Chapter 1

Stochastic Modeling Techniques in Software Aging and Rejuvenation Phenomena

Andrea Bobbio[1], Antonio Puliafito[2], Marco Scarpa[2] and Miklós Telek[3,4]

[1] *DiSit, Università del Piemonte Orientale, Italy*
[2] *Dipartimento di Ingegneria, Università di Messina, Italy*
[3] *Budapest University of Technology and Economics, Hungary*
[4] *MTA-BME Information Systems Research Group, Hungary*

{andrea.bobbio@uniupo.it; antonio.puliafito, mscarpa@unime.it; telek@hit.bme.hu}

Quantitative modeling of software aging and rejuvenation has been the object of a vast literature in the last decades. Due to the probabilistic nature of the phenomena to which these studies are devoted, the models rely on the theory of stochastic processes. The present Chapter is intended to provide an overview of the methods and analytical solution techniques related to the stochastic processes that are more often encountered in the literature on software aging and rejuvenation. The Chapter is particularly addressed to researchers who want to have an initial approach to these topics.

1. Introduction

The phenomenon of software aging has come to light in the last decades as a cause of the degradation of the software performance in time. Although we adopt the phrase software aging, it should be clear that no deterioration of the software system per se is implied but rather, the software appears to age due to the degradation of the operating environment (for example, gradual depletion of resources).[1] Software rejuvenation has been proposed for the first time in the seminal paper by Huang et al.[2] as a proactive maintenance technique to counteract aging. Software rejuvenation involves occasionally stopping the running software, cleaning its internal state or its environment, and restarting it. Likewise aging, software rejuvenation actually does not imply any modification of the software but refers to rejuvenation of the environment in which the software is executing.[3]

The aging of the software and the action of rejuvenation can be captured and analyzed through stochastic models that aim to characterize their behavior according to quantitative parameters such as the mean time to a failure condition, the optimal rejuvenation interval, the probability to be in a fully working condition and so on.

In their original paper, Huang et al.[2] consider the software aging as a two-step process. From the clean state the system jumps into a "degraded" state from which two further actions are possible: rejuvenation (with return back to the clean state) or transition to the complete failure state. The authors model a three-state process as a *continuous-time Markov chain (CTMC)* for which they derive and discuss only the steady-state behavior.

The problem connected to software rejuvenation has then been investigated by many research groups introducing more refined modeling assumptions by abandoning the exponential assumption in the time evolution of the phenomenon and considering the interaction of the software aging with the workload on the system. Further, rejuvenation can be activated at constant intervals, at random intervals or at time epochs related to the software degradation.

Garg et al.[4,5] consider the effect of the workload by including the arrival and queuing of jobs in the system, and the time dependency of the load and of the service time on the age of the system. With this addition, the arrival and service rates are time-varying and the overall model becomes a three-state non-homogeneous Markov process. Zheng et al.[6] investigate workload-based policies where the arrival stream of jobs follows a Markovian arrival process (MAP).

In the analytical approach, different authors have assumed general transition time distributions, and have developed Markov, semi-Markov (SMP), Markov regenerative processes (MRGP), to compute and optimize system availability or related measures. For instance, semi-Markov models were used in[7,8] and Markov regenerative models were considered in.[9,10]

Stochastic Petri net models and their variants have been often invoked to generate the system model like Markov regenerative stochastic Petri nets (MRSPNs) in[9] or fluid stochastic Petri nets (FSPNs) in.[11,12]

The present Chapter is intended to provide the readers, who are not particularly aware of the theory of stochastic modeling, with the basic notions, definitions and formulas that are encountered in the Chapters of the book. In each Section, some relevant references are cited and an example taken from the software rejuvenation literature is described and fully developed and solved.

Section 2 introduces the renewal theory with particular reference to the Poisson process that is usually used to model the stream of jobs arriving to a software system. A Poisson shock model, utilized in,[13,14] to characterize a software aging process is reported. In Section 3, the homogeneous Markov chain model is introduced and the basic solution equations for both transient and steady-state case are discussed. As an example, the rejuvenation model discussed in[2] is presented and analyzed.

Section 4 reports the transient and steady-state solutions for the non-homogeneous Markov process and illustrates the corresponding rejuvenation example taken from.[5]

Sections 5 and 6 deal with SMP and MRGP respectively, and provide the steady-state and transient solution equations in both cases. A SMP example taken from[15]

and an MRGP in[9] are also introduced.

Finally, Section 7 is devoted to the introduction of Petri net (PN) formalism and focuses its attention on stochastic Petri net (SPN) such as generalized Stochastic PN (GSPN), Stochstic Reward Nets (SRN) and non-Markovian SPN etc. In particular, an MRSPN model derived from[9] is provided to show how an MRGP can be generated from MRSPN.

2. Renewal Process

The stochastic process generated by a sequence of independent identically distributed random variables $(Y_1, Y_2, \ldots, Y_n, \ldots)$ (Fig. 1) with cumulative distribution function (CDF) $F(t)$, density $f(t)$, $t \geq 0$, and expected value $\mathbf{E}[Y]$ is called a *renewal process.*[16]

$$F(t) = \mathbf{P}\{Y \leq t\},\ f(t) = \frac{d\,F(t)}{d\,t},\ \mathbf{E}[Y] = m\,.$$

The corresponding Laplace transforms (LTs) are denoted by the symbol $*$:

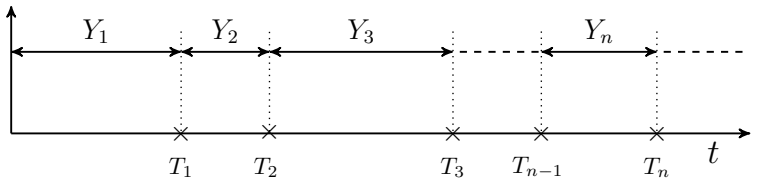

Fig. 1. A renewal process

$$F^*(s) = \mathcal{L}\,[F(t)] = \int_t^{+\infty} e^{-st} F(t)dt,\ f^*(s) = \mathcal{L}\,[f(t)] = \int_t^{+\infty} e^{-st} f(t)dt\,.$$

Let $T_k = \sum_{i=1}^{k} Y_i$, with $k \geq 1$ denote the time of the k-th event in the renewal process, and $F_k(t)$ and $f_k(t)$ denote its CDF and density, respectively. We can write:

$$F_k(t) = \mathbf{P}\{T_k \leq t\},\ f_k(t) = \frac{d\,F_k(t)}{d\,t}\,. \tag{1}$$

Applying the convolution theorem for LTs, it could be represented as Eq. (1) in the Laplace domain:

$$F_k^*(s) = \frac{1}{s}\,[f^*(s)]^k,\ f_k^*(s) = [f^*(s)]^k\,. \tag{2}$$

If $N(t)$ is the number of renewals in the interval $(0, t]$, then the following relation holds:

$$N(t) < k \quad \text{if and only if} \quad T_k > t$$

from which:

$$\mathbf{P}\{N(t) < k\} = \mathbf{P}\{T_k > t\} = 1 - F_k(t)\,. \tag{3}$$

From Equation (3), we obtain:

$$\begin{aligned}\mathbf{P}\{N(t) = k\} &= \mathbf{P}\{N(t) < k+1\} - \mathbf{P}\{N(t) < k\} \\ &= F_k(t) - F_{k+1}(t)\,. \end{aligned} \tag{4}$$

2.1. *Poisson Process*

A renewal process in which the generating random variable Y is exponentially distributed is called a Poisson Process. Assume that Y, is exponentially distributed with rate λ, then:

$$\begin{aligned} F(t) = 1 - e^{-\lambda t} \quad &\Longrightarrow \quad F^*(s) = \frac{\lambda}{s\,(s+\lambda)} \\ f(t) = \lambda\, e^{-\lambda t} \quad &\Longrightarrow \quad f^*(s) = \frac{\lambda}{s+\lambda}\,. \end{aligned}$$

The CDF and density of the time up to the k-th arrival T_k can be obtained from Equation (2), in LT domain:

$$F_k^*(s) = \frac{\lambda^k}{s\,(s+\lambda)^k},\; f_k^*(s) = \left(\frac{\lambda}{s+\lambda}\right)^k. \tag{5}$$

Transforming back to the time domain the LT density (5) for $k = 1, 2, \ldots$ we obtain

$$\begin{aligned} f_1(t) &= \mathcal{L}^{-1}\left[\frac{\lambda}{s+\lambda}\right] = \lambda\, e^{-\lambda t} \\ f_2(t) &= \mathcal{L}^{-1}\left[\frac{\lambda^2}{(s+\lambda)^2}\right] = \lambda^2\, t\, e^{-\lambda t} \\ &\cdots\cdots\cdots \\ f_k(t) &= \mathcal{L}^{-1}\left[\frac{\lambda^k}{(s+\lambda)^k}\right] = \frac{\lambda\,(\lambda t)^{k-1}}{(k-1)!}\, e^{-\lambda t} \end{aligned}$$

and its CDF of order k:

$$F_k(t) = \int_0^t f_k(u)\,du = 1 - \sum_{i=0}^{k-1} \frac{(\lambda t)^i}{i\,!}\; e^{-\lambda t}$$

which is actually the CDF of the k-stage Erlang random variable with parameter λ.[17,18]

Denote $P_k(t)$ as the probability of having k renewals in a time duration of length t, then from Equation (4):

$$P_k(t) = \mathbf{P}\{N(t) = k\} = F_k(t) - F_{k+1}(t)\,.$$

Taking LTs:

$$P_k^*(s) = \frac{\lambda^k}{s\,(s+\lambda)^k} - \frac{\lambda^{k+1}}{s\,(s+\lambda)^{k+1}} = \frac{\lambda^k}{(s+\lambda)^{k+1}}\,.$$

Inverting again to the time domain, we obtain the Poisson probability mass function (pmf):

$$P_k(t) = \mathbf{P}\{N(t) = k\} = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \tag{6}$$

or, in a recursive form:

$$P_k(t) = \mathbf{P}\{N(t) = k\} = \frac{\lambda t}{k} P_{k-1}(t). \tag{7}$$

In a Poisson process, the number of renewals at time t follows a Poisson distribution of parameter λt.

2.2. *Poisson Shock Process*[19]

The aging process of a software system consists of a sequence of additive random shocks and is monitored by observing a degradation level of the system performance, whose value at time t is denoted by $S(t)$. The shock events occur according to a Poisson process with rate λ. Each shock increases the degradation level by a positive damage denoted as variable X ($X \geq 0$) with CDF $F_X(x)$. All demages X_1, X_2, ... from successive shocks are mutually independent with the common CDF $F_X(x)$. Damages accumulate additively, and the system survives if the amount of damage accumulated does not exceed a threshold a, representing the maximum tolerated value of the degradation level. We call T_a the random time at which the degradation level reaches the threshold a. The CDF $H_{T_a}(t) = \mathbf{P}\{T_a < t\} = \mathbf{P}\{S(t) > a\}$ is the probability that at time t the degradation level has reached level a and $\overline{H}_{T_a}(t) = 1 - H_{T_a}(t) = \mathbf{P}\{S(t) < a\}$ is the probability that at time t the degradation level is below the threshold a, i.e., the survival probability.

