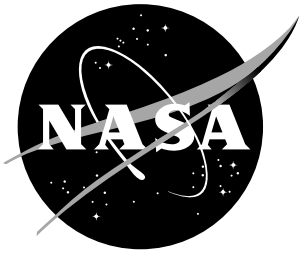
NASA/CR-2000-210296



Analysis of Phase-Type Stochastic Petri Nets With Discrete and Continuous Timing

Robert L. Jones
ASRC Aerospace Corporation, Greenbelt, Maryland

November 2000

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

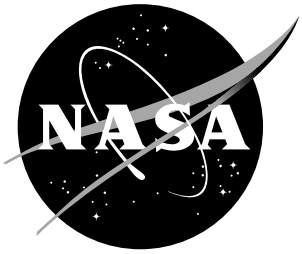
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR-2000-210296



Analysis of Phase-Type Stochastic Petri Nets With Discrete and Continuous Timing

Robert L. Jones
ASRC Aerospace Corporation, Greenbelt, Maryland

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-99124

November 2000

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The Petri net formalism has proven its usefulness in modeling discrete-state systems that move between states in discrete or continuous time and that may be characterized as sequential or concurrent, synchronous or asynchronous, deterministic or stochastic, or any combination for that matter. Through higher levels of abstraction, a Petri net (PN) permits a compact specification of the underlying mathematical model (usually a stochastic process) that is amenable to computer analysis. But applying PN modeling to arbitrarily complex systems requires the solution of difficult problems from a computational point of view.

A common problem is the computational complexity often required to solve stochastic PN (SPN) models with realistic assumptions about the logical and timed behavior, as opposed to simple behavior that restricts the applicability to toy models. SPNs are most easily solved when firing delays of transitions are either exponentially or geometrically distributed. Then, the underlying stochastic process is a continuous-time Markov chain in the former and a discrete-time Markov chain in the latter. Efficient solution techniques for such Markov chains are well known. However, such modeling assumptions may be unrealistic for many systems, leading to inaccurate results when adopted.

A more general approach is to allow the transition firing delays to have general distributions. In their full generality, such SPNs are classified as non-Markovian and specify generalized semi-Markov processes. Unfortunately, in the absence of any restrictions the study of such nets by analytical or numerical means is so computationally expensive that simulation becomes the only practical means to a solution. With some conveniently-chosen restrictions, however, the solution to the model can be made efficient in some cases while still lending itself to useful applications. The investigation of efficient exact and approximate solutions to such classes of non-Markovian PNs is the focus of our research.

Much of our research finds its foundation in the early work of Molloy [1], Bobbio and Cumani [2], and Marsan et al. [3, 4]: Molloy introducing execution policies, the combination of deterministic and discrete-time random behavior, and the expansion of phase-type firing delays at the state-space level, and Bobbio, Marsan, et al. doing the same for continuous time. Complete specifications of the semantics of a stochastic Petri net requires consideration about how (and in what order) enabled transitions are selected to fire and what happens to the remaining firing time of other enabled transitions when another transition fires. An

execution policy formally defines such semantics by specifying the policy used to select the enabled transition that fires and the way memory is kept of the past history of the net [4].

In [1], Molloy provided a comprehensive overview of the SPN semantics and behavior as a function of the execution policies. He also showed that when the probability distributions of SPN transitions are discrete phase-type, an otherwise non-Markovian underlying process can be transformed into a homogeneous discrete-time Markov chain defined over an expanded state space. Particularly relevant to this work is that Molloy showed how transitions with deterministic firing delays can be combined with geometric firing-delay transitions with the restriction that the deterministic delays are equal to the *basic step* of the geometric distributions. These ideas were brought up-to-date by Ciardo [5] while introducing the discrete deterministic and stochastic PN (DDSPN) formalism that allows firing delays with discrete phase-type distributions, all sharing a basic step, and having an underlying discrete-time Markov chain. The notion of a "basic step" is important to our proposed research as well. The basic step period, denoted by τ, is defined as the sojourn time in each state of the underlying discrete-time Markov chain.

In [3], Marsan and Chiola introduced the deterministic and stochastic PN (DSPN), and marks the first time deterministic behavior was integrated with continuous-time random behavior. Bobbio and Cumani in [2] showed how continuous-time, phase-type firing delays can be expanded at the state-space level to form a continuous-time Markov chain. This was discussed again by Marsan et al. in [4] while providing an extensive discussion of execution policies for generally distributed firing delays.

1.2 Objective

Our research involves the formal development of a new class of non-Markovian SPNs based on phase-type firing delays in both discrete and continuous time, present simultaneously in the *same* model. We build upon the extended SPN formalism, one that includes constructs that increase its modeling power, from a logical point of view, to that of a Turing machine as well as features that provide modeling conveniences. Such modern extensions include inhibitor arcs, transition priorities, transition-enabling guards, marking-dependent arc multiplicities, and marking-dependent execution policies.

When possible, efficient and exact solution algorithms will be developed with certain restrictions; otherwise, approximate solution algorithms will be investigated. In this way, we anticipate that this new SPN formalism may also prove to be useful in many modeling problems while still affording an efficient solution.

1.3 Organization and Assumptions

Relevant background material is provided in Chapter 2, which provides the foundation for our chosen approach. Topics include discrete- and continuous-time Markov chains, semi-Markov chains, semi-regenerative processes, generalized semi-Markov processes, characteristics specific to each, and known solution methods. These are the classes of underlying stochastic processes for popular SPN formalisms used today, and the classification of these formalisms is closely related to the stochastic process which they can specify. The generality and so-

lution complexity associated with these stochastic processes determine the modeling power and efficiency, and therefore, the SPN's applicability and practical usefulness. Therefore, by also discussing the complexity issues germane to the solution methods, the background chapter also serves to motivate the proposed approach towards our objective: developing a SPN formalism that lends itself to useful modeling applications *and* efficient numerical analysis.

Chapter 3 provides an outline of the proposed research. Preliminary research results are provided in Chapter 4, which includes the formalization of the new SPN class and analysis theory, culminating into an exact, stationary solution algorithm. Time-dependent analysis is shown to be difficult except for a special case, making a strong argument for approximate solutions, which is planned for later. The chapter ends with a comparative analysis of the new SPN formalism with other noteworthy extensions in terms of modeling power and solution complexity. The preliminary results are followed by the plan towards completing the research in Chapter 5.

1.4 Notation

For the definitions and methods that follow, we restrict ourselves to homogeneous (time invariant) models, and we assume that a race execution policy is employed; that is, the transition with the earliest firing time is selected to fire next. Nevertheless, a preselection execution policy can be modeled by our formalism since a mechanism to resolve "contemporary" events is still required. We assume that transition firing events are atomic (no time elapses) and always sequential even if the firing times are contemporary. From a modeling perspective, contemporary firings still occur at the same time. It is only that we choose to impose a sequential ordering policy so that new markings can be unambiguously determined. As for timing, we may sometimes allow the probability distribution functions associated with transition firings to be marking dependent.

Sets are denoted with calligraphic letters. Vectors and matrices (usually lower and upper case letters, respectively) are denoted with bold text and the corresponding elements are denoted with (usually subscripted) plain text. The notion of "state" for the models and stochastic processes presented herein is actually a vector, dimensioned on the set of natural numbers $\mathbb{N}$ and sometimes paired with supplementary information, also vectors dimensioned on $\mathbb{N}$ or the set of real numbers $\mathbb{R}$. But our model solutions take the form of probability distribution vectors that either provide the state-occupancy probabilities at certain times or at steady state, or provide cumulative probabilities of occupying states over intervals of time. In either case, each (vector) state $\boldsymbol{i} \in \mathbb{N}^n$ must be mapped to some index $i \in \mathbb{N}$ that is associated with the state's lexicographic position in the solution vector $\boldsymbol{p} = [p_i] \in \mathbb{R}^{|\mathcal{S}|}$ on the complete set of states $\mathcal{S}$, also known as the state space. When we refer to state "$\boldsymbol{i}$" in bold text, we mean its vector form, and when we refer to state "i" in plain text, we are referring to its lexicographic index, unless otherwise stated.

Chapter 2

Background

2.1 Petri Nets

A *Petri net*, such as the one pictured in Figure 2.1, is a directed bipartite graph described by the tuple $\mathbf{PN} = (\mathcal{P}, \mathcal{T}, \mathcal{A}, A^-, A^+, A^\circ, g, \prec, \boldsymbol{m}_0)$ with finite vertex sets $\mathcal{P}$ (*places*) and $\mathcal{T}$ (*transitions*) and a finite set of arcs (drawn as directed line segments), $\mathcal{A} \subseteq \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$. Places (drawn as circles) can contain an integer number of *tokens* (drawn as dots or denoted by a number inside the place). We denote the marking of the net by a row vector $\boldsymbol{m} \in \mathbb{N}^{|\mathcal{P}|}$ that contains as entries the number of tokens in each place. Hence, m_p denotes the number of tokens in place p of marking $\boldsymbol{m} = [\, m_1, m_2, \ldots, m_{|\mathcal{P}|} \,]$. The vector $\boldsymbol{m}_0$ denotes the initial marking.

Markings can be altered when enabling rules are satisfied at transitions (drawn as rectangles), permitting one or more transitions to *fire*, thereby removing tokens from input places and depositing them to output places according to the connecting arc multiplicities. *Arc multiplicities* are defined on arcs as either nonnegative integer constants or marking dependent functions that return a nonnegative integer; the semantics depend on the type of arc. For *input arcs*, the multiplicity, denoted by $A^-_{tp}(\boldsymbol{m})$, specifies the minimum number of tokens needed in place p before transition t can become enabled in marking $\boldsymbol{m}$; this number of tokens is then removed if the transition is indeed chosen to fire. Special input arcs, called *inhibitor arcs* (drawn as directed lines with a circle at the end instead of an arrow), have a complementary effect on the enabling of transitions. For inhibitor arcs, the multiplicity, denoted by $A^\circ_{tp}(\boldsymbol{m})$, is the minimum number of tokens needed in place p to disable transition t in marking $\boldsymbol{m}$. For *output arcs*, the multiplicity, denoted by $A^+_{tp}(\boldsymbol{m})$, specifies the number of tokens that will be deposited in place p when transition t fires in marking $\boldsymbol{m}$.

Guards, denoted by g and defined for transitions, are functions $\mathbb{N}^{|\mathcal{P}|} \rightarrow \{\, \mathit{true}, \mathit{false} \,\}$ that conveniently specify additional firing rules on transitions. Given some marking $\boldsymbol{m}$, $g_t(\boldsymbol{m})$ must return *true* to enable transition t.

The **PN** component $\prec \subset \mathcal{T} \times \mathcal{T}$ specifies an acyclic, preselection, priority relation, which can resolve conflicts between competing transitions attempting to fire.

Either inhibitor arcs, guards, priorities, or marking-dependent multiplicities alone increases the modeling power of a PN to that of a Turing machine (so we can represent any computational model), and hence are referred to as Turing extensions [6, 7]. Including all four Turing extensions merely provides additional modeling conveniences since the modeling

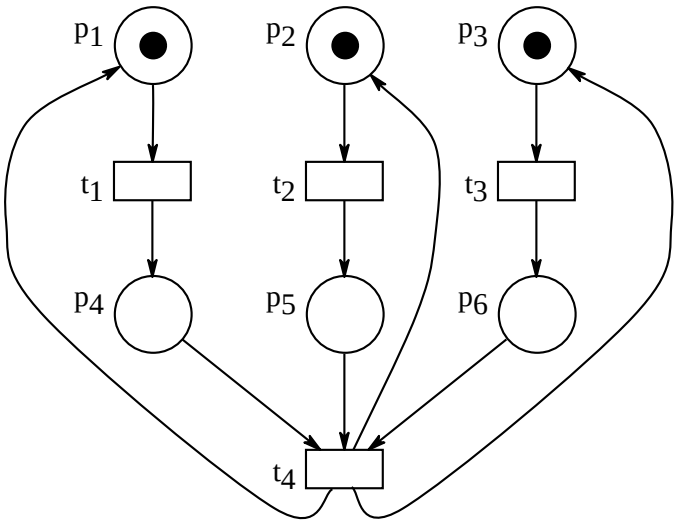


Figure 2.1: Example Petri net model.

power can no longer increase.

A transition $t \in \mathcal{F}(\boldsymbol{m})$, the *set of enabled transitions* in marking $\boldsymbol{m}$, if all of the following hold:

1. $g_t(\boldsymbol{m}) = true$
2. all of its input places p contain at least as many tokens as the corresponding input arc multiplicity $A^-_{tp}(\boldsymbol{m})$:

$$\forall p \in \mathcal{P},\, m_p \geq A^-_{tp}(\boldsymbol{m})$$

3. all of its inhibitor arc places p contain fewer tokens than the arc multiplicity $A^\circ_{tp}(\boldsymbol{m})$:

$$\forall p \in \mathcal{P},\, m_p < A^\circ_{tp}(\boldsymbol{m})$$

4. no other transition with higher priority $\prec$ is enabled:

$$\forall u \in \mathcal{T},\, u \not\prec t \text{ or } u \notin \mathcal{F}(\boldsymbol{m})$$

The firing of a transition is assumed to be atomic, consuming zero time. And, timing constraints aside, the PN defined above can evolve through markings originating from the initial marking by firing enabled transitions in any order. A transition $t \in \mathcal{F}(\boldsymbol{m}_0)$ can fire thereby changing the marking, $\boldsymbol{m}_0 \xrightarrow{t} \boldsymbol{m}_1$, where $\boldsymbol{m}_1$ is obtained by consuming tokens from input places and depositing tokens to output places according to the input and output arc multiplicities $A^-_{t\bullet}$ and $A^+_{t\bullet}$, respectively. By treating the $A^-_{t\bullet}$ and $A^+_{t\bullet}$ as vectors, we can write an equation for the next marking as

$$\begin{aligned}\boldsymbol{m}_1 &= \boldsymbol{m}_0 + A^+_{t\bullet}(\boldsymbol{m}_0) - A^-_{t\bullet}(\boldsymbol{m}_0) \\ &= \boldsymbol{m}_0 + \mathbf{1}_t\, \boldsymbol{A}(\boldsymbol{m}_0)\end{aligned}$$

where $\boldsymbol{A}(\boldsymbol{m}) = \boldsymbol{A}^{+}(\boldsymbol{m}) - \boldsymbol{A}^{-}(\boldsymbol{m})$ is called the *incidence matrix* and $\mathbf{1}_t$ is a unit vector with a 1 at the t^{th} position and 0 everywhere else. It is convenient to extend this next-marking computation to one that takes a sequence $s \in \mathcal{T}^*$ of transition firings as input, where $\mathcal{T}^*$ denotes the set of transition sequences obtained by concatenating zero or more transitions from $\mathcal{T}$. We do this by defining the *next-marking* function $M : \mathcal{T}^* \times \mathbb{N}^{|\mathcal{P}|} \to \mathbb{N}^{|\mathcal{P}|}$ to operate on a transition firing sequence $s = (\, t_1, t_2, \ldots, t_n \,) \in \mathcal{T}^n$ and a marking $\boldsymbol{m}$ and return a new marking. With $\epsilon = (\,)$ denoting the *null* sequence, the next marking function M is defined recursively as

$$\begin{aligned} M(\, \epsilon, \boldsymbol{m} \,) &= \boldsymbol{m}, \\ M(\, (t_1, t_2, \ldots, t_n), \boldsymbol{m} \,) &= M(\, (t_2, \ldots, t_n), \boldsymbol{m} + \mathbf{1}_{t_1} \boldsymbol{A}(\boldsymbol{m}) \,) \quad \text{if } t_1 \in \mathcal{F}(\boldsymbol{m}), \end{aligned}$$

and is undefined otherwise.

The PN behavior characterized by the set of markings reachable from the initial marking and the transition firings that cause the net to enter one marking from another can be represented as a directed graph with vertices corresponding to markings and arcs corresponding to the firing of transitions, completely constructed using M. Such a graph is called the *reachability graph*. The *reachability set*, $\mathcal{R}$, the set of reachability graph vertices, is the set of all markings reachable by a sequence of transition firings starting from the initial marking $\boldsymbol{m}_0$:

$$\mathcal{R} = \{\, \boldsymbol{m} : \exists\, s \in \mathcal{T}^*, \boldsymbol{m} = M(s,\, \boldsymbol{m}_0) \,\}$$

The reachability graph of the example PN model is portrayed in Figure 2.2 assuming for the moment that the net is untimed. The possible state space is subject to the number of tokens that can reside in each place $p_1, p_2, \ldots, p_6$ and the possible sequence of transition firings that move the net between markings starting from the initial marking $(m_1 m_2 \ldots m_6) = (111000)$. Without timing constraints, transitions t_1, t_2, and t_3 are concurrent with each other and each can fire asynchronously. However, synchronization is imposed after these three transitions fire before transition t_4 can become enabled and fire, returning the net to the initial marking. Because all transitions have a fair chance of firing, the reachability graph contains all possible markings and all possible transition firing sequences.

Petri nets as defined are useful in the study of many types of systems, with or without concurrency, with or without synchronization. But without the inclusion of time, we are limited to the qualitative analysis of properties like liveness, deadlock, boundedness, and invariants [8]. To broaden the applicability of PNs, the notion of time has been incorporated into the Petri net by various researchers with various generalities by requiring that an enabled transition *delay* some amount of time before firing. Ultimately, the specification captured by the Petri net must be transformed into an (underlying) mathematical model that can be solved to obtain quantitative measures. When the *firing delays* are specified as random variables (or even if deterministic but contemporary transition firings are allowed), the underlying model is a stochastic process, the solution of which governs the overall complexity of the model solution. As one would expect, the tractability of the solution decreases as the generality of the model increases.

Extended Petri nets with the most convenient stochastic models include those with geometrically distributed (Geom) firing delays [1] having an underlying discrete-time Markov

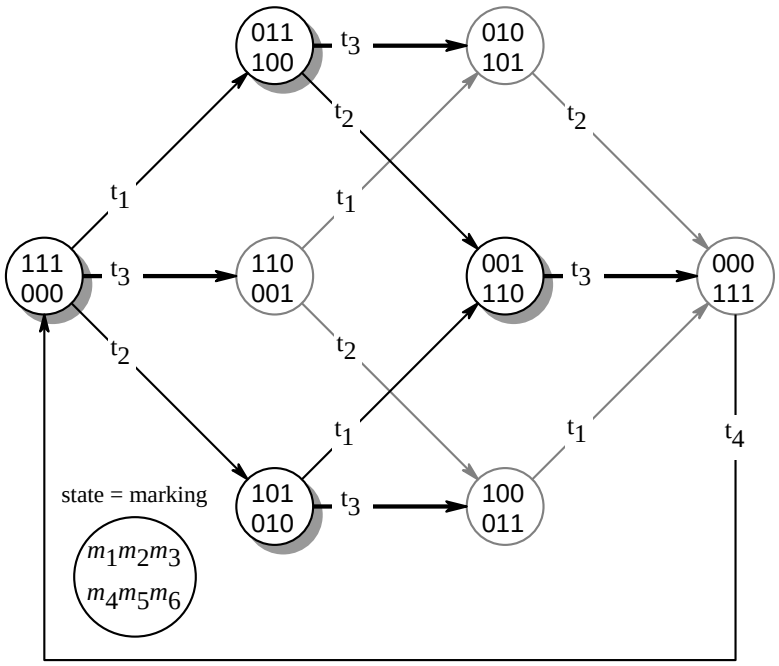


Figure 2.2: Reachability graph of example PN model.

chain and those with exponentially distributed (Expo) firing delays [9] having an underlying continuous-time Markov chain. These Markovian extensions have proven useful in the years for studying discrete-event systems with random behavior. But with the usefulness of these models to more complex and realistic systems in question, more recent extensions have tried to incorporate non-Markovian behavior. Noteworthy extensions and associated underlying processes are the phase-type SPN with an underlying, expanded continuous-time Markov chain [2], the extended SPN (ESPN) [10] with an underlying semi-Markov chain, the deterministic and stochastic PN (DSPN) [3] and the Markov regenerative SPN (MRSPN) [11, 12, 13] with an underlying semi-regenerative process, and the discrete deterministic and stochastic PN (DDSPN) [5] with an underlying, expanded discrete-time Markov chain. However, the complexity of solving more general underlying stochastic processes oftentimes limits in practice their usefulness to problems with small dimensions.

Consider now the SPN where the states reachable from the initial state are subject to the possible state transitions under the constraints imposed by the *timed* execution. Sometimes the reachability graph of the PN is isomorphic to the underlying stochastic process that models its timed execution. It is important to know when this property holds because it determines when and how the reachability graph of markings and the underlying stochastic (marking) process can be constructed. When they are isomorphic, there is a one-to-one correspondence between states in the reachability set $\mathcal{R}$ and the state space of the stochastic process, denoted by $\mathcal{S}$. This is the case for SPNs with Geom or Expo firing delays, which have underlying Markov chains. In such cases, it may be convenient to build the reachability graph first and then construct the (matrix) equations for the stochastic process from it second [14]. Also when they are isomorphic, one may wish to perform the qualitative analysis by operating

on the incidence matrix or reachability graph separately from studying the stochastic marking process [8]. If we know that isomorphism is not guaranteed or does not exist, then $\mathcal{S}$ may be a strict subset of $\mathcal{R}$. Consequently, the results from reasoning about the reachability set $\mathcal{R}$ independently from the timing specification is less meaningful, and possibly misleading.

The semantics of a SPN model also depends on the chosen execution policy. The execution policy specifies two things: 1) how the next transition to fire is selected among those enabled and 2) how memory is kept regarding the *remaining firing time* (RFT), or similarly the "age", of transitions with non-memoryless distributions.

In regard to selecting transitions to fire, the policy most frequently assumed is the *race* policy where the transition selected to fire is the one with the minimum remaining firing time over all enabled transitions. But it is also possible to perform the selection on the basis of additional specifications that do not depend on the duration of the activities associated with the transitions that are enabled. One such policy is called *preselection* where transitions are selected to fire among the enabled set according to *a priori* information, independent of the firing delay distributions. For example, a probability mass function (pmf) can be defined over the set of enabled transitions in a given marking, and used to choose the transition that fires next. This preselection can be done globally, defined for all markings of the net, which implies serialization of all activities. Alternatively, the preselection can be done locally, defined over transition groupings, not necessarily disjoint, within which a preselection policy is applied. Local preselection can be done in concert with the race policy as follows. In a marking that enables transitions belonging to these groups, the next transition to fire is identified by selecting first, with time independent criteria, an enabled transition (if one exists) from each of the groups, and then by choosing among the preselected transitions the one whose firing delay is minimal [4].

We allow a combination of race policy with pre- and post-selection in terms of priorities. Under the race policy, we select among the enabled transition whose sampled firing delay is (statistically) the shortest. This policy provides very useful models of systems that exhibit concurrency where multiple activities compete such that the first to finish determines the change in system state. We assume that *immediate transitions*, those that fire in zero time, have a higher priority of firing over timed transitions. Thus, we implicitly employ a preselection policy between timed and immediate transitions. Other execution policies like pre- or post-selection among timed transitions are discussed at length in [4] and may be required as well to resolve confusion, which is discussed later.

In regard to how memory is kept about the age, or RFT, of transitions, the memory policy is only meaningful for transitions with non-memoryless firing-delay distributions since these are the only transition that can "age". Transitions that have memoryless distributions (the Expo in continuous time and the Geom in discrete time, the Const(0) is a special case of the Geom) are not affected since their firing delays can equivalently be sampled after every transition firing. Memory policies are as important to the semantics of non-Markovian SPNs as the net topology and the policy used to select the next transition to fire. There are three policies that are most-frequently used in modeling applications, namely, *resampling*, *enabling memory*, and *age memory* [1, 4]. The chosen memory policy need not be global; a different policy can be associated with different transitions and be marking dependent [15]. A resampling policy requires that transitions obtain a new firing delay, sampled from the respective distribution functions, after some transition fires, including itself. Since each

transition firing causes a state change, a resampling policy for all transitions results in a semi-Markov process, which enjoys an absence of memory after each state change. Such a policy (with race execution) is useful for modeling competing activities (modeled of course by transitions where the amount of work performed is represented by the firing delay) in which the next state of the system is decided by the activity that finishes first. Consequently, the work performed by the losing transitions is lost. Alternatively, an enabling-memory policy causes the firing delays to be resampled only when a transition becomes enabled again after being disabled. The enabling memory policy is useful in modeling activities where work is performed until either completion or termination by another event causing the work performed to be lost. Finally, the age-memory policy causes the firing delay to be resampled only after the transition itself fires, even though it may have been disabled and re-enabled many times. Thus, the work performed by age-memory activities is never lost, which makes such transitions useful in modeling tasks that can be preempted and then resumed at the same point.

2.2 Markov Models

When the SPN firing delays are defined by random variables, the SPN provides a compact specification of an underlying stochastic process. Hereafter, we will denote this stochastic process as $\{ X(\theta) : \theta \geq 0 \}$: a collection of random variables defined over the same probability space, indexed by a time parameter θ, and taking on values in a state space $\mathcal{S}$, which may be finite or infinite as well as continuous or discrete. A stochastic process that has a discrete state space is called a *chain*. The index (time) parameter θ can also be continuous or discrete. Hereafter, we will denote discrete-time processes by $\{ X_\theta : \theta \in \mathbb{N} \}$.

A stochastic process (chain) $\{X(\theta) : \theta \geq 0\}$ with the property

$$\Pr\{ X(\upsilon + \theta) = j \mid X(\upsilon) = i, X(\zeta) = x(\zeta), 0 \leq \zeta < \upsilon \} =$$
$$\Pr\{ X(\upsilon + \theta) = j \mid X(\upsilon) = i \}$$

$\forall\, i, j, x(\zeta) \in \mathcal{S}$, $\theta \geq 0$, $\upsilon \geq 0$, is called a *Markov process* (*chain*) and the property is referred to as the "memoryless" or *Markov property*. Thus, the determination of the state the process will transition to next depends solely on the current state and not the past history of the process. When the value of that conditional probability is independent of υ, the process is said to be *homogeneous* or time invariant; this is one of our assumptions. Also, because of the discrete nature of the PN markings, we will mostly concern ourselves with discrete-state processes or chains. The acronyms DTMC and CTMC are used for discrete-time and continuous-time Markov chains, respectively. There will be occasions, presented later, when the "state" is supplemented with additional, continuous-valued information for modeling purposes thereby giving rise to a continuous-state process.

The amount of time that a process spends in a state is referred to as its *sojourn time*. Clearly, if the evolution of a Markov process depends only on the current state, it should not matter how long the process remains in the current state before making a transition. Thus, the sojourn time is geometrically distributed (Geom) for DTMCs and exponentially distributed (Expo) for CTMCs. This is expected since the Geom and Expo random variables are the only ones that exhibit the "memoryless property" for discrete and continuous random

variables, respectively. Without limiting ourselves to only Expo and Geom firing delays, the efficient analysis of SPNs using DTMCs and CTMCs will be the main focus of our research.

It follows from the memoryless property of Markov chains, letting

$$P_{ij}(\theta) = \Pr\{\, X(\theta) = j \mid X(0) = i \,\},$$

that

$$P_{ij}(\theta + \upsilon) = \sum_{k \in \mathcal{S}} P_{ik}(\theta) P_{kj}(\upsilon).$$

This is known as the Chapman-Kolmogorov equation for Markov chains and is key in formulating the analytical solutions of Markovian models.

2.2.1 Discrete-Time Markov Chains

Consider a DTMC defined by its transition matrix $\boldsymbol{\Pi} = [\Pi_{ij}]$, $i, j \in \mathcal{S}$, where

$$\Pi_{ij} = \Pr\{\, X_1 = j \mid X_0 = i \,\}$$

gives the conditional transition probabilities between states in one step or *jump*. It is often the case with DTMC models that the time spent in each state is of no concern, only the states that can be occupied after a given number of "jumps" is of interest. But we can also imagine that the DTMC remains in each state a fixed amount of time τ, referred to as its *basic step* time.

A fundamental property of DTMCs is that the Chapman-Kolmogorov equation takes the form

$$P_{ij}(\theta) = \left[\boldsymbol{\Pi}^{\theta}\right]_{ij}$$

where $\theta \in \mathbb{N}$. However, we do not have to, nor would we want to, compute the matrix $\boldsymbol{\Pi}^{\theta}$. Instead, we can compute the unconditional probability vector

$$\boldsymbol{x}^{(\theta)} = [\Pr\{\, X_\theta = i \,\}] = \boldsymbol{x}^{(0)} \boldsymbol{\Pi}^{\theta}$$

iteratively using the recursive relation

$$\boldsymbol{x}^{(\theta)} = \boldsymbol{x}^{(\theta-1)} \boldsymbol{\Pi}$$

known as the *power method* where $\boldsymbol{x}^{(0)} = [\Pr\{\, X_0 = i \,\}] \in \mathbb{R}^{|\mathcal{S}|}$ is given by the initial probability distribution.

We can also compute the cumulative probability vector $\boldsymbol{y}^{(\theta)} = \left[y_j^{(\theta)}\right] = \int_0^{\theta} \boldsymbol{x}^{(\lfloor u \rfloor)}\, du$, defined as

$$y_j^{(\theta)} = \mathrm{E}[\,\text{number of visits to } j \in \mathcal{S} \text{ until time } \theta \mid X_0 = i\,]$$

with an *extended power method*:

$$\begin{aligned} \boldsymbol{y}^{(n)} &= \boldsymbol{y}^{(n-1)} + \boldsymbol{x}^{(n-1)} \\ \boldsymbol{x}^{(n)} &= \boldsymbol{x}^{(n-1)} \boldsymbol{\Pi} \end{aligned}$$

for $n \leftarrow 1$ to θ with initial condition $\boldsymbol{y}^{(0)} = \mathbf{0}$ (the vector of all zeros).

If the DTMC contains transient states that lead to strongly-connected recurrent states (including absorbing states), then we can partition the state space into $\mathcal{S}_T$ and $\mathcal{S}_R$, the set of transient and recurrent states, respectively, such that $\mathcal{S} = \mathcal{S}_T \cup \mathcal{S}_R$. Then, by defining a new matrix $\tilde{\boldsymbol{\Pi}}$ by restricting $\boldsymbol{\Pi}$ to states in $\mathcal{S}_T$ only, we can compute the cumulative probability vector $\tilde{\boldsymbol{y}} = [\tilde{y}_j] \in \mathbb{R}^{|\mathcal{S}_T|}$ defined as

$$\tilde{y}_j = \lim_{\theta \to \infty} y_j^{(\theta)} = \mathrm{E}[\,\text{number of visits to } j \in \mathcal{S}_T \text{ until absorption} \mid X_0 = i\,]$$

with the same extended power method except that matrix $\tilde{\boldsymbol{\Pi}}$ is used instead of $\boldsymbol{\Pi}$, $\tilde{\boldsymbol{x}} \in \mathbb{R}^{|\mathcal{S}_T|}$ is used, and we stop when probability mass remaining in $\mathcal{S}_T$, determined from the vector norm $\| \tilde{\boldsymbol{x}} \|_1$, becomes small enough.

The power method can also be used to compute the stationary or steady-state solution $\boldsymbol{x} = [x_i]$, $i \in \mathcal{S}$, of DTMCs where $x_i = \lim_{\theta \to \infty} \Pr\{\, X_\theta = i \,\}$, by iterating long enough for the sequence $\left\{ x^{(n)} \right\}_{n=0}^{\infty}$ to converge to $\boldsymbol{x}$.

For the limiting measures where $\theta \to \infty$, convergence utilizing the power method may take a long time, making for a poor method in practice. Alternatively, we can observe that a stationary solution $\boldsymbol{x}$, if it exists, would satisfy the equation $\boldsymbol{x}\boldsymbol{\Pi} = \boldsymbol{x}$. This is just the case for ergodic DTMCs (those that are irreducible, aperiodic, and positive recurrent). Fortunately, ergodic DTMCs have a *unique* stationary solution that satisfies the system of equations

$$\boldsymbol{x}\boldsymbol{\Pi} = \boldsymbol{x} \quad \text{subject to} \quad \sum_{i \in \mathcal{S}} x_i = 1. \tag{2.1}$$

Of course, we could utilize direct methods like standard Gaussian elimination or LU decomposition to solve for $\boldsymbol{x}$. But because the coefficient matrix based on $\boldsymbol{\Pi}$ would be modified and susceptible to fill-in and because $\boldsymbol{\Pi}$ is typically very large for realistic models, direct methods are rarely used in practice. Even though the DTMC specified by $\boldsymbol{\Pi}$ (and most any stochastic model for that matter) is *large*, it is at least *sparse* in general. So iterative methods like Gauss-Seidel and successive overrelaxation (SOR), which have much faster convergence than the iterative power method, are usually employed to compute $\boldsymbol{x}$. These methods do not modify the iteration matrix based on $\boldsymbol{\Pi}$ and if sparse matrix storage is used, the time complexity is $O(N\eta)$ where N is the number of iterations needed for convergence and η is the number of nonzero entries in matrix $\boldsymbol{\Pi}$. Unfortunately, N may be unbounded since these iterative methods do not guarantee convergence for all initial guesses for $\boldsymbol{x}$. This is because $\boldsymbol{\Pi}$ is a stochastic matrix (each row sum is one), which makes the spectral radius (the largest eigenvalue) equal to one. Iterative methods have guaranteed convergence for any initial guess only when the spectral radius is less than one [16].

With these iterative methods, we can also solve the following system of linear equations for the cumulative probability vector $\tilde{\boldsymbol{y}}$ when $\mathcal{S}$ contains transient states:

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{x}}^{(0)} + \tilde{\boldsymbol{y}}\tilde{\boldsymbol{\Pi}}$$

or equivalently

$$\tilde{\boldsymbol{y}}(\mathbf{I} - \tilde{\boldsymbol{\Pi}}) = \tilde{\boldsymbol{x}}^{(0)}. \tag{2.2}$$

While the use of iterative methods to solve Equation 2.1 may not converge for all initial guesses, convergence is instead guaranteed for Equation 2.2, since $(\mathbf{I} - \tilde{\boldsymbol{\Pi}})$ is an M-matrix, (nonsingular, elements are less than or equal to zero, and having a nonnegative inverse [16]), which always has a spectral radius less than one. Therefore, solving Equation 2.2 with iterative methods has guaranteed convergence, regardless of the initial guess [17].

