where the validation of our method can be done directly from the real phenomenon instead of the CTMC describing it. In other words, we want investigate the use of quasi product form as an independent stochastic process. This idea is enforced by the existence of several common points between the product form approximation and the framework of *Markovian agents* described by Bobbio, Gribaudo and Cerotti in [42, 43]. This framework considers each modelled object as an independent Markov chain that communicates with the rest of the net according to mean-field approximations, as it happens in case of a product form.

Bibliography

[1] O. Andrei and M. Calder. Trend-Based Analysis of a Population Model of the AKAP Scaffold Protein. *T. Comp. Sys. Biology*, 14:1–25, 2012.

[2] A. Andreychenko, P. Crouzen, and V. Wolf. On-the-fly Uniformization of Time-Inhomogeneous Infinite Markov Population Models. In Mieke Massink and Gethin Norman, editors, *QAPL*, volume 57 of *EPTCS*, page 1, 2011.

[3] A. Angius and A. Horváth. Analysis of stochastic reaction networks with Markov reward models. In François Fages, editor, *CMSB*, pages 45–54. ACM, 2011.

[4] A. Angius and A. Horváth. Product Form Approximation of Transient Probabilities in Stochastic Reaction Networks. *Electronic Notes on Theoretical Computer Science*, 277:3–14, 2011.

[5] A. Angius, A. Horváth, and M. Colledani. Moments of Cumulated Output and Completion Time of Unreliable General Markovian Machines. In *IFAC World Congress '11, to appear*, 2011.

[6] A. Angius, A. Horváth, and V. Wolf. Quasi product form approximation for Markov models of reaction networks. *Transactions on Computational Systems Biology*, 7625(XIV), 2012.

[7] M. Arns, P. Buchholz, and A. Panchenko. On the Numerical Analysis of Inhomogeneous Continuous-Time Markov Chains. *Informs Journal on Computing*, 22(3):416–432, 2010.

[8] J. Babar, M. Beccuti, S. Donatelli, and A. S. Miner. GreatSPN Enhanced with Decision Diagram Data Structures. In *Petri Nets*, pages 308–317, 2010.

[9] C. Baier and J. Katoen. *Principles of Model Checking.* MIT Press, 2008.

[10] G. Balbo, M. Ajmone Marsan, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with generalized stochastic Petri nets.* Wiley Series in Parallel Computing . John Wiley and Sons, 1995.

[11] G. Balbo, S. C. Bruell, and M. Sereno. Product Form Solution for Generalized Stochastic Petri Nets. *IEEE Trans. Software Eng.*, 28(10):915–932, 2002.

[12] P. Ballarini and M. L. Guerriero. Query-based verification of qualitative trends and oscillations in biochemical systems. *Theor. Comput. Sci.*, 411(20):2019–2036, April 2010.

[13] F. Baskett, K.M. Chandy, R.R. Muntz, and G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.

[14] F. Bause and P. S. Kritzinger. *Stochastic Petri nets - an introduction to the theory (2. ed.).* Vieweg, 2002.

[15] P. Bazan and R. German. Approximate transient analysis of large stochastic models with WinPEPSY-QNS. *Computer Networks*, 53:1289–1301, 2009.

[16] Benaim, M. and Weibull, J. Mean-field approximation of stochastic population processes in games. Working Papers hal-00435515, HAL, 2009.

[17] R. J. Boucherie. *Product-form in queueing networks.* PhD thesis, Vrije Universiteit, Amsterdam, 1992. Th. : stochastic operations research.

[18] R. J. Boucherie and P.G. Taylor. Transient product form distributions in queueing networks. *Discrete Event Dynamic Systems: Theory and Applications*, 3:375–396, 1993.

[19] P. Buchholz. Markovian Process Algebra: Composition and Equivalence. In *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, pages 11–30, 1994.

[20] P. Buchholz, J. Katoen, P. Kemper, and C. Tepper. Model-checking large structured Markov chains. *J. Log. Algebr. Program.*, (1-2):69–97, 2003.

[21] G. Casale. Approximating passage time distributions in queueing models by Bayesian expansion. *Perform. Eval.*, 67(11):1076–1091, 2010.

[22] F. Castella, G. Dujardin, and B. Sericola. Moments'Analysis in Homogeneous Markov Reward Models. *Methodology And Computing In Applied Probability*, 11(4):583–601, 2009.

[23] K. M. Chandy, U. Herzog, and L. Woo. Parametric Analysis of Queueing networks. *IBM Journal of Research and Development*, 19(1):36–42, 1975.

[24] H. Chen and A. Mandelbaum. Discrete Flow Networks: Bottleneck Analysis and Fluid Approximations. *Mathematics of Operations Research*, 16(2):408–446, 1991.

[25] G. Ciardo, G. Lüttgen, and A. S. Miner. Exploiting interleaving semantics in symbolic state-space generation. *Formal Methods in System Design*, 31(1):63–100, 2007.

[26] G. Clark and J. Hillston. Product Form Solution for an Insensitive Stochastic Process Algebra Structure, 2002.

[27] D. R. Cox and P. A. W. Lewis. *The statistical analysis of series of events*. 1966.

[28] T. Dao-Thi, M. Tran, and J. Fourneau. Multiple Class Symmetric G-networks with Phase Type Service Times. *Comput. J.*, 54(2):274–284, 2011.

[29] T. Dayar, L. Mikeev, and V. Wolf. On the Numerical Analysis of Stochastic Lotka-Volterra Models. In *Proc. of the Workshop on Computer Aspects of Numerical Algorithms (CANA10)*, pages 289–296, 2010.

[30] J. D. Diener. *Empirical comparison of uniformization methods for continuous-time Markov chains.* PhD thesis, University of Arizona, 1994.

[31] S. Donatelli. Superposed Stochastic Automata: A Class of Stochastic Petri Nets with Parallel Solution and Distributed State Space. *Perform. Eval.*, 18(1):21–36, 1993.

[32] S. Donatelli. Superposed Generalized Stochastic Petri Nets: Definition and Efficient Solution. In Robert Valette, editor, *Application and Theory of Petri Nets*, volume 815. Springer, 1994.

[33] S. Donatelli, S. Haddad, and J. Sproston. Model Checking Timed and Stochastic Properties with CSL^{TA}. *IEEE Trans. Software Eng.*, 35(2):224–240, 2009.

[34] S. Engblom. Computing the moments of high dimensional solutions of the master equation. *Appl. Math. Comput*, 180:498–515, 2006.

[35] E. Gelembe. Réseaux neuronaux aléatoires stables. *Comptes Rendus de l'Académie des Sciences 309, Série II*, 310:177–180, 1990.

[36] E. Gelenbe. Product-form queueing networks with negative and positive customers. *Journal of Applied Probability*, 28:656–663, 1991.

[37] E. Gelenbe. G-networks with signals and batch removal. *Probability in the Engineering and Informational Sciences*, 7:335–342, 1993.

[38] E. Gelenbe. G-Networks with Triggered Customer Movement. *Journal of Applied Probability*, 30(3):742–748, 1993.

[39] E. Gelenbe and J. Fourneau. G-networks with resets. *Perform. Eval.*, 49(1/4):179–191, 2002.

[40] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[41] W.J. Gordon and G.F. Newell. Cyclic queueing networks with exponential servers. *Operations Research*, 15(2):254–265, 1967.

[42] M. Gribaudo, A. Bobbio, and D. Cerotti. Disaster Propagation in Heterogeneous Media via Markovian Agents. In *CRITIS*, pages 328–335, 2008.

[43] M. Gribaudo, D. Cerotti, and A. Bobbio. Analysis of On-off policies in Sensor Networks Using Interacting Markovian Agents. In *PerCom*, pages 300–305, 2008.

[44] S. Haddad, P. Moreaux, M. Sereno, and M. Silva. Product-form and stochastic Petri nets: a structural approach. *Performance Evaluation*, 59(4):313–336, 2005.

[45] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Number 1409 in Lecture Notes in Mathematics. Springer-Verlag, 1989.

[46] J. M. Harrison and A. J. Lemoine. A note on networks of infinite-server queues. *J. Appl. Probab.*, 18(2):561–567, 1981.

[47] P. G. Harrison. Transient Behaviour of Queueing Networks. *Journal of Applied Probability*, 18(2):482–490, 1981.

[48] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theor. Comput. Sci.*, 411(22-24):2260–2297, 2010.

[49] A. Heindl and A. van de Liefvoort. Moment conversions for discrete distributions. In *Proc. 6th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*, 2003.

[50] T. A. Henzinger, M. Mateescu, and V. Wolf. Sliding Window Abstraction for Infinite Markov Chains. In *CAV*, pages 337–352, 2009.

[51] T. A. Henzinger, L. Mikeev, M. Mateescu, and V. Wolf. Hybrid numerical solution of the chemical master equation. In *CMSB*, pages 55–65, 2010.

[52] H. Hermanns, J. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *STTT*, (2):153–172, 2003.

[53] J. Hespanha. Moment closure for biochemical networks. pages 142–147, 2008.

[54] J. Hillston. Fluid Flow Approximation of PEPA models. In *Quantitative Evaluation of Systems*, pages 33–43.

[55] G. Horváth, S. Rácz, Á. Tari, and M. Telek. Evaluation of Reward Analysis Methods with MRMSolve 2.0. In *QEST*, pages 165–174, 2004.

[56] J.R. Jackson. Jobshop-Like Queueing Systems. *Management Science*, 10(1):131–142, 1963.

[57] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.

[58] P. Kemper. Numerical Analysis of Superposed GSPNs. *IEEE Transactions on Software Engineering*, 22:52–61, 1995.

[59] D. G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, 24(3):338–354, 1953.

[60] V. G. Kulkarni. *Modeling and analysis of stochastic systems.* Chapman & Hall, Ltd., London, UK, UK, 1995.

[61] T. G. Kurtz. Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes. *Journal of Applied Probability*, 1(7):49–58, 1970.

[62] T. G. Kurtz. The Relationship between Stochastic and Deterministic Models for Chemical Reactions. *J Chem Phys*, 57(7):2976–2978, 1972.

[63] A. Loinger, A. Lipshtat, N. Q. Balaban, and O. Biham. Stochastic simulations of genetic switch systems. *Phys. Rev. E*, 75:021904, Feb 2007.

[64] A. J. Lotka. *Elements of Mathematical Biology.* Williams and Wilkins Company, 1924.

[65] S. I. Martins, A. T. Martinus, and M. A. Van Boekel. Kinetic modelling of Amadori N-(1-deoxy–fructos-1-yl)-glycine degradation pathways. Part II–Kinetic analysis. *Carbohydrate Research*, 338(16):1665–1678, 2003.

[66] W. A. Massey and W. Whitt. A probabilistic generalization of Taylor's theorem. *Statistics & Probability Letters*, 16(1):51–54, January 1993.

[67] W. A. Massey and W. Whitt. Networks of infinite-server queues with nonstationary Poisson input. *Queueing Systems*, 13:183–250, 1993.

[68] T. I. Matis and R. M. Feldman. Transient Analysis of State-Dependent Queueing Networks via Cumulant Functions. *Journal of Applied Probability*, 38(4):841–859, 2001.

[69] C. Moler and C. Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. *SIAM Review*, 45(1):3–49, 2003.

[70] M.K. Molloy. *On the integration of delay and throughput measures in distributed processing models.* PhD thesis.

[71] S. Natkin. *Les Reseaux de Petri Stochastiques et leur Application a l'Evaluation des Systemes Informatiques.* PhD thesis, 1980.

[72] V. F. Nicola, V. G. Kulkarni, and K. S. Trivedi. Queueing Analysis of Fault-Tolerant Computer Systems. In *SIGMETRICS*, 1986.

[73] V. F. Nicola, V. G. Kulkarni, and K. S. Trivedi. Queueing Analysis of Fault-Tolerant Computer Systems. *IEEE Trans. Software Eng.*, 13(3):363–375, 1987.

[74] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut fur instrumentelle Mathematik, Bonn, 1962.

[75] B. Plateau. *De l'evalutation du paralleslisme et de la syncronization.* 1984.

[76] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, SIGMETRICS '85, pages 147–154, New York, NY, USA, 1985. ACM.

[77] W. H. Sanders and P. Buchholz. Approximate Computation of Transient Results for Large Markov Chains. In *Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, QEST '04, pages 126–135, Washington, DC, USA, 2004. IEEE Computer Society.

[78] I. H. Segel. *Enzyme kinetics: behavior and analysis of rapid equilibrium and steady state enzyme systems.* New York: Wiley, 1993.

[79] M. Sereno. Towards a Product Form Solution for Stochastic Process Algebras. *The Computer Journal*, 38(7):622–632, 1995.

[80] A. Singh and J.P. Hespanha. Moment closure techniques for stochastic models in population biology. *American Control Conference*, pages 4730–4735, 2006.

[81] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1995.

[82] Á. Tari, M. Telek, and P. Buchholz. A Unified Approach to the Moments Based Distribution Estimation - Unbounded Support. In *EPEW/WS-FM*, pages 79–93, 2005.

[83] M. Telek, A. Bobbio, and M. Gribaudo. Analysis of Large Scale Interacting Systems by Mean Field Method. In *QEST*, pages 215–224, 2008.

[84] M. Telek and S. Rácz. Numerical analysis of Large Markovian reward models. *Performance Evaluation*, 36&37:95–114, Aug 1999.

[85] M. Tribastone. *Scalable Analysis of Stochastic Process Algebra Models.* PhD thesis, School of Informatics, The University of Edinburgh, 2010.

[86] M. Tribastone and S. Gilmore. *Rigorous Software Engineering for Service-Oriented Systems—Results of the SENSORIA project on Software Engineering for Service-Oriented Computing*, chapter Scaling Performance Analysis using Fluid-Flow Approximation. Springer-Verlag, 2010.

[87] N. M. Van Dijk. Uniformization for nonhomogeneous Markov chains. *Oper. Res. Lett.*, 12(5):283–291, November 1992.

[88] A. P. A. van Moorsel and Katinka W. Numerical Solution of Non-Homogeneous Markov Processes through Uniformization. In Richard N. Zobel and Dietmar P. F. Möller, editors, *ESM*, pages 710–717. SCS Europe, 1998.

[89] V. Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 1924.

[90] M. Wan, G. Ciardo, and A. S. Miner. Approximate steady-state analysis of large Markov models based on the structure of their decision diagram encoding. *Perform. Eval.*, 68(5):463–486, May 2011.

[91] W. Whitt. Decomposition Approximations for Time-Dependent Markovian Queueing Networks. *Operations Research Letters*, 24:97–103, 1999.

[92] J. X. Zhou, L. Brusch, and S. Huang. Predicting Pancreas Cell Fate Decisions and Reprogramming with a Hierarchical Multi-Attractor Model. *PLoS ONE*, 6(3):16, 2011.