After k shocks, the degradation level will be $S_k = \sum_{i=0}^{k} X_i$, with distribution $F_X^{\dagger k}(x) = \mathbf{P}\{S_k < x\}$ where $F_X^{\dagger k}(x)$ can be obtained form $F_X(x)$ by the k-fold convolution $F_X^{\dagger k}(x) = \int_0^x F_X^{\dagger k-1}(x-y) dF_X(y)$ (or in Laplace domain as in (5)). Since shocks occur according to a Poisson process with rate λ, $N(t)$, the survival probability is given by

$$\overline{H}_{T_a}(t) = \mathbf{P}\{S(t) < a\} = \sum_{i=0}^{\infty} \mathbf{P}\{N(t) = k\} \mathbf{P}\{S_k < a\} = \sum_{i=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} F_X^{\dagger k}(a).$$

The above equation is essentially a compound Poisson distribution. A software aging model based on a Poisson shock process has been utilized in.[13,14]

3. Homogeneous Continuous-Time Markov Chain

A stochastic process is a family of random variables $X(t)$ on a sample space. The value assumed by $X(t)$ are called *states*, and the set of all the possible states is the *state space*. When the value of $X(t)$ changes, the process undergoes a *state transition*. The state space of a stochastic process can be either discrete or continuous.

If the state space is discrete, we call the process a *chain*. The time parameter of a stochastic process can be either discrete or continuous.

A stochastic process can be classified by the dependence of its state at a particular time on the states at previous time. If the state of a stochastic process depends only on the immediately preceding state, we have a *Markov process*. Discrete state Markov processes are known as Markov chains. We distinguish among Discrete-time (DTMC) Markov chain and continuous-time Markov chain, depending on the nature of the time parameter. Let $\mathbf{P}\{X(t_n) = j\}$ be the probability that the process is in the state j at the time t_n. $X(t)$ is a Markov chain if, for any ordered time sequence $t_1 < t_2 < ... < t_n$ and all $i_n \in S$, the conditional probability of being in any state j is such that:

$$\begin{aligned}&\mathbf{P}\{X(t_n) = j|X(t_{n-1}) = i_{n-1}, X(t_{n-2}) = i_{n-2}, \ldots, X(t_1) = i_1\} = \\ &= \mathbf{P}\{X(t_n) = j|X(t_{n-1}) = i_{n-1}\}. \end{aligned} \tag{8}$$

The condition in Equation (8) defines what is called the *Markov property* that says that the state of a Markov chain after a transition may (but does not have to) depend on the state immediately before it, but it cannot depend on any states before that. In other words, at the time of a transition, the entire past history is summarized by the current state. If the conditional probability is invariant with respect to the time origin u, the Markov chain is said to be homogeneous. To be more specific, a Markov chain is homogeneous if for any t and u,

$$\mathbf{P}\{X(t) = j|X(u) = i\} = \mathbf{P}\{X(t-u) = j|X(0) = i\}. \tag{9}$$

For homogeneous Markov chains, if $X(u) = j$, then the probability of finding the system in state j at time $t > u$ depends on state j but not on the time the system has been in state j before u. The process can therefore be said to be *memoryless*, as regards not only the states it has occupied in the past, but also the amount of time it has already spent in the current state. This implies that the time a homogeneous Markov chain spends in a state (the *sojourn time*) cannot be arbitrarily distributed, but it can only follow a memoryless distribution, which is the geometric distribution for discrete time models and the exponential distribution for continuous time models.

3.1. *Basic Equations*

Without loss of generality, the state space of a Markov chain is given by $S = \{0, 1, 2, \ldots\}$. A purpose of the analysis of a Markov chain is to compute the state probabilities $\pi_j(t) = \mathbf{P}\{X(t) = j\}$, for all states j, i.e., the vector of state probabilities is $\boldsymbol{\pi}(t) = [\pi_0(t), \pi_1(t), \ldots]$, both for finite values of t (transient solution) and for $t \to \infty$ (steady-state solution). We define a *transition probability* to be

$$p_{ij}(t-u) = \mathbf{P}\{X(t) = j|X(u) = i\}, \tag{10}$$

the probability that the system is in state j at time t, given that it was in state i at time u; most often, we will have $u = 0$. The matrix $\boldsymbol{P}(t)$ is the square matrix of

the transition probabilities $p_{ij}(t)$. For the system to be in state j at time t, it must have been in some state i at time u, which we can express as:

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(u) \cdot \boldsymbol{P}(t-u) \tag{11}$$

This is a form of the *Chapman-Kolmogorov equation* that will be used for calculating $\boldsymbol{\pi}(t)$ for both the discrete and continuous parameter cases.

3.2. *Discrete-time Markov Chains*

Let the points at which a DTMC changes state be $\{0, 1, \cdots\}$. Let us define $\boldsymbol{P} = \boldsymbol{P}(1) = [p_{ij}]$ the *one-step transition probability* matrix. All the elements of $\boldsymbol{P}$ are probabilities and should lie in the range $[0, 1]$ and the elements of any row must sum to one because they describe the probability of a complete set of events, with respect to the next state of the process.

A matrix with such properties is called a *stochastic* matrix. For a homogeneous DTMC, Eq. (11) simplifies to the following recurrence relation:

$$\boldsymbol{\pi}(n+1) = \boldsymbol{\pi}(n) \cdot \boldsymbol{P}. \tag{12}$$

The state probability vector after n steps $\boldsymbol{\pi}(n)$ can be computed in terms of the initial probability vector $\boldsymbol{\pi}(0)$, as:

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0) \cdot \boldsymbol{P}^n. \tag{13}$$

If $\lim_{n\to\infty} \boldsymbol{\pi}(n)$ exists, taking the limit on both sides of Eq. (12) gives the following system of equations for computing the limiting probability vector:

$$\boldsymbol{\pi} = \boldsymbol{\pi} \cdot \boldsymbol{P} \tag{14}$$

where we impose the normalization condition,

$$\boldsymbol{\pi} \cdot \boldsymbol{e}^T = 1, \tag{15}$$

where $\boldsymbol{e}$ is the row vector with all its elements equal to one.

3.3. *Continuous-Time Markov Chains*

If we let $u = t - \Delta t$ and subtract $\boldsymbol{\pi}(t - \Delta t)$ from both sides of (11), we get

$$\boldsymbol{\pi}(t) - \boldsymbol{\pi}(t-\Delta t) = \boldsymbol{\pi}(t-\Delta t)\left[\boldsymbol{P}(\Delta t) - \boldsymbol{I}\right]. \tag{16}$$

If we then divide by Δt and take the limit as $\Delta t \to 0$, we get the following relation:

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t) \lim_{\Delta t\to 0} \frac{\boldsymbol{P}(\Delta t) - \boldsymbol{I}}{\Delta t}. \tag{17}$$

By introducing the Kronecker delta symbol δ_{ij} ($\delta_{ij} = 1$ if $i = j$, zero otherwise), in the homogeneous case, the matrix $\boldsymbol{Q} = [q_{ij}]$ can be written as

$$q_{ij} = \lim_{\Delta t\to 0} \frac{p_{ij}(\Delta t) - \delta_{ij}}{\Delta t}. \tag{18}$$

We can rewrite Eq. (17) as

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t)\,\boldsymbol{Q}\,. \tag{19}$$

The above equation is known as the *Kolmogorov differential equation*, whose initial condition is the initial behavior of the Markov chain, $\boldsymbol{\pi}(0)$. The solution of Eq. (19) is

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)e^{\boldsymbol{Q}t} = \boldsymbol{\pi}(0)\sum_{i=0}^{\infty}\frac{t^i}{i!}\boldsymbol{Q}^i. \tag{20}$$

The definition of $\boldsymbol{Q}$ implies that, for a small interval Δt, we have

$$1 - p_{ii}(\Delta t) \approx -q_{ii}\Delta t\,,$$
$$p_{ij}(\Delta t) \approx q_{ij}\Delta t,\ \ \forall i \neq j\,.$$

Thus, q_{ij} is the rate at which the system goes from state i to j and $-q_{ii}$ is the rate at which the system departs from state i, since we must have

$$\sum_{j\neq i} p_{ij}(\Delta t) + p_{ii}(\Delta t) = 1\,. \tag{21}$$

Because of this interpretation, $\boldsymbol{Q}$ is called the infinitesimal generator matrix of the CTMC. We can apply Eq. (21) and take the limit as $\Delta t \to 0$ to get:

$$\sum_{j} q_{ij} = 0. \tag{22}$$

This means that all elements in any row of $\boldsymbol{Q}$ must sum to 0, which can be written, in matrix form, as $\boldsymbol{Q}\boldsymbol{e}^T = \boldsymbol{0}^T$. Since the off- diagonal elements must be non negative ($q_{ij}\Delta t$ is a probability), we deduce that the elements along the diagonal of $\boldsymbol{Q}$ must be non-positive. In fact, the diagonal entry equals the negative sum of the off-diagonal entries in any row:

$$q_{ii} = -\sum_{j\neq i} q_{ij}, \tag{23}$$

and $q_i = -q_{ii}$ is the net rate out of state i. The average amount of time spent by the CTMC in state i during the interval $(0, t]$ is $L_i(t) = \int_0^t \pi_i(x)dx$. Integrating both sides of Eq. (19), we obtain a differential equation for the vector $\boldsymbol{L}(t) = [L_0(t), L_1(t), \dots]$:

$$\frac{d\boldsymbol{L}(t)}{dt} = \boldsymbol{L}(t)\boldsymbol{Q} + \boldsymbol{\pi}(0),\ L(0) = 0\,. \tag{24}$$

If the state transition graph of a CTMC is such that every state is reachable from every other state, the CTMC is said to be *irreducible*. For an irreducible CTMC the state probabilities reach an asymptotic value, independent of the initial condition, as the time goes to infinity.[20,21] We call the asymptotic solution the *steady-state* solution. If the steady-state solution exists, then for any state i:

$$\lim_{t\to\infty}\ \pi_i(t) \ =\ \pi_i, \qquad \lim_{t\to\infty}\ \frac{d\,\pi_i(t)}{d\,t} \ =\ 0, \tag{25}$$

and, in vector form, we immediately get

$$\lim_{t\to\infty} \boldsymbol{\pi}(t) = \boldsymbol{\pi}, \qquad \lim_{t\to\infty} \frac{d\boldsymbol{\pi}(t)}{dt} = \mathbf{0}\,, \tag{26}$$

where $\boldsymbol{\pi}$ is the steady-state value of the state probability vector and $\mathbf{0}$ is the vector of appropriate dimension with all entries equal to 0. Taking into account Eq. (19) and (26), we get accordingly

$$\boldsymbol{\pi}\cdot\boldsymbol{Q} = \mathbf{0} \quad \text{with} \quad \boldsymbol{\pi}\boldsymbol{e}^{\mathrm{T}} = 1\,. \tag{27}$$

Equation (27) is a linear homogeneous set of n equations with constant coefficients. It is easy to verify that $\boldsymbol{\pi} = 0$ is a solution for $\boldsymbol{\pi}\cdot\boldsymbol{Q} = \mathbf{0}$. If a non-zero solution exists, multiplication of the solution by any constant will still be a solution to the equation. But since $\boldsymbol{\pi}$ is a probability vector, it should also satisfy the normalization condition represented in vector form in Eq. (27). The normalized steady-state solution is then unique when the Markov chain is irreducible.