2.2.2 Continuous-Time Markov Chains

Consider now a CTMC, where for all continuous points in time, state transitions can occur and the process is memoryless. Let the rate at which the process transitions from state i to state j be denoted by $\lambda_{ij} \in \mathbb{R}^+$, $i, j \in \mathcal{S}$. The sojourn time in each state i is exponentially distributed, so by letting $\lambda_i = \sum_{j \in \mathcal{S}} \lambda_{ij}$, we can obtain the expected sojourn time from

$$\mathrm{E}[\,\text{sojourn time in state } i\,] = \lambda_i^{-1}.$$

The interpretation of the rates is such that

$$\lim_{\theta \to 0} \frac{P_{ij}(\theta)}{\theta} = \lambda_{ij}, \quad i \neq j \tag{2.3}$$

$$\lim_{\theta \to 0} \frac{1 - P_{ii}(\theta)}{\theta} = \lambda_i. \tag{2.4}$$

By observing the CTMC just after each state transition, we can construct a DTMC consisting of the possible sequence of states the process can move between over time θ. If only limiting measures where $\theta \to \infty$ are sought then we can also compute the probability of transitioning between state i and state j in the DTMC from the ratio λ_{ij}/λ_i since the sojourn times are exponentially distributed. Let $\mathrm{diag}(\lambda_{ii})$ be matrix with λ_i along the diagonal $\forall i \in \mathcal{S}$ and zero elsewhere. Then the embedded DTMC matrix $\boldsymbol{\Pi} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ constructed from the CTMC is defined as

$$\boldsymbol{\Pi} = \mathrm{diag}(\lambda_i)^{-1}[\lambda_{ij}]_{i \neq j} \tag{2.5}$$

Constructing a DTMC by observing a stochastic process (a CTMC in this case) at times when the Markov property holds is called *embedding*. Then steady-state, state-occupancy measures such as the stationary distribution or time-to-absorption (TTA) can be computed from the embedded DTMC (EMC) using Equations 2.1 and 2.2, respectively. But, since the quantities $\boldsymbol{x}$ and $\boldsymbol{y}$ from Equations 2.1 and 2.2 for the EMC are interpreted as or are based on the "number of visits" to states, we must convert these measures back to the original CTMC by appropriately scaling them. The needed "scaling factors" come from the knowledge that, with each visit, the expected sojourn time in each state i is just λ_i^{-1}. For example, the CTMC stationary distribution $\boldsymbol{p} = [p_i]$, $p_i = \lim_{\theta \to \infty} \Pr\{\, X(\theta) = i \,\}$, can be computed by using the embedded DTMC matrix (2.5) and then computing

$$\boldsymbol{x}\boldsymbol{\Pi} = \boldsymbol{x} \quad \text{subject to} \quad \sum_{i \in \mathcal{S}} x_i = 1, \qquad \tilde{x}_i = x_i\, \lambda_i^{-1}, \qquad p_j = \frac{\tilde{x}_j}{\sum\limits_{k \in \mathcal{S}} \tilde{x}_k}$$

where the last step is needed to re-normalize the distribution so that it sums to one once again.

Embedding a CTMC is applicable for steady-state solutions, not time-dependent solutions. With simple embedding, the CTMC is observed only at times when state transitions occur. Consequently, information concerning how long the process sojourns in states is lost making the EMC time-dependent solutions useless. For time-dependent analysis, we must once again make use of the Chapman-Kolmogorov equation. For steady-state analysis, embedding will once again be useful for more complicated stochastic process than CTMCs, as discussed in more detail in later sections.

By manipulating the Chapman-Kolmogorov equation:

$$\begin{aligned} P_{ij}(\theta+\upsilon) &= \sum_{k\in\mathcal{S}} P_{ik}(\theta)P_{kj}(\upsilon) \\ P_{ij}(\theta+\upsilon) - P_{ij}(\theta) &= \sum_{k\in\mathcal{S}} P_{ik}(\theta)P_{kj}(\upsilon) - P_{ij}(\theta) \\ P_{ij}(\theta+\upsilon) - P_{ij}(\theta) &= \sum_{k\neq i} P_{ik}(\theta)P_{kj}(\upsilon) - (1-P_{jj}(\upsilon))\,P_{ij}(\theta) \end{aligned}$$

then dividing by υ and taking the limit as $\upsilon \to 0$:

$$\lim_{\upsilon\to 0} \frac{P_{ij}(\theta+\upsilon) - P_{ij}(\theta)}{\upsilon} = \lim_{\upsilon\to 0} \left\{ \frac{\sum_{k\neq i} P_{ik}(\theta)P_{kj}(\upsilon) - (1-P_{jj}(\upsilon))\,P_{ij}(\theta)}{\upsilon} \right\}$$

and finally substituting 2.3 and 2.4, we get what is known as *Kolmogorov's forward equation*:

$$\frac{d}{d\theta}P_{ij}(\theta) = \sum_{k\neq i} P_{ik}(\theta)\lambda_{kj} - P_{ij}(\theta)\lambda_j. \tag{2.6}$$

The interchange of the summation and the limit, needed to obtain Equation 2.6, is not always justified, but it does hold for most models including those with finite state spaces, as is the case here [18]. Similarly, we can derive the *Kolmogorov's backward equation*,

$$\frac{d}{d\theta}P_{ij}(\theta) = \sum_{k\neq i} \lambda_{ik}P_{kj}(\theta) - \lambda_i P_{ij}(\theta), \tag{2.7}$$

by looking backwards in time from a given state.

By defining what is called the *infinitesimal generator matrix*, $\boldsymbol{Q} = [Q_{ij}] \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$, as

$$\boldsymbol{Q} = [\lambda_{ij}]_{i\neq j} - \mathrm{diag}(\lambda_i)$$

and letting $\boldsymbol{P}(\theta) = [P_{ij}(\theta)] \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$, Kolmogorov's forward and backward equations can be rewritten in matrix form as

$$\frac{d}{d\theta}\boldsymbol{P}(\theta) = \boldsymbol{P}(\theta)\boldsymbol{Q} \tag{2.8}$$

and

$$\frac{d}{d\theta}\boldsymbol{P}(\theta) = \boldsymbol{Q}\boldsymbol{P}(\theta) \tag{2.9}$$

respectively. These differential equations can sometimes be solved with conventional, direct or indirect means (such as Runge-Kutta or Laplace transforms) but this restricts the usefulness of CTMC models to small problems. Instead, we could make use of the known solution

$$\boldsymbol{P}(\theta) = e^{\boldsymbol{Q}\theta},$$

which is a matrix exponential computed from

$$e^{\boldsymbol{Q}\theta} = \sum_{n=0}^{\infty} \frac{(\boldsymbol{Q}\theta)^n}{n!}. \tag{2.10}$$

However, the matrix exponential method is susceptible to subtractive cancellation errors due to the positive and negative entries in $\boldsymbol{Q}$, which makes the method unstable.

In practice, the time-dependent solution of the CTMC is usually computed using Jensen's method, also known as *uniformization*. The basic idea behind uniformization is to perform time-dependent analysis on a DTMC constructed from the CTMC in a way similar to "embedding" except that all states are forced to have the same expected sojourn time by imposing selfloops on states where necessary. By uniformizing the CTMC, the CTMC is observed at times of state transitions, when they occur naturally, and at more frequent times, when self-transitions occur. Hence, information is retained about how long the CTMC occupies each state, unlike the embedding method.

The basic uniformization algorithm defines a DTMC matrix, $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$, as

$$\boldsymbol{A} = q^{-1}\boldsymbol{Q} + \mathbf{I} \tag{2.11}$$

where $q \geq \max_i Q_{ii}$ is chosen as the "sampling" rate, at least as large as the maximum outgoing rate of all CTMC states. The sampling rate q is normally chosen to be slightly *larger* than the maximum outgoing rate to ensure that the DTMC will not be periodic. Also, all $\boldsymbol{A}$ entries are non-negative. So, substituting

$$\boldsymbol{Q} = (\boldsymbol{A} - \mathbf{I})\, q,$$

derived from 2.11, into the matrix exponential 2.10 provides an efficient and much more stable computation of $\boldsymbol{P}(\theta)$, for a given initial probability distribution $\boldsymbol{x}^{(0)} = [\Pr\{\, X_0 = i \,\}] \in \mathbb{R}^{|\mathcal{S}|}$, that makes use of the Poisson random variable:

$$\begin{aligned}
\boldsymbol{x}^{(0)}\boldsymbol{P}(\theta) &= \boldsymbol{x}^{(0)} e^{\boldsymbol{Q}\theta} \\
&= \boldsymbol{x}^{(0)} e^{(\boldsymbol{A}-\mathbf{I})q\theta} \\
&= \boldsymbol{x}^{(0)} e^{\boldsymbol{A}q\theta} e^{-\mathbf{I}q\theta} \\
&= \boldsymbol{x}^{(0)} \sum_{n=0}^{\infty} \frac{\boldsymbol{A}^n (q\theta)^n}{n!} \cdot \sum_{n=0}^{\infty} \frac{\boldsymbol{I}^n (-q\theta)^n}{n!} \\
&= \boldsymbol{x}^{(0)} \sum_{n=0}^{\infty} \frac{\boldsymbol{A}^n (q\theta)^n}{n!} \cdot \sum_{n=0}^{\infty} \frac{(-q\theta)^n}{n!} \\
&= \boldsymbol{x}^{(0)} \sum_{n=0}^{\infty} \frac{(q\theta)^n e^{-q\theta}}{n!} \boldsymbol{A}^n.
\end{aligned}$$

The summation can be easily computed with

$$\sum_{n=0}^{\infty} \boldsymbol{x}^{(n)} \text{Poiss}(n, q\theta) \tag{2.12}$$

where

$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^{(n-1)} \boldsymbol{A}$$

and

$$\text{Poiss}(n, q\theta) = \text{Poiss}(n-1, q\theta)\frac{q\theta}{n}, \qquad \text{Poiss}(0, q\theta) = e^{-q\theta}.$$

So we see that uniformization is another version of the power method, only extended for suitability in studying CTMCs. Essentially, uniformization subordinates the uniformized process with a Poisson birth process. This has a nice interpretation. The Poisson process determines the probability that the uniformized process (the DTMC) can make n jumps within fixed time θ. Given n jumps, $\boldsymbol{A}^n$ determines the set of states that can be occupied conditioned on the initial state. By doing this for all possible n (and summing the probabilities), we can uncondition on n and obtain our desired solution for the original CTMC process.

In practice, computing $\boldsymbol{x}^{(0)} e^{\boldsymbol{Q}\theta}$ requires that we truncate the infinite series 2.12 and sum the remaining terms in a numerically stable way. To do this, we employ the Fox-Glynn algorithm [19], which defines left and right truncation points, L_1 and R_1, respectively, so that

$$\boldsymbol{x}^{(0)} e^{\boldsymbol{Q}\theta} \approx \sum_{n=L_1}^{R_1} \boldsymbol{x}^{(n)} \text{Poiss}(n, q\theta)$$

and the error is bounded by 10^{-d} (d digits of precision) if

$$L_1 = \max_{k \in \mathbb{N}} \left\{ \sum_{n=0}^{k} \text{Poiss}(n, q\theta) \leq \frac{10^{-d}}{2} \right\}$$
$$R_1 = \min_{k \in \mathbb{N}} \left\{ 1 - \sum_{n=L_1}^{k} \text{Poiss}(n, q\theta) \leq 10^{-d} \right\}.$$

Note that although the Poisson probabilities are only computed in the range $L_1 \leq n \leq R_1$, the vector-matrix multiplications must be done for $0 \leq n \leq R_1$.

Just as in the power method for DTMCs, we can compute the same $\boldsymbol{y}^{(\theta)}$ and $\tilde{\boldsymbol{y}}$, defined in the previous section as

$$\boldsymbol{y}^{(\theta)} = \int_0^{\theta} \boldsymbol{x}^{(u)}\, du = \boldsymbol{x}^{(0)} \int_0^{\theta} e^{\boldsymbol{Q}u}\, du$$
$$\tilde{\boldsymbol{y}} = \lim_{\theta \to \infty} \boldsymbol{y}^{(\theta)}$$

at the same time $\boldsymbol{x}^{(0)}e^{\boldsymbol{Q}\theta}$ is computed [20]. Substitution of the uniformization computation for $e^{\boldsymbol{Q}u}$ yields

$$\boldsymbol{x}^{(0)} \int_0^\theta e^{\boldsymbol{Q}u}\, du = \int_0^\theta \left(\sum_{n=0}^{\infty} \boldsymbol{x}(n) \cdot \text{Poiss}(n, qu) \right) du$$

and after factoring the summation series over n, we have

$$\sum_{n=0}^{\infty} \boldsymbol{x}^{(n)} \int_0^\theta \text{Poiss}(n, qu)\, du,$$

which can be written equivalently as

$$\frac{1}{q} \sum_{n=0}^{\infty} \boldsymbol{x}^{(n)} \left(1 - \sum_{\ell=0}^{n} \text{Poiss}(\ell, q\theta) \right).$$

using integration by parts. Although we must begin at $n = L_2 = 0$ here, a right truncation point R_2 can be found with bounded error. Since the total sojourn in all states over an interval $[0, \theta]$ must be θ, so that

$$\left\| \boldsymbol{x}^{(0)} \int_0^\theta e^{\boldsymbol{Q}u}\, du \right\|_1 = \theta,$$

we can stop when the difference between θ and $(1 - \sum_{\ell=0}^{n} \text{Poiss}(\ell, q\theta))$ becomes small. Thus, we have the right truncation point

$$R_2 = \min_{k \in \mathbb{N}} \left\{ \theta - \frac{1}{q} \sum_{n=0}^{k} \left(1 - \sum_{\ell=0}^{n} \text{Poiss}(\ell, q\theta) \right) \right\} \leq 10^{-d}$$

When computing both $\boldsymbol{x}^{(0)} \int_0^\theta e^{\boldsymbol{Q}u}\, du$ and $\boldsymbol{x}^{(0)}e^{\boldsymbol{Q}\theta}$, the smallest left truncation point $L = 0$ and largest right truncation point $R = \max(R_1, R_2)$ are chosen.

In case the CTMC with generator $\boldsymbol{Q}$ is ergodic, we can also test for stationary conditions to check whether θ is large enough for the DTMC to have reached steady state. Detecting stationary conditions that occur before the right truncation is reached, and halting, can result in significant performance gains.

But if stationary solutions are sought then there is a better way just as in the DTMC case. When stationary or steady-state equilibrium is reached, the change in probability mass between states becomes zero. So we can set the derivative in Equation 2.8 to zero and obtain global balance equations for the CTMC:

$$\boldsymbol{x}\boldsymbol{Q} = \boldsymbol{0} \tag{2.13}$$

where $\boldsymbol{x}$ satisfies the stationary solution $\lim_{\theta \to \infty} \boldsymbol{x}^{(\theta)}$ independent of the initial probability distribution when the CTMC is ergodic. The same argument used in the previous section against employing direct methods apply here as well; iterative methods are better even though convergence is not guaranteed for all initial guesses.

With iterative methods, we can also solve the following system of linear equations for the cumulative probability vector $\tilde{\boldsymbol{y}} = \lim_{\theta\to\infty} \boldsymbol{y}^{(\theta)}$:

$$\tilde{\boldsymbol{y}}\tilde{\boldsymbol{Q}} = -\tilde{\boldsymbol{x}}^{(0)}. \tag{2.14}$$

Like in Equation 2.2 for the DTMC case, the use of iterative methods for Equation 2.2 enjoys guaranteed convergence since $-\tilde{\boldsymbol{Q}}$ is also an M-matrix. Because the use of Equations 2.13 and 2.14 has the same complexity as Equations 2.1 and 2.2, respectively, steady-state solutions are computed directly from the CTMC in practice.

Because the uniformization algorithm is integral to the solution algorithms developed later, we provide it here as Algorithm 2.2.1, without steady-state detection. The algorithm computes the transient probability vector $\boldsymbol{\pi}^{(\theta)} \in \mathbb{R}^{|\mathcal{S}_i|}$ on the CTMC state space $\mathcal{S}_i$ originating from the initial state i where $\pi_j = \Pr\{\, X(\theta) = j \mid X(0) = i \,\}$ and the cumulative probability vector $\boldsymbol{\sigma} = \int_0^\theta \boldsymbol{\pi}^{(u)}\, du$. The algorithm assumes that $\boldsymbol{\pi}$ contains the initial probability distribution when the algorithm is invoked.

Algorithm 2.2.1 Extended uniformization algorithm

Given the solution time θ, generator $\boldsymbol{Q} \in \mathbb{R}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$, and
initial probability vector $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{S}_i|}$,
Let $q \leftarrow 1.02 \cdot \max_i |Q_{ii}|$ and $\boldsymbol{A} \leftarrow q^{-1}\boldsymbol{Q} + \mathbf{I}$
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{\pi}$
$\boldsymbol{\pi} \leftarrow \mathbf{0}$
$\boldsymbol{\sigma} \leftarrow \mathbf{0}$
$s \leftarrow 1$
Choose L and R for desired precision 10^{-d}
Compute $\text{Poiss}(n), \forall\, n, L \leq n \leq R$ using Fox-Glynn algorithm
for $n \leftarrow 0$ to $L - 1$ **do**
    $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + \hat{\boldsymbol{x}}$
    $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}}\boldsymbol{A}$
end for
for $n \leftarrow L$ to R **do**
    $s \leftarrow s - \text{Poiss}(n)$
    $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} + \text{Poiss}(n) \cdot \hat{\boldsymbol{x}}$
    $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + s \cdot \hat{\boldsymbol{x}}$
    $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}}\boldsymbol{A}$
end for
$\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma}/q$
return solutions $\boldsymbol{\pi}$, $\boldsymbol{\sigma}$

2.3 Phase-Type Models

Phase-type random variables are defined as the time-to-absorption of Markov chains with at least one absorbing state. An absorbing CTMC (via a rate matrix) is used for *continuous phase-type distributions* (let PH denote this family hereafter) and an absorbing DTMC (via a stochastic matrix) is used for *discrete phase-type distributions* (denoted hereafter by DPH). Each must also include a specification of the initial state occupancy probabilities that the absorbing Markov chain assumes when a new random variable is "sampled". Special cases of PH, as shown in Figure 2.3, include: exponential (Expo), Erlang, hyper-exponential (Hyper), and hypo-exponential (Hypo). Special cases of DPH, as shown in Figure 2.4, include: geometric (Geom), constant integer multiples of τ (Const), and discrete uniform (Equiprob).

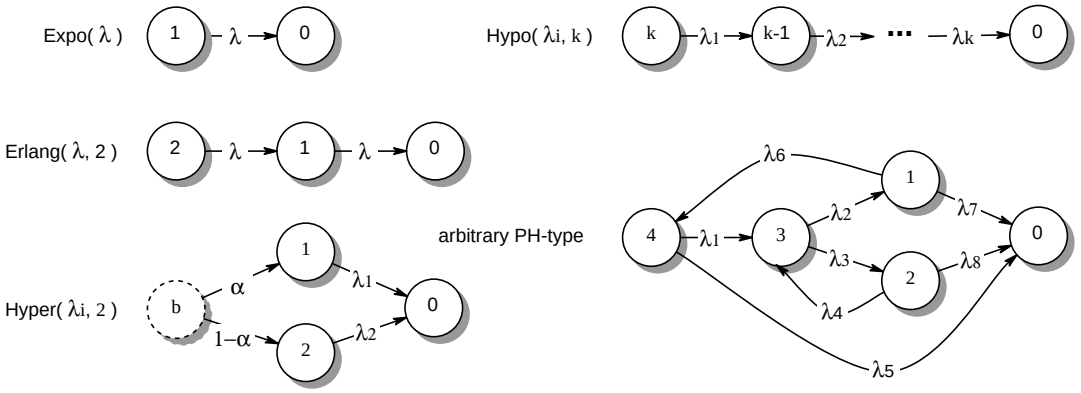


Figure 2.3: Example continuous-time phase-type (PH) random variables.

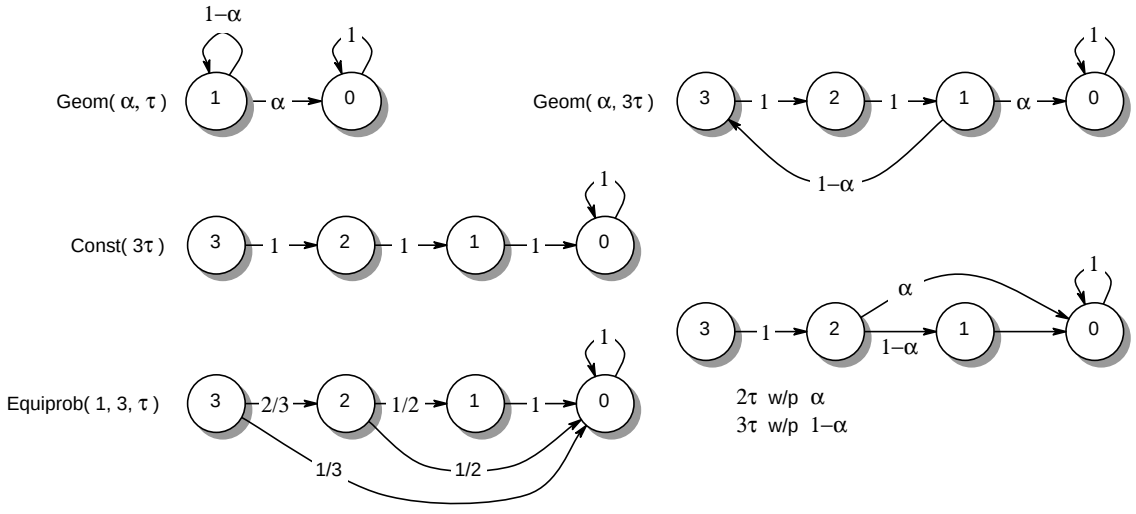


Figure 2.4: Example discrete-time phase-type (DPH) random variables with step τ.

The reachability graph and corresponding state space of a SPN with PH or DPH firing delays is constructed by *expanding* the state space and state transitions so as to remember the RFT of each enabled transition until one or more transitions can fire. The RFT for phase-type firing delays is naturally discretized, and so the pairing of each possible phase vector $\boldsymbol{\phi}$ together with each possible, discrete PN marking vector $\boldsymbol{m}$ produces a state $(\boldsymbol{m}, \boldsymbol{\phi})$ within an expanded Markov chain. By including enough information in the current state about the past evolution of the process, the past can be forgotten, essentially *Markovianizing* the process. Because the idea of supplemented states is important to our research, we will soon revisit this topic in greater detail. After firing a transition t, the execution policies that define what happens to the RFT of transitions still enabled are then applied to create a new phase vector $\boldsymbol{\phi}'$ paired with the new marking $M(t, \boldsymbol{m})$.

If the transition t that fires is once again enabled in the new marking, then a new firing delay is sampled from its PH or DPH distribution function (an absorbing Markov chain) and included in $\boldsymbol{\phi}'$ of the new state. If the firing transition t is not enabled in the marking, then its phase component, ϕ'_t, within vector $\boldsymbol{\phi}'$ is unspecified, a "don't care". This procedure continues until no new state is discovered or until no transitions can fire (this event results in an absorbing state in the reachability graph).

For all phase-type transitions t and all reachable markings $\boldsymbol{m}$, the possible combination of phases can be obtained by performing the Cartesian product of the *phase space*, denoted by $\mathcal{D}^t$, of each absorbing Markov chain that specifies the phase-type firing delay: a stochastic matrix $\boldsymbol{D}^t(\boldsymbol{m})$ for DPH phases or a rate matrix $\boldsymbol{E}^t(\boldsymbol{m})$ for PH phases. We allow the firing delays to be marking dependent by letting these matrices be functions of the marking. For DPH phases, the result is a *Cartesian product* of the constituent phase spaces and an *arithmetic product* of all corresponding one-step probabilities. For PH phases, the result is a similar Cartesian product and an *arithmetic sum* of all corresponding rates. Hereafter, we let $\mathcal{D}$ denote the potential phase space in the expanded model. When discrete-time and continuous-time models are considered separately, an expanded DTMC results from DPH phases and an expanded CTMC results from PH phases.

For DPH models, the expanded DTMC states and transition probabilities are specified formally using the *Kronecker multiplication*. Letting $\boldsymbol{A} \in \mathbb{R}^{r \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{s \times s}$, the *Kronecker product* $\otimes$ is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1r}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2r}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1}\boldsymbol{B} & a_{r2}\boldsymbol{B} & \cdots & a_{rr}\boldsymbol{B} \end{bmatrix}$$

the result being of order rs [21]. Then, for each DPH transition t, the expanded DTMC on the total, *potential* phase space $\mathcal{D}$ is given by

$$\boldsymbol{D}(\boldsymbol{m}) = \bigotimes_t \boldsymbol{D}^t(\boldsymbol{m}) \tag{2.15}$$

with order $\prod_t |\mathcal{D}^t|$. The stochastic matrix $\boldsymbol{D}(\boldsymbol{m})$ completely specifies all conditional next phases and associated probabilities that can occur in one basic step. By referring to a

"potential" phase space, we emphasize that not all combinations of discrete phases given by the matrix equation 2.15 is reachable.

For PH models, the expanded CTMC states and transition rates are specified formally using the *Kronecker addition*. Using $\otimes$ and the identity matrices $\boldsymbol{I}_r$ and $\boldsymbol{I}_s$ of order r and s, respectively, to obtain the correct dimension for summing $\boldsymbol{A}$ and $\boldsymbol{B}$, the *Kronecker sum* $\oplus$ is defined as

$$\boldsymbol{A} \oplus \boldsymbol{B} = \boldsymbol{A} \otimes \boldsymbol{I}_s + \boldsymbol{I}_r \otimes \boldsymbol{B}$$

Then, for each PH transition t, the expanded CTMC on the total, potential phase space $\mathcal{D}$ is given by

$$\boldsymbol{E}(\boldsymbol{m}) = \bigoplus_t \boldsymbol{E}^t(\boldsymbol{m}) \tag{2.16}$$

with order $\prod_t |\mathcal{D}^t|$. The rate matrix $\boldsymbol{E}(\boldsymbol{m})$ completely specifies all conditional next phases and associated transition rates that can occur in continuous time.

An example reachability graph isomorphic to an expanded DTMC is shown in Figure 2.5 assuming DPH transition timing: $t_1 \sim \text{Geom}(p, 1)$, $t_2 \sim \text{Geom}(q, 3)$, $t_3 \sim \text{Const}(2)$, and $t_4 \sim \text{Const}(1)$. The states are composed of both marking and RFT information corresponding to the discrete, firing delay phases of each enabled transition. Phases of transitions not enabled are of no importance and consequently are indicated by "$\bullet$" symbols.

Note that the state transitions associated with "ϵ" are those where no transition actually fires, but merely update the phase information. The one-step transition probabilities originating from state $(110001, 11 \bullet\bullet)$ are shown in Figure 2.6. Here we see the presence of the simultaneous firing of t_1 and t_2 between states $(110001, 11 \bullet\bullet)$ and $(000111, \bullet\bullet\bullet 1)$. Unlike the continuous-time PH models where the probability of any two transition having the same firing time is zero, such contemporary firings are possible, indeed likely, with DPH models.

Contemporary firings not only have the potential of making the reachability graph more dense (more state transitions) than one with PH transitions, but can create *confusion*. Two transition are said to be *concurrent* when each can fire and the firing of one does not affect the firing of the other. Two transitions are said to be in *conflict* when the firing of one prevents the firing of the other. When we have both concurrency and conflict, we *may* have confusion [8].

Consider the examples of possible confusion in Figure 2.7 where contemporary firings are possible and where there exists a mix of concurrency and conflict. In both examples, transitions a and c are concurrent, and transitions b and c are in conflict. In the bottom example, there is also conflict between transitions a and b. Even though contemporary firings are possible, a firing sequence must be chosen and applied to the function M to determine the next marking. PN confusion can occur when the next marking depends on the order in which transition firings are chosen. When considering the marking

$$(m_1 m_2 m_3 m_4 m_5) = (10100)$$

shown in the top example, we only see the concurrency between transitions a and c, but if we choose to fire transition a first, and move to the next marking

$$(m_1 m_2 m_3 m_4 m_5) = (01100),$$

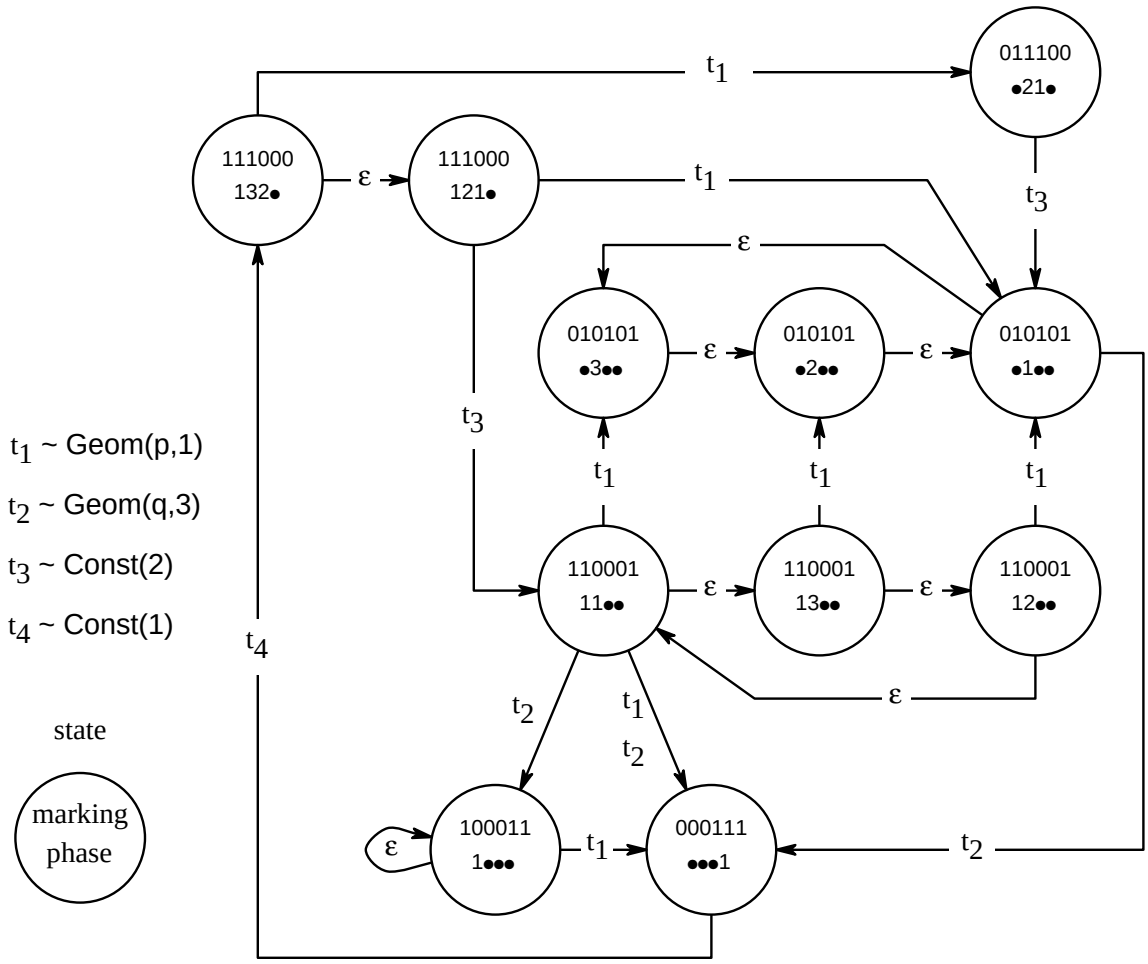


Figure 2.5: Markovianized process of the example SPN of Figure 2.1.

we encounter a conflict between transitions b and c. If we instead choose to fire transition c first, and move to the next marking

$$(m_1 m_2 m_3 m_4 m_5) = (10001),$$

then transition b is prevented from firing whether or not transition a fires next in the same marking. Thus, different outcomes arise depending on which transition, a or c, is chosen first as shown by the reachability graph in the top-right corner of Figure 2.7. The confusion about which transition to fire first must be resolved by either avoiding the confusion in the first place, by preventing the bothersome enabling of transitions, or by forcing a particular firing sequence. Either fix can be accomplished by employing a different net specification, guards, or preselection priority. Preselection priorities can resolve conflicts in untimed PNs or within timed PNs for immediate transitions that can fire in zero time. Using preselection priorities for example, different probabilities can be assigned to each sequence leading to the three new states, such that all probabilities sum to one. This results in a stochastic model even though the PN may be untimed.

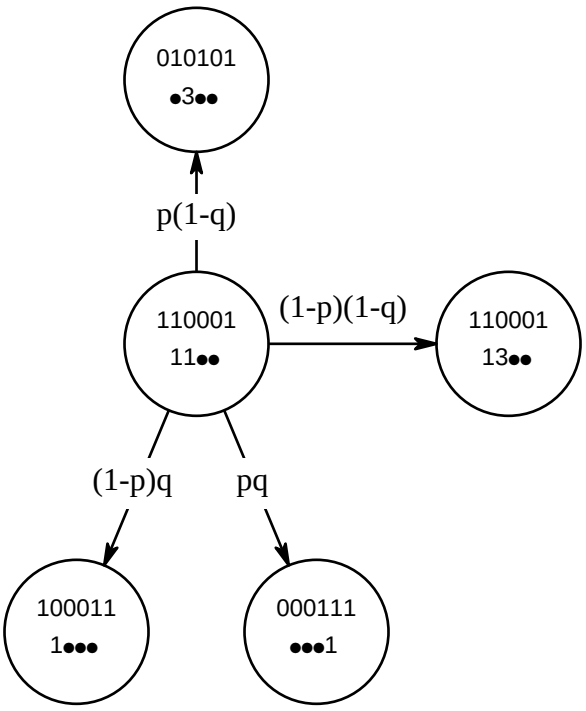


Figure 2.6: Transition probabilities for a portion of Figure 2.5.