3.4. *Example*

The paper by Huang et al.,[2] that initiated the study of software aging and rejuvenation, proposed the CTMC model shown in Fig. 2, where we have maintained the same symbols of the original figure. The system first starts in a highly robust

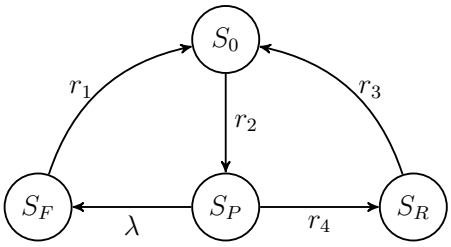

Fig. 2. The CTMC rejuvenation model in Huang et al.[2]

state S_0 and then goes into a failure probable state S_p due to software aging with transition rate r_2. From state S_P two actions are possible: a transition to the failure state S_F with rate λ and a transition to the rejuvenation state S_R with rate r_4. r_1 is the repair rate that takes back the system from a failure state to the highly robust state, and r_3 is the rejuvenation rate of the transition from rejuvenation state to the highly robust state. In Huang et al.[2] the rejuvenation is assumed to be periodic with mean period equaul to t so that $r_4 = 1/t$. Applying Eq. (27), the steady-state solution is:

$$\pi_P = \frac{1}{1 + \frac{\lambda}{r_1} + \frac{r_4}{r_3} + \frac{\lambda + r_4}{r_2}}\;; \quad \pi_0 = \frac{\lambda + r_4}{r_2}\,\pi_P\;; \quad \pi_F = \frac{\lambda}{r_1}\,\pi_P\;; \quad \pi_R = \frac{r_4}{r_3}\,\pi_P$$

3.5. *Markov Reward Model (MRM)*

In a Markov reward model (MRM), we attach to each state of a CTMC with state space S a non-negative real variable, called *reward rate*, that indicates the level of performance or the cost incurred by the system in that state. In particular, the reward rates are defined based on the system requirements, be it availability-, reliability-, or task-oriented. If r_i is the reward rate attached to state i, then, a reward $r_i\,\Delta t$ is accumulated when the CTMC spends a time Δt in state i. If $R(t)$ is the instantaneous reward rate of the MRM at time t and $Y_R(t)$ the total accumulated reward up to time t, we have the expected instantaneous reward rate at time t, $E[R(t)]$, and the expected accumulated reward up to time t, $E[Y_R(t)]$, as below:

$$
\begin{aligned}
E[R(t)] &= \sum_{i\in S} r_i \pi_i(t)\,, \\
Y_R(t) &= \int_0^t R(u)\,du\,, \\
E[Y_R(t)] &= \sum_{i\in S} r_i \int_0^t \pi_i(u)\,du \;=\; \sum_{i\in S} r_i\, L_i(t).
\end{aligned}
$$

4. Non-Homogeneous Continuous-Time Markov Chain

Non-homogeneous continuous-time Markov chains (NHCTMCs) are discrete-state, continuous-time stochastic processes that satisfy the Markov property, but not the homogeneity property. In these models, the transition rates are dependent on the global time, i.e. the variable taking care of the flow of time since the starting of the whole process. Thus, the sojourn time distributions are not exponential anymore and do not experience the memoryless property. We note here the distinction between the Markov property and the memoryless property: the former is a property of some stochastic processes (e.g., DTMC, CTMC, NHCTMC) while the latter is the property of some distributions (e.g., the exponential and the geometric). To distinguish between homogeneous and non-homogeneous CTMC, we refer in this Section to the first type as HCTMC and to the latter as NHCTMC.

4.1. *Basic Equations*

Similarly to HCTMCs, the transient behavior of a NHCTMC is defined by the system of Kolmogorov ordinary differential equations (ODE):[17,18]

$$
\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t)\boldsymbol{Q}(t) \tag{28}
$$

whose initial condition is the initial behavior of the Markov chain, $\boldsymbol{\pi}(0)$. Equation (28) is very similar to Eq. (19), defined for a HCTMC, with one difference: the generator matrix $\boldsymbol{Q}(t) = [q_{ij}(t)]$ is now time-dependent.

The solution of Eq. (28) can be computed with various numerical ODE solver methods. Apart of general ODE solver approaches like Euler method, Runge-Kutta method,[22] specific methods (e.g., TR-BFD2) have been developed for improving the numerical properties of the computation.[23,24]

For special NHCTMC models various alternative methods have been proposed for the transient analysis such as the convolution integration approach, the uniformization and the piecewise constant approximation (PCA) approach. Yet another approach is based on using a phase-type (PH) expansion of the non-exponential distributions.[18]

A special case of NHCTMC could be considered where the matrix $\boldsymbol{Q}(t)$ can be factored so that $\boldsymbol{Q}(t) = g(t)\,\boldsymbol{W}$, with matrix $\boldsymbol{W}$ having all its entries independent of time. In this case the solution to the NHCTMC can be written down as:

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)\, e^{(\int_0^t g(\tau)d\tau)\boldsymbol{W}} = \boldsymbol{\pi}(0)\, e^{\boldsymbol{W} g^* t}\,, \tag{29}$$

where $g^* = (\int_0^t g(\tau)d\tau)/t$ and $\boldsymbol{W}\, g^*$ is the generator matrix of an HCTMC. Thus by solving an "equivalent" HCTMC, we can obtain the solution of the original NHCTMC in this special case. We refer to this method of solution of an NHCTMC as the *Equivalent-HCTMC method*.

When the above factorization in not possible and the NHCTMC is acyclic, convolution integration approach can be recommended,[17,25] which is based on the following equation:

$$p_{ij}(t) = \delta_{ij} e^{-\int_0^t q_{ii}(\tau)d\tau} + \int_0^t \sum_k p_{ik}(x) q_{kj}(x) e^{-\int_x^t q_{jj}(\tau)d\tau} dx, \tag{30}$$

where δ_{ij} is the Kronecker delta function defined by $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The corresponding equation for unconditional state probability is thus:

$$\pi_i(t) = \pi_i(0) e^{-\int_0^t q_{ii}(\tau)d\tau} + \int_0^t \sum_{k\neq i} \pi_k(x) q_{ki}(x) e^{-\int_x^t q_{ii}(\tau)d\tau} dx. \tag{31}$$

The PCA method is based on approximating a continuous function of time by a stair-case function whose value remains constant in certain intervals. We divide the interval $(0, t]$ over which the original continuous function should be evaluated, into n smaller intervals, such that the function can be considered approximately constant over any of the n small intervals. In this way, computing the value of the original function in the midpoint of each small interval, we can generate the approximating stair-case function. The approximation improves as n increases and the length of each small interval decreases.

The PH expansion method consists in replacing the non-exponential distributions present into the model by a PH distribution, which is defined by the probability distribution of the absorbing time in a HCTMC. Since the PH distribution family is dense, the PH distribution can approximate any probability distribution with high precision. In this way, the original non-exponential model can be approximated by an expanded HCTMC that can be solved using the methods of Section 3.

4.2. *Example*

A system consists of a server type software to which transactions arrive according to a Poisson process of rate λ (Section 2.1).[5] The software system is subject to aging

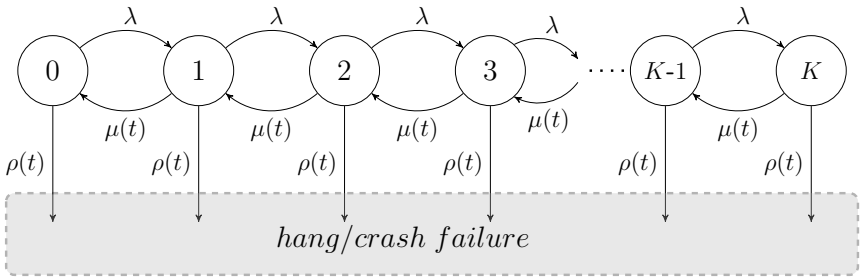

Fig. 3. A NHCTMC model for a software system with aging and failure

and hang/crash failures that are assumed to be stochastically independent processes. Rejuvenation (or Preventive Maintenance) to counteract aging, and recovering to restore from failure are regenerative actions that take back the system to a *as good as new* state. Each transaction receives service for a random period. The effect of aging may be captured by using decreasing service rate and increasing failure rate, where the decrease or the increase, respectively, can be a function of time since the last rejuvenation/recovery action. Assuming that the software queue may allocate up to K transactions, the state transition diagram of the software system is given in Fig. 3, where t is the time from the last renewal. We have maintained the notation in Garg et al.[5]

5. Semi-Markov process (SMP)

The discrete-state continuous-time random process, $X(t)$, is a Semi-Markov process (SMP) if it is time-homogeneous and it experiences the Markov property at the state transition instances (see Fig. 4).

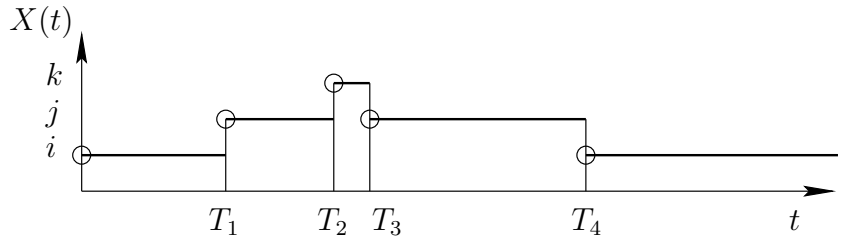

Fig. 4. Semi-Markov process, which experiences the Markov property at the indicated time points

The name of SMP comes from the fact that an SMP does not always experience the Markov property (during the sojourn in a state), but only at the state transition instances. Specifically, during a sojourn in state i, both the remaining time in that state and the next visited state depend on the elapsed time since the process is in state i. The state transition instances where the Markov property holds are marked with a circle in Fig. 4.

The most important consequences of the definition of SMP are:

- the sojourn time in a state can be any positive random variable;
- the distribution of the next state and the time spent in a state are not independent.

Consequently, to define an SMP the following joint distribution has to be given. It is usually done by the definition of the kernel matrix of the process $\boldsymbol{K}(t) = [K_{ij}(t)]$ (we adopt the notation from[18]). Let $X(t) \in S$ be a continuous-time SMP, $T_1, T_2, T_3, ...$ be the state transition instances, then, the (i, j) entry of the kernel matrix is:

$$K_{ij}(t) = \mathbf{P}\{X(T_{i+1}) = j,\, T_{i+1} - T_i \leq t \,|\, X(T_i) = i\}.$$

Utilizing the time homogeneity property of the process we further have for T_i that

$$\mathbf{P}\{X(T_{i+1}) = j, T_{i+1} - T_i \leq t \,|\, X(T_i) = i\} = \mathbf{P}\{X(T_1) = j, T_1 \leq t \,|\, X(0) = i\}.$$

The analysis of an SMP is based on the results of renewal theory and the analysis of embedded Markov chain (EMC) built on the sequence of the transition instances. The definition of an SMP requires the knowledge of the kernel matrix $\boldsymbol{K}(t) = \{K_{ij}(t)\}$ (for $t \geq 0$) and an initial distribution. It is commonly assumed that $X(t)$ experiences the Markov property at time $t = 0$.

5.1. *Analysis based on State Transitions*

We intend to compute the transient state transition probability $V_{ij}(t) = \mathbf{P}\{X(t) = j \mid X(0) = i\}$ assuming that the sojourn in the first state finishes at time h $(T_1 = h)$, that is

$$V_{ij}(t|T_1 = h) = \mathbf{P}\{X(t) = j \mid X(0) = i, T_1 = h\}.$$

In this case

$$V_{ij}(t|T_1 = h) = \begin{cases} \delta_{ij} & h > t \\ \displaystyle\sum_{k \in S} \mathbf{P}\{X(T_1) = k \mid X(0) = i, T_1 = h\}\; V_{kj}(t-h) & h \leq t, \end{cases} \tag{32}$$

where $\mathbf{P}\{X(T_1) = k \mid X(0) = i, T_1 = h\}$ is the probability that the process starts from state i at time $t = 0$ and it is in state k right after the state transition at time T_1 assuming $T_1 = h$. In contrast with CTMCs this probability depends on the

sojourn time in state i:

$$\begin{aligned}
&\mathbf{P}\{X(T_1) = k \mid X(0) = i, T_1 = h\} \\
&= \lim_{\Delta \to 0} \frac{\mathbf{P}\{X(T_1) = k, h < T_1 \leq h + \Delta \,|\, X(0) = i\}}{\mathbf{P}\{h < T_1 \leq h + \Delta \,|\, X(0) = i\}} \\
&= \lim_{\Delta \to 0} \frac{K_{ik}(h+\Delta) - K_{ik}(h)}{H_i(h+\Delta) - H_i(h)} = \frac{dK_{ik}(h)}{dH_i(h)} \quad ,
\end{aligned} \tag{33}$$

where $H_i(h)$ denotes the distribution of time spent in state i,

$$\begin{aligned}
&H_i(t) = \mathbf{P}\{T_1 \leq t \,|\, X(0) = i\} = \\
&\sum_{k \in S} \mathbf{P}\{X(T_1) = k, T_1 \leq t \,|\, X(0) = i\} = \sum_{k \in S} K_{ik}(t)\,.
\end{aligned} \tag{34}$$

It is commonly assumed that the state transitions are real, which means that after staying in state i a state transition moves the process to a different state. It means that $K_{ii}(t) = 0,\ \forall i \in S$. It is also possible to consider virtual state transitions from state i to state i, but it does not expand the set of SMPs and we do not consider this case here. Note that the meaning of a diagonal element of a SMP kernel matrix is completely different from the meaning of a diagonal element of an infinitesimal generator of a CTMC. One of the technical consequences of this difference is the fact that we do not need to exclude the diagonal element from the summations over the set of states.

There are two cases considered in Eq. (32):

- If the time point of interest, t, is before the first state transition of the process (i.e. $t < h$), then the conditional state transition probability is either 0 or 1 depending on the initial and the final state. If the initial state i is identical with the final state j then the transition probability is 1, because there is no state transition up to time t, otherwise it is 0.
- If the time point of interest, t is after the first state transition of the process (i.e. $h \leq t$) then we need to evaluate the distribution of the next state k assuming that the state transition occurs at time h, and after that the state transition probability from the new state k to the final state j during time $t - h$, using the Markov property of the process at time h. The probability that the process moves to state k assuming it occurs at time h is $\dfrac{dK_{ik}(h)}{dH_i(h)}$ and the probability of moving from state k to state j during an interval of length $t - h$ is $V_{kj}(t-h)$.

The distribution of the condition of Eq. (32) is known. The distribution of the sojourn time in state i is $H_i(h)$. Using the law of total probability we obtain

$$\begin{aligned}
V_{ij}(t) &= \int_{h=t}^{\infty} \delta_{ij}\ dH_i(t) + \int_{h=0}^{t} \sum_{k \in S} \frac{dK_{ik}(h)}{dH_i(h)}\ V_{kj}(t-h)\ dH_i(h) \\
&= \delta_{ij}\ (1 - H_i(t)) + \int_{h=0}^{t} \sum_{k \in S} V_{kj}(t-h)\ dK_{ik}(h).
\end{aligned} \tag{35}$$

The analysis of an SMP based on the first state transition resulted in a Volterra integral equation. The transient behaviour of SMPs can be computed using numerical methods for Volterra integral equations.[26]

Transform domain description: We take the Laplace-Stieltjes transform (LST) of both sides of the Volterra integral equation (35). The only non-trivial term is a convolution integral on the right hand side.

$$V_{ij}^{\sim}(s) = \delta_{ij}\ (1 - H_i^{\sim}(s)) + \sum_{k \in S} K_{ik}^{\sim}(s)\ V_{kj}^{\sim}(s),$$

where the LST functions are defined as $f^{\sim}(s) = \int_0^{\infty} e^{-st} df(t)$.

Introducing the diagonal matrix $\boldsymbol{E}^{\sim}(s)$ composed by the elements $1 - H_i^{\sim}(s)$, that is $\boldsymbol{E}^{\sim}(s) = \mathrm{diag}\langle 1 - H_i^{\sim}(s)\rangle$, the LST of the state transition probabilities are obtained in matrix form:

$$\boldsymbol{V}^{\sim}(s) = \boldsymbol{E}^{\sim}(s) + \boldsymbol{K}^{\sim}(s)\boldsymbol{V}^{\sim}(s),$$

from which

$$\boldsymbol{V}^{\sim}(s) = [\boldsymbol{I} - \boldsymbol{K}^{\sim}(s)]^{-1}\boldsymbol{E}^{\sim}(s).$$

Stationary behaviour: Let the transition probability matrix of the EMC be $\boldsymbol{P} = [p_{ij}]$. It is obtained from the kernel matrix through the following relation

$$p_{ij} = \mathbf{P}\{X(T_1) = j \,|\, X(0) = i\} = \lim_{t \to \infty} \mathbf{P}\{X(T_1) = j, T_1 \leq t \,|\, X(0) = i\} = \lim_{t \to \infty} K_{ij}(t).$$

The stationary distribution of the EMC $\boldsymbol{\nu} = [\nu_i]$ is the solution of the linear system $\boldsymbol{\nu} = \boldsymbol{\nu P}, \sum_i \nu_i = 1$. The stationary distribution of the SMP is given by

$$\pi_i = \frac{\nu_i\, h_i}{\sum_j \nu_j\, h_j}, \tag{36}$$

where h_i is the mean time spent in state i and can be computed from the kernel matrix through $h_i = \int_0^{\infty} (1 - H_i(t)) dt$.

5.2. *Example*

The system we study is taken from Chen and Trivedi[15] and consists of a server type software that is available in state $\boldsymbol{U}$. The software can fail, upon which, recovery procedure is started. The state, in which the software is recovering and is unavailable for service, is denoted as state $\boldsymbol{D}$. The software occasionally undergoes rejuvenation, or preventive maintenance (PM). This state is denoted as state $\boldsymbol{R}$. When the system stays in the available state $\boldsymbol{U}$, it will undergo a rejuvenation and enter state $\boldsymbol{R}$ after a random time with distribution function $F_0(t)$ or fail and enter state $\boldsymbol{D}$ with a general distribution function $F_2(t)$. The distribution function for the duration of a preventive maintenance is $F_1(t)$, and the distribution function for the duration of repair is $F_3(t)$. We assume that all the states are regenerative, that

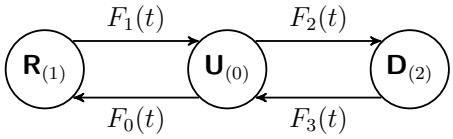

Fig. 5. An SMP model for a software system with preventive maintenance (rejuvenation) and failure

is, any transition instance experiences the Markov property. Hence, the process is an SMP to which we can apply the transient and steady-state analysis of Section 5. The kernel matrix $\boldsymbol{K}(t)$ has the following structure:

$$\boldsymbol{K}(t) = \begin{bmatrix} 0 & K_{01}(t) & K_{02}(t) \\ K_{10}(t) & 0 & 0 \\ K_{20}(t) & 0 & 0 \end{bmatrix}$$

With

$$\begin{aligned} K_{01}(t) &= \mathbf{P}\{\text{PM occurs before failure within time t}\} \\ &= \int_0^t (1 - F_2(x))\, dF_0(x)\,, \end{aligned}$$

$$\begin{aligned} K_{02}(t) &= \mathbf{P}\{\text{failure occurs before PM within time t}\} \\ &= \int_0^t (1 - F_0(x))\, dF_2(x)\,, \end{aligned}$$

$$K_{10}(t) = \mathbf{P}\{\text{PM completes within time t}\} = F_1(t)\,,$$

$$K_{20}(t) = \mathbf{P}\{\text{repairs completes within time t}\} = F_3(t)\,.$$

It follows that the one-step transition probability matrix of the EMC $\boldsymbol{P} = \lim_{t\to\infty} \boldsymbol{K}(t)$ becomes:

$$\boldsymbol{P} = \begin{bmatrix} 0 & p_{01} & p_{02} \\ p_{10} & 0 & 0 \\ p_{20} & 0 & 0 \end{bmatrix}$$

Solving the EMC steady-state equations $\boldsymbol{\nu} = \boldsymbol{\nu P}, \sum_i \nu_i = 1$ we get

$$\nu_0 = 1/2,\ \nu_i = p_{0i}/2,\ i = 1, 2$$

Further the expected sojourn time in all states are

$$\begin{aligned} h_0 &= \mathbf{E}[H_0] = \int_0^\infty (1 - F_0(t))(1 - F_2(t))\, dt\,, \\ h_1 &= \mathbf{E}[H_1] = \int_0^\infty (1 - F_1(t))\, dt\,, \\ h_2 &= \mathbf{E}[H_2] = \int_0^\infty (1 - F_3(t))\, dt\,. \end{aligned}$$

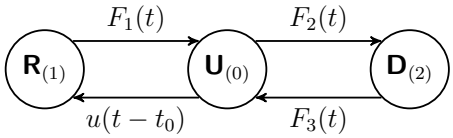

Fig. 6. An SMP model for a software system with rejuvenation triggered at constant time t_0

Applying Eq. (36), we obtain the steady-state availability of the system as:

$$A = \pi_0 = \frac{h_0}{h_0 + h_1 p_{01} + h_2 p_{02}}. \tag{37}$$

In a special case of great importance in realistic applications where the rejuvenation (or the preventive maintenance) is executed periodically with a deterministic time period t_0, we have $F_0(t) = u(t - t_0)$ (see Fig. 6), and the one-step transition probabilities simplify to

$$\begin{aligned} p_{01} &= \mathbf{P}\{\text{PM triggers before failure occurs}\} = 1 - F_2(t_0)\,, \\ p_{02} &= \mathbf{P}\{\text{failure occurs before PM triggers}\} = F_2(t_0)\,, \\ p_{10} &= p_{20} = 1\,. \end{aligned}$$

The mean sojourn times h_1 and h_2 remain unchanged and

$$h_0 = \mathbf{E}[H_0] = \int_0^{t_0} (1 - F_2(t))\, dt\,.$$

To proceed further and compute the availability in Eq. (37), the distribution functions $F_i(t),\ (i = 1 - 3)$ should be assigned. Examples assuming Weibull, Two-stage Hypoexponential or Log-normal distributions are given in.[15,18]

6. Markov Regenerative Process

The discrete-state, continuous-time, and time-homogeneous stochastic process $X(t)$ is a Markov regenerative process (MRGP) if there exists a random time series $T_0, T_1, T_2, \ldots$ $(T_0 = 0)$ such that the process $X(t)$ experiences the Markov property at time epochs $T_0, T_1, T_2, \ldots$[21,27] (see Fig. 7). The time epochs $T_0, T_1, T_2, \ldots$ that satisfy the Markov property are called *regeneration time points* (RTPs).