When analyzing timed PNs, we may also have stochastic confusion. In the bottom example of Figure 2.7, PN confusion does not exist since the net reaches the same marking no matter which sequence is chosen. However, if impulse rewards* $\gamma : \mathcal{T} \to \mathbb{R}$ are used such that $\gamma(a) + \gamma(c) \neq \gamma(b)$ then stochastic confusion results since the reward measure depends on the chosen sequence. If the stochastic outcome differs depending on the order in which transitions in a contemporary firing sequence are selected to fire, then the model is not "well defined". Such confusion may be resolved the same way as with PN net confusion or with the addition of *postselection* priorities. Rather than preventing the simultaneous enabling of transitions that may lead to confusion, we may instead leave them be, let them delay, and if their simultaneous firing leads to confusion, decide then which transition gets to fire first using postselection priorities. The understanding of this problem and possible solutions, including the use of postselection priorities, have already been discussed in [22, 23, 24] and can be brought to bear on the research proposed here.

2.4 Semi-Markov Models

If the sojourn times in states are Expo or Geom random variables, we have a Markov chain: a CTMC in the former, which is memoryless for all time, and a DTMC in the latter, which is memoryless at times multiple of the basic step. Alternatively, if the sojourn times in states are *all* equal to the *same* constant $\tau \in \mathbb{N}$, we still have a DTMC. From the previous

*Each time a state transition occurs due to the firing of t, a reward of $\gamma(t)$ is accumulated.

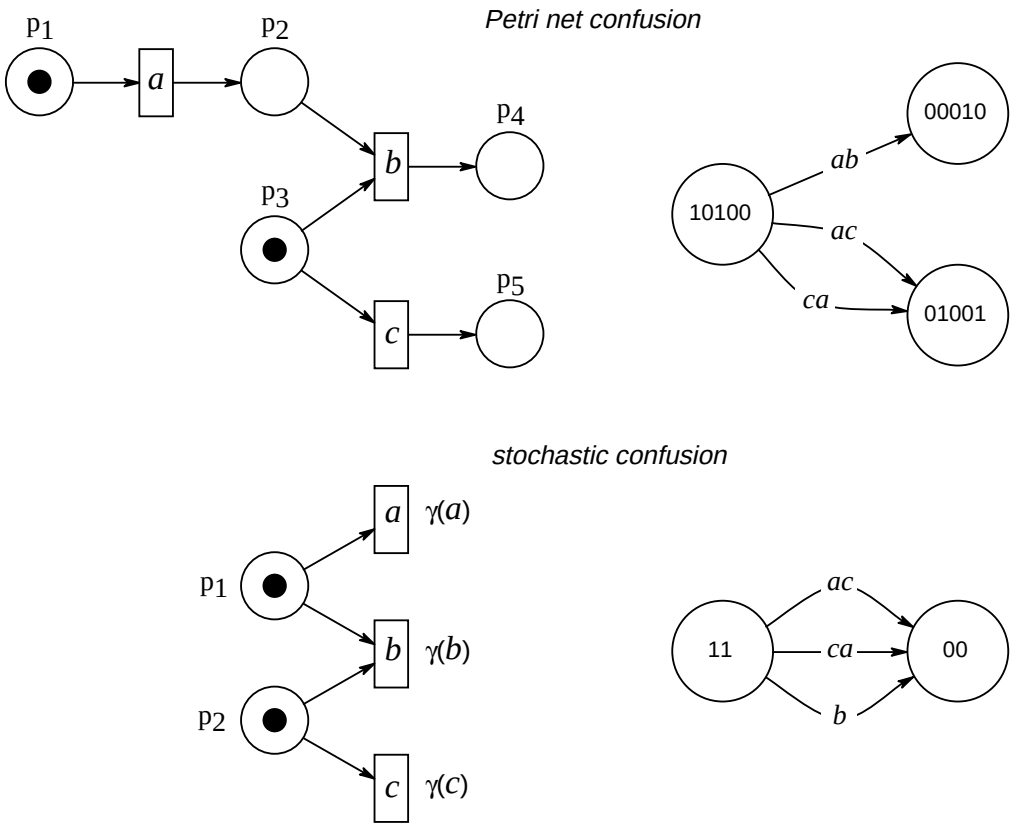


Figure 2.7: Examples of possible Petri net and stochastic confusion.

section, we know that although such models are restricted, they lend themselves to efficient time-dependent and stationary analysis.

In more general cases, semi-Markov processes satisfy the Markov property at times of jumps when state transitions occur, but not necessarily between jumps. Consequently, the sojourn times for semi-Markov processes can be arbitrary, nonmemoryless random variables. Satisfying the Markov property in at least a "semi" way affords efficient, stationary analysis just as strict Markov chains, but time-dependent analysis becomes difficult. The following theory used to study semi-Markov chains is provided below in preparation for the more general theory needed to study semi-regenerative processes, of which Markov and semi-Markov chains are special cases.

Consider a random variable X_n, defined for each $n \in \mathbb{N}$ and taking values from the state space $\mathcal{E}$, and a random variable T_n, likewise defined for each n but taking values in $\mathbb{R}^+$ such that $T_0 = 0$ and $T_n \leq T_{n+1}, \forall\, n \in \mathbb{N}$. The process $\{\,(X_n, T_n) : n \in \mathbb{N}\,\}$ is called a *Markov renewal process* (MRP) with state space $\mathcal{E}$ if the following holds $\forall\, n \in \mathbb{N},\ \forall\, j \in \mathcal{E}$:

$$\Pr\{\, X_{n+1} = j, T_{n+1} - T_n \leq \theta \mid X_0, X_1, \ldots, X_n; T_0, T_1, \ldots, T_n \,\} =$$
$$\Pr\{\, X_{n+1} = j, T_{n+1} - T_n \leq \theta \mid X_n \,\} = \Pr\{\, X_1 = j, T_1 \leq \theta \mid X_0 \,\}$$

The sequence $\{X_n : n \in \mathbb{N}\}$ is a DTMC. An example MRP sample path is portrayed in Figure 2.8.

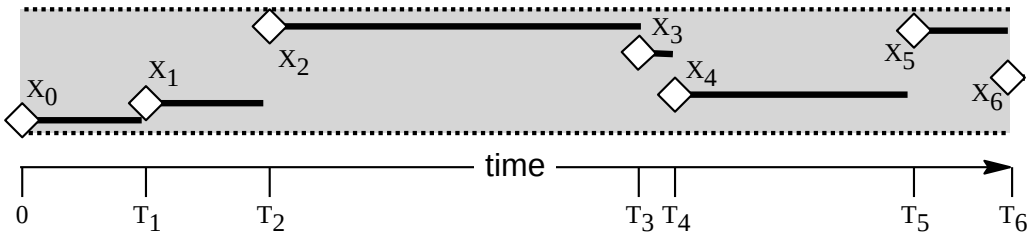


Figure 2.8: Markov renewal process.

Consider a stochastic process $\{ X(\theta) : \theta \geq 0 \}$ with state space $\mathcal{S}$ that has an *embedded* MRP with state space $\mathcal{E} \subseteq \mathcal{S}$. That is, observing $X(\theta)$ at certain, random times T_n and recording the state X_n occupied at those times produces a MRP. Analogous to the CTMC embedding discussed in the previous section, if we can determine the transition probability matrix $\boldsymbol{\Pi}$ of the embedded DTMC (EMC) and its stationary solution $x_i = \lim_{n\to\infty} \Pr\{ X_n = i \}$, $i \in \mathcal{E}$, then we can easily compute the stationary distribution $p_j = \lim_{\theta\to\infty} \Pr\{ X(\theta) = j \}$, $j \in \mathcal{S}$, of the complete process. For a semi-Markov process, one that enjoys a *renewal* after every state transition, as portrayed in Figure 2.8, we have $\mathcal{S} = \mathcal{E}$ and $\mathrm{E}[\, T_1 \mid X_0 = i \,] = \mathrm{E}[\,\text{sojourn in } i\,]$, $i \in \mathcal{E}$, and

$$\tilde{x}_i = x_i \cdot \mathrm{E}[\,\text{sojourn in } i\,], \qquad p_j = \frac{\tilde{x}_j}{\sum\limits_{k\in\mathcal{S}} \tilde{x}_k}. \tag{2.17}$$

This well known technique of "embedding" is based on the following reasoning. The stationary probability distribution can be interpreted as the fraction of time the process resides in each state. For there to be a unique stationary solution, the EMC must be ergodic; i.e., it is aperiodic, positive recurrent, and irreducible. The aperiodic property ensures that we can compute an unique stationary solution, given that the other two properties also hold. The positive recurrent property means that the process after leaving some state will eventually return to the same state in some finite time. The irreducible property means that every state can reach every other state. So to determine the expected cycle time of the stationary process, we need only pick a single reference state. Let state k be this reference state. Then after determining the stationary solution of the EMC, x_i, $\forall\, i \in \mathcal{E}$, we can interpret the ratio x_i/x_k as the expected number of visits to state i between two visits to state k. The expected sojourn time in state i within a stationary cycle is just $\mathrm{E}[\,\text{sojourn in } i\,] \cdot x_i/x_k$ and the expected cycle time is just $\sum_{j\in\mathcal{E}} \mathrm{E}[\,\text{sojourn in } j\,] \cdot x_j/x_k$. Equations 2.17 then follows from the interpretation that the stationary probability distribution is the fraction of time the process resides in each state within the expected cycle time.

The embedding method makes intuitive sense as well when we think of it as just scaling the stationary probability distribution of the EMC according to the expected sojourn times and then normalizing so that the new distribution sums to one. This idea of "scaling" is useful in understanding the theory that follows.

2.5 Semi-Regenerative Models

For a *semi-regenerative process*, one that becomes a probabilistic replica of itself at certain random times T_n, given the same state X_n, the stationary solution is computed in a way similar to semi-Markov processes except that $\boldsymbol{\Pi}$ and $\mathrm{E}[\,\text{sojourn in } k \text{ during } [0, T_1) \mid X_0 = i\,]$, $i \in \mathcal{E}$, $k \in \mathcal{S}_i$, must be computed by studying the *subordinate process*, the process with state space $\mathcal{S}_i$ that evolves between renewals [25]:

$$\tilde{x}_k = \sum_{i \in \mathcal{E}} x_i \cdot \mathrm{E}[\,\text{sojourn in } k \text{ during } [0, T_1) \mid X_0 = i\,], \qquad p_j = \frac{\tilde{x}_j}{\sum\limits_{\ell \in \mathcal{S}} \tilde{x}_\ell}. \tag{2.18}$$

An example sample path of a semi-regenerative process is portrayed in Figure 2.9.

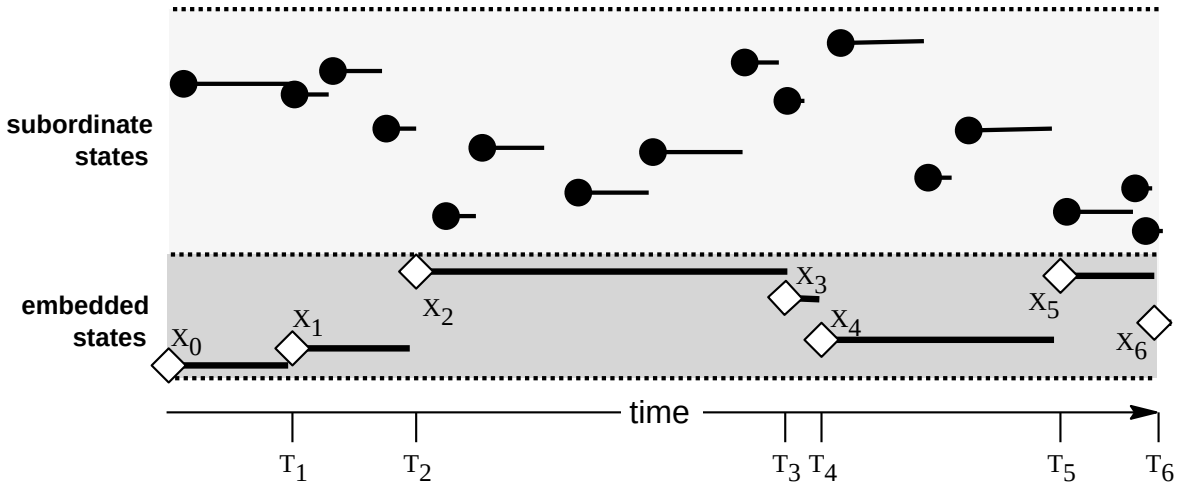


Figure 2.9: Semi-regenerative process sample path.

The problem of solving a semi-regenerative process using Markov renewal theory is reduced to studying the stochastic process between the T_n points in time when the process enjoys an absence of memory concerning its past. Unlike the semi-Markov process, the semi-regenerative process can occupy many states (within the subordinate process) between renewals, referred to hereafter as *regeneration* times. So the embedding technique used for semi-Markov processes by "scaling" is applicable to semi-regenerative processes except that we proportionally redistribute the EMC stationary distribution over all the states visited in the subordinate process between regenerations, followed by normalization. For semi-regenerative processes, the distribution, scaling, and normalization required to compute the stationary solution from the EMC is referred to hereafter as *conversion*. We will return to the subject of embedding a semi-regenerative process after we present the key aspects of Markov renewal theory, which is needed to compute the EMC and the conversion factors.

In general, Markov renewal theory depends on the specification of the following two quantities:

$$\begin{aligned} G_{ij}(\theta) &= \Pr\{\, X_{n+1} = j, T_{n+1} - T_n \leq \theta \mid X_n = i \,\} \\ &= \Pr\{\, X_1 = j, T_1 \leq \theta \mid X_0 = i \,\}, \end{aligned}$$

$$H_{ik}(\theta) = \Pr\{\, X(\theta) = k, T_1 > \theta \mid X_0 = i \,\} \qquad i, j \in \mathcal{E},\ k \in \mathcal{S}_i$$

where the elimination of n is justified by assuming homogeneity. $H_{ik}(\theta)$ gives the transient probability of occupying state $k \in \mathcal{S}_i \subseteq \mathcal{S}$ at time θ before the next regeneration given the initial state $i \in \mathcal{E}$ entered at the last regeneration. The subordinate process evolves during the interval $[T_n, T_{n+1})$ or equivalently $[0, T_1)$ when $X_0 = X_n$. $G_{ij}(\theta)$ gives the state transition probability of the EMC between two consecutive regenerations jointly with the distribution of the regeneration period T_1.

The quantities G and H can be recursively combined similar to the Chapman-Kolmogorov equation to obtain

$$P_{ij}(\theta) = H_{ij}(\theta) + \sum_{k \in \mathcal{E}} \int_0^\theta dG_{ik}(\upsilon) P_{kj}(\theta - \upsilon) \qquad i, j \in \mathcal{S} \tag{2.19}$$

known as the *Markov renewal equation*. Markov renewal theory is essentially the application of this equation to aid the study of semi-regenerative processes.

If $\mathcal{E}$ is finite, the Markov renewal equation is satisfied by the unique solution

$$P_{ij}(\theta) = \sum_{k \in \mathcal{E}} \int_0^\theta dR_{ik}(\upsilon)\, H_{kj}(\theta - \upsilon), \tag{2.20}$$

where R is the *Markov renewal function* [25]. The Markov renewal function $R_{ij}(\theta)$ is defined as the expected number of renewals observed at a fixed state $j \in \mathcal{E}$ starting from state $i \in \mathcal{E}$ within a fixed interval $[0, \theta]$:

$$R_{ij}(\theta) = \sum_{n=0}^{\infty} \Pr\{\, X_n = j, T_n \leq \theta \mid X_0 = i \,\} = \sum_{n=0}^{\infty} G_{ij}^n(\theta).$$

where $G_{ij}^n(\theta)$ is the n-fold convolution of $G_{ij}(\theta)$ with itself.

Finding the transient probability distribution $P_{ij}(\theta)$ that satisfies Equations 2.19 or 2.20 is not a trivial task in general. For models with large dimensions, direct solution in the time domain is expensive and is susceptible to numerical difficulties or instabilities. Alternatively, Equation 2.19 can be solved as a linear system of equations in the s-domain by utilizing the Laplace-Stieltjes transform [26]. However, the numerical inversion necessary to obtain the time-domain solution afterwards is also complex if numerical instabilities are to be avoided. So it would seem that the difficulties in obtaining an exact, time-dependent solution of semi-regenerative processes makes a good case for either reasonable, simplifying restrictions that make R easier to compute, or approximations [27].

Fortunately for stationary analysis, the method of embedding indirectly produces a solution that satisfies

$$\lim_{\theta \to \infty} P_{ij}(\theta) = \lim_{\theta \to \infty} \sum_{k \in \mathcal{E}} \int_0^\theta dR_{ik}(\upsilon)\, H_{kj}(\theta - \upsilon)$$

for any initial state $i \in \mathcal{S}$ (assumed to be an embedded state at time 0) with much less difficulty than solving Equation 2.19 or 2.20 directly [25]. To do this, we determine the EMC transition matrix from

$$\Pi_{ij} = \lim_{\theta \to \infty} G_{ij}(\theta) \qquad i, j \in \mathcal{E} \tag{2.21}$$

where G itself is determined from H by observing the subordinate process at regeneration times. At the same time, the conversion factors, denoted hereafter by $\boldsymbol{h} = [h_{ik}] \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{S}|}$, are computed $\forall\, i \in \mathcal{E},\ k \in \mathcal{S}_i$, from

$$h_{ik} = \mathrm{E}[\,\text{sojourn in } k \text{ during } [0, T_1) \mid X_0 = i\,] = \int_0^\infty H_{ik}(\theta)\, d\theta \tag{2.22}$$

from which we can also compute the expected value of a typical regeneration period

$$\mathrm{E}[\,T_1 \mid X_0 = i\,] = \sum_{k \in \mathcal{S}_i} h_{ik}.$$

Consequently, the complexity of the method is dictated by the complexity of studying the subordinate process and the EMC. The solution complexity of these two subproblems depends to a large degree on the regeneration points that are sampled—which ones and how many. The PDPN solution algorithms we propose in Chapter 4 are developed with these considerations in mind.

We now focus on the application of Markov renewal theory to the deterministic and stochastic PN (DSPN) [3] and its generalization, the Markov regenerative SPN (MRSPN) [12]. We assume for clarity that the probability distribution function or PDF (also known as the cumulative distribution function or CDF) for the firing delay of a generally distributed transition t,

$$F^t(\theta) = \Pr\{\,\text{transition } t \text{ firing delay } \leq \theta\,\},$$

is not marking dependent and that t cannot be preempted by the firing of another transition. While marking dependent PDFs and preemption policies have been addressed, in [28, 15, 29, 30] for example, such situations introduce an unnecessary complication to the following discussion. Although we assume marking independent PDFs for our work, we do allow certain kinds of preemption and so this topic is discussed later in Chapter 4.

Starting with $G_{ij}(\theta)$ and conditioning on the events $\{\,T_1 = \upsilon\,\}$ and $\{\,X(\upsilon) = k\,\}$, we can equivalently write, for $i, j \in \mathcal{E}$,

$$G_{ij}(\theta) = \sum_{k \in \mathcal{S}} \int_0^\theta \Pr\{\,X_1 = j \mid X(\upsilon) = k\,\} \Pr\{\,X(\upsilon) = k \mid X_0 = i\,\}\, d\Pr\{\,T_1 \leq \upsilon \mid X_0 = i\,\}. \tag{2.23}$$

For MRSPNs, which are restricted so that at most one generally distributed transition is enabled in any marking, and with our assumptions, regeneration points are observed at times when generally distributed transitions either become enabled or fire. When only Expo transitions are enabled, regeneration points are observed just after each state transition as a

result of Expo transition firings. This means that the subordinate process is a general CTMC at worst, when a generally distributed transition is enabled, and a single state CTMC at best, when no generally distributed transitions are enabled.

Consider a regeneration period starting with a known embedded state $i \in \mathcal{E}$. For the simple case when only Expo transitions are enabled, let λ_i^{-1} denote the expected sojourn time in state i and let λ_{ij} be the rate that the exponentially distributed process transitions between state i and some next state j. Then Π_{ij} is given in closed form as $\lambda_{ij}\lambda_i^{-1}$.

For the more complex case, when a generally distributed transition t is enabled, we can substitute the PDF for transition t, $F^t(\upsilon)$, in place of $\Pr\{\, T_1 \leq \upsilon \mid X_0 = i \,\}$ and recognize that $\Pr\{\, X(\upsilon) = k \mid X_0 = i \,\}$ is just the solution of the subordinate CTMC at time υ when t may fire. The quantity $\Pr\{\, X_1 = j \mid X(\upsilon) = k \,\}$ is simply the probability of entering marking j when t fires in marking k, possibly followed by a firing sequence of immediate transitions, all occurring in zero time. We denote this *switching probability* with Δ_{kj}^t. If no immediate transition firing can occur, $\Delta_{kj}^t = 1$ if $k = j$ and 0 otherwise. These substitutions into Equation 2.23 yields

$$G_{ij}(\theta) = \sum_{k \in \mathcal{S}} \int_0^\theta \left[e^{\boldsymbol{Q}_i \upsilon}\right]_{ik} dF^t(\upsilon) \Delta_{kj}^t \qquad i, j \in \mathcal{E} \tag{2.24}$$

for the MRSPN where $\boldsymbol{Q}_i$ denotes the CTMC generator with state space $\mathcal{S}_i$ and initial state $i \in \mathcal{E}$.

When only exponential transitions are enabled in state $i \in \mathcal{E}$, we know that the expected sojourn time in i is λ_i^{-1} from Equation 2.4, and so $H_{ii}(\theta) = e^{-\lambda_i \theta}$. For the more complex case, we can rewrite $H_{ik}(\theta)$ equivalently as

$$\begin{aligned} H_{ik}(\theta) &= \Pr\{\, X(\theta) = k, T_1 > \theta \mid X_0 = i \,\} \\ &= \Pr\{\, X(\theta) = k \mid T_1 > \theta, X_0 = i \,\} \Pr\{\, T_1 > \theta \mid X_0 = i \,\} \qquad i \in \mathcal{E},\ k \in \mathcal{S}_i \end{aligned}$$

and again for the MRSPN when a generally distributed transition t is enabled in state $i \in \mathcal{E}$, we can substitute $\Pr\{\, X(\theta) = k \mid T_1 > \theta, X_0 = i \,\}$ with the transient solution of the subordinate CTMC at time θ and obtain $\Pr\{\, T_1 > \theta \mid X_0 = i \,\}$ from the PDF of t to get

$$H_{ik}(\theta) = \left[e^{\boldsymbol{Q}_i \theta}\right]_{ik} \left(1 - F^t(\theta)\right). \tag{2.25}$$

The DSPN is a special case of the MRSPN where the non-exponential transitions t have deterministic firing delays, specified by $\text{Const}(\tau_t)$. With the substitution $F(\upsilon)^t = \text{Const}(\tau_t)$ in Equations 2.24 and 2.25 we can obtain the Equations 2.26 and 2.27 for the EMC $\boldsymbol{\Pi} = [\Pi_{ij}]$ and conversion factors $\boldsymbol{h} = [h_{ik}]$, respectively:

$$\begin{aligned} \Pi_{ij} &= \lim_{\theta \to \infty} G_{ij}(\theta) \\ &= \sum_{k \in \mathcal{S}} \int_0^\infty \left[e^{\boldsymbol{Q}_i \upsilon}\right]_{ik} dF^t(\upsilon)\, \Delta_{kj}^t \\ &= \sum_{k \in \mathcal{S}} \int_0^\infty \left[e^{\boldsymbol{Q}_i \upsilon}\right]_{ik} \delta(\tau_t)\, d\tau_t\, \Delta_{kj}^t \\ &= \sum_{k \in \mathcal{S}} \left[e^{\boldsymbol{Q}_i \tau_t}\right]_{ik} \Delta_{kj}^t \end{aligned} \tag{2.26}$$

and

$$\begin{aligned} h_{ik} &= \int_0^\infty H_{ik}(\theta)\, d\theta \\ &= \int_0^\infty \left[e^{\boldsymbol{Q}_i \theta}\right]_{ik} \left(1 - F^t(\theta)\right)\, d\theta \\ &= \int_0^\infty \left[e^{\boldsymbol{Q}_i \theta}\right]_{ik} \left(1 - 1(\theta - \tau_t)\right) d\theta \\ &= \int_0^{\tau_t} \left[e^{\boldsymbol{Q}_i \theta}\right]_{ik} d\theta \end{aligned} \tag{2.27}$$

where $\delta(\cdot)$ is the unit impulse function and $1(\cdot)$ is the unit step function. We will find Equations 2.26 and 2.27 useful in the analysis of the non-Markovian SPN proposed in the remaining chapters.

We end this section by providing in Figure 2.10 the underlying semi-regenerative process for our running example when all transitions but t_3 are exponentially distributed. The semi-regenerative process *graph* shown here is the same in structure, but not timing, of course, whether we consider the model to be a MRSPN or a DSPN by assuming transition t_3 to be generally distributed or deterministic, respectively. States in the EMC are shadowed to distinguish them from the states in the subordinate Markov chain that evolve due to the firing of transitions t_1 and t_2 while t_3 is enabled. Once t_3 has fired, all states visited are considered embedded states until t_3 becomes enabled once again. The initial state (111000) is considered to be both an embedded state and subordinate state by definition. Note that the semi-regenerative process graph is isomorphic to the PN reachability graph given in Figure 2.2. This is not happenstance. This is always the case for DSPNs or MRSPNs when at most one deterministic or general transition, respectively, is enabled in any marking.

A *sufficient* condition for isomorphism between the PN reachability set $\mathcal{R}$ and the underlying stochastic state space $\mathcal{S}$ is if all firing delays have continuous distributions with infinite support $[0, \infty)$. But this is not a necessary condition as shown by the fact that DSPNs, with finite-support, deterministic firing delays, have reachability sets that *are* isomorphic to the underlying stochastic state space. The necessary conditions for isomorphism ensure that the timing specification allow every enabled transition to have a chance of firing. That is, $\forall\, \boldsymbol{m} \in \mathcal{R}$:

1. The quantity $\Pr\{\, t \text{ fires, firing delay} \leq \theta \mid \text{history}\,\}$, at the present, is uniquely determinable by observing the past evolution constituting the *history*,
2. $\sum\limits_{t \in \mathcal{F}(\boldsymbol{m})} \lim\limits_{\theta \to \infty} \Pr\{\, t \text{ fires, firing delay} \leq \theta \mid \text{history}\,\} = 1$, and
3. $\forall\, t \in \mathcal{F}(\boldsymbol{m}),\ \Pr\{\, t \text{ fires} \mid \text{history}\,\} = \lim\limits_{\theta \to \infty} \Pr\{\, t \text{ fires, firing delay} \leq \theta \mid \text{history}\,\} > 0$

are true [4]. Simply put, (1) and (2) ensure unambiguous determination of future states given the past even when timing constraints are imposed, and (3) ensures that every transition has an *opportunity* to fire in every marking that enables it. When these criteria are met, the determination of $\mathcal{R}$ is independent of the firing-delay distributions. These conditions are satisfied by DSPNs and MRSPNs since at most one deterministic or general, respectively, transition t is enabled in any given marking. Because all other enabled (Expo) transitions

have continuous distribution functions with infinite support starting at 0, transition t is able to fire before any other enabled transition, and the other transitions are able to fire before transition t. Hence, every enabled transition has an opportunity to fire.

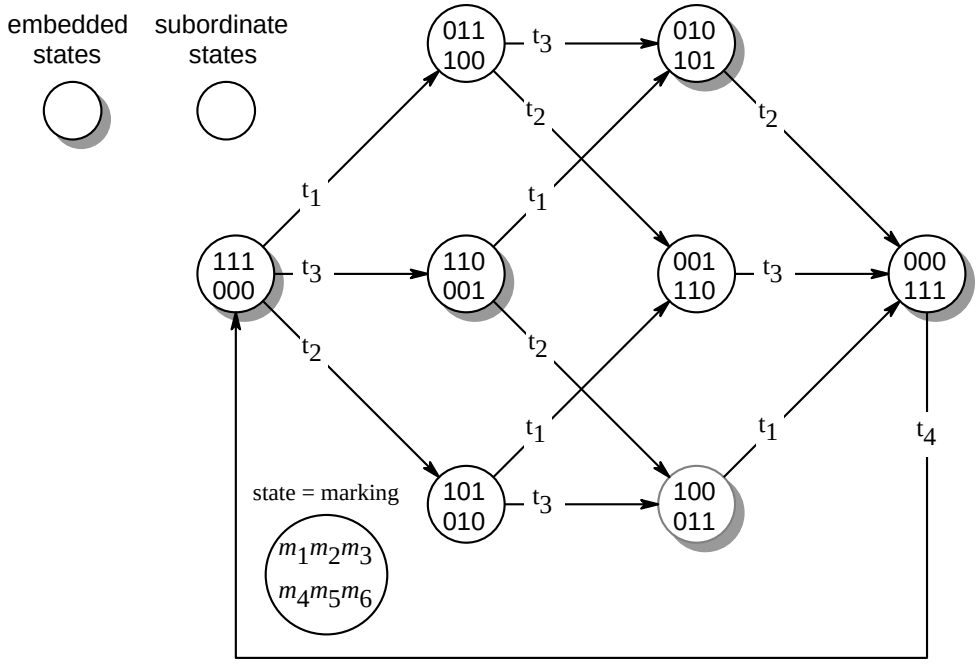


Figure 2.10: Semi-regenerative process for generally distributed t_3 and all others Expo.

The complexity of studying semi-regenerative processes increases with the complexity of the subordinate process. Models with at most one generally distributed transition enabled are convenient because they restrict the subordinate process to a CTMC. But when we allow the simultaneous enabling of multiple, generally distributed transitions, the subordinate process is more complicated. For example, the MRSPNs was extended in [29] by allowing multiple general transitions to be simultaneously enabled, provided that only one of the general transitions defines the next regeneration point. But the subordinate process becomes a semi-Markov chain, which is more difficult to solve in the transient than a CTMC. With unrestricted models in general, the subordinate process may be a semi-Markov chain or worse, perhaps even a semi-regenerative process by itself.

For example, consider our running example model with two generally distributed and two exponentially distributed transitions: $t_1 \sim F^{t_1}(\theta)$, $t_2 \sim F^{t_2}(\theta)$, $t_3 \sim \text{Expo}(\lambda)$, and $t_4 \sim \text{Expo}(\mu)$. Since t_1 and t_2 are simultaneously enabled in markings (111000) and (110001), we cannot be sure, in general, that the underlying process is semi-regenerative. But if, for instance, the PDFs were such that t_1 *always delays longer than* t_2 then, because they both become enabled simultaneously, we know that once transition t_1 fires, everything about the past since t_1 and t_2 became enabled can be forgotten. Because the stochastic marking process regenerates itself at this point in time, and depends on the state reached when t_1 fires, the process is semi-regenerative.

Starting in the initial state (111000), assumed to be a regeneration point, the next regeneration point would coincide with the firing of transition t_1 and the subordinate process

that evolves in between (due to the firing of t_2 and t_3) is itself a semi-regenerative process. So we have a two-level semi-regenerative process hierarchy—regeneration when t_2 fires at level 2 and regeneration when t_1 fires at level 1. This underlying process is portrayed in Figure 2.11 where the embedded states at level 1 (the ones we wish to place in $\mathcal{E}$) are shadowed.

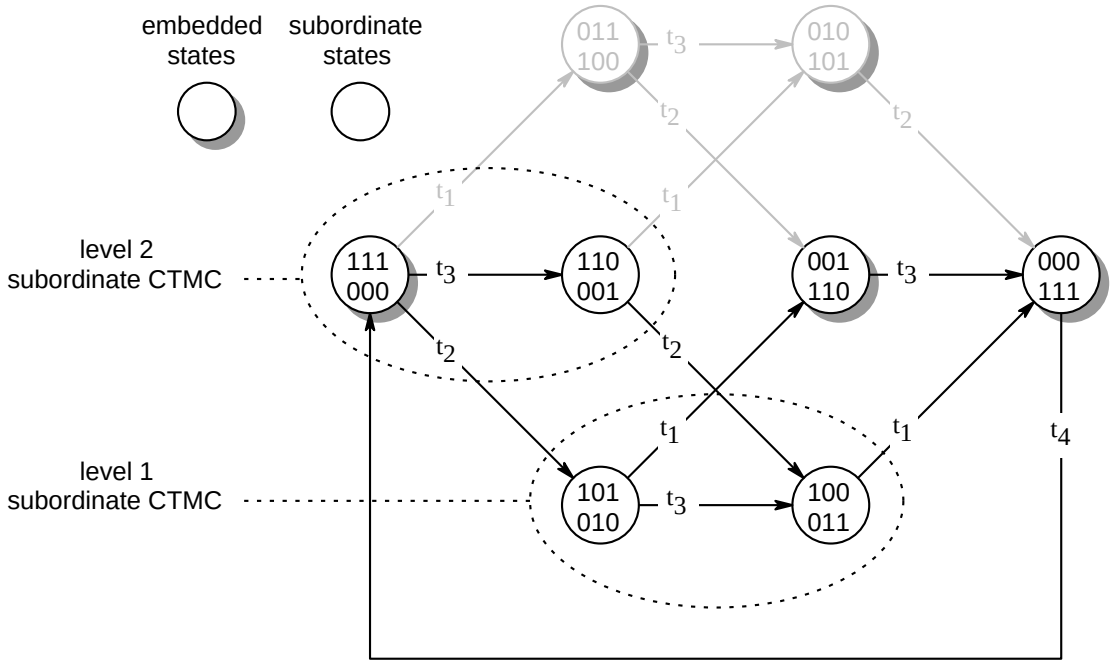


Figure 2.11: Two-Level Semi-regenerative process.