An MRGP is a generalization of many stochastic processes.[21,27] In a CTMC any time epoch t satisfies the Markov property (Eq. 8), in an SMP the Markov property is satisfied at all the time instances at which the process undergoes a state transition, in an MRGP the Markov property holds only when the process enters a subset of specific states called *regeneration states*.

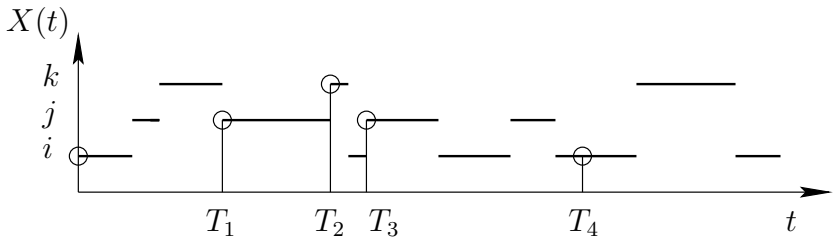

Fig. 7. Markov regenerative process – circles denote the points with Markov property

Compared to the properties of SMP, where the process experiences the Markov property at all state transition instances, the definition of MRGPs is less restrictive since the MRGP experiences the Markov property only at some transition instances, i.e. the RTPs. The analysis of MRGPs is based on the occurrence of the RTPs where the process experiences the Markov property.

The definition of an MRGP does not prescribe how the behaviour of the process between two consecutive RTPs $T_0, T_1, T_2, \ldots$ should be. Hence, MRGPs can be characterized by any kind of stochastic process between two consecutive RTPs. In practice, the use of renewal theorem for the analysis of these processes is meaningful only when the stochastic behavior between consecutive time points $T_0, T_1, T_2, \ldots$ is analytically tractable. The common analysis method of MRGPs is based on the analysis of the next time point with Markov property (T_1).

We denote by $Y_0, Y_1, Y_2, \ldots$ the series of states reached by the MRGP right after an RTP: i.e. $Y_i = X(T_i)$. The series of random variables $\{Y_n, T_n; n \geq 0\}$ is a time-homogeneous *Markov renewal series* since it satisfies the following relation

$$\begin{aligned}
&\mathbf{P}\{Y_{n+1} = y, T_{n+1} - T_n \leq t \,|\, Y_0, \ldots, Y_n, T_0, \ldots, T_n\} \\
&= \mathbf{P}\{Y_{n+1} = y, T_{n+1} - T_n \leq t \,|\, Y_n\} \\
&= \mathbf{P}\{Y_1 = y, T_1 - T_0 \leq t \,|\, Y_0\}
\end{aligned}$$

for all $n \geq 0$, $y \in \Omega$ and $t \geq 0$. From the definition of Markov renewal series, the sequence of states $Y_0, Y_1, Y_2, \ldots$ forms a DTMC.

The analysis of an MRGP is based on this *embedded* Markov renewal series. To this end the joint distribution of the next RTP and the state in that RTP has to be known, and it is denoted as

$$K_{ij}(t) = \mathbf{P}\{Y_1 = j, T_1 - T_0 \leq t \,|\, Y_0 = i\}\,, \quad i, j \in S$$

Matrix $\boldsymbol{K}(t) = \{K_{ij}(t)\}$ is referred to as the global kernel of the MRGP and completely characterizes the stochastic properties of the MRGP at the RTPs. The description of the process between RTPs is complex, but for the transient analysis (more precisely for computing transient state probabilities) it is enough to know the transient state probabilities between consecutive RTPs. It is given by the local

kernel matrix of the MRGP $\boldsymbol{E}(t) = \{E_{ij}(t)\}$, whose elements are

$$E_{ij}(t) = \mathbf{P}\{X(t) = j,\, T_1 > t,\, |\, Y_0 = i\} \ .$$

$E_{ij}(t)$ is the probability that the process starts in state i, the first RTP is later than t and the process stays in state j at time t.

6.1. *Transient Analysis based on the Embedded Markov Renewal Series*

Let the transient state transition probability matrix be $\boldsymbol{V}(t)$, whose elements are

$$V_{ij}(t) = \mathbf{P}\{X(t) = j \,|\, X(0) = i\} \ .$$

Assuming that $T_1 = h$ we can compute the conditional state transition probability as follows

$$V_{ij}(t \,|\, T_1 = h) = \begin{cases} \mathbf{P}\{X(t) = j \,|\, T_1 = h,\, X(0) = i\}, & h > t, \\ \displaystyle\sum_{k \in S} \mathbf{P}\{X(T_1) = k \mid X(0) = i, T_1 = h\} \cdot V_{kj}(t-h), & h \le t. \end{cases} \tag{38}$$

Similar to the transient analysis of SMPs, the Eq. (38) describes two exhaustive and exclusive events $h \le t$ and $h > t$. In case of SMPs the $h > t$ case resulted in 0 or 1, while in case of a MRGP the conditional probability for $h > t$ can be different from 0 or 1, because the process can have state transitions also before T_1.

Using the distribution of T_1 and the formula of total probability we obtain

$$\begin{aligned} V_{ij}(t) = &\int_{h=t}^{\infty} \mathbf{P}\{X(t) = j \,|\, T_1 = h,\, X(0) = i\}\, dH_i(h) \\ &+ \int_{h=0}^{t} \sum_{k \in S} \frac{dK_{ik}(h)}{dH_i(h)}\, V_{kj}(t-h)\, dH_i(h)\,, \end{aligned} \tag{39}$$

where $H_i(t) = \sum_j K_{ij}(t)$ (see Eq. (34)) is the distribution of time spent in state i. Let us consider the first term of the right hand side

$$\begin{aligned} &\int_{h=t}^{\infty} \mathbf{P}\{X(t) = j \,|\, T_1 = h,\, X(0) = i\}\, dH_i(h) \\ &= \int_{h=t}^{\infty} \lim_{\Delta \to 0} \mathbf{P}\{X(t) = j \,|\, h \le T_1 < h + \Delta,\, X(0) = i\}\ dH_i(h) \\ &= \int_{h=t}^{\infty} \lim_{\Delta \to 0} \frac{\mathbf{P}\{X(t) = j,\, h \le T_1 < h + \Delta \,|\, X(0) = i\}}{\mathbf{P}\{h \le T_1 < h + \Delta,\, |\, X(0) = i\}}\ dH_i(h) \\ &= \int_{h=t}^{\infty} \frac{d_h\ \mathbf{P}\{X(t) = j,\, T_1 < h \,|\, X(0) = i\}}{dH_i(h)}\ dH_i(h) \\ &= \mathbf{P}\{X(t) = j,\, t < T_1 \,|\, X(0) = i\} \end{aligned}$$

from which

$$V_{ij}(t) \;=\; E_{ij}(t) \;+\; \sum_{k\in S} \int_{h=0}^{t} V_{kj}(t-h)\, dK_{ik}(h)\,, \tag{40}$$

which is a Volterra integral equation.

Similar to the transient analysis of CTMCs and SMPs, we obtained the above Volterra equation for the transient analysis of MRGPs.

Transform domain description: The LST of Eq. (40) is

$$V^{\sim}_{ij}(s) \;=\; E^{\sim}_{ij}(s) + \sum_{k\in\Omega} K^{\sim}_{ik}(s)\, V^{\sim}_{kj}(s), \tag{41}$$

which can be written in matrix form

$$\boldsymbol{V}^{\sim}(s) \;=\; \boldsymbol{E}^{\sim}(s) \;+\; \boldsymbol{K}^{\sim}(s)\, \boldsymbol{V}^{\sim}(s). \tag{42}$$

The solution of Eq. (42) is

$$\boldsymbol{V}^{\sim}(s) \;=\; [\boldsymbol{I} - \boldsymbol{K}^{\sim}(s)]^{-1}\, \boldsymbol{E}^{\sim}(s). \tag{43}$$

Based on Eq. (43), numerical inverse Laplace methods can also be used for the transient analysis of MRGPs.

Stationary behaviour: In spite of the differences between SMP and Markov regenerative processes their stationary analysis goes along the same steps. The state transition probability of the DTMC embedded in the RTPs $\boldsymbol{P} = [p_{ij}]$ is computed from:

$$\begin{aligned} p_{ij} &= \mathbf{P}\{X(T_1) = j \,|\, X(0) = i\} = \lim_{t\to\infty} \mathbf{P}\{X(T_1) = j, T_1 \le t \,|\, X(0) = i\} \\ &= \lim_{t\to\infty} K_{ij}(t)\ . \end{aligned}$$

The stationary distribution of the EMC $\boldsymbol{\nu} = [\nu_{ij}]$ is the solution of

$$\boldsymbol{\nu} = \boldsymbol{\nu}\boldsymbol{P}, \qquad \sum_i \nu_i = 1. \tag{44}$$

Now we need to compute the mean time spent in different states during the interval (T_0, T_1). Fortunately, the local kernel carries the necessary information. Let α_{ij} be the mean time the process spends in state j during the interval (T_0, T_1) assuming that it starts from state i ($X(T_0) = i$). Then

$$\alpha_{ij} = \mathbf{E}\left[\int_{t=0}^{\infty} \mathcal{I}_{\{X(t)=j, T_1>t\}} \;|\; X(T_0) = i\, dt\right] \tag{45}$$

where $\mathcal{I}_{\{\bullet\}}$ is the binary indicator variable of event $\bullet$, i.e. $\mathcal{I}_{\{\bullet\}} = 1$ when event $\bullet = true$ and $\mathcal{I}_{\{\bullet\}} = 0$ when event $\bullet = false$. When the process stays in state j at time t and T_1 is greater than t then $\mathcal{I}_{\{X(t)=j, T_1>t\}} = 1$, that is, the integral in (45) counts the time the process spends in state j during the interval (T_0, T_1). This is a

random time which depends on the evolution of the process and its mean is needed for the stationary analysis. From Eq.(45) we derive:

$$\begin{aligned}\alpha_{ij} &= \int_{t=0}^{\infty} \mathbf{E}\big[\mathcal{I}_{\{X(t)=j, T_1>t\}} \mid X(T_0)=i\big]\, dt \\ &= \int_{t=0}^{\infty} \mathbf{P}\{X(t)=j, T_1>t \mid X(T_0)=i\} dt \\ &= \int_{t=0}^{\infty} E_{ij}(t) dt,\end{aligned}$$