The subordinate processes for levels 1 and 2 are partitioned into two groups. The level 2 subordinate process, a 2-state CTMC, has initial state (111000) with probability one; it is also an embedded state by definition. The level 1 subordinate process, also a 2-state CTMC, has an initial probability distribution subject to the transient analysis at level 2, which gives the probability of entering either of the two states (101010) or (100011) when t_2 fires. The embedded state entered at level 1 depends on the subordinate state occupied when t_1 finally fires. The grayed portion of the graph indicates the states that are no longer reachable from (111000) because of the imposed timing, i.e., transition t_1 must fire after t_2. Unlike the previous examples, the reachability graph is obviously not isomorphic to the stochastic process and $\mathcal{S} \subset \mathcal{R}$.

In cases like this, studying the subordinate process between regeneration points becomes as difficult as studying the actual process. Moreover, under different assumptions, regeneration points may be rare or the process may not even be semi-regenerative. Without simplifying assumptions, the underlying process of an SPN may be a generalized semi-Markov process.

2.6 Generalized Semi-Markov Models

By definition, the future markings of an untimed PN depends only on the current marking, not the past markings, and so the Markov property holds. It is the nature of the timed marking process, a stochastic process, that complicates matters when any transition t has a PDF, $F^t(\cdot)$, that is not memoryless, and hence the Markov property does not hold. Without restrictions, the underlying process of an SPN is known as a *generalized semi-Markov process* (GSMP).

By including in the GSMP state the RFT information for each transition along with the current marking, everything about the past that is needed to determine the future evolution of the stochastic process is contained in the current state. Hence, the past can be forgotten. Alternatively, we could instead augment markings with *age* information, which records the times since transitions became enabled without firing. Either way, by effectively *Markovianizing* the process, customary Markov techniques can be applied, albeit to a larger, possibly continuous, state space and, perhaps, a more complicated reachability graph. This "Markovianizing" of the underlying process, which can then be solved using a generalization of Kolmogorov's forward (or backward) equations, is called the *method of supplementary variables*, which was first proposed by Cox in [31]. However, the new, Markovianized process is a *continuous-state* process in general because of the continuous nature of the age or RFT information that are augmented with the discrete marking information.

The method of supplementary variables has been applied to DSPNs and MRSPNs in [32] as an alternative to Markov renewal theory. Unlike Markov renewal theory, this method is still applicable when the process is a GSMP. A fourth-order, stationary solution algorithm has been proposed in [33] for DSPN and MRSPN models. Unfortunately, the solution of such system of equations is usually too numerically challenging for anything other than models with very small dimensions. We will back up this claim while describing the method of supplementary variables using our running example.

When employing the method of supplementary variables, there is some freedom of choice in how the generalized Kolmogorov's equations are constructed. We can choose to look forward or backward, use age or RFT variables, and specify differential or integral formulas. The published literature regarding the solution of extended DSPNs and MRSPNs with the method of supplementary variables tend to use forward equations and *age* variables, and we choose to do so here as well.

Generalized Kolmogorov's equations, forward or backward, require the use of a stationary probability density function (pdf) on the state and age variable jointly:

$$q_k(\upsilon) = \lim_{\theta\to\infty} \frac{d}{d\upsilon}\Pr\{\, X(\theta) = k,\ a \le \upsilon \,\}$$

where $\upsilon \in \mathbb{R}^+$, $k \in \mathcal{S}$, and $a \in \mathbb{R}^+$ denotes the age variable. A multidimensional pdf, $q_k(a_1, a_2, \ldots, a_n)$, is used for states that enable more than one generally distributed transition. Instead of the constant rates given by the infinitesimal generator matrix, the generalized Kolmogorov's equations also require the use of *instantaneous rate functions*, defined as

$$\lambda^t(\upsilon) = \frac{f^t(\upsilon)}{1 - F^t(\upsilon)}$$

where $f^t(t) = \frac{d}{dv}F^t(v)$ is the pdf of the firing delay for transition t. Since $\lambda^t(\cdot)$ is a conditional probability function (not a PDF however), this allows the age information a to be taken into consideration when computing the state transition rates. That is, if a denotes the age of transition t then $\lambda^t(a)da$ is interpreted as the conditional probability that t fires in the next da interval given that it has not fired in time a since becoming enabled. Of course, $\lambda^t(\cdot) = \lambda$, a constant, if t is exponentially distributed with rate λ. So when only exponentially distributed transitions are enabled, the forward equations will degenerate to the familiar system of equations (Equation 2.8) presented earlier for CTMCs.

As a consequence of the supplementary variable conditioning, we must be careful of the initial value and boundary conditions when constructing the state equations. Moreover, the Kolmogorov's forward equations must be *partial differential equations* when more than one transition with general firing delays are enabled simultaneously, coinciding with states having multiple age variables.

Consider the last example model used at the end of the previous section but without any special assumptions about $F^{t_1}(\cdot)$ and $F^{t_2}(\cdot)$. In this case, the underlying process is a GSMP. Let $a_1 \in \mathbb{R}$ and $a_2 \in \mathbb{R}$ be the *ages* of transitions t_1 and t_2, respectively. Of course, age information for transitions t_3 and t_4 can be omitted since the Expo PDF is the same when conditioned with age (or RFT) information; hence, nothing concerning elapsed time needs to be remembered. Only states that enable t_1, t_2, or both are supplemented with a_1, a_2, or both, respectively. The GSMP for our example is shown in Figure 2.12 while also introducing a different depiction of "supplemented states", enumerated by $k \in \{1, 2, \ldots, 8\} = \mathcal{S}$. That is, the supplemented states are portrayed as two circles: the smaller, raised one contains the marking and the larger one contains the age information. This depiction will be used in Chapter 4 when presenting the solution algorithms for our new SPN class, except that instead of continuous age, discrete information concerning the RFT of phase-type transitions will be recorded. Referring to Figure 2.12, the state equations using the method of supplementary "age" variables are constructed as follows.

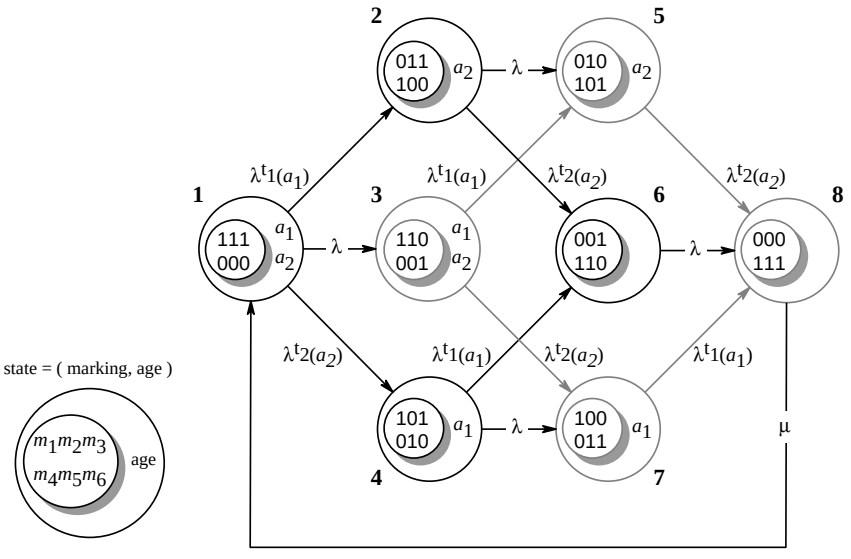


Figure 2.12: GSMP where t_1, t_2 are generally distributed and all others Expo.

First, we must consider the initial values for the age variables, which are both reset to zero only upon entering state 1 from state 8. We assume that $F^{t_1}(\cdot)$ an $F^{t_2}(\cdot)$ do not have mass at the origin and so t_1 and t_2 must delay some positive value before firing, and so no other events causing outflow of probability mass needs capturing. Consequently, the initial value condition is simply

$$q_1(0,0) = q_8\mu.$$

Second, we have the state equations for positive-valued age variables, constructed in the spirit of Kolmogorov's forward equation:

$$\begin{aligned}
\frac{d}{da_1}q_4(a_1) &= \int_0^\infty q_1(a_1,a_2)\lambda^{t_2}(a_2)\,da_2 - q_4(a_1)\lambda, \quad a_1 > 0 \\
\frac{d}{da_1}q_7(a_1) &= \int_0^\infty q_3(a_1,a_2)\lambda^{t_2}(a_2)\,da_2 + q_4(a_1)\lambda, \quad a_1 > 0 \\
\frac{d}{da_2}q_2(a_2) &= \int_0^\infty q_1(a_1,a_2)\lambda^{t_1}(a_1)\,da_1 - q_2(a_2)\lambda, \quad a_2 > 0 \\
\frac{d}{da_2}q_5(a_2) &= \int_0^\infty q_3(a_1,a_2)\lambda^{t_1}(a_1)\,da_1 + q_2(a_2)\lambda, \quad a_2 > 0 \\
\left(\frac{\partial}{\partial a_1} + \frac{\partial}{\partial a_2}\right) q_3(a_1,a_2) &= q_1(a_1,a_2)\lambda, \qquad\qquad a_1 > 0, a_2 > 0
\end{aligned}$$

The integrals are needed to uncondition on the age variable associated with the transition that fires, thereby causing the state transition of interest.

Third, we have equations for states without age variables, which resemble the familiar flow-balance equations for CTMCs:

$$\begin{aligned}
\int_0^\infty q_4(a_1)\lambda^{t_1}(a_1)\,da_1 + \int_0^\infty q_2(a_2)\lambda^{t_2}(a_2)\,da_2 - q_6\lambda = 0 \\
\int_0^\infty q_7(a_1)\lambda^{t_1}(a_1)\,da_1 + \int_0^\infty q_5(a_2)\lambda^{t_2}(a_2)\,da_2 + q_6\lambda - q_8\mu = 0
\end{aligned}$$

Fourth, since we are only interested in the stationary probability distribution of markings, we also need equations that eliminate the age variables:

$$p_k = \begin{cases}
\iint_0^\infty q_k(a_1,a_2)\,da_1\,da_2 & : \quad k \in \{\,1,3\,\} \\
\int_0^\infty q_k(a_2)\,da_2 & : \quad k \in \{\,2,5\,\} \\
\int_0^\infty q_k(a_1)\,da_1 & : \quad k \in \{\,4,7\,\} \\
q_k & : \quad k \in \{\,6,8\,\}
\end{cases}$$

where $p_k = \lim_{\theta\to\infty} \Pr\{\, X(\theta) = k\,\}$, $k \in \mathcal{S}$.

And finally, we need one last equation to normalize the solution:

$$\sum_{k\in\mathcal{S}} p_k = 1.$$

Of course, all of the above equations need to be satisfied simultaneously. Numerical solution of such system of equations ultimately requires the discretization of the continuous

variable differential-integro equations so that finite difference equations can be solved instead. We must also assume that $F^{t_1}(\cdot)$ and $F^{t_2}(\cdot)$ have finite support so that a finite dimension mesh can be constructed. But even then, computing the solution is numerically challenging. We do not intend to solve models this way. The real purpose of this exercise was to show that even with this small, simple model, solving general SPN models with underlying GSMPs is computationally challenging and costly.

Alternative solution techniques for GSMP models have been investigated. One of these, presented in [34], observes the process at fixed intervals and records the marking and RFT at these times. This procedure gives rise to an embedded general state-space Markov chain from which state equations can be written and then transformed into a system of Volterra equations. These Volterra equations permit the specification of state transitions subject to the clock readings at the equidistant time intervals and can be simpler to solve numerically than the method of supplementary variables.

When faced with practical considerations, the modeler is usually limited to simulation. SPN-specific simulation techniques have also been investigated. In [35], structural properties and conditions imposed on the SPN model are exploited that ensures the underlying process is regenerative so that faster regenerative simulation can be employed. A regenerative process is in fact a special case of semi-regenerative processes, one that regenerates itself probabilistically but also independently to the state entered at each regeneration time, thus it is more restricted. For example, a semi-regenerative process with only a single state is a regenerative process. A different simulation technique can be found in [36] where time-averaged statistics can be obtained using methods based on standardized time series, in particular, the method of batch means with the number of batches fixed. This method can analyze simulated output where the regeneration methods are not applicable.

The disadvantages to using simulation for general SPN models is that the results can be inaccurate, limited to confidence intervals, and require long simulation times. Also, the state space is never explored exactly, a shortfall that may preclude logical analysis and model-based formal verification. That is, just because some state specific property (such as deadlock) or sequence-specific property (such as an undesirable chain of events) is not observed in the simulation run does not mean that such properties do not exist in the model. It could simply mean that the simulation was not run long enough.

Chapter 3

Proposed Research

The concept essential to studying a non-Markovian process using Markovian techniques is the inclusion of additional information in the state for the purpose of remembering the time each enabled transition has to delay before its scheduled firing. This time is initially sampled from its probability distribution function when a transition first becomes enabled. Compared with general firing delays, phase-type firing delays have the advantage that the extra RFT information included in the state is conveniently *discretized* into states of an absorbing Markov chain, greatly simplifying the solution when either PH or DPH are used alone.

PH or DPH random variables can approximate any general random variable arbitrarily well. The use of phase-type firing delays has had success in the past as a way of broadening the applicability of SPN models. Indeed, continuous PH random random variables like Erlang can even approximate discrete, constant random variables by including enough stages, since the variance diminishes as the number of stages increases. Similarly, the discrete Geom random variable can approximate the continuous Expo random variable arbitrarily well by reducing the basic step size. But approximating general timing behavior by allowing either PH or DPH alone in a given model offers less modeling convenience and, possibly, requires more computational work.

Limiting a model to either PH or DPH timing may also require more computational work. Capturing a deterministic activity into a model with only PH-type timing requires an approximation using an Erlang random variable and may require many stages, and hence adding to the state space. possibly, requires more computational work.

Fortunately, such restrictions are unnecessary.

We propose to extend the SPN definition of the previous chapter to include non-Markovian timing that may prove useful to many problems while still affording an efficient, numerical solution. To this end, we elect to extend the phase-type SPN formalism for use in both discrete and continuous time, present simultaneously in the *same* model. This research effort will develop a new class of SPN that permits transition firing delays with both PH and DPH distributions. We call this new formalism a *Phased Delay Petri Net* (PDPN). The contribution herein includes the first time that PH behavior has been combined with DPH behavior in the same model, thereby extending the aforementioned research that considered each separately. Alone, PH and DPH models enjoy the "memoryless" property of underlying CTMCs and DTMCs, respectively, making efficient solutions possible. Together, the reasoning about the combined behavior becomes complicated. The proposed research will

show that the underlying process is semi-regenerative at best (perhaps degenerating to a DTMC, CTMC, or semi-Markov chain at times) and a generalized semi-Markov process at worst [25, 37]. So, our contribution also includes a formalized understanding of the stochastic process underlying this new class of SPN and the development of efficient algorithms for its solution.

After presenting the PDPN in a general setting, we intend to show how certain simplifying assumptions restrict the underlying stochastic process to one that is manageable with efficient solutions. Investigations will be conducted into techniques that offer efficiencies in exact solutions when practical. Otherwise, approximate solution algorithms will be sought that give heuristically good results with high fidelity and efficiency. Although these assumptions may also restrict the set of problems that can be modeled, we anticipate nevertheless that this will not preclude the usefulness of the PDPN to many real-world applications.

While the PDPN approach has the benefit in fidelity that comes from mixing PH and DPH behavior, the expansion of the state space required by this approach will undoubtedly compound the well known "state-space explosion problem". However, this extra burden on memory can be alleviated by utilizing advanced data structures and manipulation algorithms. For instance, decision diagrams proposed in [38, 39] offer very compact storage with a fraction of the memory requirements as conventional sparse storage structures where the memory usage grows linearly with the number of states. These advanced data structures should be just as applicable to storing the PDPN reachability graph (for performance analysis) as well as the reachability set (for logical analysis).

We can also exploit the convenient formulation of the PH and DPH state space expansion using Kronecker addition and multiplication, respectively. By employing data structures and manipulation algorithms similar to the aforementioned ones, we can take advantage of the Kronecker representations to efficiently store the expanded state space implicitly. That is, the sparse PH and DPH Markov chains can be stored in isolation and the expanded state space and matrix entries can be constructed as needed using the Kronecker operators. Such techniques have already been investigated in [40, 41, 42], and we feel that these recent advancements in compact state-space and matrix storage techniques make phase-expansion approaches worth revisiting.

Chapter 4

Preliminary Research

Before efficient analysis techniques can be investigated, the PDPN model must first be formalized, its inherent properties understood, and interesting subclasses defined to aid the investigation of solution algorithms. In this section, we discuss the underlying stochastic process of the PDPN, how to study it, and at the same time define three subclasses to the general PDPN base class. The subclasses restrict the underlying process to one that can be more easily studied. We discuss the findings of our preliminary research and the implications to solution complexity and applicability of PDPNs. We end this chapter with a proposed stationary solution algorithm and its analysis.

4.1 Analyzing the Underlying Stochastic Process

For convenience, we partition the set of transitions $\mathcal{T}$ into the set $\mathcal{T}_C$ having PH distributions, the set $\mathcal{T}_D$ having DPH distributions, and the set $\mathcal{T}_Z$ of immediate transitions having Const(0) distributions. Although the "immediate" transitions in $\mathcal{T}_Z$ are, in fact, special cases of DPH transitions, we consider them separately since they are given higher priority in firing over the timed transitions in $\mathcal{T}_C \cup \mathcal{T}_D$, and hence are usually handled separately during the analysis.

As already stated, when PH (DPH) is used alone (possibly in conjunction with immediate transitions), an otherwise non-Markovian process becomes a CTMC (DTMC). When PH and DPH are allowed to coexist, we have complicated matters, but not as badly as allowing completely general distributions. Towards constructing a PDPN reachability graph and the corresponding stochastic process specification, we must determine the possible combination of phases that can occur, ultimately leading to a phase that allows some transition t to fire. While at least one $t \in \mathcal{T}_D$ is enabled, phase changes occur at discrete instants of time $n\tau$, where $n \in \mathbb{N}$ and τ denotes the basic step interval common to all DPH transitions, referred to hereafter as the *clock*. Because PH distributions are continuous, the probability of observing a PH phase change at any particular point in time, in particular at times $n\tau$, is zero. Therefore, we can consider the DPH phase changes separately from PH phase changes. The possible combination of DPH and PH phases for each marking $\boldsymbol{m}$ can be obtained from Equations 2.15 and 2.16, respectively. Hereafter, we let $\mathcal{D}$ denote the potential phase space. The actual state space of the stochastic process will be denoted by $\mathcal{S} \subseteq \mathcal{R} \times \mathcal{D}$.

An example reachability graph assuming that transition t_3 is either Erlang$(\cdot, 2) \in$ PH or Const$(2) \in$ DPH and all others are Expo $\in$ PH is given in Figure 4.1. The states are

shadowed in a way that distinguishes the RFT information of transition t_3 corresponding to its firing delay *phase*.

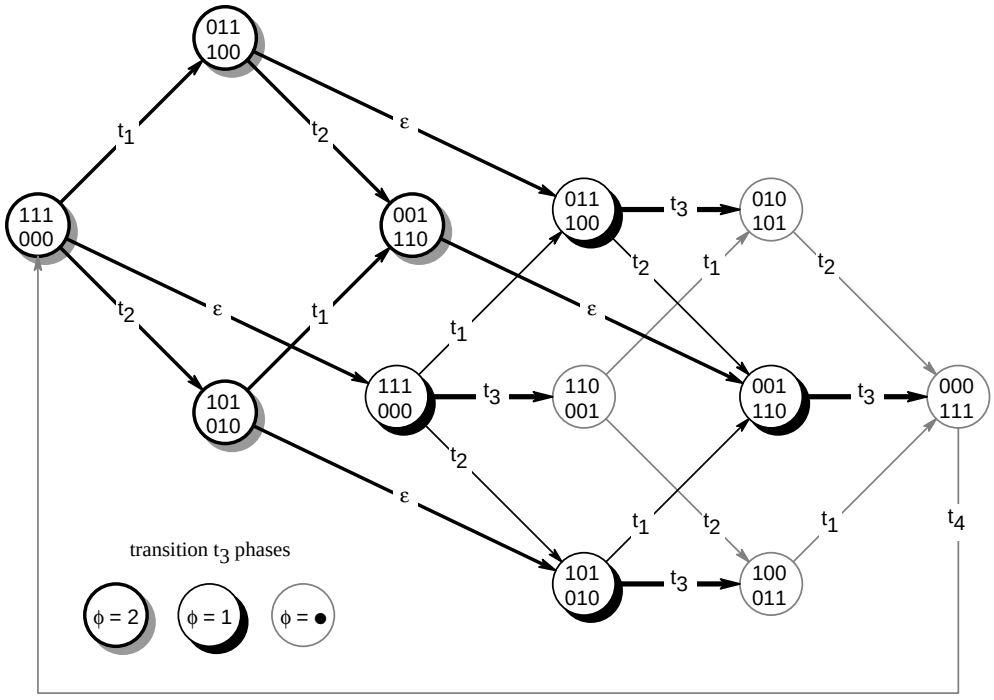


Figure 4.1: Markovianized process of the example PDPN.

PDPNs do not satisfy the conditions for isomorphism between the reachability graph and state space because multiple $\mathcal{T}_D$ transitions can be enabled and fired simultaneously, a sequence $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$, with the possibility of disabling any or all co-enabled transitions in a way that prevents their firing in certain states. Therefore, the construction of the PDPN reachability graph and the underlying stochastic process must be done in concert. As an example of how the timing constraints can restrict the reachability graph, consider the case where $t_2 \sim \text{Geom}(q, 3)$, $t_3 \sim \text{Const}(2)$, and the other transitions are Expo. The resulting stochastic marking process is portrayed in Figure 4.2 where the grayed portion indicates the states that are no longer accessible due to the time constraints. Whereas the sojourn times in markings (100011) and (000111) are exponentially distributed and therefore memoryless, this is not the case for the other markings.

In general, the underlying stochastic process of the PDPN is a generalized semi-Markov process. Although state equations can be constructed, as discussed in Chapter 2, and the method of supplementary variables applied, the method requires the solution of partial differential equations, which is computationally intensive in general. Recall that for MRSPNs, the restriction that at most one generally distributed transition is enabled in any marking simplifies the model to one consisting of ordinary differential equations, which are easier to solve. But, since this restricted marking process is a semi-regenerative process, the supplementary variable can be eliminated altogether by constructing the solution algorithm around Markov renewal theory.

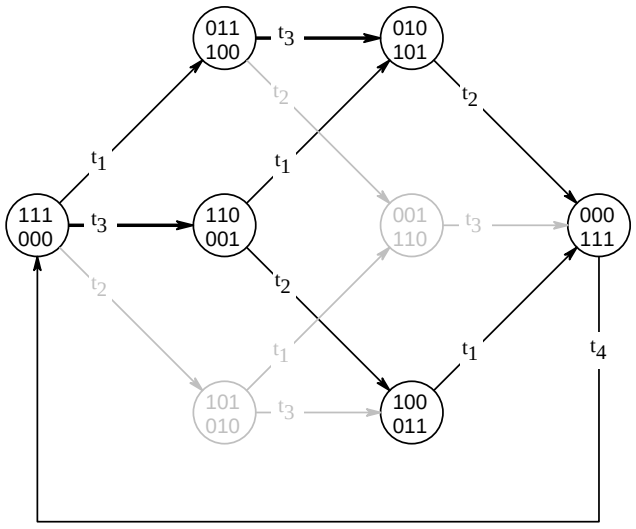


Figure 4.2: Example non-Markovian process when $t_2 \sim \text{Geom}(q, 3)$ and $t_3 \sim \text{Const}(2)$.

For PDPNs, we can eliminate the clock variables from the state, thereby reducing the state-space memory along with the computational costs if we restrict the PDPN so that *synchronization* is maintained among all enabled $\mathcal{T}_D$ transitions. By synchronizing the $\mathcal{T}_D$ transitions, the analysis is simplified, subject to only a single clock, thus requiring the storage of just one clock variable in each state. Even better, the clock variable can be eliminated altogether for this restricted PDPN by recognizing the underlying stochastic process as a semi-regenerative process. The conditions necessary for such synchronization in any marking $\boldsymbol{m} \in \mathcal{R}$ are:

1. transitions $t \in \mathcal{T}_D$ are never enabled by a firing sequence $s \in \mathcal{T}_C\mathcal{T}_Z^*$ except when $\mathcal{F}(\boldsymbol{m}) \cap \mathcal{T}_D = \emptyset$, and

2. if a firing sequence $s \in \mathcal{T}_C\mathcal{T}_Z^*$ resets the current phase of a transition $t \in \mathcal{F}(\boldsymbol{m}) \cap \mathcal{T}_D$ then it must do so for all transitions in $\mathcal{F}(\boldsymbol{m}) \cap \mathcal{T}_D$.

Theorem 4.1.1 *The stochastic process underlying a synchronized PDPN is semi-regenerative.*

Proof. Let $X = \{ X(\theta) : \theta \geq 0 \}$ denote the underlying stochastic process. Clearly, if only $\mathcal{T}_D$ transitions are enabled, X is a DTMC, and if only $\mathcal{T}_C$ transitions are enabled, X is a CTMC, both of which are special cases of the semi-regenerative class. Consider periods when both $\mathcal{T}_D$ and $\mathcal{T}_C$ transitions are enabled. Because the firing sequences $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ are expanded into Expo and Const(0) state transitions, which are memoryless for all time, we need only observe the successive times when the $\mathcal{T}_D$ transitions become enabled, fire, or undergo phase advancements. Restrictions (1) and (2) ensure that all such events for different $\mathcal{T}_D$ are synchronized and therefore occur at successive jump times $T_n = T_{n-1} + \tau$ at which time state X_n is entered. The sequence of states $\{ X_n : n \geq 0 \}$ form a DTMC, and together the sequence $\{(X_n, T_n) : n \geq 0\}$ forms a Markov renewal process. Therefore it follows, by definition, that X is a semi-regenerative process. □

The process shown in Figure 4.2 is semi-regenerative since t_2 and t_3 maintain synchronization with respect to the basic clock advancements. As such, Markov renewal theory can be applied to solve the model. While it has been shown that solution algorithms based on Markov renewal theory has the same asymptotic costs as the method of supplementary variables [11], we believe Markov renewal theory can be more intuitive in some ways, interpretation of the results to the original process can be preserved, and opportunities to eliminate phase information from the state can be exploited. But to make Markov renewal theory applicable, we must impose the above conditions on the PDPN model to ensure that the underlying stochastic process is semi-regenerative.

4.1.1 Theory Applied to PDPNs in General

We already know that, separately, PH and DPH based SPNs enjoy the efficient solution of underlying CTMCs and DTMCs, respectively. We also know that mixing PH and DPH behavior requires the solution of a semi-regenerative process, thereby complicating matters. However, by assuming synchronization between $\mathcal{T}_D$ transition when enabled and with the aid of Markov renewal theory, we have reduced the analysis problem to one of studying multiple CTMCs (the subordinate processes), one for each embedded state in $\mathcal{E}$, and one DTMC (the EMC). To this end, we must simultaneously study the evolution of both the DTMC and each CTMC in turn as they interact with one another.

Figure 4.3 shows a sample path observation of a typical PDPN regeneration period aided by Markov renewal theory. Because isomorphism between the timed and untimed PDPN reachability graph is not guaranteed and because of the potential interaction between the DTMC and CTMC models, we take the approach of constructing the stochastic matrix, $\boldsymbol{\Pi}$, of the EMC, one row at a time.

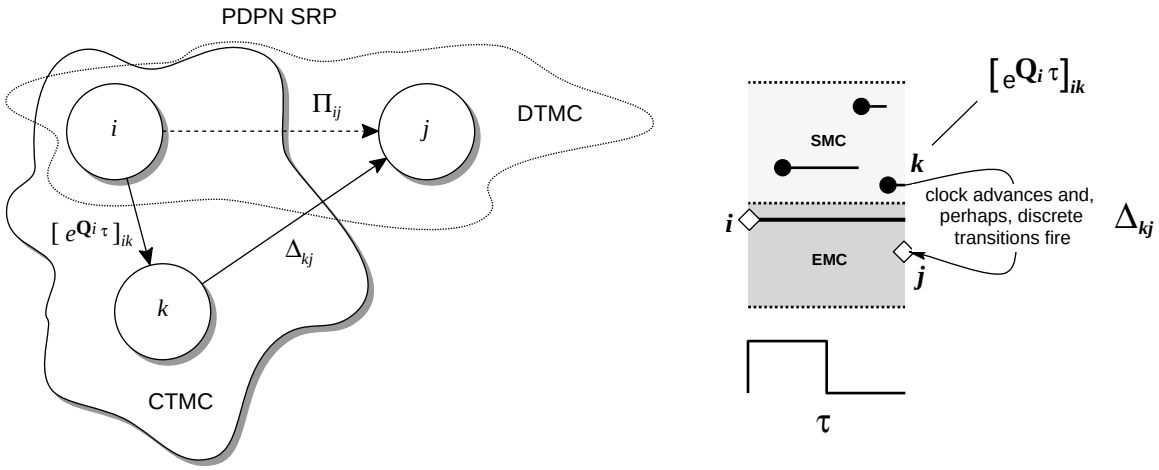


Figure 4.3: Studying a PDPN regeneration period.

The basic algorithm to construct $\boldsymbol{\Pi}$ can be as follows. Starting with a known embedded state $i \in \mathcal{E}$, we observe the subordinate CTMC (SMC) up to the next clock advance, a period of *at most* length τ. The regeneration period T_1 defined here is actually a random variable over the range $(0, \tau]$. T_1 is exactly τ if at least one $t \in \mathcal{T}_D$ remains enabled during the entire period. However, if all transitions in $\mathcal{T}_D$ become disabled or they all are forced to simultaneously reset to a new firing delay (resample) due to the firing of a transition from $\mathcal{T}_C \cup \mathcal{T}_Z$, then T_1 will be less than τ. Assuming that $T_1 = \tau$, we simply solve the SMC (with generator $\boldsymbol{Q}_i$ and state space $\mathcal{S}_i$ originating from the embedded state i) at time τ. The state occupied at time τ, when the clock advance occurs, is applied to the next state *switching matrix* $\boldsymbol{\Delta}$ that computes the set of states reachable after some $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$ following the clock advance. This set of next states are new embedded states and are added to $\mathcal{E}$. This procedure repeats until no new embedded states are found. Formally, the analysis is expressed by the application of Markov renewal theory with the PDPN properties in mind:

$$H_{ik}(\theta) = \left[e^{\boldsymbol{Q}_i \theta} \right]_{ik} \tag{4.1}$$

$$\Pi_{ij} = \sum_{k \in \mathcal{S}_i} \left[e^{\boldsymbol{Q}_i \tau} \right]_{ik} \Delta_{kj} \tag{4.2}$$

where, of course, $e^{\boldsymbol{Q}_i \theta}$ is the transient solution of the SMC at time θ, one solution for each $i \in \mathcal{E}$. Notice that these equations are the same as for DSPNs except that for PDPNs we have a fixed deterministic delay τ. The following equations that are needed for the stationary solution of PDPNs are the same as well and are repeated here for convenience.

We can distinguish the two cases of $\{T_1 = \tau\}$ and $\{T_1 < \tau\}$ by appropriately constructing the SMC. That is, the CTMC states reached that also coincide with $\mathcal{T}_D$ simultaneous disabling or resampling are made absorbing and are regarded as embedded states in $\mathcal{E}$. The set of absorbing states, which are formed in this way, will be denoted hereafter as the set $\mathcal{E}_i$. In this way, we *trap* such events and associated probability mass in these absorbing states when solving the CTMC at time τ. If the total probability mass absorbed is α, then because $\Pr\{\, T_1 > \tau \,\} = 0$, we know that $\Pr\{\, T_1 < \tau \,\} = \alpha$ and $\Pr\{\, T_1 = \tau \,\} = 1 - \alpha$. For stationary analysis where we are interested in constructing the EMC matrix $\Pi_{ij} = \lim_{\theta \to \infty} G_{ij}(\theta)$, the exact value of T_1 is of no consequence, only its expected value is important. With the appropriately constructed absorbing CTMC, the expected value of T_1 is determined from the cumulative probabilities

$$h_{ik} = \mathrm{E}[\,\text{sojourn in } k \text{ during } [0, T_1) \mid X_0 = i\,] = \left[\int_0^\tau e^{\boldsymbol{Q}_i v}\, dv \right]_{ik} \qquad i \in \mathcal{E}, k \in \mathcal{S}_i \backslash \mathcal{E}_i \tag{4.3}$$

in each CTMC state, which are computed anyway to obtain the necessary conversion factors. Then

$$\mathrm{E}[\, T_1 \mid X_0 = i \,] = \sum_k h_{ik}$$

for each $i \in \mathcal{E}$ and for all $k \in \mathcal{S}_i \backslash \mathcal{E}_i$.

The stationary solution $\boldsymbol{x} = [x_i] \in \mathbb{R}^{|\mathcal{E}|}$ of the EMC satisfies the set of balance equations

$$\boldsymbol{x}\boldsymbol{\Pi} = \boldsymbol{x} \quad \text{ subject to } \quad \sum_{i \in \mathcal{E}} x_i = 1. \tag{4.4}$$

Because we expect $\boldsymbol{\Pi}$ to be large and sparse, iterative methods like Gauss-Seidel and successive overrelaxation (SOR) should be employed for the solution of $\boldsymbol{x}$.

Finally, the stationary solution

$$p_j = \lim_{\theta \to \infty} \Pr\{\, X(\theta) = j \,\},\ j \in \mathcal{S},$$

of the semi-regenerative process underlying the PDPN is obtained through the conversion

$$p_j = \frac{\sum\limits_{i \in \mathcal{E}} x_i h_{ij}}{\sum\limits_{k \in \mathcal{S}} \sum\limits_{i \in \mathcal{E}} x_i h_{ik}} \tag{4.5}$$

We now investigate in the following sections the analysis of three classes of PDPNs, defined by certain simplifying assumptions, and all having semi-regenerative marking processes. The fourth class, discussed last, is actually the PDPN base class with no simplifying assumptions and an underlying generalized semi-Markov marking process.