The mean length of the interval (T_0, T_1) is

$$\alpha_i = \sum_{j\in\Omega} \alpha_{ij} \ .$$

Finally the stationary distribution of the process can be computed as

$$\pi_i = \frac{\sum_{j\in\Omega} \nu_j \ \alpha_{ji}}{\sum_{j\in\Omega} \nu_j \ \alpha_j} \ . \tag{46}$$

6.2. *Example*

This example is taken from Garg et al.[9] and refers to a client-server type system, where the server software is required to run continuously for very long periods, and is periodically rejuvenated at a constant interval of length δ. In their paper, the system was modeled through a Markov regenerative stochastic Petri net (MRSPN) (see Section 7.4); however, in this example we utilize the state transition model derived from the MRSPN, reported in Garg et al.,[9] and shown in Fig. 8. State 1

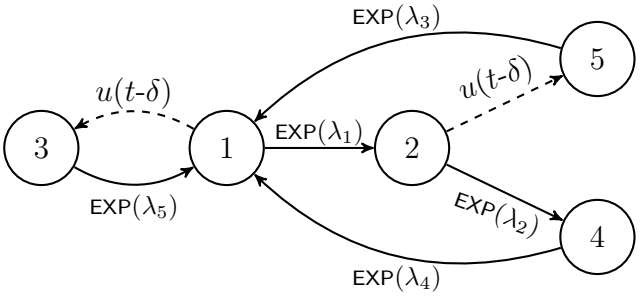

Fig. 8. Markov regenerative process

models the robust up state. The aging of the software moves the system to the failure probable state 2. From state 2 a crash failure of the software leads the system to state 4, where a restart is initiated and every other activity is suspended. Rejuvenation is activated at constant intervals of duration δ. The dashed transitions

labelled $u(t-\delta)$ indicate that the clock count is not reset by the possible jump from state 1 to state 2, and it is resumed in state 2 from the value it had before the jump. If the system is in state 1 when rejuvenation is triggered the system transits to state 3, while if the system is in the failure probable state 2, the transition is toward state 5. From both states 3 and 5 the rejuvenation leads back the system to the robust state 1, and every other activity is suspended. It is assumed that rejuvenation and repair are completely regenerative. We assume that all the transition times are exponentially distributed, but the trigger time of rejuvenation is deterministic.

Considering the presence of exponential transitions, it can he seen from Fig. 8 that the regeneration times exactly correspond to times of entering either of states 1, 3, 4 or 5. In contrast, when the process transits from state 1 to state 2 the deterministic activity initiated in state 1 goes on. Thus, in order to probabilistically determine future state of the process, the time already spent in state 1 needs to be known apart from the knowledge that the process is in state 2. The process identified in Fig. 8 is an MRGP.

The set of states at regeneration instants is $\Omega = \{1, 3, 4, 5\}$ and the global kernel matrix becomes:

$$\boldsymbol{K}(t) = \begin{bmatrix} 0 & K_{13}(t) & K_{14}(t) & K_{15}(t) \\ K_{31}(t) & 0 & 0 & 0 \\ K_{41}(t) & 0 & 0 & 0 \\ K_{51}(t) & 0 & 0 & 0 \end{bmatrix}$$

Note that the subscripts ij on $K_{ij}(t)$ denote the actual state labels in Fig. 8. For the non-zero entries, we have

$$K_{13}(t) = e^{-\lambda_1 \delta}\, u(t-\delta)\,,$$

$$K_{14}(t) = \begin{cases} 1 - \dfrac{\lambda_1}{\lambda_1 - \lambda_2} e^{-\lambda_2 t} + \dfrac{\lambda_2}{\lambda_1 - \lambda_2}\, e^{-\lambda_1 t}\,, & 0 \le t < \delta \\ 1 - \dfrac{\lambda_1}{\lambda_1 - \lambda_2} e^{-\lambda_2 \delta} + \dfrac{\lambda_2}{\lambda_1 - \lambda_2}\, e^{-\lambda_1 \delta}\,, & t \ge \delta\,, \end{cases}$$

$$K_{15}(t) = \frac{\lambda_1}{\lambda_1 - \lambda_2} \left[e^{-\lambda_2 \delta} - e^{-\lambda_1 \delta} \right] u(t-\delta)\,,$$

$$K_{i1}(t) = 1 - e^{-\lambda_i t},\ i = 3,\ 4,\ 5.$$

Since the local kernel $\boldsymbol{E}(t)$ describes the time behavior of the process between two consecutive RTPs, starting from an RTP, hence it is a 4 x 5 matrix given as:

$$\boldsymbol{E}(t) = \begin{bmatrix} E_{11}(t) & E_{12}(t) & 0 & 0 & 0 \\ 0 & 0 & E_{33}(t) & 0 & 0 \\ 0 & 0 & 0 & E_{44}(t) & 0 \\ 0 & 0 & 0 & 0 & E_{55}(t) \end{bmatrix}$$

Again, the subscripts ij on $E_{ij}(t)$ denote the actual state labels in Fig. 8. Zero entries in the matrix at ij-th location indicate that either a transition from i to j

is not possible or it results in a regeneration. The non-zero entries are

$$E_{11}(t) = e^{-\lambda_1 t}\,(u(t) - u(t-\delta))\,,$$
$$E_{12}(t) = \frac{\lambda_1}{\lambda_1 - \lambda_2}\left[e^{-\lambda_2 t} - e^{-\lambda_1 t}\right]\,(u(t) - u(t-\delta))\,,$$
$$E_{ii}(t) = e^{-\lambda_i t},\ i = 3,\ 4,\ 5.$$

$E_{11}(t)$ is the probability that the process starting in state 1, stays in it till time t. Here t lies in the interval $[0, \delta)$ as at time δ, the process transits to state 3 resulting in a regeneration. $E_{12}(t)$ is the probability that at time t the process is in state 2 given that it was in state 1 at $t = 0$. Note that t takes values in $[0, \delta)$. $E_{12}(t)$ is obtained by conditioning on time to enter state 2 and un-conditioning. $E_{ii}(t)$, for $i = 3,\ 4,\ 5$ is the probability that the corresponding enabled exponential transition does not fire by time t .

The transient solution is obtained by applying Eq. (43) and inverting the related LST. To compute the steady state distribution the non-zero α_{ij} entries are

$$\alpha_{11} = \int_0^\infty E_{11}(t)\,dt = \frac{1}{\lambda_1}[1 - e^{-\lambda_1 \delta}]\,,$$
$$\alpha_{12} = \int_0^\infty E_{12}(t)\,dt = \frac{1}{\lambda_2} - \frac{\lambda_1}{\lambda_1 - \lambda_2}\left[\frac{e^{-\lambda_2 \delta}}{\lambda_2} - \frac{e^{-\lambda_1 \delta}}{\lambda_1}\right]\,,$$
$$\alpha_{ii} = \int_0^\infty E_{ii}(t)\,dt = \frac{1}{\lambda_i},\ i = 3,\ 4,\ 5$$

and the solution of (44) is

$$\nu_1 = \frac{1}{2}\,,$$
$$\nu_3 = \frac{1}{2}e^{-\lambda_1 \delta}\,,$$
$$\nu_4 = \frac{1}{2} - \frac{\lambda_1}{2(\lambda_1 - \lambda_2)}e^{-\lambda_2 \delta} - \frac{\lambda_2}{2(\lambda_1 - \lambda_2)}e^{-\lambda_1 \delta}\,,$$
$$\nu_5 = \frac{\lambda_1}{2(\lambda_1 - \lambda_2)}\left[e^{-\lambda_2 \delta} - e^{-\lambda_1 \delta}\right]\,.$$

The stationary distribution is obtained from Eq. (46) as

$$\pi_i = \nu_i \alpha_{ii},\ i = 1, 3, 4, 5,$$
$$\pi_2 = \nu_1 \alpha_{12}\ .$$

7. Petri Nets

Petri nets (PNs) are a mathematical and graphical tool for describing complex systems with concurrent and conflicting activities. The structure of a standard Petri net (PN) is a bipartite graph that comprises a set of *places*, a set of *transitions*, and a set of *directed arcs*.[28]

Formally a PN is defined by a tuple $\mathcal{PN} = (\mathcal{P}, \mathcal{T}, I, O, H, M_0)$, where:

- $\mathcal{P}$ is the set of places with cardinality n_P;
- $\mathcal{T}$ is the set of transitions with cardinality n_T;
- $I, O, H : \mathcal{T} \rightarrow Bag(\mathcal{P})$ are respectively the sets of input, output, and inhibition functions, that map transitions to bags[a] of places;
- M_0 is the initial marking of the net, i.e., the initial distribution of tokens among the places in $\mathcal{P}$.

A place p is called (i) an *input* place for a given transition t if there is an input arc that connects p to t; (ii) an *output* place if there is an output arc that connects t to p; (iii) an *inhibitor* place if an inhibitor arc connects p to t. In the following, the symbols $I(t)$, $O(t)$ and $H(t)$ respectively denote the sets of input, output and inhibitor places of transition t.

The marking of a PN, is defined by a bag M_i with the number of tokens contained in each places. The symbol $M_i(p)$ denotes the numbers of tokens contained in place p in marking M_i. The graphical structure of a PN is a bipartite directed graph where places (drawn as circles) and transitions (drawn as bar) are the nodes, and arcs (drawn as arrows) are the edges. Markings are represented by black dots (called tokens) in the places. An example of graphical representation of a PN is shown in Fig. 9(a) with initial marking $M_0 = (1001100)$.

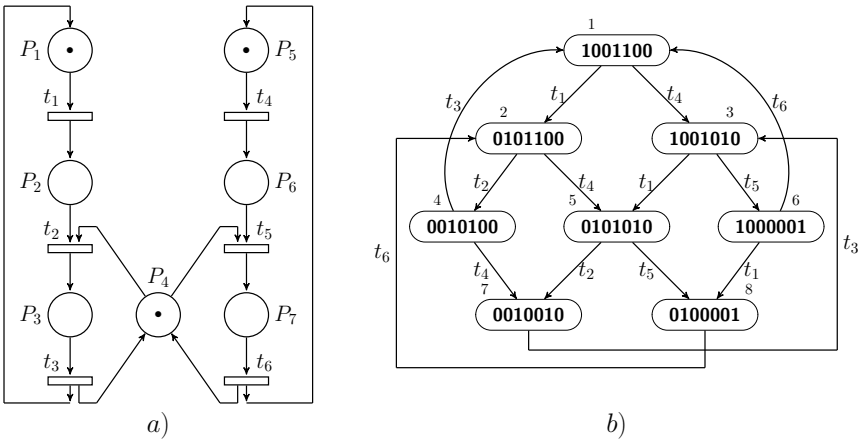

Fig. 9. A marked PN a) and its reachability graph b).

The dynamic evolution of a PN is specified by the *enabling* and *firing* rules. A transition $t \in \mathcal{T}$ is *enabled* in marking M_i if

(1) each input place contains at least as many tokens as the multiplicity of the corresponding input arc,

[a]A bag is a set with duplication of elements allowed; that is, it keeps track of "occurrence count" of each set element.