4.1.2 Isochronous PDPNs

The most restricted PDPN class we consider is one where we assume that

1. transitions in $\mathcal{T}_D$ are *always* synchronized,
2. at least one $t \in \mathcal{T}_D$ is enabled at *all* times, and
3. the clock is *never* reset by transitions in $\mathcal{T}_C$.

These assumptions cause strict *synchronous* execution that has the *same* clock period for all time. Hence, we refer to this restricted class as an *isochronous* PDPN. The most important consequence of this "isochronous" execution is that each regeneration period is deterministic, having a constant duration of τ, the basic clock period of all $\mathcal{T}_D$ transitions. We allow periods when only $\mathcal{T}_D$ transitions are active (DTMC only), when both $\mathcal{T}_D$ and $\mathcal{T}_C$ transitions are active (DTMC and CTMC), but not $\mathcal{T}_C$ transitions alone (CTMC only), as portrayed in Figure 4.4(a).

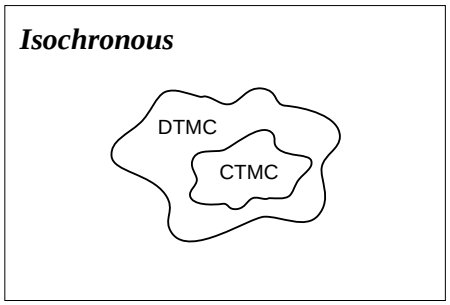
Isochronous PDPNs are very convenient since they are accompanied by a fixed timeline in which to perform time-dependent analysis, which is otherwise too difficult. Figure 4.4(b) portrays an isochronous PDPN sample path with a clocked timeline. Notice that the expected sojourn time of every embedded state is a constant τ. Consequently, the number of regenerations that can occur in some fixed time θ must be given by $N = \lfloor \theta/\tau \rfloor$. After N regenerations (jumps), the EMC must occupy the set of states with conditional probability distribution $\boldsymbol{\Pi}^N$. It follows then that the Markov renewal function can be computed from

$$\boldsymbol{R}(\theta) = \boldsymbol{\Pi}^{\lfloor \theta/\tau \rfloor}.$$

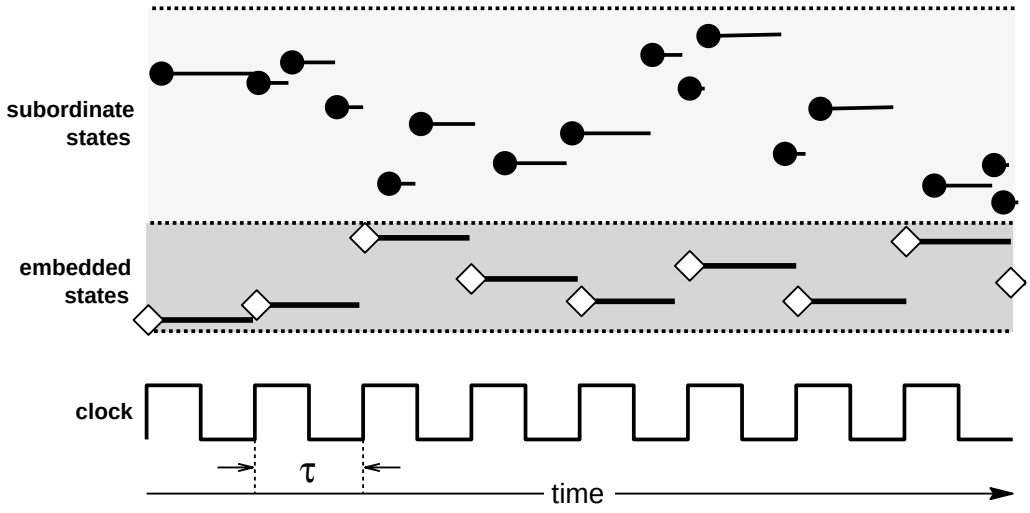
Given state $k \in \mathcal{E}$ occupied after $N = \lfloor \theta/\tau \rfloor$ regenerations, the conditional SMC occupancy at the residual time $u = \theta - N\tau$ can be computed from $e^{\boldsymbol{Q}_k u}$, and the formula

$$\sum_{k \in \mathcal{E}} \left[\boldsymbol{\Pi}^N\right]_{ik} \left[e^{\boldsymbol{Q}_k u}\right]_{kj} \tag{4.6}$$

for $i, j \in \mathcal{S}$ satisfies the Markov renewal equation (2.19), providing a rather efficient and exact time-dependent solution, $P_{ij}(\theta)$. Stationary solutions are still easily computed with the embedding method. The only impact these assumptions have on stationary analysis is that the SMC will not have absorbing states due to resampling events.



(a)

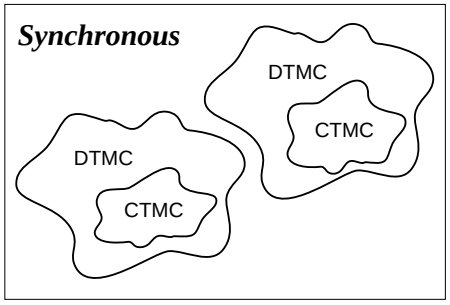


(b)

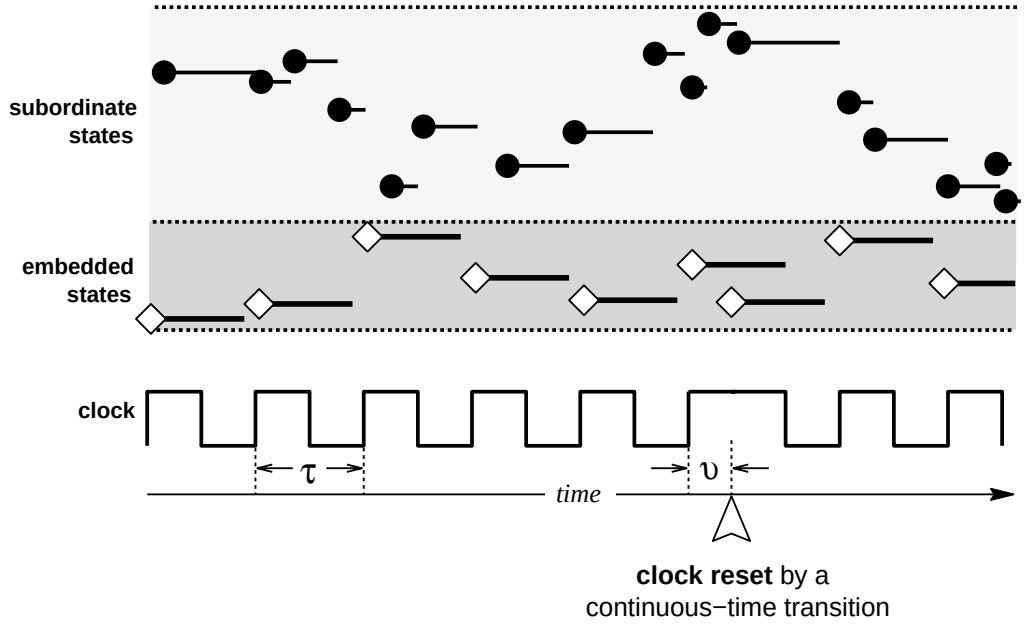
Figure 4.4: (a) Characterization and (b) sample path of the Isochronous PDPN.

4.1.3 Synchronous PDPNs

Consider now a PDPN class that is slightly less restrictive than the isochronous PDPN. Called a *synchronous* PDPN, this class is identical to the isochronous PDPN except that resampling events are permitted. That is, firing sequences $s \in \mathcal{T}_C \mathcal{T}_Z^*$ within the SMC may cause the clock governing $\mathcal{T}_D$ execution to reset, but such firing sequences without at least one $t \in \mathcal{T}_D$ enabled is not allowed. The synchronous PDPN can be characterized by Figure 4.5(a) since clock resets can be thought of as the start of a new DTMC and CTMC evolution referenced from a new time origin. Although stationary analysis is still easy with the embedding method, time-dependent analysis is now hard because regeneration periods may have durations that are less than τ by a random amount of time depending on when the clock resets occur, as portrayed in Figure 4.5(b).



(a)



(b)

Figure 4.5: (a) Characterization and (b) sample path of the Synchronous PDPN.

Time-dependent state equations can be written by considering the mutually-exclusive occasions when the clock is free to advance by amount τ and when the clock is reset. Beginning with the Markov renewal equation (2.19), we can let

$$G_{ik}(\upsilon) = \begin{cases} \sum\limits_{\ell \in \mathcal{S}_i} \left[e^{\boldsymbol{Q}_i \tau}\right]_{i\ell} \Delta_{\ell k} & i,k \in \mathcal{E} \quad \text{when the clock advances,} \\ \left[e^{\boldsymbol{Q}_i \upsilon}\right]_{ik} & i \in \mathcal{E}, k \in \mathcal{E}_i \quad \text{when the clock resets} \end{cases}$$

and substitute $H_{ij}(\theta)$ with the conditional SMC solution given by Equation 4.1 for the residual time less than τ, which serves as the boundary condition that stops the recursion. This yields the recursive state equation

$$\begin{aligned} P_{ij}(\theta) = & \sum_{k \in \mathcal{E}} \sum_{\ell \in \mathcal{S}_i} \left[e^{\boldsymbol{Q}_i \tau}\right]_{i\ell} \Delta_{\ell k} P_{kj}(\theta - \tau) + \qquad \textit{clock advances} \\ & \sum_{k \in \mathcal{E}_i} \boldsymbol{Q}_i \int_0^\tau \left[e^{\boldsymbol{Q}_i \upsilon}\right]_{ik} P_{kj}(\theta - \upsilon)\, d\upsilon + \qquad \textit{clock resets} \\ & \left[e^{\boldsymbol{Q}_i \theta}\right]_{ij} 1_{\{j \in \mathcal{S}_i\}} \Big(1 - 1(\theta - \tau)\Big) \qquad \textit{stop} \end{aligned} \tag{4.7}$$

defined as such for $i, j \in \mathcal{S}$, $\theta \geq 0$ and equal to 0 otherwise. The indicator function $1_{\{j \in \mathcal{S}_i\}}$ returns 1 if $j \in \mathcal{S}_i$ and 0 otherwise.

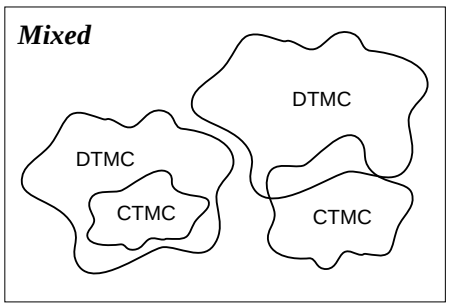
Writing time-dependent state equations is one thing, but finding efficient ways to solve them is another. Whether we attempt this in the time domain or the s-domain by using the Laplace transform, the outcome is the same for anything other than toy models: the time-dependent solution is difficult and computationally intense.

4.1.4 Mixed PDPNs

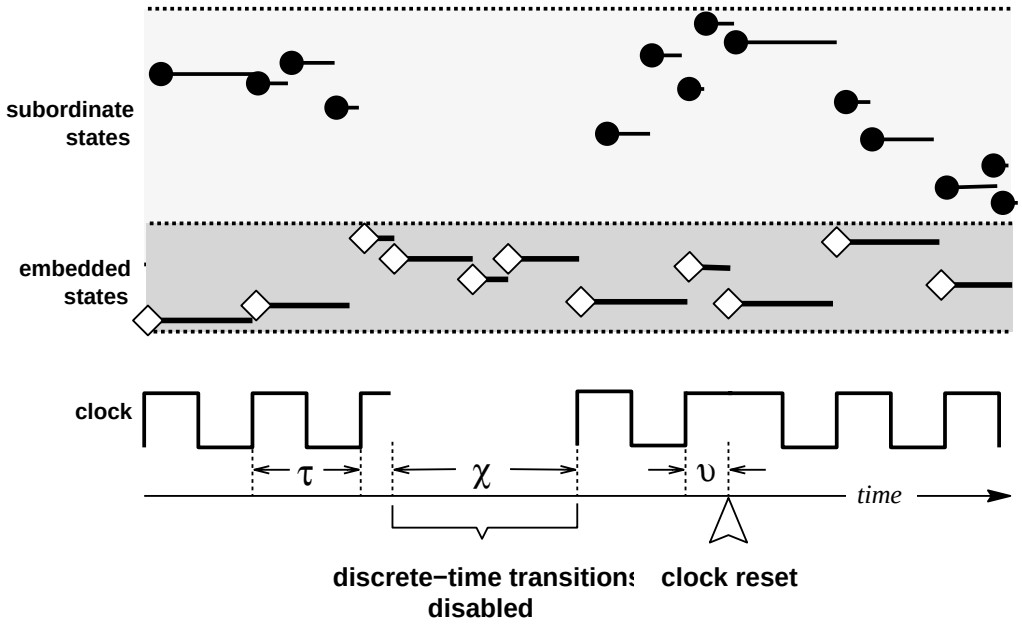
The restricted PDPN classes presented in the previous two sections assume that at least one transition from $\mathcal{T}_D$ is enabled in all reachable markings. This implies that a clock and a DTMC aspect of the model are always present. When this is not the case, we can say that the PDPN is *asynchronous* in the sense that transition firings are not always, if at all, synchronized with a clock. Instead, there would only exist a CTMC aspect due to firing sequences $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$, which can occur along a continuous timeline. Certainly, we can think of a PDPN when only transitions in $\mathcal{T}_C$ are enabled as asynchronous in the current context.

So let us now consider a PDPN that is synchronous at certain times and asynchronous at other times, referred to as a *mixed* PDPN. A mixed PDPN is a restricted class of PDPN, having the same properties as a synchronous PDPN except that all transitions in $\mathcal{T}_D$ may be disabled at times, as characterized by Figure 4.6(a). Still, interestingly enough, stationary analysis is just as easy, and time-dependent analysis is just as hard as with synchronous PDPNs.

Referring to a sample path portrayed in Figure 4.6(b), notice that during periods when only transitions in $\mathcal{T}_C$ are enabled, the EMC can be considered identical to an embedding of



(a)



(b)

Figure 4.6: (a) Characterization and (b) sample path of the Mixed PDPN.

the active CTMC. Similarly, the EMC is identical to the active DTMC when only transitions in $\mathcal{T}_D$ are enabled. Alternatively, we could observe a different EMC, one that skips over such periods of inactivity of either $\mathcal{T}_D$ or $\mathcal{T}_C$ transitions. More will be said later about alternative embeddings.

The time-dependent state equations for the mixed PDPN is the same as for the synchronous PDPN with the minor addition: the transient solution of a CTMC associated with transition firings $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ in isolation. The probability that a state transition from i with total outgoing rate λ_i does not occur—hence, the CTMC remains in the current state i—is given by

$$H_{ij}(\theta) = e^{-\lambda_i \theta} \delta_{ij} \qquad i, j \in \mathcal{S}$$

where the unit delta function δ_{ij} is equal to one for $i = j$ and zero otherwise. By embedding

the CTMC, we can compute

$$G_{ij}(\theta) = \frac{\lambda_{ij}}{\lambda_i}\left(1 - e^{-\lambda_i \theta}\right)$$

and thus

$$dG_{ij}(\theta) = \lambda_{ij} e^{-\lambda_i \theta} d\theta \qquad i, j \in \mathcal{E}.$$

So the state equations for mixed PDPNs are given by Equation 4.7 for states i where at least one $t \in \mathcal{T}_D$ is enabled, and

$$P_{ij}(\theta) = e^{-\lambda_i \theta}\delta_{ij} \; + \sum_{k \in \mathcal{S}\setminus\{i\}} \int_0^\theta \lambda_{ik} e^{-\lambda_i \upsilon} P_{kj}(\theta - \upsilon)\, d\upsilon \qquad i, j \in \mathcal{S} \tag{4.8}$$

for states $i \in \mathcal{S}$ where only transitions in $\mathcal{T}_C$ are enabled, defined for $\theta \geq 0$ and equal to 0 otherwise.

4.1.5 Asynchronous PDPNs

The previous PDPN classes maintain synchronization among transitions in $\mathcal{T}_D$ when enabled. The base class of the PDPN does not have this nor any other restriction. Without simplifying assumptions, the PDPN may have multiple DTMC executions or embedded processes, evolving simultaneously and unsynchronized with skewed timelines relative to each other. Of course, there may also be times when only DTMC or CTMC execution is present. This full generality is indicative of the PDPN base class, as portrayed in Figure 4.7(a), and referred to as an *asynchronous* PDPN.

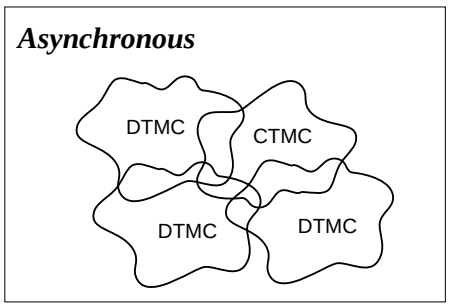
Transitions in $\mathcal{T}_D$ become unsynchronized when there is at least one transition in $\mathcal{T}_D$ enabled and

1. a firing sequence $s \in \mathcal{T}_C \mathcal{T}_Z^*$ causes one or more previously disabled transitions in $\mathcal{T}_D$ to become enabled without disabling some of the currently enabled transitions in $\mathcal{T}_D$, or

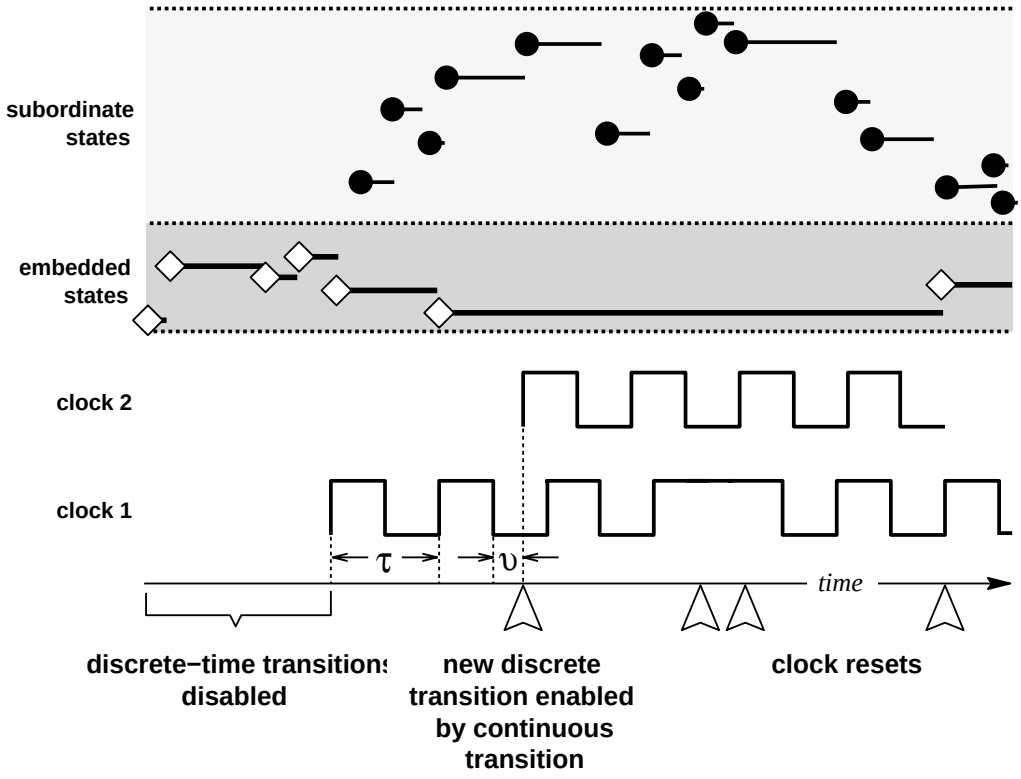
2. when a firing sequence $s \in \mathcal{T}_C \mathcal{T}_Z^*$ causes some but not all enabled transitions in $\mathcal{T}_D$ to resample a new firing delay.

In both cases, a new clock sequence will start that is out-of-sync with the current clock. Firing sequences $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$ never pose a problem since any enabling or resampling events they cause would be synchronized with the clock. But the continuous-time nature of $\mathcal{T}_C$ transitions means that the probability of any $t \in \mathcal{T}_C$ firing in-sync with the clock is zero. So the firing of any $t \in \mathcal{T}_C$ or any immediate transition following t that causes (1) or (2) above to happen will be followed by unsynchronized execution. However, once the PDPN becomes unsynchronized, it does not necessarily have to stay that way. All transitions in $\mathcal{T}_D$ may be either disabled, thereby eliminating the multiple clocks altogether, or forced to resample simultaneously, thereby allowing the synchronized execution to resume. Such examples are illustrated by the sample path shown in Figure 4.7(b).

Referring to Figure 4.7(b), notice that during the period of unsynchronized execution, there is an extended sojourn in the embedded state entered prior to the unsynchronizing



(a)



(b)

Figure 4.7: (a) Characterization and (b) sample path of the Asynchronous PDPN.

event. This is because regeneration points are typically rare, too difficult to determine, or, more likely, do not even exist during unsynchronized execution. So for asynchronous PDPNs, we may have a difficult time recognizing the underlying stochastic process as being semi-regenerative. At the very least, the subordinate process during unsynchronized execution will be much harder to solve. Consequently, even stationary analysis of asynchronous PDPNs is difficult. Without the aid of simplifying properties or assumptions, we will have to consider the underlying process to be a generalized semi-Markov process. As such, we might resort to the method of supplementary variables or fixed-interval observations for exact so-

lutions, or estimations derived from simulation. Unfortunately, these methods can be too computationally intensive. We plan to investigate, instead, approximation methods for both stationary and time-dependent solutions that are efficient and offer acceptable accuracy.

4.2 PDPN Stationary Solution Algorithm

The formalization and mathematical modeling of PDPN behavior has been developed in the previous sections. It should be clear to the reader that transient analysis of PDPNs other than the *isochronous* class is difficult while stationary analysis of those other than the *asynchronous* class is relatively easy. So, exact methods will mostly be sought for stationary analysis and approximate methods will be sought for transient analysis. We save the development of approximate solutions for the remaining research, outlined later in Chapter 5. We will now propose in this section efficient stationary solution algorithms that stem from our preliminary research, but first, some important issues will be discussed.

Given that the stochastic process state space generated by a high-level formalism like a Petri net already suffers from exponential growth, the combinatorial growth introduced by phase-type, state-space expansion only exasperates the problem. There are at least three ways to alleviate the state-space growth problem: 1) distributed computing, 2) compact state-space storage techniques, and 3) alternate embedding strategies. Of course, these strategies may also reduce computation time as well. Regarding the first, much of the work to solve a PDPN can be shared among multiple computers. This is especially true for the solution of each SMC, which can be done in parallel with little or no communication or synchronization. As for the second, we anticipate significant reductions in memory requirements by employing decision diagrams like those in [39] and [42]. But what impact these novel data structures have on computational effort remains unknown until we incorporate the data structures into the PDPN solution algorithms. Finally, in the third, we observe that the solution complexity of the EMC and SMCs depends a great deal on how regeneration points are sampled. That is, what sampling, or "embedding", should be used to balance the effort between the many SMC solutions and the one EMC solution? Because the first two options seem to be more straightforward than the latter, we will spend more time investigating alternate embedding strategies in this section.

Before continuing, we should address the subject of execution policies, and how specific policies, namely, *resampling*, *enabling memory*, and *age memory*, may influence the choice or applicability of a particular solution method. It so happens that execution policies do not affect the applicability of the solution methods discussed in the next section for states that cause a mutually-exclusive enabling of transitions in $\mathcal{T}_D$ or $\mathcal{T}_C$. The nuances of each policy on transitions in $\mathcal{T}_D$ or $\mathcal{T}_C$, separately, is imparted into the DTMC or CTMC matrices, respectively, during construction. For the same reason, the effect that firing sequences $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ have on the phases of other transitions in $\mathcal{T}_C$ enabled with transitions in $\mathcal{T}_D$ also pose no problems—the effects are included by construction. Nor is there a problem with execution policies imposed on transitions as a result of firing sequences $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$ since the effects are imparted by the switching matrix when constructing the embedded states for the new regeneration periods—the analysis of which are separate from the current regeneration period. However, the execution policies can affect the applicability of a partic-

ular solution approach under certain other situations when transitions from $\mathcal{T}_D$ and $\mathcal{T}_C$ are enabled simultaneously.

Let us consider then the regeneration periods when transitions from both $\mathcal{T}_D$ and $\mathcal{T}_C$ are enabled. If any $t \in \mathcal{T}_D$ is specified with the resampling execution policy, enacted as a result of some sequence $s \in \mathcal{T}_C\mathcal{T}_Z^*$, then to maintain synchronization, all other transitions in $\mathcal{T}_D \backslash \{t\}$ enabled in the new marking must also resample a new firing delay. As previously discussed, such events are captured simply by appropriately constructing the SMC with absorbing states, entered when resampling occurs, and regarded, rightly so, as embedded states. If, in the extreme, all $t \in \mathcal{T}_D$ are forced to resample as a result of any transition firing, each SMC would consist of a single state leading to (embedded) absorbing states, and consequently, we could enjoy closed-form expressions for associated EMC entries. Unfortunately, such models do not occur often in practice.

When an enabling-memory execution policy is imposed on an enabled transition $t \in \mathcal{T}_D$, enacted as a result of some sequence $s \in \mathcal{T}_C\mathcal{T}_Z^*$, the phase of t does not advance at the next clock increment and t must resample a new firing delay when enabled once again in the future, or even if still enabled in the new marking. While disabled, t is assigned phase "$\bullet$". The enabling-memory policy is as convenient as the resampling policy because the phase of a transition can advance as long as it is enabled, and when it is not enabled, we know with certainty that its phase is "$\bullet$".

When an age-memory execution policy is imposed on an enabled transition $t \in \mathcal{T}_D$, enacted as a result of some sequence $s \in \mathcal{T}_C\mathcal{T}_Z^*$, the phase of t does not advance at the next clock increment, yet it retains its current value. From a modeling perspective, the work expended right up to the time of preemption is retained for $t \in \mathcal{T}_C$ with age memory. However, age memory is only approximately modeled with DPH transition because of the discretization in the basic step. For $t \in \mathcal{T}_D$ with age memory, only the work expended since the last clock increment is lost, not the work performed before then. But the approximation to a true age-memory execution improves as the firing delays become large relative to the clock period τ. From a solution perspective, age-memory policies usually require more computational effort and memory. The phase of an age-memory transition can advance for as long as it is enabled before firing, but we must remember its phase at the time when it becomes disabled. Then, the phase advancements can resume from the same point when the transition is enabled once again.

In the remaining text, we consider improvements to the straightforward stationary solution method presented up to now, which advances the discrete-time phases one basic step, τ, at a time. Instead, our proposed algorithm attempts to advance the discrete-time phases ahead in increments larger than the basic step size, when no $t \in \mathcal{T}_D$ can fire, and even further along in time while recording firing opportunities until all firing opportunities are discovered. However, when transitions in both $\mathcal{T}_D$ and $\mathcal{T}_C$ are enabled simultaneously, the method is somewhat different. The initially presented solution method allows transitions with any of the previously mentioned memory policies. However, for regeneration periods where the enabled DPH transitions are not marking dependent and do not have age memory then a slightly different, more efficient procedure can be employed. After motivating and developing the solution methods in this section, we later present in Section 4.2.3 procedures that realize the solution methods along with the procedures that are applicable when only transitions in $\mathcal{T}_D$ or $\mathcal{T}_C$ are enabled.

The PDPN solutions we seek, such as the state-occupancy probability distribution, need only span the unique markings in $\mathcal{R}$. But while obtaining this solution, we must contend with the expanded state space, $\mathcal{S} \subseteq \mathcal{R} \times \mathcal{D}$, resulting from the Cartesian product of the marking space and phase space. Because of the nature of phase-type firing delays, there will be many states in $\mathcal{S}$ that serve only to provide *delay* information, not unique marking information associated with actual transition firings. It would then be desirable to consider the expanded state space in a smart way, in hopes of eliminating as many "delay" states as possible. Ideally, we would like to choose an embedding that reduces the size of the embedded state space $\mathcal{E}$ to one that closely approaches the reachable set of markings, $\mathcal{R}$. In addition to reducing the memory requirements, a reduced $\mathcal{E}$ set can have a significant effect in the per-iteration complexity and the convergence rate when computing the stationary solution of the EMC. To this end, we propose a new embedding technique that observes the PDPN stochastic process at regeneration times that coincide with actual PN transition firings as much as possible, thereby eliminating as many "delay" states as possible. We call this technique *embedding with elimination*.

The technique is straightforward when only $\mathcal{T}_D$ or only $\mathcal{T}_C$ transitions are enabled. In such cases, the problem of studying the SMC becomes a TTA problem discussed in Chapter 2 instead of the usual transient solution discussed in section 4.1.1. However, when transitions from both $\mathcal{T}_D$ and $\mathcal{T}_C$ are simultaneously enabled, the technique becomes more complicated. While the costs associated with studying the SMCs can change, for better or worse depending on the model, we predict that the reduction in the EMC solution will net an overall cost reduction in many cases. The technique will be developed in detail in the remaining text of this section, culminating into a stationary solution algorithm presented at the end.

In the following, we denote EMC states as the tuple (i, a). The "i" part identifies just the PN *marking* if only Expo transitions are enabled; otherwise, i identifies a *state* composed of the marking and continuous-time phase information of the expanded CTMC. The "a" part identifies the discrete-time phase information (the RFT) associated with transitions in $\mathcal{T}_D$ other than Geom$(\cdot, 1)$ and Const(1). RFT state information for Geom$(\cdot, 1)$ and Const(1) is unnecessary since Geom$(\cdot, 1)$ is memoryless and Const(1) has only a one-step duration within a given state. Distinguishing the phase values "1" and "$\bullet$" can be done, instead, based on $\mathcal{F}(\cdot)$ membership. Explicit consideration of continuous-time phase information is not needed since the PH behavior is just expanded into a larger CTMC, the solution of which is the same. That is, whether there are PH transitions or just Expo transitions does not affect the analysis; it only enlarges the SMC state space. The information that is essential to the analysis is the state of the SMC (the i part) and the state of a DTMC (the a part), defined on the applicable discrete-time phase space. We analyze the DTMC and CTMC components in concert to construct the EMC.

4.2.1 Embedding with Elimination

The PDPN stationary solution algorithm based on Markov renewal theory can be summarized in following steps:

1. Explore from some known embedded state $(i, a) \in \mathcal{E}$.
2. The stochastic matrix specifying the EMC can be constructed one row at a time by studying the regeneration period (the time until the next regeneration point) defined for each embedded state (i, a) on which it depends.
3. Each regeneration period is studied by performing a transient analysis on the SMC that evolves between regeneration points in time.
4. There are three possible regeneration period cases to analyze:
   - DTMC (only $t \in \mathcal{T}_D$ enabled),
   - CTMC (only $t \in \mathcal{T}_C$ enabled), or
   - DTMC interacting with a CTMC (transitions enabled in both $\mathcal{T}_D$ and $\mathcal{T}_C$)
5. Repeat for each newly discovered embedded state until no others are found.
6. Solve the EMC for its stationary solution, Equation 4.4
7. Obtain the stationary solution of the actual semi-regenerative process by conversion, Equation 4.5, with expected sojourn times in SMC states for each regeneration period, Equation 4.3.

The regeneration period is characterized by the embedded state $(i, a) \in \mathcal{E}$ entered at the time of regeneration and from which the period emanates. The "i" part alone can uniquely identify the SMC since it is from state i that the reachability graph with state space $\mathcal{S}_i$ evolves as a result of firing sequences $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$. The infinitesimal generator matrix $\boldsymbol{Q}_i$ is then constructed from the reachability graph. The state of "a" does not change until the next clock increment; it can be set aside and later re-attached to the SMC state at the next regeneration point when the appropriate firing sequence $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$ are applied to discover the next set of embedded states. Let us now consider the three possible regeneration periods characterized by having *DTMC only*, *CTMC only*, or *both DTMC and CTMC* dynamics.

■ *The DTMC only case.* We could employ a *single-step* approach, observing regenerations every τ units of time so that the DTMC defined on the phase space $\mathcal{D}_a$ (the set of phases reachable from a) of transitions in $\mathcal{T}_D$ is identical to the EMC. Then we have the nice property that $\mathrm{E}[\, T_1 \mid X_0 = (i, a)\,] = \tau$, a constant.

The state of i does not change between steps and is, therefore, implied instead of explicitly stored in the DTMC state space. But this approach would also capture state transitions in the EMC that only change the discrete phase, not the marking. Thus, we consider this approach potentially inefficient.

We could take the previous approach a bit further by observing regenerations at times when there is *at least one* state where some $t \in \mathcal{T}_D$ can fire. This approach was discussed

in [5]. Here the EMC would be identical to a discrete-time semi-Markov chain where the sojourn times in states would not be constant. This has the potential of reducing a number of the ϵ transitions (phase change only) from the EMC, but when DPH firing delays with *selfloops* and *circuits* are present, e.g., Geom distributions, there will still be state transitions (and therefore extra states) where only the phases change.

A better approach would be to observe the DTMC at times when the firing of some $t \in \mathcal{T}_D$ is certain. This approach eliminates all ϵ transitions from the EMC. The DTMC is defined only on the phase space $\mathcal{D}_a$. The random variable T_1 is defined as the TTA of the DTMC with the fire-enabling, "0 phase" states treated as absorbing states.