(2) each inhibitor place contains a number of tokens strictly smaller than the multiplicity of the corresponding inhibitor arc.

When a transition t_k is enabled in marking M_i it can *fire*, removing from each input place as many tokens as the multiplicity of the input arcs, and adding in all the output places as many tokens as the multiplicity of the output arcs. The firing of a transition in marking M_i produces a new marking M_j with a token count given by:

$$M_j = M_i - I(t_k) + O(t_k). \tag{47}$$

If two transitions are enabled in the same marking and the firing of one does not disable the other one the two transitions are said to be *concurrent*. If the firing of one of the two disables the other one, the two transitions are said to be in *conflict*.

Reachability graph - The firing rules allow to describe the evolution of a marked PN starting from its initial marking M_0. Every time an enabled transition $t_k \in \mathcal{T}$ fires in marking M_i a new marking M_j is computed; the marking M_j is said to be directly reachable from M_i, and this property is denoted with $M_i - t_k \rightarrow M_j$. Applying the firing rule to the reached marking, it is possible to find a sequence of transition firings and new markings describing the net evolution. The set of all reachable markings starting from an initial marking M_0 is called the *reachability set* and is denoted by $RS(M_0)$. The reachability set can be represented as a graph whose nodes are the markings in $RS(M_0)$ and the arcs connecting the nodes are labelled with the transitions whose firing causes the change of marking $M_i - t_k \rightarrow M_j$. This graph is called *reachability graph*, and is denoted by $R(M_0)$. The reachability graph of the PN shown in Fig. 9(a) is represented in Fig. 9(b). Starting from the initial marking M_0, shown in Fig. 9(a), and applying the enabling and firing rules, it is possible to derive the structure of Fig. 9(b).

An *execution sequence* $\mathcal{E}$ in a marked PN, is a sequence of legal markings obtained by firing a sequence of enabled transitions (note that $M_{(0)} = M_0$):

$$\mathcal{E} = \{ (M_{(0)}, t_{(0)}) ; (M_{(1)}, t_{(1)}) ; \ldots ; (M_{(j)}, t_{(j)}) ; \ldots \} .$$

An execution sequence $\mathcal{E}$ can be viewed as a connected path in the reachability graph $R(M_0)$ of the PN.

7.1. *Stochastic Petri Net (SPN)*

Original PNs did not carry any notion of time.[29] In order to use the PN formalism for the quantitative analysis of software systems, several extensions have been proposed in the literature, in particular by associating *firing times* with the transitions.[30]

A timed execution sequence $\mathcal{S}_E$ of a marked PN starting from an initial marking $M_{(0)}$, is an execution sequence $\mathcal{E}$ augmented by a non-decreasing sequence of real values representing the epochs of firing of each transition, such that consecutive

transitions $(t_{(j)}\,;\,t_{(j+1)})$ in $\mathcal{E}$ correspond to ordered epochs $\tau_j \leq \tau_{j+1}$ in $\mathcal{S}_E$. Thus formally:[31,32]

$$\mathcal{S}_E \;=\; \{\,(M_{(0)},t_{(0)},\tau_0)\,;\,(M_{(1)},t_{(1)},\tau_1)\,;\,\ldots\,;\,(M_{(j)},t_{(j)},\tau_j)\,;\,\ldots\}\,.$$

The time interval $\tau_{j+1}-\tau_j$ between consecutive epochs represents the period that the PN sojourns in marking $M_{(j)}$. In the sequel we always assume the initial epoch to be $\tau_0 = 0$. The ensemble of all the possible timed execution sequences defines the Marking Process $\mathcal{M}(t)$.

Stochastic Petri nets (SPNs)[31,33] are PNs in which the firing delays between successive transitions are exponentially distributed random variables. To this end, a transition t_k enabled in marking M_j is associated a constant, possibly marking dependent, firing rate $\lambda_k(M_j)$. Molloy[33] has proved that when the random firing times associated with PN transitions are exponentially distributed, the dynamic behavior of the PN can be mapped into a homogeneous continuous-time Markov chain with state space isomorphic to the reachability graph of the PN. The isomorphic CTMC can be automatically generated from the reachability graph $RS(M_0)$ with the following rules:

- at each marking $M_i \in RS(M_0)$ corresponds a state i in the CTMC,
- an arc between two states i and j of the CTMC has a rate given by the sum of rates of the transitions whose individual firings produce the change of marking from M_i to M_j.

From the reachability graph of Fig. 9(b) it is immediate to derive the corresponding CTMC, by replacing markings with states, and assigning to each arc the constant transition rate λ_i assigned to the PN transition t_i.

7.2. *Generalized Stochastic Petri Nets (GSPN)*

Activities whose durations differ by orders of magnitude can often be encountered in the modeling process of software systems. The corresponding CTMC becomes stiff and hard to solve numerically.[24]

To overcome this problem, Ajmone et al.[34] have introduced two classes of transitions: timed transitions with exponentially distributed firing times and untimed or *immediate* transitions. Immediate transitions fire in zero time and have higher priority over the timed ones. SPNs augmented with immediate transitions are called *generalized* SPNs (GSPNs).[34,35]

The markings where only timed transitions are enabled are referred to as *tangible* markings, in contrast, markings where at least one single immediate transition is enabled are called *vanishing* markings. Vanishing markings are traversed in zero time and do not contribute to the time evolution of the net but only influence the logic evolution.

In a vanishing marking only immediate transitions can fire since they have higher priority over the timed ones. If in a vanishing marking more than one immediate

transition is enabled, a rule to select the one that fires needs to be introduced. The choice is done in a probabilistic way associating a weight w_i to any immediate transition t_i. Given the set of immediate transitions $T_e^{(i)}$ enabled in marking M_i, the probability that transitions $t_k \in T_e^{(i)}$ fires is

$$\mathbf{P}\{t_k \text{ fires}|\text{marking } M_i\} = \frac{w_k}{\sum_{t \in T_e^{(i)}} w_t}\,. \tag{48}$$

These probabilities are called *switching probabilities* in GSPN terminology.

The evolution of a GSPN cannot be described by a CTMC, due to the presence of vanishing markings.[36] However, an automated procedure can be envisaged to progressively eliminate vanishing states from the reachability graph until only tangible states remain. A reachability graph where all the vanishing markings are eliminated is called *reduced reachability graph* and over the reduced reachability graph an isomorphic CTMC can be build. The reduction procedure, described in,[18,34,36] can be implemented in software tools so that becomes completely transparent to the analyst.

The marking process $\mathcal{M}(t)$ after elimination of vanishing states is a CTMC generated as explained in Section 7.1, and can be solved using Eqs. (19) and (27).

7.3. *Stochastic Reward Nets (SRN)*

GSPNs provide a useful high-level language but their use in the representation of large systems may lead to very intricate structures. To alleviate this problem and make the high level representation more compact and readable, several structural extensions to PNs were introduced in the literature. We refer, in particular, to the *Stochastic Reward Nets* (SRN) formslism defined in.[36,37]

The SRN extensions include guard functions, very general marking dependency, variable cardinality arcs and the superposition of a reward structure. From an SRN, a Markov reward model is thus generated, facilitating the specification and computation of important reward-based metrics (see Section 3.5). Parameters such as the firing rate of the timed transitions, the multiplicities of input/output arcs and the reward rate in a marking can be specified as functions of the number of tokens in any place in the SRN. Another important characteristic of SRN is the ability to express complex enabling/disabling conditions through guard functions that greatly simplify the graphical representations of complex systems.

To get the performance and reliability/availability measures of a system, appropriate reward rates are assigned to its SRN model. As SRN is automatically transformed into a Markov reward model (MRM), thence steady state and/or transient numerical solution of the MRM produces the required measures of the original SRN. Examples of use of SRN models in rejuvenation problems can be found in.[38–40]

7.4. *Non-Markovian Stochastic Petri Net (NMSPN)*

Non-Markovian stochastic Petri net (NMSPN) enriches the formalism of SPN, by assuming that the firing time associated to a transition can be a generally distributed random variable with known CDF. The inclusion of non-exponential distributions destroys the memoryless property of the associated marking process,[41,42] and further specifications are needed at the PN level for univocally defining the stochastic process underlying an NMSPN, given the topology of the PN and the set of CDFs associated to each timed transition. The set of specifications constitutes the *execution policy* whose semantic was first investigated by Ajmone et al.[32]

The execution policy comprises two specifications, the *firing policy* and the *memory policy*. The firing policy specifies how to select the next transition to fire. The standard choice is to adopt a *race policy*: the transition that fires is the one with minimum stochastic delay among the enabled transitions.

The memory policy specifies the way in which the process is conditioned upon the past history. In NMSPN the memory is represented by an age variable a_k associated to each timed transition t_k that increases with the time in which the transition is enabled. The way in which a_k is related to the past history determines the different memory policies. Three alternatives are considered:

- *Age memory* - The age variable a_k accounts for the total time in which t_k has been enabled from its last firing. The distribution of the firing delay in a given marking depends on the residual time needed for the transition to complete given a_k.
- *Enabling memory* - The age variable a_k accounts for the time elapsed from the last epoch in which t_k has been enabled. When transition t_k is disabled (even without firing) the corresponding enabling age variable is reset. The distribution of the firing delay in a given marking depends on the residual time needed for the transition to complete given a_k.
- *Resampling* - The age variable a_k is reset to zero at any change of marking. The firing distribution depends only on the time elapsed in the current marking.

At the entrance in a new tangible marking, the residual firing time is computed for each enabled timed transition given its age variable and memory policy and then the race policy is applied. The marking process $\mathcal{M}(t)$ does not have, in general, an analytically tractable formulation, unless various restrictions are applied.[42,43]

A *Markov regenerative stochastic Petri net (MRSPN)* is an NMSPN whose marking process $\mathcal{M}(t)$ is an MRGP (Section 6).[44] In an MRSPN an embedded sequence of RTPs should be present in the marking process, where an RTP occurs when at the entrance in a new tangible marking all the age variables are equal to 0. A sufficient condition for an NMSPN to be an MRSPN is that in each tangible marking at most one single transition with generally distributed firing time is allowed (being all the other ones exponential). Some extensions to the above rule have been studied in the literature.[45]

7.5. *Example*

The MRGP presented in Example 6.2 was generated by Garg et al.[9] by resorting to the MRSPN of Fig. 10. White rectangles represent transitions with exponentially

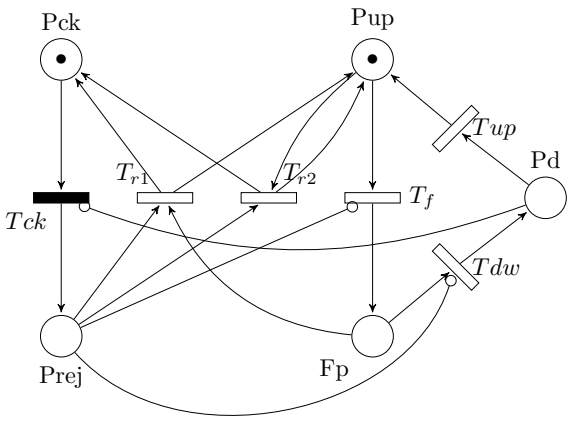

Fig. 10. The MRSPN generating the MRGP of Example 6.2

distributed firing time while the black rectangles represent a transition with generally distributed firing time. In the case of Fig. 10, the black rectangle models the clock with associated deterministic delay, which triggers rejuvenation at constant intervals. Place Pup is the up state. Transition T_f models the aging process of the software. When this transition fires the software enters the failure probable state Fp. The transition Tdw models crash failure of the software. During the restart of the software, represented by the firing of transition Tup, the clock count Tck is suspended. The transition Tck fires when the clock expires by moving a token in place Prej. The token in Prej starts the action related with software rejuvenation and, at the same time, suspends any other activity by means of inhibitors arcs. Upon the completion of rejuvenation, the net is reinitialized by putting one token again in place Pup, one in place Pck, being all the other places empty. If the software was in state Pup when Tck fired the rejuvenation action is represented by transition T_{r2}. If the software had reached the failure probable state Fp, then T_{r1} fires to complete the rejuvenation. In both cases the net is reinitialized.

As there is only one deterministic transition in the net, the condition for at most one generally distributed transition enabled at any time is automatically satisfied, and the SPN model belongs to the MRSPN class.

The generation of the corresponding MRGP from the MRSPN of Fig. 10 has been already discussed in Example 6.2, and the reachability graph has been presented in Fig. 8, where the following distributions were assigned to the PN transitions:

$$
\begin{array}{lcllcl}
T_f & \rightarrow & \mathrm{EXP}(\lambda_1) & Tdw & \rightarrow & \mathrm{EXP}(\lambda_2) \\
Tup & \rightarrow & \mathrm{EXP}(\lambda_4) & T_{r2} & \rightarrow & \mathrm{EXP}(\lambda_5) \\
T_{r1} & \rightarrow & \mathrm{EXP}(\lambda_3) & Tck & \rightarrow & u(t-\delta)
\end{array}
$$

8. Conclusions

We have provided an overview on the theory of stochastic processes and Petri nets that are mainly used in the literature on quantitative modeling of software aging and rejuvenation. We hope that the reader not particularly accustomed with this matter can find useful material to pursuing his/her investigation and research about the topics on which this book is centered.

Acknowledgements

A.B. received a financial support for this original work from the Università del Piemonte Orientale. M.T. was partially supported by the OTKA K-123914 project.

References

1. Y. Bao, X. Sun, and K. S. Trivedi. Adaptive software rejuvenation: degradation model and rejuvenation scheme. In *2003 International Conference on Dependable Systems and Networks, 2003. Proceedings.*, pp. 241–248 (June, 2003).
2. Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software rejuvenation: analysis, module and applications. In *Proceedings of the 25-th Fault Tolerant Computing Symposium (FTCS-25)*, pp. 381–390, (1995).
3. K. Vaidyanathan and K. Trivedi, A comprehensive model for software rejuvenation, *Dependable and Secure Computing, IEEE Transactions on.* **2**(2), 124–137 (April, 2005). ISSN 1545-5971.
4. S. Garg, A. Pfening, A. Puliafito, M. Telek, and K. Trivedi. Modeling and analysis of load and time dependent software rejuvenation policies. In *Proceedings 3-rd International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS3)*, pp. 35–39, (1996).
5. S. Garg, A. Puliafito, M. Telek, and T. Trivedi, Analysis of preventive maintenance in transactions based software systems, *Computers, IEEE Transactions on.* **47**(1), 96–107 (Jan, 1998).
6. J. Zheng, H. Okamura, L. Li, and T. Dohi, A comprehensive evaluation of software rejuvenation policies for transaction systems with markovian arrivals, *IEEE Transactions on Reliability.* **66**(4), 1157–1177 (Dec, 2017).
7. T. Dohi, K. Goseva-Popstojanova, and K. S. Trivedi. Analysis of software cost models with rejuvenation. In *Proceedings. Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE 2000)*, pp. 25–34, (2000).
8. W. Xie, Y. Hong, and K. Trivedi, Analysis of a two-level software rejuvenation policy, *Reliability Engineering & System Safety.* **87**(1), 13 – 22, (2005).
9. S. Garg, A. Puliafito, M. Telek, and K. Trivedi. Analysis of software rejuvenation using Markov regenerative stochastic Petri nets. In *Proceedings of the 6-th International Symposium on Software Reliability Engineering*, Toulouse, (1995).

10. G. Ning, J. Zhao, Y. Lou, J. Alonso, R. Matias, K. S. Trivedi, B. B. Yin, and K. Y. Cai, Optimization of two-granularity software rejuvenation policy based on the markov regenerative process, *IEEE Transactions on Reliability.* **65**(4), 1630–1646 (Dec, 2016). ISSN 0018-9529. doi: 10.1109/TR.2016.2570539.
11. A. Bobbio, S. Garg, M. Gribaudo, A. Horváth, M. Sereno, and M. Telek. Modeling software systems with rejuvenation, restoration and checkpointing through Fluid Stochastic Petri Nets. In *8-th International Conference on Petri Nets and Performance Models - PNPM99*, pp. 83–92. IEEE Computer Society, (1999).
12. A. Bobbio, S. Garg, M. Gribaudo, A. Horvth, M. Sereno, and M. Telek. Compositional fluid stochastic petri net model for operational software system performance. In *2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE WOSAR08)*, pp. 1–6 (Nov, 2008).
13. H. O. H. Fujio and T. Dohi, Fine-grained shock models to rejuvenate software systems, *IEICE Transactions Information & Systems.* **E-82**(1), 1–8, (1999).
14. A. Bobbio, M. Sereno, and C. Anglano, Fine grained software degradation models for optimal rejuvenation policies, *Performance Evaluation.* **46**, 45–62 (September, 2001).
15. D. Chen and K. Trivedi. Analysis of periodic preventive maintenance with general system failure distribution. In *Proc. Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 103–107, (2001).
16. D. R. Cox, *Renewal theory.* (Chapman & Hall, London, 1962).
17. K. Trivedi, *Probability & Statistics with Reliability, Queuing & Computer Science applications.* (Wiley, II Edition, 2002).
18. K. Trivedi and A. Bobbio, *Reliability and Availability Engineering: Modeling, Analysis, and Applications.* (Cambridge University Press, 2017).
19. J. D. Esary, A. W. Marshall, and F. Proschan, Shock models and wear processes, *The Annals of Probability.* **1**, 627–649, (1973).
20. D. Cox and H. Miller, *The theory of stochastic processes.* (Chapman and Hall, London, 1965).
21. V. G. Kulkarni, *Modeling and Analysis of Stochastic Systems.* (Chapman & Hall, 1995).
22. J. Lambert, *Computational Methods in Ordinary Differential Equations.* (John Wiley and Sons, New York, 1973).
23. A. Reibman and K. Trivedi, Numerical transient analysis of Markov models, *Computers and Operations Research.* **15**, 19–36, (1988).
24. M. Malhotra, J. K. Muppala, and K. S. Trivedi, Stiffness-tolerant methods for transient analysis of stiff Markov chains, *Journal of Microelectronics and Reliability.* **34**, 1825–1841, (1994).
25. W. Feller, *An Introduction to Probability Theory and its Applications - Vol. I and II.* (John Wiley & Sons, New York, 1968).
26. G. Gripenberg, S. O. Londen, and O. Staffans, *Volterra Integral and Functional Equations.* Encyclopedia of Mathematics and its Applications, (Cambridge University Press, 1990). doi: 10.1017/CBO9780511662805.
27. E. Cinlar, *Introduction to Stochastic Processes.* (Prentice-Hall, Englewood Cliffs, 1975).
28. J. Peterson, *Petri net theory and the modeling of systems.* (Prentice Hall, Englewood Cliffs, 1981).
29. T. Murata, Petri nets: properties, analysis and applications, *Proceedings of the IEEE.* **77**, 541–580, (1989).
30. A. Bobbio. System modelling with Petri nets. In eds. A. Colombo and A. S. de Bustamante, *System Reliability Assessment*, pp. 103–143. Kluwer Academic P.G., (1990).

31. G. Florin and S. Natkin, Les reseaux de Petri stochastiques, *Technique et Science Informatique*. **4**, 143–160, (1985).
32. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani, The effect of execution policies on the semantics and analysis of stochastic Petri nets, *IEEE Transactions on Software Engineering*. **SE-15**, 832–846, (1989).
33. M. Molloy, Performance analysis using stochastic Petri nets, *IEEE Transactions on Computers*. **C-31**, 913–917, (1982).
34. M. Ajmone Marsan, G. Balbo, and G. Conte, A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems, *ACM Transactions on Computer Systems*. **2**, 93–122, (1984).
35. M.Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte, Generalized stochastic Petri nets: a definition at the net level and its implications, *IEEE Transactions on Software Engineering*. **19**, 89–107, (1993).
36. G. Ciardo, J. Muppala, and K. Trivedi, On the solution of GSPN reward models, *Performance Evaluation*. **12**, 237–253, (1991).
37. G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. Trivedi. Automated generation and analysis of Markov reward models using stochastic reward nets. In eds. C. D. Meyer and R. J. Plemmons, *Linear Algebra, Markov Chains, and Queueing Models*, vol. 48, *The IMA Volumes in Mathematics and its Applications*, pp. 145–191. Springer New York, (1993). ISBN 978-1-4613-8353-6.
38. D. Wang, W. Xie, and K. S. Trivedi, Performability analysis of clustered systems with rejuvenation under varying workload, *Performance Evaluation*. **64**(3), 247 – 265, (2007).
39. F. Machida, D. S. Kim, and K. S. Trivedi, Modeling and analysis of software rejuvenation in a server virtualized system with live vm migration, *Performance Evaluation*. **70**(3), 212 – 230, (2013).
40. J. Xu, X. Li, Y. Zhong, and H. Zhang, Availability modeling and analysis of a single-server virtualized system with rejuvenation, *Journal of Software*. **9**, 129–139, (2014).
41. A. Bobbio and M. Telek, Non-exponential stochastic Petri nets: an overview of methods and techniques, *Computer Systems: Science & Engineering*. **13**(6), 339–351, (1998).
42. A. Bobbio, A. Puliafito, M. Telek, and K. Trivedi, Recent developments in non-Markovian stochastic Petri nets, *Journal of Systems Circuits and Computers*. **8**(1), 119–158 (Feb, 1998).
43. A. Bobbio and M. Telek. Computational restrictions for SPN with generally distributed transition times. In eds. D. H. K. Echtle and D. Powell, *First European Dependable Computing Conference (EDCC-1), Lecture Notes in Computer Science - LNCS 852*, pp. 131–148. Springer Verlag, (1994).
44. H. Choi, V. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. In ed. G. I. ans S.S. Lavenberg, *Proceedings International Conference PERFORMANCE'93*, pp. 339–356, (1993).
45. A. Puliafito, M. Scarpa, and K. Trivedi, Petri nets with K simultaneously enabled generally distributed timed transitions, *Performance Evaluation*. **32**(1), 1–34 (February, 1998).