Even better, we might observe the DTMC at times when the firing of some $t \in \mathcal{T}_D$ results in a new marking that enables some $u \in \mathcal{T}_C$ is certain. Regenerations at these random times T_1 are marked by the start of a CTMC. This approach not only eliminates all ϵ transitions from the EMC, but also skips over state changes that could just as easily be studied in the SMC constructed as a discrete-time semi-Markov chain (as opposed to the EMC that will be solved for its stationary solution later). Removing these states/transitions from the EMC will reduce the per iteration cost when computing the EMC stationary solution with an iterative method and should improve the convergence rate as demonstrated in [43] due to the smaller state space $\mathcal{E}$. We will discuss the convergence rate in more detail in Section 4.3 where we analyze our proposed solution algorithm. The random variable T_1 is defined here as the TTA of the DTMC with the $\mathcal{T}_C$-enabling states treated as absorbing states.

■ *The CTMC only case.* We could employ a similar *single-step* approach here as well, observing regenerations at times immediately after each state transition, as proposed in the literature for DSPNs. This implies that $\mathrm{E}[\, T_1 \mid X_0 = (i,a)\,]$ would be the expected sojourn in CTMC state (i,a). Although the (i,a) row calculation of the EMC would have a closed form, this savings could easily be lost in general when it comes time to compute the stationary solution of the much larger EMC. If the CTMC has an *expanded* state space due to PH firing delays, then we have, again, the same inefficiencies as for the single-step approach in the *DTMC only* case. The remedy is the same as well. That is, we could observe the CTMC at times when the firing of some $t \in \mathcal{T}_C$ is certain. Such an approach eliminates ϵ transitions from the EMC that only update the continuous-time phase information. Consequently, the random variable T_1 is defined as the TTA of the CTMC with the 0-phase states treated as absorbing states, just as in the related *DTMC only* case.

But again, as in the *DTMC only* case, a better approach might be to observe the CTMC at times when the firing of some $t \in \mathcal{T}_C$ results in a new marking that enables some $u \in \mathcal{T}_D$. Regenerations at these random times T_1 are marked by the start of a DTMC. This approach also eliminates all ϵ transitions from the EMC, and skips over state changes that could just as easily be studied in the SMC constructed as a continuous-time semi-Markov chain. And, removing these states/transitions from the EMC will also reduce the per iteration cost when computing the EMC stationary solution with an iterative method and should improve the convergence rate. The random variable T_1 is defined here as the TTA of the CTMC with the $\mathcal{T}_D$-enabling states treated as absorbing states.

■ *Both DTMC and CTMC case.* Here, the underlying process is semi-regenerative, formed from the interdependent, simultaneous evolution of the DTMC and CTMC associated with the transitions in $\mathcal{T}_D$ and $\mathcal{T}_C$, respectively. Although a CTMC is memoryless for all time, we must nevertheless observe the process in-sync with the $\mathcal{T}_D$ clock advancements

due to the possible interaction between the DTMC and CTMC. If we employ a *single-step* approach here as well so that regenerations are observed every τ increment in time, the subordinate process will be identical to the CTMC and the EMC can be constructed by solving the CTMC in the transient at time τ. The next embedded state is obtained by incrementing the clock and then applying the resulting CTMC state probability vector to the switching matrix $\boldsymbol{\Delta}$ of possible state changes due to firing sequences $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$.

The DTMC aspect is defined on the discrete-time phase space only, which does not change between clock increments. Therefore, the "a" part of the state is implicit and is not included in the construction of the SMC states occupied between clock increments. The expectation $\mathrm{E}[\, T_1 \mid X_0 = (i, a)\,]$ is equal to τ, a constant, for *isochronous* PDPNs only; transitions in $\mathcal{T}_D$ may be preempted in all other classes, causing $\mathrm{E}[\, T_1 \mid X_0 = (i, a)\,]$ to be less than τ. Just as in the previous single-step approaches in both the *DTMC only* and *CTMC only* cases, this approach will also capture ϵ transitions in the EMC, both discrete- and continuous-time phase changes. Worse, the embedded states at each step are formed from the Cartesian product of possible *next states* originating from CTMC states occupied at time τ and the possible *next phases* reached after the clock advance and, if transitions fire, after execution policies are applied. This approach can expand the size of the EMC in a serious way, and therefore, we consider it very inefficient.

To illustrate, consider our running example for yet another set of timing constraints specified in Figure 4.8. Figure 4.9 illustrates how the EMC would evolve from state

$$(m_1 m_2 m_3 m_4 m_5 m_6 \phi_1 \phi_2) = (11100032) \in \mathcal{E}$$

if the actual (semi-regenerative) process was observed using this single-step approach. Note that phase information $(\phi_1 \phi_2)$ for only transitions t_1 and t_2 is needed, which is shown in the larger circle offset from the marking information contained in the smaller circle. The SMC shown to the right is associated with the upper row of states, which enables the Expo transition t_3. The dotted arc labeled t_3 signifies that the sojourn time in state $(001110 \bullet\bullet)$ is exponentially distributed, unlike the other states, which have constant τ sojourn times.

Starting from $(i, a) \in \mathcal{E}$, we observe that *between* clock increments, the process can move like $(i, a) \xrightarrow{s} (k, a)$ where $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ and $k \in \mathcal{S}_i$ with probability one. Although the CTMC may disable transitions in $\mathcal{T}_D$ during this interval $[0, \tau)$, we need not consider this change to the discrete-time phases until the next time the clock advances; hence, "a" is considered unchanged until then. Also, in one clock step, the EMC may move between embedded states like $(i, a) \xrightarrow{s,\epsilon} (k, b)$, where $(i, a), (k, b) \in \mathcal{E}$, $k \in \mathcal{S}_i$, $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ and no $t \in \mathcal{T}_D$ fires (denoted by ϵ). In such cases, we know that

$$\Pr\{\, (i, a) \xrightarrow{s,\epsilon} (k, b)\,\} = \Pr\{\, i \xrightarrow{s} k \text{ in time } \tau\,\} \cdot \Pr\{\, a \to b \text{ in one clock step}\,\}$$

and by summing (and effectively lumping) over all states $k \in \mathcal{S}_i$ we have the result

$$\Pr\{\, (i, a) \xrightarrow{\epsilon} (\mathcal{S}_i, b)\,\} = \Pr\{\, a \to b \text{ in one clock step}\,\}.$$

Therefore, we can avoid the Cartesian product of CTMC states with the discrete-time phase information along sample paths with ϵ transitions by combining all CTMC states with the same discrete-time phase component into a single state as illustrated in Figure 4.10, where the arcs correspond to EMC state transitions only.

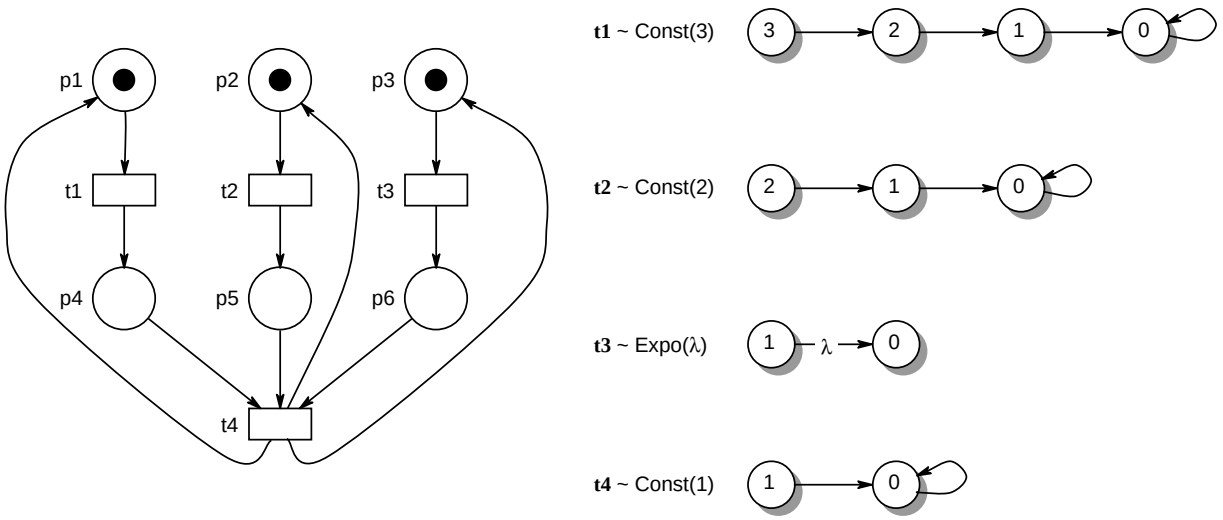


Figure 4.8: Example model.

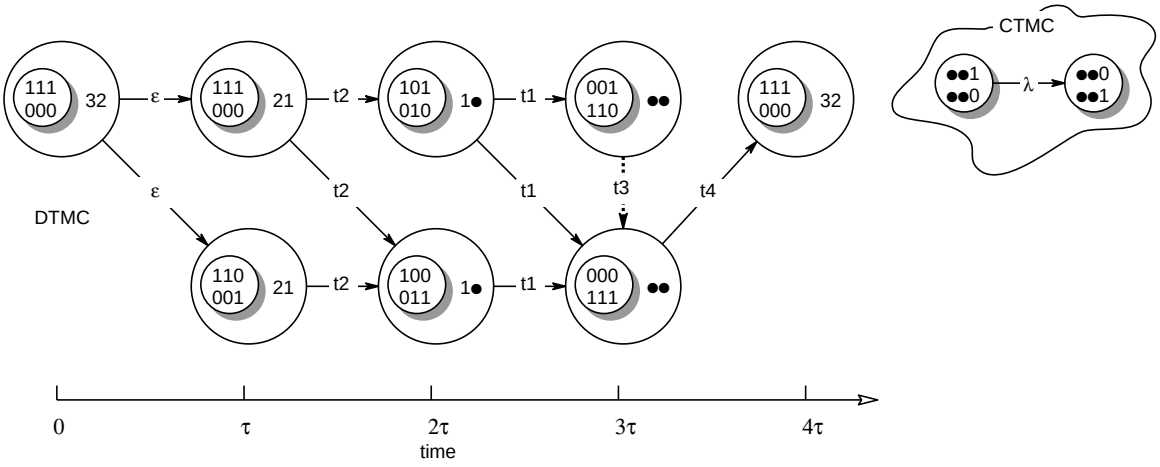


Figure 4.9: Single-step EMC model.

But, this method is most beneficial when we can study such regeneration periods over the entire possible range of T_1. More specifically, it is preferable to return to Equation 2.23 and replace $d\Pr\{\, T_1 \leq \upsilon \mid X_0 = (i,a)\,\}$ with the pmf derived from the firing time of any DPH transition enabled during the regeneration period. This pmf can be computed while concurrently advancing the stochastic matrices $\boldsymbol{D}^t$, which defines the firing-delay of each enabled $t \in \mathcal{T}_D$, until any one of the transitions can fire with nonzero probability.

Suppose $\tilde{\mathcal{T}} \subseteq \mathcal{T}_D$ is the set of DPH transitions enabled in the starting state $(i,a) \in \mathcal{E}$. Then $\tilde{\mathcal{T}}$ is the largest set of enabled DPH transitions over the regeneration period defined by embedded state (i,a). One or more $t \in \tilde{\mathcal{T}}$ may be disabled during the regeneration period but not re-enabled. Nor can any other transition $t \in \mathcal{T}_D \backslash \tilde{\mathcal{T}}$ become enabled; otherwise, the model would violate the conditions that insures a semi-regenerative process. Let $\boldsymbol{\phi}^t$ denote the probability vector of phases for each enabled $t \in \mathcal{T}_D$ that requires phase information.

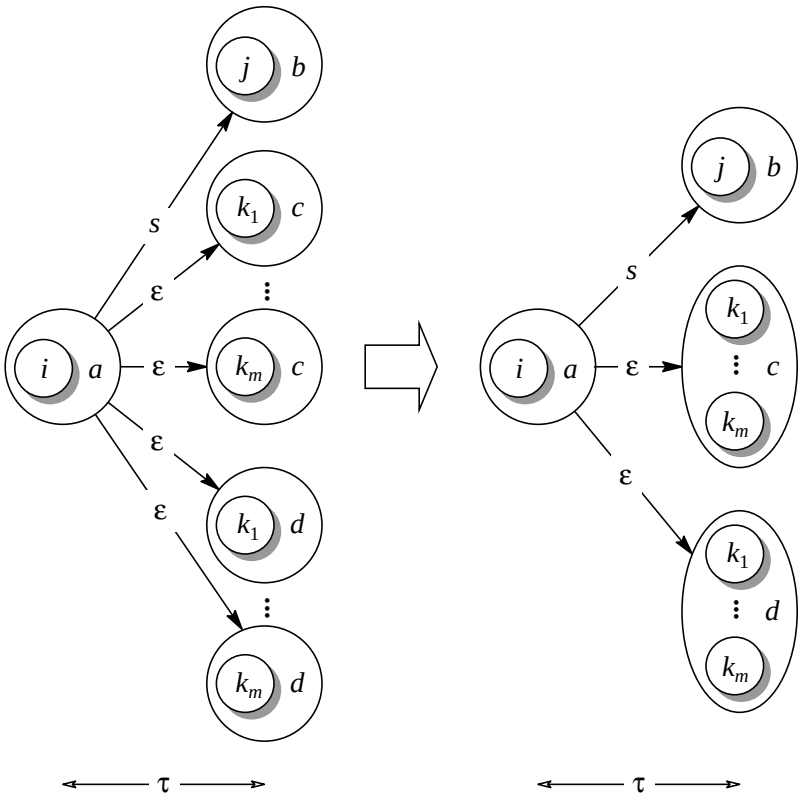


Figure 4.10: Reducing the EMC.

The vector $\mathbf{1}_\bullet$, indicating that the phase is "$\bullet$" with probability one, and zero for all other phases, can be substituted in place of $\boldsymbol{\phi}^t$ for $t \in \mathcal{T}_D \backslash \tilde{\mathcal{T}}$, which are not enabled. For each $u \in \tilde{\mathcal{T}}$, $\boldsymbol{\phi}^u$ is initialized to $\mathbf{1}_o$ for some appropriate phase "o" so that the Kronecker product, $\otimes_t \boldsymbol{\phi}^t$, $\forall\, t \in \mathcal{T}_D$, corresponds to the phase vector identified by "a". Then starting from the initial phase at time 0 (the start of the regeneration period), the respective phases at time step $\theta = n\tau$, $n \in \mathbb{N}$, can be computed using the power method,

$$\boldsymbol{\phi}^t(\theta) = \boldsymbol{\phi}^t(\theta - \tau)\, \boldsymbol{D}^t,$$

until at least one $t \in \tilde{\mathcal{T}}$ is ready to fire; that is, its phase is zero and it is enabled in the CTMC state, k, occupied at time θ:

$$\exists\, t \in \tilde{\mathcal{T}} : \phi_0^t > 0 \wedge t \in \mathcal{F}(k).$$

Of course, more than one transition in $\tilde{\mathcal{T}}$ may be able to fire at the same time.

Accordingly, the random variable T_1 can take on any such value of θ that coincides with a nonzero probability of some $t \in \tilde{\mathcal{T}}$ firing given that $X_0 = (i, a)$. The conditional pmf of T_1 is then the probability of entering all such firing states over an interval that accumulates a total probability mass of one, or at least close enough for practical purposes. The full state vector and associated probability mass at the time when a firing can occur, is determined by appropriately combining the probabilities of the CTMC states with that of the discrete-time phases. CTMC states, k, and associated probabilities, π_k, that enable transitions in $\tilde{\mathcal{T}}$, can

be combined with the new phase vectors which result from incrementing the clock, except for the transitions $t \in \tilde{\mathcal{T}}$ that can now fire: these phase components must be set to $\boldsymbol{\phi}_0^t$, a vector with the 0^{th} element equal to ϕ_0^t and all other elements equal to zero. The respective phase component of transitions in $\tilde{\mathcal{T}}$ not enabled in the same CTMC states are set to $\phi_\bullet^t$. The probability mass for each such firing event found must then be discarded before incrementing the clock forward in time in search of the next firing event. In this way, the probability mass associated with each of the states allowing transitions in $\tilde{\mathcal{T}}$ to fire will form a conditional pmf for T_1 and will equal one when summed together.

In general, the DTMC aspect of our model may be affected by the SMC evolution, the CTMC aspect. State transitions in the CTMC that cause the enabled DPH transitions to resample is one way. But these effects are easily taken into consideration by treating the CTMC states entered from such transitions as absorbing states. As previously discussed, the resampling events and probabilities are then trapped and the absorbing states are considered as embedded states, members of set $\mathcal{E}_i$ presented earlier, thereby concluding any further consideration along such sample paths. For later reference, we let set $\mathcal{E}_i$ contain all such absorbing states. Another possible effect on the DTMC behavior comes from the disabling of DPH transitions as the CTMC evolves. This implies that the advancement of each $\boldsymbol{D}^t$ over time is conditioned on $t \in \tilde{\mathcal{T}}$ being enabled at the time of each clock increment, a function of the CTMC state occupied at such times. In the extreme, there may exist CTMC states that disable all transitions in $\tilde{\mathcal{T}}$. These CTMC states would be made absorbing when constructing the CTMC since they can also be considered embedded states. Consequently, these absorbing states too are made members of $\mathcal{E}_i$. Finally, the DTMC may be marking dependent, and therefore depend on the state of the CTMC at each clock increment. That is, the phase advancements are *conditioned* on the state of the CTMC at the time of each clock advance. In such situations, the CTMC must be solved first at time τ so that the discrete-time phases can be advanced based on the state of the CTMC.

When constructing the state vectors and associated probability mass for each $t \in \tilde{\mathcal{T}}$ firing opportunity at some time θ, we must be sure to combine the discrete-time phase vector associated with the firing, as previously defined, with the state probabilities $k \in \mathcal{S}_i$ in which the discrete-time phase is valid. Let us suppose that $\mathcal{T}_D = \{t_1, t_2, \ldots, t_{|\mathcal{T}_D|}\}$. Then, by defining

$$\boldsymbol{\psi}_k^t = \begin{cases} \boldsymbol{\phi}_0^t & \text{if } t \in \mathcal{F}(k) \wedge \phi_0^t > 0 \\ \boldsymbol{\phi}^t & \text{otherwise} \end{cases}$$

where $\boldsymbol{\phi}_0^t = [\phi_0^t, 0, \ldots, 0]$, the complete firing state and associated probability mass is determined from

$$\sum_{k \in \mathcal{S}_i} \boldsymbol{\pi}_k \otimes \boldsymbol{\psi}_k^{t_1} \otimes \boldsymbol{\psi}_k^{t_2} \otimes \cdots \otimes \boldsymbol{\psi}_k^{t_{|\mathcal{T}_D|}}$$

where $\boldsymbol{\pi}_k$ is a vector of all zeros except for the k^{th} entry, which is π_k instead.

The CTMC must be observed at times coincident with the clock increments to ensure that all DPH transition firing opportunities can be discovered and carried out along the way in case the outcomes happen to modify the CTMC specification. During the regeneration interval of interest, the CTMC can change only as a result of $t \in \tilde{\mathcal{T}}$ firings, not by mere

phase advancements. By devising a procedure that computes the conditional pmf for T_1 first, setting aside the consideration of the $t \in \tilde{\mathcal{T}}$ firings, we are certain that the CTMC is the same while incrementing the clock in search of more $t \in \tilde{\mathcal{T}}$ firing opportunities.

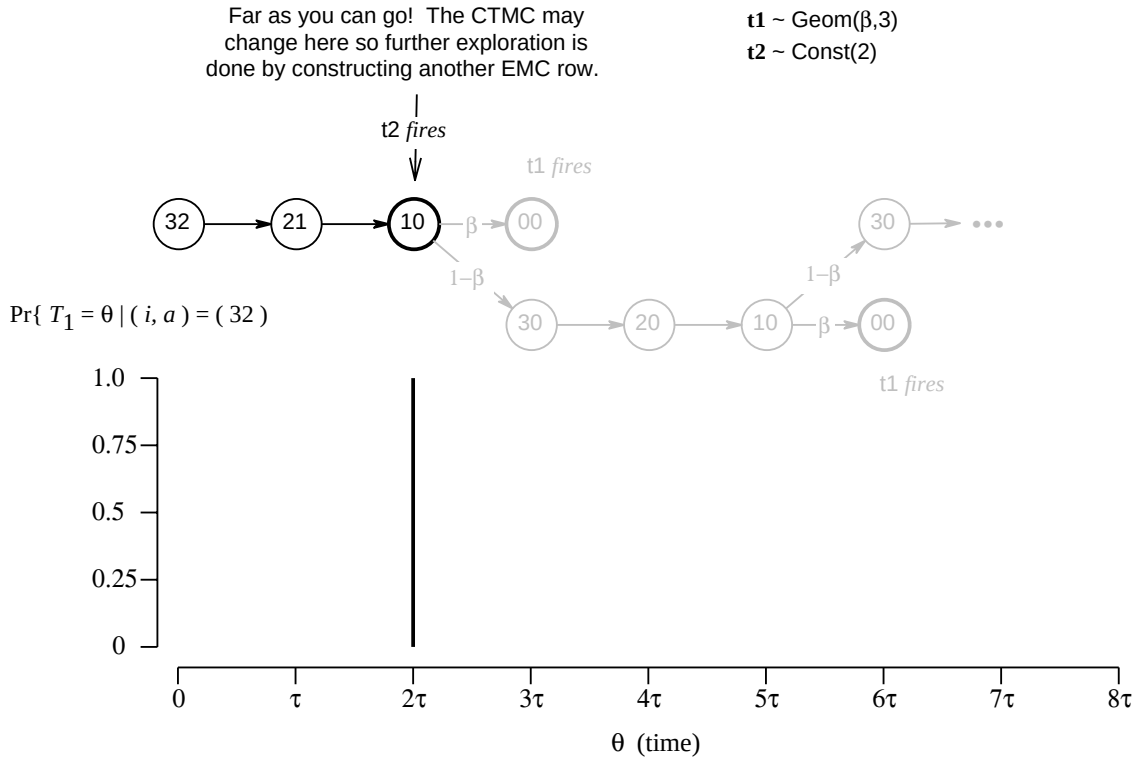


Figure 4.11: Multi-step observation example.

Consider the example PDPN in Figure 4.8 except assume $t_1 \sim \text{Geom}(\beta, 3)$ and $t_2 \sim \text{Const}(2)$. Figure 4.11 shows the DTMC evolution given the initial state

$$(m_1 m_2 m_3 m_4 m_5 m_6 \phi_1 \phi_2) = (11100032).$$

Using our multi-step approach, with $\tilde{\mathcal{T}} = \{t_1, t_2, t_3\}$, we would stop at time step 2τ since here the probability of observing the firing of a transition $t \in \tilde{\mathcal{T}}$ is *one*. In this case, t_2 fires and we must explore the grayed portion starting from the embedded state reached as a result of the firing. The EMC originating from state (i, a) is portrayed in Figure 4.12. Although there is only one possible next phase, the process can be in any of the two CTMC states when the clock advances, thereby resulting in two possible next states. The expected sojourn time in (i, a) is 2τ.

Again, consider the running example except that now we assume $t_1 \sim \text{Geom}(\beta, 3)$ and $t_2 \sim \text{Geom}(\alpha, 2)$. Figure 4.13 portrays the exploration of the DTMC on the phase space starting from the same embedded state as before, and shows the resulting, conditional pmf for T_1 when $\alpha = 0.4$ and $\beta = 0.2$ out to 8 clock steps. The probability mass beyond 8τ diminishes quickly to zero. In this example, we need not stop at time step 2τ since the

probability of observing the firing of a transition $t \in \tilde{\mathcal{T}}$ (t_2 again) is $\alpha < 1$. If we stop the construction of the EMC row (i, a) at this point, we would have entries for $(\phi_1\phi_2) = (12)$ which would have to be explored in other EMC rows. So if we need to explore from phase vector $(\phi_1\phi_2) = (12)$ anyway, we may as well do it while building row (i, a). This way, the intermediate "delay" states are eliminated from the EMC, potentially a real advantage when it comes time to compute its stationary solution.

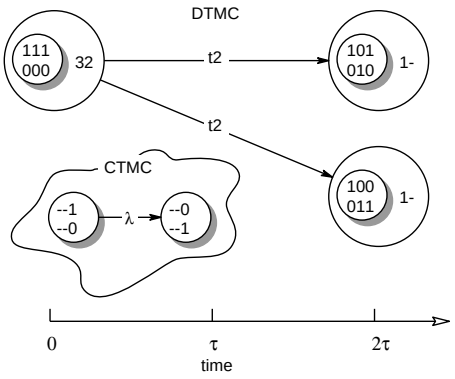


Figure 4.12: Multi-step EMC model.

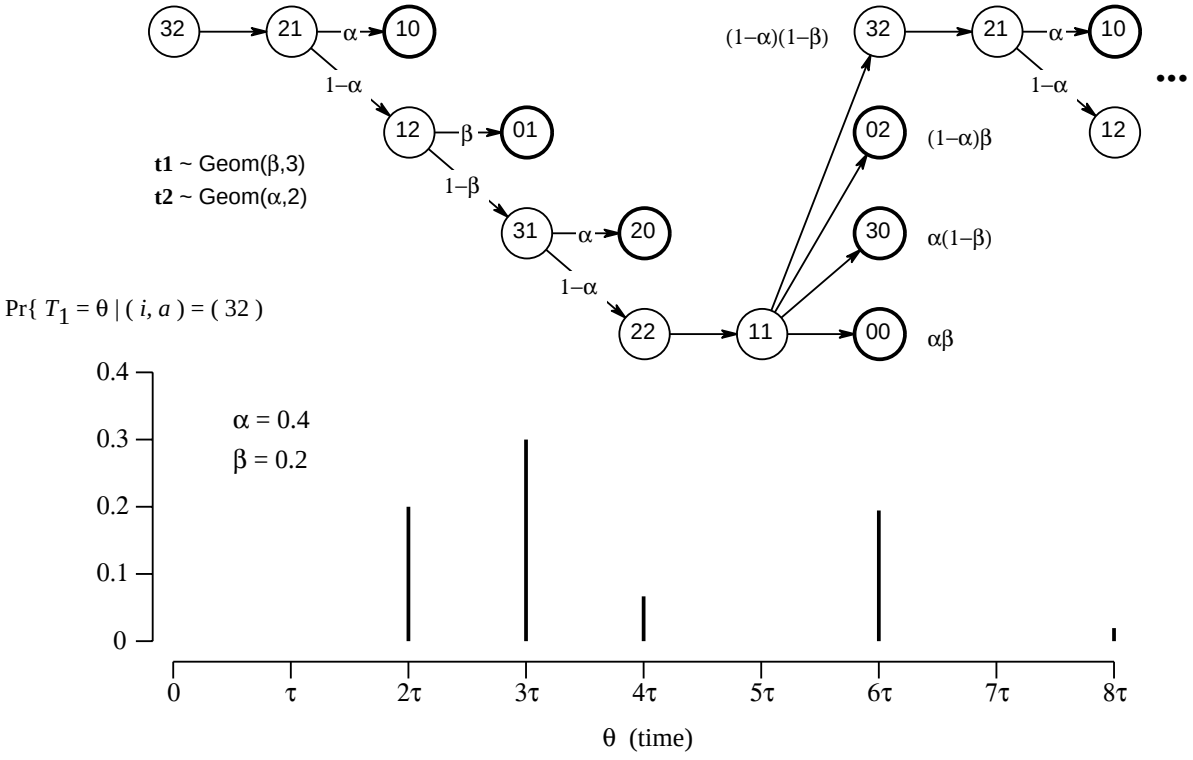


Figure 4.13: Multi-step example with two Geom transitions.

4.2.2 Conversion Matrix

Before presenting the stationary solution algorithm developed from the previous investigation, we should consider the matter of converting the stationary solution of the EMC to that of the actual process. Recall that the conversion matrix, $\boldsymbol{h} = [h_{ik}]$, $i \in \mathcal{E}$, $k \in \mathcal{S}_i$, defined as

$$h_{ik} = \mathrm{E}[\,\text{sojourn in } k \text{ during } [0, T_1) \mid X_0 = (i, a)\,],$$

is usually a real matrix of dimension $|\mathcal{E}| \times |\mathcal{S}|$. However, if this mapping is utilized, the probability distribution over states $i \in \mathcal{E}$ would be redistributed onto all reachable states, including those reached via ϵ transitions. In other words, the stationary solution vector of the actual semi-regenerative process would contain every state $i \in \mathcal{E}$ and every state $k \in \mathcal{S}_i$ reachable between regenerations. This mapping to such a large solution vector would be memory inefficient, causing a loss of some of the benefit obtained by eliminating the extra "delay" states during the regeneration period studies. Obviously, in the end, all we wish to know is the probability distribution among the *unique* reachable markings within $\mathcal{R}$, on which most measures (other than impulse rewards) are defined.

Of course, we can simply *compress* the solution vector by accumulating the probabilities of states with the *same* marking, but different RFT content, into a single, *lumped* state. However, waiting until the end to do this is too late since we have already consumed the memory with the large conversion matrix.

Alternatively, we can employ the measure-based technique presented in [44] that "distills" the *rewards** of all states into rewards for the embedded states alone. In this way, we need not store a conversion matrix at all, only vectors of size $\mathcal{E}$. But, this technique has the disadvantage in that the results are limited to just the rewards defined when the solution is computed.

We could instead avoid some of the memory inefficiency while still allowing for the storage of a probability vector that does not limit the measures that can be computed from it later by utilizing a conversion matrix on $\mathcal{E} \times \mathcal{R}$, which maps to the set of unique reachable markings. This can be done *during* the study of each regeneration period, which constructs a row of the EMC. We simply accumulate the expected sojourn times of SMC states with the *same* marking together. If the conversion matrix $\boldsymbol{h}$ is still too large to keep in memory, we can at least find solace in knowing that $\boldsymbol{h}$ is constructed one row at a time and is used just once at the end. Therefore, the $\boldsymbol{h}$ matrix can be cached to disk if necessary, and only a minor performance impact should be observed because of the sequential construction and later recall of its entries [45].

*We refer to the *rate* rewards, quantities $\rho_i \in \mathbb{R}$ accumulated during the entire time state $i \in \mathcal{S}$ is occupied and are usually defined on markings at the net level.

4.2.3 Presentation of the Algorithm

This section proposes a stationary solution algorithm applicable to PDPNs with underlying semi-regenerative processes. We assume that the PDPN model is restricted to one of the classes: isochronous, synchronous, or mixed PDPN. Checks can be made during the construction and analysis of the underlying stochastic process to ensure that it is indeed semi-regenerative; however, those checks are omitted here for the sake of clarity. Checks on the PDPN incidence matrix may be investigated in the future that would ensure that transitions in $\mathcal{T}_D$ maintain synchronization during execution. In this way, examination of the PDPN model itself would determine which solution algorithm is appropriate. This solution algorithm realizes the "embedding with elimination" method presented earlier. We have decomposed the overall solution algorithm into four parts: the main algorithm and the three procedures SolveBoth, SolveDTMC, and SolveCTMC, which consider the three possible solution approaches as outlined in Section 4.2.1.

The main algorithm (4.2.1) is responsible for exploring and building the embedded state space $\mathcal{E}$ by analyzing each regeneration period that originates from known embedded states, constructing the EMC probability matrix $\boldsymbol{\Pi}$ one row at a time, and placing newly discovered embedded states in $\mathcal{U}$ for later consideration. The algorithm calls one of the three procedures in turn to perform the actual analysis of each regeneration period. The conversion matrix $\boldsymbol{h}$ is constructed within each of the three procedures. After constructing the EMC and computing its stationary solution, the stationary probability distribution of the actual process with state space $\mathcal{S}$ is finally computed using $\boldsymbol{h}$. The state space $\mathcal{S}$ is implicitly constructed during this conversion.

Some comments should be made before proceeding. First, note that in Algorithm 4.2.1, i itself is assumed to represent both marking and phase (PH and DPH), but in the procedures that follow, i may represent only a component of the state: either the marking and continuous-time phase information or the marking alone, as discussed in the beginning of Section 4.2. The semantics will be made clear in each case to avoid confusion. Notice also in Algorithm 4.2.1 that the parameter to the set of enabled transitions, $\mathcal{F}$, is a *state*, consisting of marking and phase information, instead of just a marking as previously defined. Of course, only the marking information is considered when constructing the set. Second, the switching matrix $\boldsymbol{\Delta}$, used explicitly in procedure SolveBoth and implicitly in procedure SolveDTMC should be defined using the Traverse algorithm presented in [24] for DDSPNs. Constructing $\boldsymbol{\Delta}$ using the Traverse algorithm ensures that the PDPN model is "well defined" with respect to contemporary transition firings. Third, in the SolveBoth procedure, we use an extended version of the switching matrix $\boldsymbol{\Delta}$ defined earlier. The extended version considers the firing outcome of multiple transitions on a given set of markings instead of the firing outcome of just one transition. In this way, given a probability distribution of states that enable different sets of transitions, the multiple outcomes from simultaneously firing all such enabled-transition sets can be determined in a single step.

The procedures SolveDTMC and SolveCTMC are much alike. They analyze regeneration periods when only a DTMC or CTMC exist, respectively, following the "embedding with elimination" approach of Section 4.2.1. Consequently, the procedures consider states within the subordinate DTMC or CTMC that change the situation to an interacting DTMC and CTMC, from a DTMC to a CTMC, or vice versa. Such states, referred to as "stopping

states", are by definition embedded states. The "stopping states" are treated as absorbing states and the vector $\tilde{\boldsymbol{\sigma}}$ of expected times in the nonabsorbing states until absorption into an embedded state is computed. Then, the probability Π_{ij} of transitioning from the embedded state i, entered at the start of the regeneration period, to one of the next embedded states j, marking the next regeneration period, can be computed from $\tilde{\boldsymbol{\sigma}}$. This method eliminates all states other than the absorbing ones from the EMC, which would otherwise be included if the EMC were observed after each state transition during the same period.

Algorithm 4.2.1 PDPN stationary solution algorithm

1: $\mathcal{E} \leftarrow \emptyset$ — set of embedded states
2: Let $\boldsymbol{\Pi} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ where $\Pi_{ij} = \Pr\{\, X_1 = j \mid X_0 = i \,\}$ — EMC matrix
3: Let $\mathcal{U}$ contain *known* embedded states — set of unexplored states
4: **while** $\mathcal{U} \neq \emptyset$ **do**
5: Choose and remove $i \in \mathcal{U}$ and place in $\mathcal{E}$ — construct row i of $\boldsymbol{\Pi}$
6: **if** $\mathcal{F}(i) \cap \mathcal{T}_C = \emptyset$ **then** — DTMC only
7: SolveDTMC
8: **else if** $\mathcal{F}(i) \cap \mathcal{T}_D = \emptyset$ **then** — CTMC only
9: SolveCTMC
10: **else** — both DTMC and CTMC
11: SolveBoth
12: **end if**
13: Place in $\mathcal{U}$ all j not already in $\mathcal{E} \cup \mathcal{U}$ such that $\Pi_{ij} > 0$
14: **end while**
15: Solve $\boldsymbol{x}\boldsymbol{\Pi} = \boldsymbol{x}$ subject to $\sum_{i\in\mathcal{E}} x_i = 1$ — stationary solution of EMC
16: Compute the stationary solution $p_j \leftarrow \lim_{\theta\to\infty} \Pr\{\, X(\theta) = j \,\},\ j \in \mathcal{S}$ from

$$p_j \leftarrow \frac{\sum_{i\in\mathcal{E}} x_i h_{ij}}{\sum_{k\in\mathcal{S}} \sum_{i\in\mathcal{E}} x_i h_{ik}}$$

Procedure 4.2.2 Regeneration period solution for DTMC only case

procedure SolveDTMC **is**

1: Construct stochastic matrix $\tilde{\boldsymbol{\Phi}}$ of the expanded DTMC
by applying firing sequences $s \in (\mathcal{T}_D \cup \mathcal{T}_Z)^*$ starting from $(i,a) \in \mathcal{E} \subseteq \mathcal{R} \times \mathcal{D}$
until *stopping states* are reached that enabled some $t \in \mathcal{T}_C$;
Make the *stopping states* in $\tilde{\boldsymbol{\Phi}}$ absorbing.

2: Solve $\tilde{\boldsymbol{\sigma}}(\mathbf{I} - \tilde{\boldsymbol{\Phi}}) = \mathbf{1}_{(i,a)}$

3: **for** all (k,c) elements of $\tilde{\boldsymbol{\sigma}}$, $k \in \mathcal{R}$, $c \in \mathcal{D}$, **do**

4: $\quad h_{(i,a)k} \leftarrow h_{(i,a)k} + \tilde{\sigma}_{(k,c)}$

5: $\quad$ **for** all (j,b) such that $\mathcal{F}(j) \cap \mathcal{T}_C \neq \emptyset$ **do**

6: $\quad\quad \Pi_{(i,a)(j,b)} \leftarrow \Pi_{(i,a)(j,b)} + \tilde{\sigma}_{(k,c)} \cdot \Pr\{\, (k,c) \rightarrow (j,b) \,\}$

7: $\quad$ **end for**

8: **end for**

Procedure 4.2.3 Regeneration period solution for CTMC only case

procedure SolveCTMC **is**

1: Construct generator matrix $\tilde{\boldsymbol{Q}}$ of the expanded CTMC
by applying firing sequences $s \in (\mathcal{T}_C \cup \mathcal{T}_Z)^*$ starting from $(i,a) \in \mathcal{E} \subseteq \mathcal{R} \times \mathcal{D}$
until *stopping states* are reached that enabled some $t \in \mathcal{T}_D$;
Make the *stopping states* in $\tilde{\boldsymbol{Q}}$ absorbing.

2: Solve $\tilde{\boldsymbol{\sigma}}\tilde{\boldsymbol{Q}} = -\mathbf{1}_{(i,a)}$

3: **for** all (k,c) elements of $\tilde{\boldsymbol{\sigma}}$, $k \in \mathcal{R}$, $c \in \mathcal{D}$, **do**

4: $\quad h_{(i,a)k} \leftarrow h_{(i,a)k} + \tilde{\sigma}_{(k,c)}$

5: $\quad$ **for** all (j,b) such that $\mathcal{F}(j) \cap \mathcal{T}_D \neq \emptyset$ **do**

6: $\quad\quad \Pi_{(i,a)(j,b)} \leftarrow \Pi_{(i,a)(j,b)} + \tilde{\sigma}_{(k,c)} \cdot \mathrm{rate}\{\, (k,c) \rightarrow (j,b) \,\}$

7: $\quad$ **end for**

8: **end for**

In SolveDTMC, the initial embedded state $(i,a) \in \mathcal{R} \times \mathcal{D}$ represents a marking, the "i" part, and discrete-time phase information, the "a" part. Information about continuous-time phases is omitted since all transitions in $\mathcal{T}_C$ are disabled. After constructing the absorbing DTMC, $\tilde{\boldsymbol{\sigma}}$ is computed in line 2 using Equation 2.2. Then each nonabsorbing state (k,c), a marking and DPH phase, is considered in turn with line 4 accumulating the results into each unique marking k. Lines 5 and 6 accumulate the EMC transition probabilities from (i,a) to each new embedded state (j,b). The transition probability $\Pi_{(i,a)(j,b)}$ for DTMCs is given by

$$\begin{aligned}
&\sum_{(k,c)} \mathrm{E}[\, \textit{visits} \text{ to } (k,c) \text{ until absorption} \,|\, \text{start in state } (i,a)\,] \cdot \Pr\{\, (k,c) \to (j,b)\,\} \\
&= \sum_{(k,c)} \sigma_{(k,c)} \cdot \Pr\{\, (k,c) \to (j,b)\,\},
\end{aligned}$$

which is computed in line 6.

The procedure SolveCTMC is almost identical to SolveDTMC except for a few subtle differences. First, the initial embedded state $(i,a) \in \mathcal{R} \times \mathcal{D}$ represents a marking, the "i" part, and continuous-time phase information, the "a" part. Information about discrete-time phases is omitted since all transitions in $\mathcal{T}_D$ are disabled. Second, $\tilde{\boldsymbol{\sigma}}$ is computed in line 2 using Equation 2.14. And third, the transition probability $\Pi_{(i,a)(j,b)}$ for CTMCs is given by

$$\begin{aligned}
&\sum_{(k,c)} \mathrm{E}[\, \textit{time} \text{ in } (k,c) \text{ until absorption} \,|\, \text{start in state } (i,a)\,] \cdot \mathrm{rate}\{\, (k,c) \to (j,b)\,\} \\
&= \sum_{(k,c)} \sigma_{(k,c)} \cdot \mathrm{rate}\{\, (k,c) \to (j,b)\,\},
\end{aligned}$$

which uses a "rate" instead of a probability since $\tilde{\boldsymbol{\sigma}}$ actually contains the expected *time* spent in states instead of the expected *visits* as in the DTMC case.

Procedures SolveDTMC and SolveCTMC are the easy ones; procedure SolveBoth is harder when making the same attempt towards eliminating embedded states. The reason, as discussed in Section 4.2.1, is that the underlying process is now semi-regenerative, unlike the special cases of a DTMC or CTMC. While realizing the technique referred to earlier as the "multi-step method", procedure SolveBoth must analyze the interdependent, simultaneous evolution of a DTMC on the discrete-time phase space and a CTMC, possibly expanded on the marking and continuous-time phase space. Of course, here the "DTMC" stems from the combined, possibly marking dependent, discrete-time phase evolution of enabled DPH transitions, $\tilde{\mathcal{T}}$, during the regeneration period. The same notation presented when developing this procedure in Section 4.2.1 is maintained here for clarity.

Procedure 4.2.4 General solution procedure when both DTMC and CTMC are active

procedure SolveBoth **is**

1: Given $(i,a) \in \mathcal{E}$, let $\mathcal{S}_i$ be the states reachable from i and construct the associated, possibly expanded, CTMC with state space $\mathcal{S}_i \cup \mathcal{E}_i$ and generator matrix $\boldsymbol{Q}_i$.
2: Given $\mathcal{T}_D = \{t_1, t_2, \ldots, t_{|\mathcal{T}_D|}\}$, let $\tilde{\mathcal{T}} = \mathcal{F}(i) \cap \mathcal{T}_D$ be the set of DPH transitions initially enabled at the same time in state i.
3: For each $t \in \tilde{\mathcal{T}}$, let $\boldsymbol{D}^t(\cdot) \in \mathbb{R}^{|\mathcal{D}^t| \times |\mathcal{D}^t|}$ be the, possibly marking dependent, firing-delay specification (a stochastic matrix) defined on the phase space $\mathcal{D}^t$, and set the 0^{th} row to zero, $\boldsymbol{D}^t_0(\cdot) \leftarrow \boldsymbol{0}$.
4: Let $\boldsymbol{\phi}^t \in \mathbb{R}^{|\mathcal{D}^t|}$ be the individual phase vectors $\forall\, t \in \mathcal{T}_D$ and initialize each so that vector $\otimes_t \boldsymbol{\phi}^t$ corresponds to phase "a".
5: Let $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{S}_i|}$ be the CTMC probability vector, and initialize, $\boldsymbol{\pi} \leftarrow \boldsymbol{1}_i$.
6: Let $\boldsymbol{\sigma} \in \mathbb{R}^{|\mathcal{S}_i|}$ be the CTMC cumulative probability vector, and initialize, $\boldsymbol{\sigma} \leftarrow \boldsymbol{0}$.
7: Let $\boldsymbol{\nu} \in \mathbb{R}^{|\mathcal{S}_i \times \mathcal{D}_a|}$ be the complete state probability vector, which accumulates state occupancy probabilities when some $t \in \tilde{\mathcal{T}}$ can fire, set initially to zero, $\boldsymbol{\nu} \leftarrow \boldsymbol{0}$.
8: **while** $\exists\, t \in \tilde{\mathcal{T}} : \left\|\boldsymbol{\phi}^t\right\|_1 > 10^{-d}$ **do** ... for as long as the possibility for $\tilde{\mathcal{T}}$ firings exist
9: $\quad \boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + \boldsymbol{\pi} \int_0^{\tau} e^{\boldsymbol{Q}_i u}\, du$ accumulate time spent in each CTMC state
10: $\quad \boldsymbol{\pi} \leftarrow \boldsymbol{\pi} e^{\boldsymbol{Q}_i \tau}$ CTMC state at *this* clock step
11: $\quad$ **for each** $t \in \tilde{\mathcal{T}} : \|\boldsymbol{\phi}^t\|_1 > 0$ **do**
12: $\quad\quad \boldsymbol{w} \leftarrow \boldsymbol{0}$
13: $\quad\quad$ **for each** $k \in \mathcal{S}_i$ **do**
14: $\quad\quad\quad \boldsymbol{v} \leftarrow \boldsymbol{\phi}^t\, \boldsymbol{D}^t(k)$ advance phase conditioned on current CTMC state
15: $\quad\quad\quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{v}\, \pi_k$ compute unconditional next-phase vector
16: $\quad\quad\quad$ **if** $v_0 > 0$ **then** Can transition t fire?
17: $\quad\quad\quad\quad \boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + \boldsymbol{\pi}_k \otimes \boldsymbol{\psi}_k^{t_1} \otimes \boldsymbol{\psi}_k^{t_2} \otimes \cdots \otimes \boldsymbol{\psi}_k^{t_{|\mathcal{T}_D|}}$,

$$\text{where } \boldsymbol{\psi}_k^t = \begin{cases} \boldsymbol{v}_0 & \text{if } t \in \mathcal{F}(k) \wedge v_0 > 0 \\ \boldsymbol{\phi}^t & \text{otherwise} \end{cases}$$

if yes then build state vector
18: $\quad\quad\quad$ **end if**
19: $\quad\quad$ **end for**
20: $\quad\quad \boldsymbol{\phi}^t \leftarrow \boldsymbol{w}$ new phase after clock increment
21: $\quad$ **end for**
22: **end while**
23: $\boldsymbol{\Pi}_{(i,a)} \leftarrow \boldsymbol{\nu}\boldsymbol{\Delta} + \boldsymbol{\pi}_{\{\mathcal{E}_i\}} \otimes \boldsymbol{\phi}^{(0)}$ construct EMC row (i,a) by applying switching matrix
24: $\boldsymbol{h}_{(i,a)} \leftarrow \boldsymbol{\sigma}$ the (i,a) row of the conversion matrix is identical to $\boldsymbol{\sigma}$

The generator matrix of the expanded CTMC, originating from state i, is denoted by $\boldsymbol{Q}_i$. During the regeneration interval of interest, the CTMC specified by $\boldsymbol{Q}_i$ can change only as a result of a DPH transition firing, not by mere phase advancements. Since we process the DPH transition firings at the end of the procedure, we are certain that the CTMC is

the same while advancing the clock in search of DPH transition firing opportunities, thereby computing the conditional pmf of T_1 marking the start of a new regeneration period.

The probability distribution of CTMC states is stored in the vector $\boldsymbol{\pi}$ and the cumulative probabilities in the same state up to the current time step are stored in the vector $\boldsymbol{\sigma}$. The phase vector $\boldsymbol{\phi}^t$ for each $t \in \mathcal{T}_D$ is initialized so that the Kronecker product will match the vector identified by "a". Then, the entire probability mass of the DTMC is initially located in the a^{th} component of the phase vector, and the entire probability mass of the CTMC is initially located in the i^{th} component of vector $\boldsymbol{\pi}$. After the initialization, the procedure SolveBoth proceeds to effectively advance in time from embedded state (i, a), moving the probability mass around the CTMC and individual DTMCs, while looking for $t \in \tilde{\mathcal{T}}$ firing opportunities. Unlike the phase vectors for $t \in \tilde{\mathcal{T}}$, which will evolve during the procedure, the phase vectors of each $t \in \mathcal{T}_D \backslash \tilde{\mathcal{T}}$ will remain the same.

Ignoring the **while** loop for the moment, lines 9 and 10 solve the CTMC at time τ, the time of the next clock increment, which can be computed at the same time using the uniformization algorithm (2.2.1) presented in Chapter 2. By using the vector $\boldsymbol{\pi}$ itself as the initial vector in the equations, the transient analysis resumes each time from the previous observation. Of course, we assume that $\mathcal{T}_D$ firing delays are nonzero so that at least one phase advance is needed, otherwise the transition(s) would be immediate and reside instead in set $\mathcal{T}_Z$. Lines 11-21 then advance the phases of each $t \in \tilde{\mathcal{T}}$ based on each CTMC state occupied at the next clock increment. The conditional next phase is stored in vector $\boldsymbol{v}$ whereas the unconditional next phase is stored in vector $\boldsymbol{w}$ once all possible CTMC states have been considered. For each occurrence of the condition $v_0 > 0$, indicating that the firing of $t \in \tilde{\mathcal{T}}$ at the current clock step is possible, the corresponding state vector is constructed and stored in vector $\boldsymbol{\nu}$. The vector $\boldsymbol{v}_0$ has v_0 as the 0^{th} element, and all other elements are zero. By accumulating such "firing" state probabilities over all firing occurrences during the entire regeneration period, we effectively set aside the processing of the firings until the end. The state probability mass at each $t \in \tilde{\mathcal{T}}$ firing opportunity is given by the Kronecker product of CTMC states and DTMC phases that make the firing possible.

Some comments about the marking-dependent $\boldsymbol{D}^t(\cdot)$ matrix are needed. At the very least, for each $k \in \mathcal{S}_i$, $\boldsymbol{D}^t(k)$ is defined as the identity matrix $\mathbf{I}$ when $t \notin \mathcal{F}(k)$. In this way, the phase given by $\boldsymbol{\phi}^t$ is unchanged when t is disabled in the marking associated with state k. In addition, $\boldsymbol{D}^t(k)$ may also depend on the state (marking) k; the matrix entries, the matrix structure, or both may be marking dependent. Of course, if we allow the matrix structure to change, we must at least maintain a consistent phase space to avoid ill-defined models. Most important to the construction of $\boldsymbol{D}^t(\cdot)$, we set the "0 phase" row elements to zero. Doing this discards the probability mass in each ϕ_0^t when line 14 is executed the next time around. Only the probability mass contained after its use in the other $\boldsymbol{D}^t(\cdot)$ states is retained when the clock is incremented to search for other firing opportunities in the future time horizon. Hence, the probability mass contained within each $\boldsymbol{D}^t(\cdot)$ diminishes over time; once the mass in all diminishes below the desired precision (d decimal places), there is no practical need to search for any further firings, and the **while** loop can exit. Until then, the procedure repeats. After exiting the **while** loop, the vector $\boldsymbol{\nu}$ will have a row sum practically equal to one, within the precision 10^{-d}. Then, $\boldsymbol{\nu}$ can be multiplied with the switching matrix $\boldsymbol{\Delta}$ to compute the embedded states reached from state (i, a). If total disablement or resetting of the transitions in $\tilde{\mathcal{T}}$ is possible during the regeneration period,

the vector $\boldsymbol{\pi}_{\{\mathcal{E}_i\}}$, consisting of (absorbing state) entries π_k if $k \in \mathcal{E}_i$ and zero otherwise, will contain the (trapped) probability of such events, and thus the probability of entering the additional set of embedded states $\mathcal{E}_i$. The phase vector $\boldsymbol{\phi}^{(0)}$ denotes the new, *resampled* phases of transitions enabled in the set of embedded states $\mathcal{E}_i$. Row (i, a) of the conversion matrix is identical to the vector $\boldsymbol{\sigma}$.

For regeneration periods where

1. $\tilde{\mathcal{T}}$ consists of transitions with *only* resampling or enabling memory policies and
2. the $\boldsymbol{D}^t$ matrix for each $t \in \tilde{\mathcal{T}}$ is *not* marking dependent,

we can be more efficient. In such situations, the DTMC is fixed and we do not need to remember the phase of age-memory transitions at the previous clock increment just in case it becomes disabled before the next. By the construction that ensures a semi-regenerative process, if some $t \in \tilde{\mathcal{T}}$ is enabled in state k at precisely time θ then it must have been enabled over the entire interval leading up to θ. This convenience allows us to advance the clock forward by more than one increment in time until a firing opportunity presents itself. Then the CTMC can be solved at this larger time interval, followed by the construction of the "firing" state probability vectors. The enhanced version of the SolveBoth procedure is presented as Procedure 4.2.5.

After setting the variable θ to zero in line 9, the **repeat** loop searches for firing opportunities while incrementing the clock one τ step at a time (lines 10 through 15) while the time is kept in θ. Again, we assume that $\mathcal{T}_D$ firing delays are nonzero so that at least one phase advance is needed. The **repeat** loop exits when the firing of at least one $t \in \tilde{\mathcal{T}}$ is possible, and consequently, θ will contain the largest forward-step in time when the embedded process can be observed without the possibility of missing a DPH transition firing. The CTMC is then advanced ahead to this time θ and the expected sojourn times in CTMC states are computed over this same interval from the last probability distribution of state occupancy (lines 16 and 17). The state probability vector $\boldsymbol{\nu}$ is constructed in lines 18-20 as before except that the phases of disabled transitions in $\tilde{\mathcal{T}}$ are set to "$\bullet$", as expected by definition.

Procedure 4.2.5 Enhanced solution procedure when both DTMC and CTMC are active and age DPH transitions are not enabled.

procedure EnhancedSolveBoth **is**

Given $(i,a) \in \mathcal{E}$, let $\mathcal{S}_i$ be the states reachable from i and construct the associated, possibly expanded, CTMC with state space $\mathcal{S}_i \cup \mathcal{E}_i$ and generator matrix $\boldsymbol{Q}_i$.

Given $\mathcal{T}_D = \{t_1, t_2, \ldots, t_{|\mathcal{T}_D|}\}$, let $\tilde{\mathcal{T}} = \mathcal{F}(i) \cap \mathcal{T}_D$ be the set of DPH transitions initially enabled at the same time in state i.

For each $t \in \tilde{\mathcal{T}}$, let $\boldsymbol{D}^t \in \mathbb{R}^{|\mathcal{D}^t| \times |\mathcal{D}^t|}$ be the firing-delay specification (a stochastic matrix) defined on the phase space $\mathcal{D}^t$, and set the 0^{th} row to zero, $\boldsymbol{D}^t_0 \leftarrow \boldsymbol{0}$.

Let $\boldsymbol{\phi}^t \in \mathbb{R}^{|\mathcal{D}^t|}$ be the individual phase vectors $\forall\, t \in \mathcal{T}_D$ and initialize each so that vector $\otimes_t \boldsymbol{\phi}^t$ (with $\mathbf{1}_\bullet$ used instead for $t \in \mathcal{T}_D \backslash \tilde{\mathcal{T}}$) and phase "$a$" are one in the same.

Let $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{S}_i|}$ be the CTMC probability vector, and initialize, $\boldsymbol{\pi} \leftarrow \mathbf{1}_i$.

Let $\boldsymbol{\sigma} \in \mathbb{R}^{|\mathcal{S}_i|}$ be the CTMC cumulative probability vector, and initialize, $\boldsymbol{\sigma} \leftarrow \boldsymbol{0}$.

Let $\boldsymbol{\nu} \in \mathbb{R}^{|\mathcal{S}_i \times \mathcal{D}_a|}$ be the complete state probability vector, which accumulates state occupancy probabilities when some $t \in \tilde{\mathcal{T}}$ can fire, set initially to zero, $\boldsymbol{\nu} \leftarrow \boldsymbol{0}$.

while $\exists\, t \in \tilde{\mathcal{T}} : \left\| \boldsymbol{\phi}^t \right\|_1 > 10^{-d}$ **do** ... for as long as the possibility for $\tilde{\mathcal{T}}$ firings exist

  $\theta \leftarrow 0$

  repeat advance the clock to the next $\tilde{\mathcal{T}}$ firing opportunity

    for each $t \in \tilde{\mathcal{T}} : \|\boldsymbol{\phi}^t\|_1 > 0$ **do**

      $\boldsymbol{\phi}^t \leftarrow \boldsymbol{\phi}^t \, \boldsymbol{D}^t$

    end for

    $\theta \leftarrow \theta + \tau$

  until $\exists\, t \in \tilde{\mathcal{T}} : \phi^t_0 > 0$ phase = 0 indicates that transition t can fire

  $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + \boldsymbol{\pi} \int_0^\theta e^{\boldsymbol{Q}_i u} \, du$ accumulate time spent in each CTMC state

  $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} e^{\boldsymbol{Q}_i \theta}$ CTMC state at *this* clock step

  for each $k \in \mathcal{S}_i$ **do**

    $\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + \boldsymbol{\pi}_k \otimes \boldsymbol{\psi}_k^{t_1} \otimes \boldsymbol{\psi}_k^{t_2} \otimes \cdots \otimes \boldsymbol{\psi}_k^{t_{|\mathcal{T}_D|}}$, state vector associated with firing

    where $\boldsymbol{\psi}^t_k = \begin{cases} \boldsymbol{\phi}^t & \text{if } t \notin \tilde{\mathcal{T}} \\ \boldsymbol{\phi}^t_0 & \text{if } t \in \mathcal{F}(k) \cap \tilde{\mathcal{T}} \wedge \phi^t_0 > 0 \\ \mathbf{1}_\bullet & \text{otherwise} \end{cases}$

  end for

end while

$\boldsymbol{\Pi}_{(i,a)} \leftarrow \boldsymbol{\nu} \boldsymbol{\Delta} + \boldsymbol{\pi}_{\{\mathcal{E}_i\}} \otimes \boldsymbol{\phi}^{(0)}$ construct EMC row (i,a) by applying switching matrix

$\boldsymbol{h}_{(i,a)} \leftarrow \boldsymbol{\sigma}$ the (i,a) row of the conversion matrix is identical to $\boldsymbol{\sigma}$

4.3 Complexity Analysis

The previous classification of the PDPNs with certain simplifying assumptions and the formulation of time-dependent state equations allows us to assess the power-complexity trade-off between these PDPN classes and how they compare with other SPN extensions. Figure 4.14 shows a plot that roughly estimates the modeling power versus solution complexity for each of the SPN classes. On the two extremes, we have the untimed PN, which is not accompanied by a stochastic process and therefore has the least power but is easiest to solve (in terms of qualitative analysis), and the general SPN, which has the most power but is the hardest to solve (in terms of quantitative analysis). Although the DTMC and CTMC based SPNs have the same modeling power, the DTMC-SPN is considered a little more difficult to solve due to the contemporary transition firings; hence, it is sitting above the CTMC-SPN. The same reasoning was applied to the DDSPN and PH-SPN.

From the plot, we can see that progress is being made towards achieving more modeling power but at the cost of additional solution complexity. It is hoped that the approximate solutions discovered by this research effort will allow the general (asynchronous) class of PDPN to be reasonably powerful and accurate with much less effort than the most general SPN.

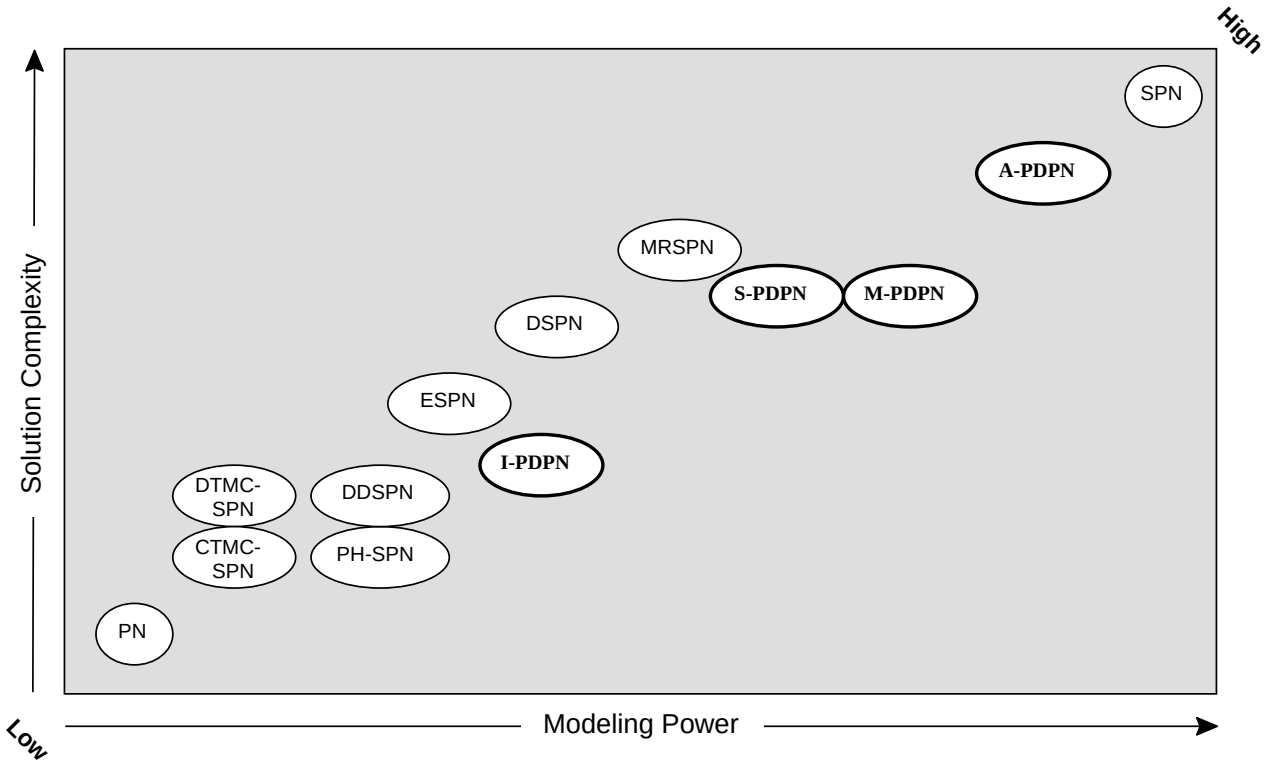


Figure 4.14: Performance comparison.

Returning to our stationary solution algorithm, we should be curious about its computation time, in particular, the convergence of line 15 of Algorithm 4.2.1 and line 2 of procedures SolveDTMC and SolveCTMC using iterative methods. Although the use of iterative methods like Gauss-Seidel and SOR to compute line 15 of Algorithm 4.2.1 does not guarantee convergence for all initial guesses, convergence can be guaranteed for line 2 of both SolveDTMC and SolveCTMC, since both $-\tilde{\boldsymbol{Q}}$ and $(\mathbf{I}-\tilde{\boldsymbol{\Phi}})$ are M-matrices. Because the SMC solutions, at least, enjoy bounded computation, this new stationary PDPN algorithm may be no worse, relatively speaking, than the algorithms proposed in the literature, e.g., [11, 46], for DSPNs and MRSPNs. However, just how much computation is really involved is another matter since it depends on the number of iterations needed to achieve the desired accuracy. We do have the advantage that we expect the size of $(\mathbf{I}-\tilde{\boldsymbol{\Phi}})$ and $\tilde{\boldsymbol{Q}}$ to be small since they are constructed from the combination of absorbing DTMCs and CTMCs, respectively, which themselves will usually be small. If the matrices are small enough, we could even employ direct methods such as Gaussian elimination or LU decomposition to solve for $\tilde{\boldsymbol{\sigma}}$.

Similarly for the procedures SolveBoth and EnhancedSolveBoth, the number of **while** loop iterations required to achieve the desired accuracy depends on the PDPN model, the type of DPH transitions, and each particular regeneration period. Many iterations may be necessary if the DPH specifications are such that the probability mass tends to occupy selfloops and circuits within the DTMC. Nevertheless, the **while** loop will exit since the probability mass will ultimately drain from each $\boldsymbol{D}^t$ matrix, $t \in \tilde{\mathcal{T}}$. Actually, the residual probability mass goes to zero like ω^k as the number of loop iterations, k, increases, where $\omega = \max_{t\in\tilde{\mathcal{T}}}(\omega_1^t) < 1$ and ω_1^t is the largest eigenvalue, or spectral radius, of matrix $\boldsymbol{D}^t$. The reason is two-fold. One, making the "0 phase" row entries of each $\boldsymbol{D}^t$ matrix identically zero ensures that the spectral radius of each $\boldsymbol{D}^t$ matrix is less than one. Without the zero row, each $\boldsymbol{D}^t$ matrix would be stochastic and therefore have a spectral radius of one. Making one of the rows identically zero reduces the largest eigenvalue to zero, which would otherwise equal one, thereby making the second-largest eigenvalue, which *is* less than one, the largest. Two, for iterative methods such as this (essentially the power method), the convergence rate is proportional to the largest eigenvalue of the iteration matrix. More will be said about this momentarily.

The number of iterations can always be shortened by exiting the **while** loop early, however, doing so will increase the number of states and transitions within the EMC that only represent "delay" information. The additional EMC states may also worsen the convergence rate of the iterative method used to compute its stationary solution. Clearly, a heuristic is needed to find an acceptable trade-off between computation time and space.

We now conclude this section (and chapter) with a rough analysis that considers the trade-off of complexity between the EMC and SMC solutions. For the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in general, or $(\mathbf{I}-\boldsymbol{\Pi})\,\boldsymbol{x} = \mathbf{0}$ in particular, an iterative method like Gauss-Seidel or SOR takes an initial guess, $\boldsymbol{x}^{(0)}$, for the exact solution, $\boldsymbol{x}$, and generates a sequence of vectors $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ that, hopefully, converges to $\boldsymbol{x}$ as $k \to \infty$. The intent of an iterative method is to transform the original system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to an equivalent system $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}$ that not only produces a sequence $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ that converges to the exact solution, $\boldsymbol{x}$, but also makes the

spectral radius, ρ, of $\boldsymbol{B}$ and the spectral radius, ρ', of $\boldsymbol{A}$ satisfy

$$\rho < \rho' \leq 1.$$

The technique is guaranteed to converge for any initial guess if $\rho < 1$, however convergence is not guaranteed for all guesses when $\rho = 1$ [16]. The error between $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}$ goes to zero like ρ^k as k increases. For a desired precision of 10^{-d}, the rate of convergence is then defined as $-d/\log\rho$. Any reduction of the spectral radius, ρ, by a fraction, x, will improve the rate of convergence by an additive amount $\log(1-x)$ in the denominator.

With this knowledge, we approach the analysis with the following reasoning. Let us suppose that if the embedded process was observed at the most frequent opportunities, i.e., the "single-step" embedding method is used as discussed in Section 4.2.1, then the EMC matrix $\boldsymbol{\Pi}$ would contain η entries. For regeneration periods where procedures SolveDTMC or SolveCTMC are used, the SMC is identical to the EMC, each containing a fraction, y, of the total $\boldsymbol{\Pi}$ matrix entries, and a much simpler SMC analysis is traded for an anticipated, simpler EMC analysis. That is, if the "single-step" method was used instead, the (single state) SMC solution for each entry in $\boldsymbol{\Pi}$ would require $O(1)$ time, due to the closed-form expressions, but $y\eta$ such solutions would be required. With SolveDTMC or SolveCTMC, we study the SMC as a TTA problem, one time, and the time complexity is $O(y\eta K)$ if it requires K iterations. The reduction of $(1-y)\eta$ entries in the matrix $\boldsymbol{\Pi}$ not only reduces the per iteration cost by the same amount but should also improve the rate of convergence by reducing the spectral radius of the iteration matrix derived from $\boldsymbol{\Pi}$, as observed in [43]. We have a similar trade-off for regeneration periods when procedure SolveBoth, or EnhancedSolveBoth, is used, except that the trade is between having $(1-y)\eta$ less entries in $\boldsymbol{\Pi}$ and having to perform the power method on essentially a DTMC matrix $\otimes_{t\in\tilde{\mathcal{T}}}\boldsymbol{D}^t$ that has, perhaps, $O(y\eta)$ entries. The SMC and generator matrix is the same whether employing the "multi-step" method of SolveBoth or, instead, the "single-step" method.

Assume that a simple SMC solution results in the worse-case EMC complexity of η matrix entries and requires

$$K_1 = \frac{-d}{\log\rho_1}$$

iterations where ρ_1 is the spectral radius of the iteration matrix. The time complexity of computing the EMC stationary solution is therefore $O(\eta K_1)$. With our approach, a more complex SMC solution that reduces ρ_1 by a fraction x and reduces η by a fraction y would yield a *reduced* time complexity of $O((1-y)\eta K_2)$ for the EMC solution where

$$K_2 = \frac{-d}{\log\left((1-x)\rho_1\right)}$$

but should typically require an *additional* effort of $O(y\eta K_3)$ in solving the SMC due to the K_3 iterations required of the more complex SMC solution with $y\eta$ matrix entries. Hence, the net reduced effort is given by

$$1 - \left(\frac{(1-y)\eta K_2 + y\eta K_3}{\eta K_1}\right).$$

Substituting

$$\frac{K_2}{K_1} = \frac{-d/\log((1-x)\rho_1)}{-d/\log\rho_1} = \frac{\log\rho_1}{\log(1-x)+\log\rho_1}$$

and, similarly

$$\frac{K_3}{K_1} = \frac{\log\rho_1}{\log\rho_2}$$

where ρ_2 is the spectral radius of the SMC matrix, we get a reduced effort of

$$r(x, y, \rho_1, \rho_2) = 1 - \log\rho_1 \left(\frac{1-y}{\log(1-x)+\log\rho_1} + \frac{y}{\log\rho_2} \right).$$

The net reduced effort, r, is plotted in Figure 4.15 for $\rho_1 = \rho_2 = 0.9$. From the plot, we can see that the overall complexity is reduced for nonzero reductions in ρ_1 and for values of y less than unity. The plot shows that the solution algorithm may enjoy significant cost savings for even the smallest x reduction in ρ_1 as long as y is reasonably small too. This is also true for other, equal values of ρ_1 and ρ_2 that are large, which is typically the case. As shown in Figure 4.16 for $\rho_1 = \rho_2 = 0.2$, the cost savings is less dramatic as the spectral radii become small. This is expected since the iterative methods would converge quickly for small spectral radii; therefore, moving the effort from the EMC to the SMC, via the elimination of embedded states, would have less impact on performance.

Our predictions indicate that a non-negative cost savings can be realized for all values of x, y, ρ_1, and ρ_2 when $\rho_1 \geq \rho_2$. As ρ_1 becomes larger than ρ_2, the potential cost savings becomes even more dramatic. For example, Figure 4.17 shows the savings when $\rho_1 = 0.9$ and $\rho_2 = 0.8$, which is similar to Figure 4.15 except the savings levels off at 50 percent as y approaches unity, as opposed to 0 percent.

It seems that "embedding with elimination" will *always* improve performance. Unfortunately, this is not the case. At the risk of ending on a bad note, let us consider when $\rho_1 < \rho_2$. This would mean that the EMC solution would most likely enjoy better performance than the SMC solutions when left alone. Moving some of the complexity from the EMC to the SMC has the potential of worsening the overall performance if the degradation in the SMC solution performance is not offset by even better EMC solution performance. As an example, Figure 4.18 shows the cost savings when $\rho_1 = 0.8$ and $\rho_2 = 0.9$. Although cost savings can still be realized, there is now the possibility of making matters worse, perhaps even 100 percent worse if too much complexity (the y fraction) is moved to the SMC. Clearly, a heuristic is needed to work with the "embedding with elimination" algorithm to ensure that an acceptable trade-off between the EMC solution and the many SMC solutions can be realized in most problem cases.

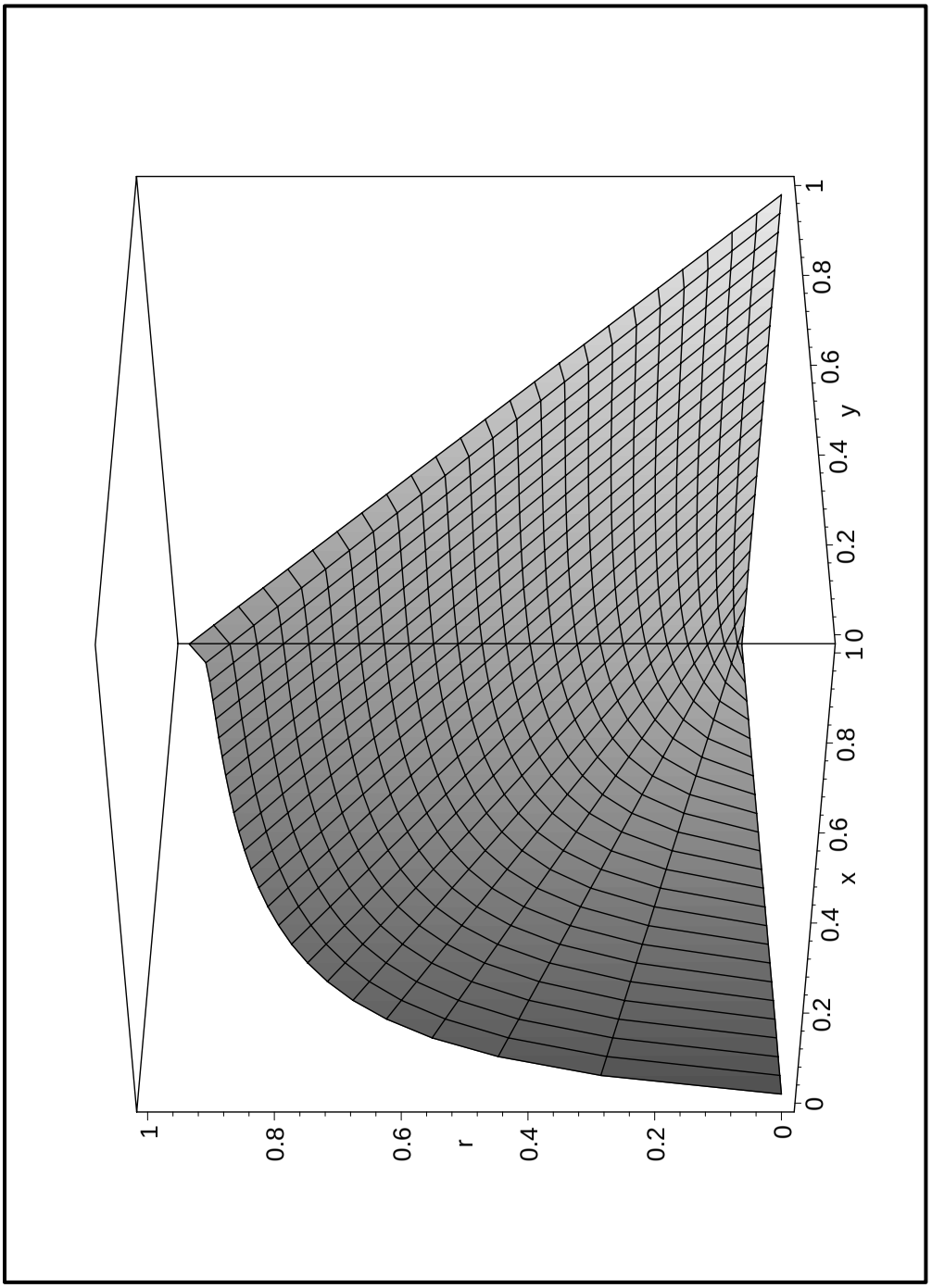


Figure 4.15: Reduced effort for $\rho_1 = \rho_2 = 0.9$.

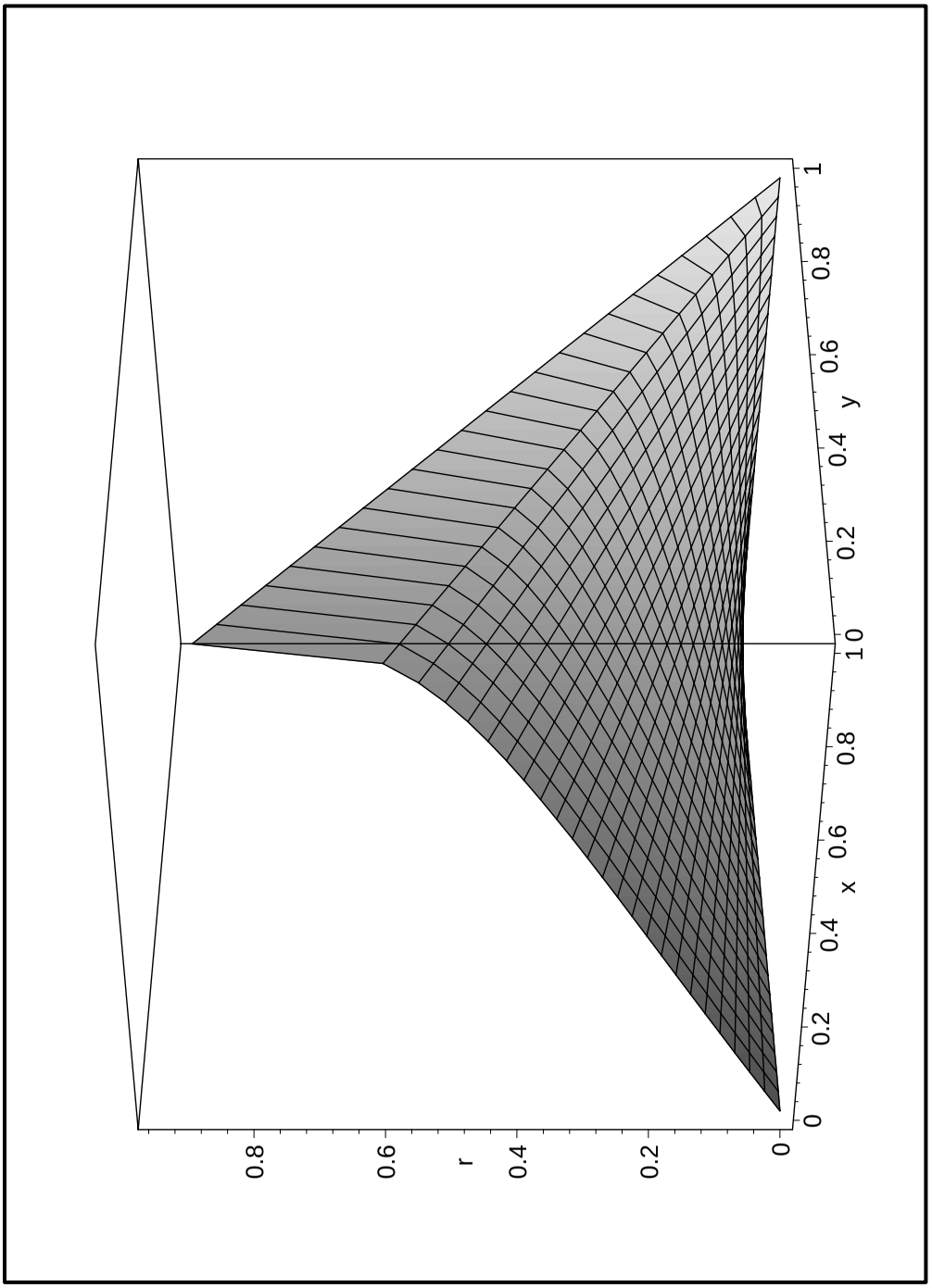


Figure 4.16: Reduced effort for $\rho_1 = \rho_2 = 0.2$.

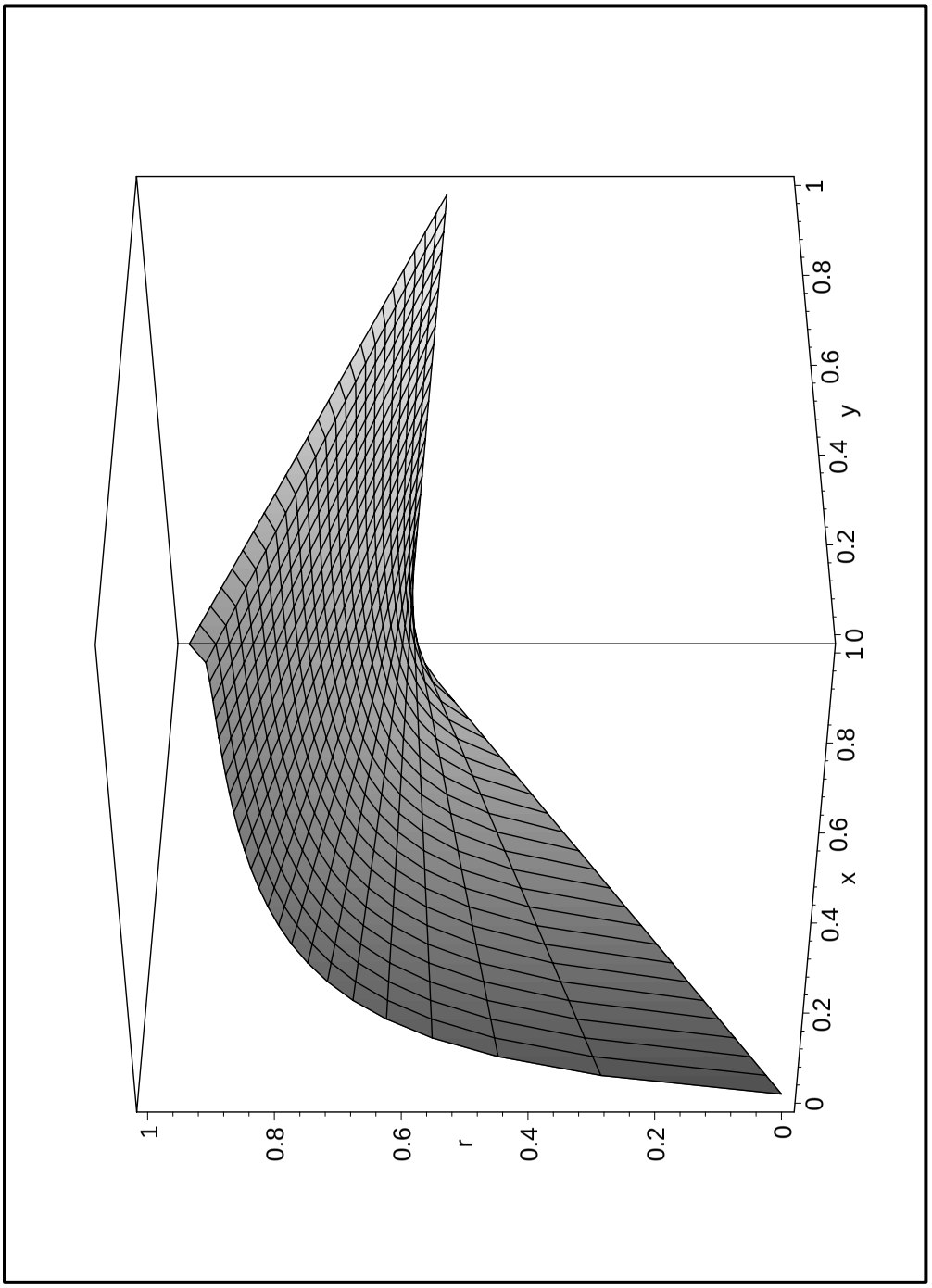


Figure 4.17: Reduced effort for $\rho_1 = 0.9, \rho_2 = 0.8$.

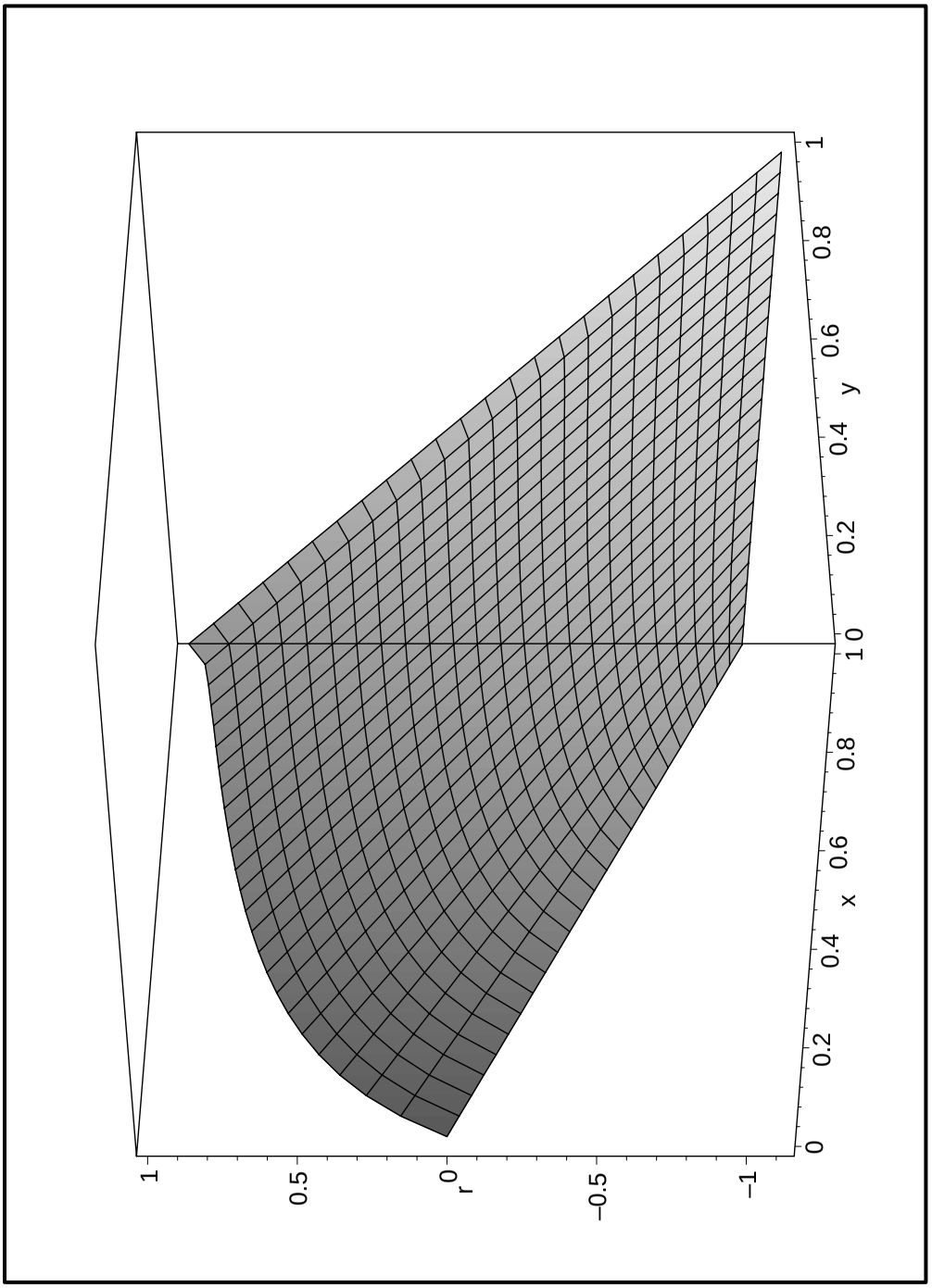


Figure 4.18: Reduced effort for $\rho_1 = 0.8, \rho_2 = 0.9$.

4.4 Summary of Current Accomplishments

We have shown that SPNs are easily solved when firing delays of transitions are either exponentially or geometrically distributed. These models have an underlying CTMC or DTMC, respectively, for which efficient solution techniques are available. However, such modeling assumptions by themselves may be unrealistic for many systems, leading to inaccurate results when adopted. In their full generality, SPNs specify generalized semi-Markov processes whose numerical analysis is impractical. The goal of our research is to improve SPN modeling fidelity in a way that preserves, as much as possible, the solution efficiencies enjoyed by CTMC and DTMC models.

Our approach has led to the formal development of a new class of non-Markovian SPN employing phase-type firing delays in both discrete and continuous time. We refer to this new SPN class as a phased delay Petri net or PDPN, and to our knowledge, this is the first time discrete and continuous phase-type firing delays have been combined simultaneously in the same model. With phase-type firing delays, the notion of state is composed of the net marking and the discretized, firing-delay *phases* of enabled transitions.

Our investigations into the characteristics and analysis of the PDPN has led to the identification of PDPN subclasses that afford different combinations of modeling power and solution complexity. In decreasing generality, these subclasses are identified as *asynchronous*, *mixed*, *synchronous*, and *isochronous*. The essential difference between the unrestricted asynchronous PDNP and the latter three subclasses is that the phase advancements of discrete phase transitions maintain synchronization in the latter subclasses. Maintaining synchronization among discrete phase transitions results in an underlying semi-regenerative process and is key to the existence and discovery of efficient solution algorithms. With the aid of Markov renewal theory, we can decompose a semi-regenerative process into interacting CTMCs and DTMCs, each of which can be solved efficiently, and the intermediate results can be combined to obtain the solution to the original, more complex problem. Full understanding of these subclasses should help us find efficient algorithms for their exact or approximate solutions, as appropriate, culminating into a set of solution methods. Given a particular modeling application, the solution engine may then classify the type of PDPN and employ a capable-enough solution algorithm having the least amount of complexity.

To this end, our preliminary investigation has produced efficient algorithms for the exact stationary solution of all PDPN subclasses except for the asynchronous PDPN, and the exact transient solution of the isochronous PDPN. Compared to similar stationary solution algorithms for DSPNs and MRSPNs, our stationary solution algorithm is new in that it proposes an "embedding with elimination" approach, which attempts to reduce the size of the embedded state space and the number of state transitions that update the phase information without actually firing a transition.

A complete "embedding with elimination" algorithm and theoretical complexity analysis was presented. Our stationary algorithm shows much promise when there exists opportunities to reduce the embedded state space. Our theoretical investigation has also provided evidence that approximate solution algorithms are more needed for transient analysis than for stationary analysis and may be the only practical means, other than simulation, towards solving asynchronous PDPNs. The investigation and development of such solution algorithms is the focus of our future research.

Chapter 5

Future Work

Through our preliminary research, we have begun to understand the PDPN well enough to formalize its definition, semantics, and underlying stochastic process. Towards this understanding, we have determined that the general class of problem, which we call an *asynchronous* PDPN, also has three useful subclasses denoted by the adjectives *isochronous*, *synchronous*, and *mixed* (having both synchronous and asynchronous characteristics). Restricting PDPN models to one of these three subclasses, ensures that the underlying process is semi-regenerative. As such, we can employ Markov renewal theory in an attempt to find efficient solutions, either stationary or time dependent.

Our preliminary research has allowed us to acquire an understanding of Markov renewal theory and to apply it to PDPN models. Consequently, we now have an initial stationary solution algorithm that shows promise in terms of efficiency and fidelity.

Markov renewal theory has also shown us how difficult it is to obtain time-dependent solutions. So other than the exact solution algorithm given here for the *isochronous* PDPN, we will seek approximate solution algorithms that offer higher efficiency and reasonable accuracy.

The plan towards completing our proposed research and development is as follows:

- Improve our proposed stationary solution algorithm, to investigate heuristics that select the best embedding strategy based on each particular PDPN model. Further extensions to the uniformization algorithm may also be needed.

- Investigate and develop approximate solution algorithms for time-dependent solutions that are both efficient and reasonably accurate, preferably with upper and lower bounds. Again, heuristics will be sought to ensure good performance for most PDPN models that one would typically encounter.

- We also plan to investigate the fidelity of the developed PDPN modeling approach to other contemporary approaches including simulation while applying our results to relevant applications. Such comparisons with other approaches will also include what impact the choice of PH or DPH, continuous or discrete time, has on solution accuracy with respect to solution complexity.

- Finally, all PDPN analysis algorithms and data structures will be implemented into the software tool SMART (Simulation and Markovian Analyzer for Reliability and Timing) under development at the College of William & Mary.

Bibliography

[1] M. K. Molloy, "Discrete time stochastic Petri nets," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 417–423, April 1985.

[2] A. Bobbio and A. Cumani, "Discrete state stochastic systems with phase type distributed transition times," in *Proc. ASME Int. Conf. Modelling and Simulation*, (Athens, Greece), pp. 173–192, 1984.

[3] M. A. Marsan and G. Chiola, *On Petri nets with deterministic and exponentially distributed firing time*, vol. 266, pp. 132–145. Springer-Verlag, 1987.

[4] M. A. 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*, vol. 15, pp. 832–845, 1989.

[5] G. Ciardo, "Discrete-time Markovian stochastic Petri nets," in *Computations with Markov Chains* (W. J. Stewart, ed.), pp. 339–358, Boston, MA: Kluwer, 1995.

[6] M. Hack, "Decidability questions for Petri nets," Technical Report 161, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, June 1976.

[7] T. Agerwala, "Putting Petri nets to work," *IEEE Comp.*, pp. 85–94, Dec. 1979.

[8] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, Apr. 1989.

[9] M. A. Marsan, G. Balbo, and G. Conte, "A Class of Generalized Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Transactions on Computer Systems*, vol. 2, pp. 93–122, May 1989.

[10] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended stochastic Petri nets: Applications and analysis," in *Proceedings of Performance '84*, (Paris, France), pp. 507–519, December 1984.

[11] R. German, *Analysis of Stochastic Petri Nets with Nonexponentially Distributed Firing Times*. PhD thesis, Technical Univ. of Berlin, Berlin, Germany, Jan. 1994.

[12] H. Choi, V. G. Kulkarni, and K. Trivedi, "Markov regenerative stochastic petri nets," *Performance Evaluation*, vol. 20, pp. 337–357, 1994.

[13] A. Bobbio and M. Telek, "Markov regenerative SPN with non-overlapping activity cycles," in *Proceedings of the International Computer Performance and Dependability Symposium*, (Erlangen, Germany), pp. 124–133, April 1995.

[14] G. Ciardo, *Analysis of Large Stochastic Petri Net Models*. PhD thesis, Duke University, Durham, NC, Apr. 1989.

[15] G. Ciardo and C. Lindemann, "Analysis of deterministic and stochastic Petri nets," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models (PNPM'93)*, (Toulouse, France), pp. 160–169, IEEE Computer Society Press, October 1993.

[16] D. Young, *Iterative Solution of Large Linear Systems*. New York: Academic Press, 1971.

[17] W. J. Stewart and A. Goyal, "Matrix methods in large dependability models," Tech. Rep. RC-11485, IBM T.J. Watson Res. Center, Yorktown Heights, NY, Nov. 1985.

[18] S. M. Ross, *Stochastic Processes*. John Wiley & Sons, 1983.

[19] B. L. Fox and P. W. Glynn, "Computing Poisson probabilities," *Communications of the ACM*, vol. 31, pp. 440–445, April 1988.

[20] A. Reibmann and K. S. Trivedi, "Transient analysis of cumulative measures of markov model behaviour," *Comm. Statist.-Stochastic Models*, vol. 5, no. 4, pp. 683–710, 1989.

[21] G. W. Stewart, "Stochastic automata, tensors operation, and matrix formulas," Tech. Rep. TR-3598, Department of Computer Science, Univ. of Maryland, College Park, MD, January 1996.

[22] W. H. Sanders, *Construction and solution of performability models based on Stochastic Activity Networks*. PhD thesis, Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI, 1988.

[23] G. Ciardo and R. Zijal, "Well-defined stochastic Petri nets," in *Proc. 4th Int'l Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, (San Jose, CA), pp. 278–284, IEEE Computer Society Press, Februrary 1996.

[24] R. Zijal, G. Ciardo, and G. Hommel, "Discrete deterministic and stochastic Petri nets," in *9. ITG/GI-Fachtagung: Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen (MMB'97), (Measurement, Modeling, and Valuation of Computer- and Communication-Systems)*, (Freiberg, Germany), pp. 103–117, VDE-Verlag, Sept. 1997.

[25] E. Çinlar, *Introduction to Stochastic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[26] H. Choi, V. G. Kulkarni, and K. S. Trivedi, "Markov regenerative stochastic Petri nets," in *Performance '93* (G. Iazeolla and S. S. Lavenberg, eds.), North-Holland, Sept. 1993.

[27] G. Ciardo and G. Li, "Approximate transient analysis for subclasses of deterministic and stochastic Petri nets," *Performance Evaluation*, vol. 35, pp. 109+, May 1999.

[28] C. Lindemann and R. German, "Modeling discrete-event systems with state-dependent deterministic service times," *Discrete event dynamic systems: theory and applications*, vol. 3, pp. 249–270, July 1993.

[29] M. Telek and A. Bobbio, "Markov regenerative stochastic Petri nets with age type general transitions," in *Proceeding of the 16th International Conference on Application and Theory of Petri Nets*, (Turin), pp. 471–489, June 1995.

[30] M. Telek, A. Bobbio, and A. Puliafito, "Steady state solution of MRSPN with mixed preemption policies," in *Proc. IEEE International Computer Performance and Dependability Symposium (IPDS'96)*, (Urbana-Champaign, IL, USA), pp. 106–115, IEEE Comp. Soc. Press, Sept. 1996.

[31] D. R. Cox, "The Analysis of Non-Markov Stochastic Processes by the Inclusion of Supplementary Variables," *Proceedings of the Cambridge Philosophical Society (Math./ and Phys./ Sciences)*, vol. 51, pp. 433–441, 1955.

[32] R. German and C. Lindemann, "Analysis of stochastic petri net by the method of supplementary variables," *Performance Evaluation*, vol. 20, pp. 317–335, 1994.

[33] A. Heindl and R. German, "A Fourth-Order Algorithm with Automatic Stepsize Control for the Transient Analysis of DSPNs," in *Petri Nets and Performance Models*, (Saint Malo, France), pp. 60–69, June 3-6 1997.

[34] C. Lindemann and G. S. Shedler, "Numerical analysis of deterministic and stochastic Petri nets with concurrent deterministic transitions," *Performance Evaluation*, vol. 27&28, pp. 565–582, 1996.

[35] P. J. Haas and G. S. Shedler, "Regenerative stochastic Petri nets," *Performance Evaluation*, vol. 6, pp. 189–204, 1986.

[36] P. J. Haas, "Estimation Methods for Stochastic Petri Nets Based on Standardized Time Series," in *Petri Nets and Performance Models*, (Saint Malo, France), pp. 194–204, June 3-6 1997.

[37] G. Ciardo, R. German, and C. Lindemann, "A characterization of the stochastic process underlying a stochastic Petri net," *IEEE Transactions on Software Engineering*, vol. 20, pp. 506–515, July 1994.

[38] G. Ciardo and A. S. Miner, "Storage alternatives for large structured state spaces," in *Proc. 9th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation* (R. Marie, B. Plateau, M. Calzarossa, and G. Rubino, eds.), LNCS 1245, (St. Malo, France), pp. 44–57, Springer-Verlag, June 1997.

[39] A. S. Miner and G. Ciardo, "Efficient reachability set generation and storage using decision diagrams," in *Application and Theory of Petri Nets 1999, Lecture Notes in Computer Science 1639, Proc. 20th Int. Conf. on Applications and Theory of Petri Nets, Williamsburg, VA, USA* (H. Kleijn and S. Donatelli, eds.), pp. 6–25, Springer-Verlag, June 1999.

[40] P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper, "Complexity of Kronecker Operaions on Sparse Matrices with Application to the Solution of Markov Models," Tech. Rep. 97–66, Institute for Computer Applications in Science and Engineering, Hampton, VA, Dec. 1997.

[41] M. Scarpa and A. Bobbio, "Kronecker Representation of Stochastic Petri Nets with Discrete PH Distributions," in *Proceedings of the International Computer Performance and Dependability Symposium*, (Durham, NC), pp. 52–61, 1998.

[42] G. Ciardo and A. S. Miner, "A data structure for the efficient Kronecker solution of GSPNs," in *Proc. 8th Int. Workshop on Petri Nets and Performance Models (PNPM'99)* (P. Buchholz, ed.), (Zaragoza, Spain), IEEE Comp. Soc. Press, Sept. 1999. To appear.

[43] G. Ciardo, A. Blakemore, P. F. J. Chimento, J. K. Muppala, and K. S. Trivedi, "Automated generation and analysis of markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models* (C. Meyer and R. J. Plemmons, eds.), vol. 48 of *IMA Volumes in Mathematics and its Applications*, pp. 145–191, Springer-Verlag, 1993.

[44] G. Ciardo and C. Lindemann, "Analysis of deterministic and stochastic Petri nets," in *Proc. 5th Int. Workshop on Petri Nets and Performance Models (PNPM'93)*, (Toulouse, France), pp. 160–169, IEEE Comp. Soc. Press, Oct. 1993.

[45] L. M. Malhis and W. H. Sanders, "An efficient two-stage iterative method for the steady-state analysis of Markov regenerative stochastic Petri net models," *Performance Evaluation*, vol. 27&28, pp. 583–601, 1996.

[46] C. Lindemann, "An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic petri net models," *Performance Evaluation*, vol. 18, pp. 79–95, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 2000	Contractor Report

4. TITLE AND SUBTITLE
Analysis of Phase–Type Stochastic Petri Nets With Discrete and Continuous Timing

5. FUNDING NUMBERS
NAS1–99124
519–51–41–01

6. AUTHOR(S)
Robert L. Jones

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ASR Aerospace Corporation
6301 Ivy Lane, Suite 300
Greenbelt, MD 20770

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-2199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR-2000-210296

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Plesent W. Goode

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 64 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
The Petri net formalism is useful in studying many discrete–state, discrete–event systems exhibiting concurrency, synchronization, and other complex behavior. As a bipartite graph, the net can conveniently capture salient aspects of the system. As a mathematical tool, the net can specify an analyzable state space. Indeed, one can reason about certain qualitative properties (from state occupancies) and how they arise (the sequence of events leading there). By introducing deterministic or random delays, the model is forced to sojourn in states some amount of time, giving rise to an underlying stochastic process, one that can be specified in a compact way and capable of providing quantitative, probabilistic measures. We formalize a new non–Markovian extension to the Petri net that captures both discrete and continuous timing in the same model. The approach affords efficient, stationary analysis in most cases and efficient transient analysis under certain restrictions. Moreover, this new formalism has the added benefit in modeling fidelity stemming from the simultaneous capture of discrete– and continuous–time events (as opposed to capturing only one and approximating the other). We show how the underlying stochastic process, which is non–Markovian, can be resolved into simpler Markovian problems that enjoy efficient solutions. Solution algorithms are provided that can be easily programmed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Computer modeling; Stochastic Petri nets; Stochastic processes; Non–Markovian processes; Semi–regenerative processes; Markov renewal theory	91
	16. PRICE CODE: A05

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL