MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Parameter Synthesis in Continuous-Time Stochastic Systems

PH.D. THESIS

Ľuboš Korenčiak

Brno, December 2017

Declaration

Hereby I declare, that this thesis is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

In Brno, December 18, 2017
Ľuboš Korenčiak

Supervisor: prof. RNDr. Antonín Kučera, Ph.D.

Acknowledgement

I would like to express my deepest gratitude to my advisors Vojtěch Řehák and Tomáš Brázdil and supervisor prof. Antonín Kučera for their helpful comments, suggestions and guidance throughout the work on this thesis. Especially I would like to thank to Vojtěch Řehák who helped and influenced me the most.

I thank my parents, siblings, and friends for their support during my studies. Moreover, I thank my girlfriend Kristína Stromková for her love and good cooking. I am grateful to Jan Krčál and Petr Novotný for their patience and leadership during the times when I needed it the most. I am thankful to my co-authors Christel Baier, Martin Bezděka, Clemens Dubslaff, and Adrian Farmadin for the excellent and enjoyable collaboration. Finally, I would like to thank my colleagues and good friends Fanda Blahoudek, Martin Chmelík, Ondřej Bouda, Jakub Gajarský, Martin Jonáš, Jan Křetínský, Matúš Madzin, Jan Obdržálek, and Jan Strejček for their enthusiastic attitude that brought a lot of laugh to my life.

Abstract

Continuous-time Markov chains (CTMCs) is an attractive formalism for specification of stochastic systems where discrete events occur in continuous time. The class of CTMCs is suitable for performance evaluation thanks to the Markov property, which simplifies the analysis but limits the modeling power as it requires exponential distribution for all events. Continuous-time Markov chains with alarms (ACTMCs) additionally allow for alarm events that have generally distributed delays, given that at most one alarm is active at any time. We consider *parametric* ACTMCs where the (continuous) parameters of alarm-delay distributions are not given explicitly and can be subjects of parameter synthesis. This is very useful in practice, e.g., to automatically synthesize timeouts in network protocols.

We present two sets of algorithms solving the ε-optimal parameter synthesis problem for parametric ACTMCs with total accumulated reward or long-run average optimization objectives. All algorithms are based on reduction to the problem of finding optimal policies in partially observable Markov decision processes (POMDPs) and sufficient discretization of parameter space (that corresponds to the action space of the POMDP). The algorithms of the first set explicitly construct the action space and call the state-of-the-art POMDP solution algorithms. Since the set of actions in the discretized POMDP can be very large, the explicit algorithms have high time and space requirements to solve even simple instances of the problem. The second set of presented algorithms uses an enhanced policy iteration on a symbolic representation of the action space. The soundness of the symbolic algorithms is established for parametric ACTMCs with alarm-delay distributions satisfying mild assumptions, that are shown to hold for uniform, Dirac, exponential, and Weibull distributions in particular, but are satisfied for many other distributions as well. An experimental implementation shows that the symbolic approach substantially improves the efficiency of the explicit algorithms and allows to solve instances of a large size.

Keywords

parameter synthesis, continuous-time Markov chain with alarms, generalized semi-Markov process, stochastic Petri net, partially observable Markov decision process, mean payoff, total reward, policy iteration

Contents

Chapter 1

Introduction

Nowadays, during the development of computer systems testing is much more used than formal analysis. However, there are areas where the formal methods are crucial. For example, in the aerospace industry a specification for a new functionality usually contains performance requirements, e.g., that the probability of failure should be lower than a given threshold. The formal analysis needed to compute performance properties in order to provide the evidence that the system satisfies the requirements. One of the ways to compute the properties is to model the system in a suitable formalism, express the property using some specification language, and use an adequately accurate method which supports computation of the property (in a reasonable time).

Some systems can be properly modeled by discrete-time and discrete-space formalisms. However, systems such as temperature or real-time control are naturally continuous and modeling them with discrete time and space can be inappropriate. We are interested in formalisms with discrete-state space where the state-changing events occur in continuous time. Concretely, the change of a state is instantaneous, whereas the time between the changes can be randomly distributed. These formalisms are usually referred to as *discrete-event systems* [43]. One well-known example of such a formalism is a *continuous-time Markov chain* (CTMC).

A CTMC is simply a finite set of states where the change of a state is driven by a finite set of transitions. Each transition in a CTMC has specified initial and terminal state and a parameter called *rate*. When a state of the CTMC is entered, each outgoing transition is assigned a waiting time according to the exponential distribution with the corresponding rate. Each transition waits until its time runs up. Then the transition with the shortest waiting time is executed and the state is changed to its terminal state. The CTMC formalism is analytically tractable because it possesses the Markov property, i.e., at any point of time, the future behavior of the

CTMC depends only on the current state. The exponential distribution is the only continuous-time distribution that can be used in a CTMC in order to keep the Markov property. However, in practice, there are many systems that contain also non-exponential distributions. To deal with such distributions, phase-type approximation technique [124] is usually applied, which results in a CTMC model with an increased state space. Unfortunately, as already noted in [124], some distributions cannot be efficiently fit with the phase-type approximation. In particular, Dirac-distributed events, i.e., the events that occur after a fixed amount of time with probability 1, form a distinguished example of this phenomenon where either a state space explosion or a large error is caused by the approximation [98]. However, as non-exponential distributions play a crucial role in many systems, for example in communication protocols [126] or time-driven real-time scheduling [132], they should be handled correctly in modeling and analysis.

Continuous-time Markov Chains with Alarms. We study *continuous-time Markov chains with alarms (ACTMCs)*, which are an extension of the continuous-time Markov chains where besides the standard exponentially-distributed events we allow for non-exponentially-distributed *alarm events*. Each alarm is equipped with a set of active states, a cumulative distribution function, and a stochastic transition kernel that specifies the discrete probability of reaching some state from a given state. We assume that at most one alarm is active at any time during a system execution and that non-alarm events can disable an alarm. The ACTMC semantics can be intuitively described as follows. Imagine that the underlying CTMC is now equipped with a timer. When the timer is turned off, the ACTMC behaves exactly as the underlying CTMC. Whenever a state with an active alarm is visited and the timer is off at that moment, it is turned on and set to ring after randomly chosen time according to the cumulative distribution function associated to the active alarm. Subsequently, the process keeps behaving as the underlying CTMC until either

- a state is visited, where the alarm is not active (in which case the timer is turned off), or
- the accumulated time from the moment of turning the timer on equals to the value when the timer *rings* (then the state is changed according to the stochastic transition kernel of the active alarm and the current state).

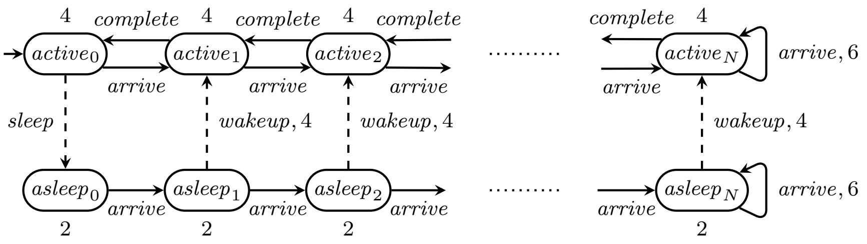


Figure 1.1: Dynamic power manager of a disk drive.

Example 1.0.1. *To get some intuition about the modeling formalisms, consider a dynamic power management of a disk drive inspired by [131]. The behavior of the disk drive can be described as follows (see Figure 1.1): At every moment, the drive is either* active *or* asleep*, and it maintains a queue of incoming I/O operations of capacity N. The events of* arriving *and* completing *an I/O operation have exponential distributions with rates* 1.39 *and* 12.5*, respectively. When the queue is full, all newly arriving I/O operations are rejected. The I/O operations are performed only in the* active *mode. When the drive is* active *and the queue becomes empty, an internal clock is set to d_s. If no further I/O request is received within the next d_s time units, the sleep event changes the mode to* asleep. *When the drive is* asleep *and some I/O operation arrives, the internal clock is set to d_w and after d_w time the wakeup event changes the mode to* active*. The rest of the model will be explained later.*

An ACTMC model of a real-world system is provided in Example 1.0.1. This can be seen as the first evidence that the class of ACTMCs is expressive and allows to model systems with practical relevance. The class of ACTMCs is a good trade-off between modeling power and analytical tractability, since all the performance measures analyzable in CTMCs can be efficiently approximated up to an arbitrary small error for ACTMCs using simple algorithms. All generally considered richer classes have either more complicated algorithms to compute the performance measures or the solution methods support only statistical error guarantees.

In a *parametric ACTMC,* we assume that every alarm distribution depends on one single parameter ranging over a given interval of eligible values. A *parameter function* assigning an eligible parameter value to every alarm yields a (non-parametric) ACTMC. Thus, the semantics of a parametric ACTMC is a family of ACTMCs induced by some parameter function. Naturally, there is a question which parameter function would be optimal with respect to some optimization objective. In this thesis, we consider an optimal parameter problem for parametric ACTMCs with mean-payoff and total accumulated reward optimization objectives. As the parameter space is dense, we cannot expect an algorithm that would compute optimal strategies for parametric ACTMCs exactly. Thus, we aim towards an algorithm which for an arbitrarily small positive number ε yields an ε-optimal parameter function.

Example 1.0.1 (cont.). *We annotate rewards in terms of energy consumption per time unit or instantaneous energy losses for events. I.e., rewards here are considered as costs and the aim is to minimize them. The power consumption is* 4 *and* 2 *per time unit in the states* active *and* asleep, *respectively. Moving from* asleep *to* active *requires* 4 *units of energy. Rejecting a newly arrived I/O request when the queue is full is undesirable, what is penalized by a loss of* 6 *units of energy. All other transitions consume* 1 *energy unit. Obviously, the designer of the disk drive controller has some freedom in choosing the delays* d_s *and* d_w*, i.e., they are free parameters of a Dirac distribution. However,* d_w *cannot be lower than the minimal time required to wake up the drive, which is constrained by the physical properties of the hardware used in the drive. Further, there is also a natural upper bound on* d_s *and* d_w *given by the capacity of the internal clock. Observe that if* d_s *is too small, then many costly transitions from* asleep *to* active *are performed; and if* d_s *is too large, a lot of time is wasted in the* active *state that consumes more power. Similarly, if* d_w *is too small, a switch to the* active *mode is likely to be invoked with a few I/O operations in the queue, and more energy could have been saved by waiting somewhat longer; and if* d_w *is too large, the risk of rejecting newly arriving I/O operations increases. Now we may ask the following instance of an optimal parameter synthesis problem we deal with in this thesis:*

> What values should a designer assign to the delay parameters d_s and d_w such that the long-run average power consumption is minimized?

1.1 Problem Statement

To express specification performance measures we enriched the ACTMC with a standard reward structure: a specific subset of goal states, a rate reward that is obtained for each time unit spent in a state, and an impulse reward that is obtained when an exponential or alarm transition is taken. Observe that costs can be modeled by rewards since they are basically the same object with an exception that costs are generally perceived to be "bad" and the aim is to minimize them, and rewards are perceived to be "good" and the aim is to maximize them [104]. Given a parametric ACTMC $\mathcal{N}$ with a parameter function $\mathbf{d}$ and a fixed reward structure, we consider the following two performance measures:

- The *expected total accumulated reward* denoted as $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$, i.e., the expected reward paid on runs from a given initial state of $\mathcal{N}$ until reaching any goal state of the reward structure,
- The *expected mean payoff* denoted by $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$ is the expected reward paid in a long run, i.e., the expectation of the following random variable

$$\mathrm{MP} \overset{\text{def}}{=} \limsup_{n\to\infty} \frac{\text{reward accumulated up to } n \text{ state changes}}{\text{time elapsed until } n \text{ state changes}}.$$

For a given performance measure $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}] \in \{\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}], \mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]\}$ and an error bound $\varepsilon > 0$, we say that a parameter function $\mathbf{d}$ is *ε-minimal* iff

$$\left| \mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}] - \inf_{\mathbf{d}'} \mathbb{E}^{\mathbf{d}'}_{\mathcal{N}}[\mathrm{PM}] \right| < \varepsilon.$$

Here, $\mathbf{d}'$ ranges over all eligible parameter functions. An ε-maximal parameter function can be defined similarly. We use the term optimal if we want to refer to both maximal and minimal cases. The *ε-optimal parameter synthesis problem* for a given parametric ACTMC, a reward structure, a performance measure $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}] \in \{\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}], \mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]\}$, and an error bound $\varepsilon > 0$ is to compute an ε-optimal parameter function.

1.2 Contribution of the Thesis

The contribution of the thesis is as follows:

- We provide *explicit algorithms* that solve our ε-optimal parameter synthesis problems for ACTMCs satisfying two mild assumptions. The algorithms are based on a reduction to problems of finding an ε-optimal policy in a *partially observable Markov decision process* (POMDP) with uncountably many actions (one for each pair of alarm and its eligible parameter value). Hence each policy of the POMDP corresponds to a unique parameter function of the ACTMC and vice versa. Then the POMDP is discretized using a sufficiently small discretization step $\delta(a)$ for each alarm a and error tolerance κ such that an ε-optimal policy exists even when the policy space is restricted to the discretized parameter values and effects of actions are computed with precision κ. Finally, we use standard algorithms for POMDPs to find an optimal policy in the discretized POMDP, that yields an ε-optimal parameter function for the given ACTMC.

- Moreover, we provide *symbolic algorithms* that solve our ε-optimal parameter synthesis problems for ACTMCs that satisfy additional assumptions. The symbolic algorithms are much faster than the explicit algorithms, since they do not explicitly compute (usually enormous) discretized set of actions of POMDPs. Instead, we represent the actions symbolically and modify a specific algorithm for solving Markov decision processes called *policy iteration*. This allows us to explicitly compute only a small set of necessary actions on the fly.
- We show that all the abstractly formulated assumptions are fulfilled for timeout events modeled by Dirac distributions, uniformly distributed alarms (employed, e.g., in variants of the CSMA/CD protocol [21]), exponential distributions (used, e.g., to specify the biochemical reactions [88]), and Weibull distributions (used to model hardware failures [121]). We argue that the approach is applicable to a large set of distributions where the densities of distribution functions can be effectively approximated by polynomials.
- We provide theoretical bounds for $\delta(a)$ and κ that work for every ACTMC. Moreover, we present heuristic algorithms that compute significantly better values of $\delta(a)$ and κ for most of the given ACTMCs.
- We provide an experimental evaluation of the algorithms on practically motivated models.

1.3 Author's Publications

In this section, we list all of the author's scientific publications along with his contribution. All listed papers are published in proceedings of recognized international conferences.

[22] M. Bezděka, O. Bouda, Ľ. Korenčiak, M. Madzin, and V. Řehák. "Sequence chart studio". In: *Application of Concurrency to System Design (ACSD)*. IEEE Computer Society Press, 2012, pp. 148–153.

This is a tool paper on the Sequence Chart Studio program, which I was developing with my colleagues for 4 years. The tool is designed for modeling and verification of systems in message sequence charts formalism. I participated in discussions and in writing the main body of the paper, thus I evaluate my contribution to be approximately 25%.

[34] T. Brázdil, Ľ. Korenčiak, J. Krčál, J. Křetínský, and V. Řehák. "On time-average limits in deterministic and stochastic Petri nets". In: *International Conference on Performance Engineering (ICPE)*. Poster paper. ACM Press, 2013, pp. 421–422.

This paper extends results presented in [36]. I wrote part of the paper and participated in proofreading the paper. My contribution is approximately 10%.

[98] Ľ. Korenčiak, J. Krčál, and V. Řehák. "Dealing with zero density using piecewise phase-type approximation". In: *European Performance Engineering Workshop (EPEW)*. Vol. 8721. Lecture Notes in Computer Science. Springer, 2014, pp. 119–134.

The paper presents an extension of the phase-type approximation technique (see, e.g., [23]). Significant amount of work was devoted to designing and performing experiments that demonstrate feasibility of the new approach. I was mainly participating in the experiments and in proofreading the paper. My contribution is approximately 35%.

[32] T. Brázdil, Ľ. Korenčiak, J. Krčál, P. Novotný, and V. Řehák. "Optimizing performance of continuous-time stochastic systems using timeout synthesis". In: *Quantitative Evaluation of Systems (QEST)*. Vol. 9259. Lecture Notes in Computer Science. Springer, 2015, 141–159.

The publication contains theoretical results and an explicit algorithm for ε-minimal total-reward parameter synthesis in ACTMC with alarms that have Dirac distributions. I participated in discussions, proved the core results and wrote roughly 70% of a very long appendix. Thus my contribution is approximately 25%.

[100] Ľ. Korenčiak, V. Řehák, and A. Farmadin. "Extension of PRISM by synthesis of optimal timeouts in fixed-delay CTMC". In: *integrated Formal Methods (iFM)*. Vol. 9681. Lecture Notes in Computer Science. Springer, 2016, pp. 130–138.

The paper presents an extension of PRISM model checker to support specification of ACTMCs with Dirac-distributed alarms and ε-minimal total-reward parameter synthesis using the explicit algorithm of [32]. I developed most of the source code (including several prototype implementations). I participated in discussions, writing of the paper, and proofreading. My contribution is approximately 60%.

[99] Ľ. Korenčiak, A. Kučera, and V. Řehák. "Efficient timeout synthesis in fixed-delay CTMC using policy iteration". In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society Press, 2016, pp. 367–372.

The paper presents a symbolic algorithm for ε-minimal total-reward parameter synthesis in ACTMCs with Dirac-distributed alarms. I originated the main idea of the paper, formulated the symbolic algorithm, and programmed the algorithm for experimental evaluation. I participated in discussions, in writing of the paper, and in proofreading. My contribution is approximately 50%.

[15] C. Baier, C. Dubslaff, Ľ. Korenčiak, A. Kučera, and V. Řehák. "Mean-payoff optimization in continuous-time Markov chains with parametric alarms". In: *Quantitative Evaluation of Systems (QEST)*. Vol. 10503. Lecture Notes in Computer Science. Springer, 2017, pp. 190–206.

This paper extends the symbolic algorithm of [99] to support also minimization of expected mean payoff in general ACTMCs that satisfy abstractly formulated criteria. I originated the main idea of the paper, heavily participated in discussions, reprogramming the algorithm for experimental evaluation, writing of the paper, and proofreading. My contribution is approximately 40%.

[16] C. Baier, C. Dubslaff, Ľ. Korenčiak, A. Kučera, and V. Řehák. "Synthesis of optimal resilient control strategies". In: *Automated Technology for Verification and Analysis (ATVA)*. Vol. 10482. Lecture Notes in Computer Science. Springer, 2017, pp. 417–434.

The paper presents an elegant approach for identifying multi-objective optimal strategies in Markov decision processes. I wrote a part of the main result proof and heavily participated in discussions and in proofreading the paper. My contribution is approximately 20%.

This thesis provides a complete and well-structured presentation of the state-of-the-art results about ε-optimal parameter synthesis of both expected total reward and mean payoff in parametric ACTMC. Namely it contains results from the conference papers [15, 32, 99, 100]. Moreover, some material was added or improved, namely:

- More appropriate definitions of ACTMCs and POMDPs are provided.
- Errors in proofs for computation of theoretical bounds on $\delta(a)$ and κ were identified and fixed.
- The synthesis algorithms were extended to support general parametric ACTMCs when optimizing the expected total reward (previously it was only for ACTMCs with Dirac-distributed alarms).
- Moreover, the synthesis algorithms were extended to support maximization of performance measures. This enabled us to introduce new heuristics for obtaining better values of $\delta(a)$ and κ when optimizing both expected mean payoff or total reward.
- The symbolic algorithm for optimization of the expected mean payoff was generalized to support a larger class of ACTMCs.

1.4 Outline of the Thesis

In Chapter 2, we state all the necessary definitions. Then we review related formalisms, performance measures, and verification and synthesis approaches in Chapter 3. The rest of the thesis follows the order of bullets in the list of Section 1.2. In Chapter 4, we present the reduction to POMDP problems and the explicit algorithms. Then we present the symbolic algorithms in Chapter 5. In Chapter 6, we provide formulas and heuristics to compute the discretization steps $\delta(a)$ and error tolerance κ and we show that the abstract assumptions of our synthesis algorithms are fulfilled for Dirac, uniform, exponential, Weibull, and other distributions. In Chapter 7, we present the experimental evaluation of the presented algorithms. We conclude the thesis and provide suggestions for future research topics in Chapter 8.

Chapter 2

Foundations

In this section we formally define formalisms and problems we will deal with thorough this thesis. We first provide a definition of parametric alarm continuous-time Markov chains (ACTMCs) and then we define well studied formalism of partially observable semi-Markov decision processes that will be used for synthesis of optimal parameters in ACTMCs. We will now define some basic notation.

Let $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Q}_{\geq 0}$, $\mathbb{Q}_{>0}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ denote the set of all positive integers, non-negative integers, non-negative rational numbers, positive rational numbers, non-negative real numbers, and positive real numbers, respectively. For clarity of presentation we denote vectors as functions, i.e., vector $\mathbf{d} \in \mathbb{R}^A$ is denoted as $\mathbf{d}\colon A \to \mathbb{R}$. We use boldface for vectors in places where we want to emphasize that they are vectors. Furthermore, for a countable set A, we denote by $\mathcal{D}(A)$ the set of discrete probability distributions over A, i.e., functions $\mu\colon A \to \mathbb{R}_{\geq 0}$ such that $\sum_{a \in A} \mu(a) = 1$. The *support* of μ is the set of all $a \in A$ with $\mu(a) > 0$. A *probability matrix* over some finite A is a function $M\colon A \times A \to \mathbb{R}_{\geq 0}$ where $M(a, \cdot) \in \mathcal{D}(A)$ for all $a \in A$.

We also use general distributions on time, in this thesis. Each such distribution will be specified by its cumulative distribution function (CDF) $F : \mathbb{R} \to [0,1]$. A *probability density function* (PDF) of a continuous distribution with a CDF F is a function $f : \mathbb{R} \to \mathbb{R}$ such that $f(\tau) = \mathrm{d}F(\tau)/\mathrm{d}\tau$ for each $\tau \in \mathbb{R}$. E.g., the *exponential distribution* has CDF $F(\tau) = 1 - e^{-\lambda\tau}$ and PDF $f(\tau) = \lambda \cdot e^{-\lambda\tau}$, where the parameter $\lambda \in \mathbb{R}_{>0}$ is called *rate*. In Table 2.1 we list the CDFs and parameters of distributions that will be often used in this thesis.

A measurable space is a pair $(\Omega, \mathcal{F})$, where Ω is a non-empty set called *sample space* and $\mathcal{F} \subseteq 2^\Omega$ is a *σ-field* of Ω, i.e., $\Omega \in \mathcal{F}$ and $\mathcal{F}$ is closed under complements and countable unions. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathrm{Pr})$, where $(\Omega, \mathcal{F})$ is a measurable space and $\mathrm{Pr}\colon \mathcal{F} \to [0,1]$ is a *prob-*

Name	CDF	Parameters
exponential	$F(\tau) = 1 - e^{-\lambda\tau}$	$\lambda \in \mathbb{R}_{>0}$
Weibull	$F(\tau) = 1 - e^{-(\tau/\lambda)^k}$	$\lambda, k \in \mathbb{R}_{>0}$
Dirac	$F(\tau) = \begin{cases} 0 & \text{if } \tau < d \\ 1 & \text{if } \tau \geq d \end{cases}$	$d \in \mathbb{R}_{>0}$
uniform	$F(\tau) = \max\left\{0, \min\left\{1, \frac{\tau-a}{b-a}\right\}\right\}$	$a, b \in \mathbb{R}_{\geq 0}, a < b$

Table 2.1: Cumulative distribution functions.

ability measure, i.e., $\Pr(\Omega) = 1$ and it is countably additive. For more details see [133].

Let $A \subseteq B$ be sets. Then $\mathbf{1}_A \colon B \to \{0,1\}$ is an *indicator vector (function)* such that $\mathbf{1}_A(a) = 1$ if and only if $a \in A$. For $a \in B$ we set $\mathbf{1}_a \overset{\text{def}}{=} \mathbf{1}_{\{a\}}$. The function $\mathbf{0}_B : B \to \{0\}$ assigns zero to each element of B. Encoding size of an object O is denoted by $||O||$.

2.1 Continuous-Time Markov Chain

A *continuous-time Markov chain (CTMC)* is a triple $\mathcal{C} = (S, Q, s_{\text{in}})$, where S is a finite set of states, $Q : S \times S \to \mathbb{R}_{\geq 0}$ is a matrix of rates such that $\sum_{s' \in S} Q(s, s') > 0$ for each $s \in S$, and $s_{\text{in}} \in S$ is an initial state. For simplicity we define an *exit rate* $\lambda_s \in \mathbb{R}_{>0}$ of a state $s \in S$ as $\sum_{s' \in S} Q(s, s')$. A run of a CTMC $\mathcal{C}$ is an infinite alternating sequence of states and times $\omega = s_0 t_0 s_1 t_1 \ldots$ where $s_0 = s_{\text{in}}$, s_i is the i-th state, and $t_i \in \mathbb{R}_{>0}$ is the time spent in s_i. For each $i \in \mathbb{N}_0$ the time t_i and the next state s_{i+1} are chosen as follows:

- the time t_i is chosen randomly according to the exponential distribution with rate λ_s and
- the next state s_{i+1} is chosen randomly according to the discrete distribution $Q(s_i, \cdot)/\lambda_s$.

We do not provide the formal semantics of CTMCs by (defining the probability measure for sets of runs) since it can be easily obtained from the formal semantics of ACTMCs that is presented later. Instead, we illustrate the definitions on an example.

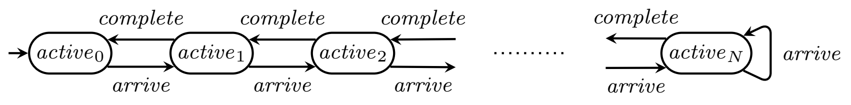


Figure 2.1: CTMC model of an M/M/1/N queue.

Example 2.1.1. *Assume a CTMC model of an M/M/1/N queue depicted in Figure 2.1. The execution starts in the initial state* active_0*. Here, there is only the* arrive *transition possible with rate say* 1.39*. Then, the time* t_0 *spent in* active_0 *is chosen randomly according to density* $1.39 \cdot e^{-1.39 t_0}$ *and the next state is* active_1 *with probability* 1*. There are* arrive *and* $\mathit{complete}$ *transitions possible in state* active_1 *with rates* 1.39 *and* 12.5*, respectively. Then, the time* t_1 *spent in* active_1 *is chosen randomly according to distribution* $13.89 \cdot e^{-13.89 t_1}$ *and the next state is* active_2 *or* active_0 *with probability* $1.39/13.89 \approx 1/10$ *and* $12.5/13.89 \approx 9/10$*, respectively. The execution proceeds in the same manner.*

CTMCs attracted a lot of attention because they possess Markov (or memoryless) property, i.e., the conditional probability of reaching future states in some time depends only on the current state s and not on the previous history including the amount of time already spent in s. The Markov property follows directly from the memoryless property of exponential distributions. There are alternative definitions of CTMC semantics, that are equivalent to ours thanks to the memoryless property. I.e., one can define the next state s_{i+1} and the time t_i spent in current state s_i as follows:

- For each $s \in S$ that has $Q(s_i, s) > 0$ a variable t_s is set randomly according to the exponential distribution with rate $Q(s_i, s)$.
- Let s be the state with minimal t_s. Then the time t_i is set to t_s and the next state s_{i+1} is set to s.

The alternative definition is more intuitive for description of M/M/1/N queue. We informally show how it works in Example 2.1.2.

Example 2.1.2. *We illustrate the alternative definition on a CTMC of an M/M/1/N queue depicted in Figure 2.1. The execution is the same as in Example 2.1.1 until reaching state* active_1*. There are* arrive *and* $\mathit{complete}$ *transitions possible in* active_1 *with rates* 1.39 *and* 12.5*, respectively. Here, times* t_{arrive} *and* t_{complete} *are chosen randomly according to rates* 1.39 *and* 12.5*, respectively. If* $t_{\mathit{arrive}} < t_{\mathit{complete}}$ *the* arrive *transition executes. If* $t_{\mathit{arrive}} > t_{\mathit{complete}}$ *the* $\mathit{complete}$ *transition executes. Observe that there is probability* 0 *that* $t_{\mathit{arrive}} = t_{\mathit{complete}}$*. The execution proceeds in the same manner.*

2.2 Continuous-Time Markov Chain with Alarms

We extend CTMCs by generally distributed events called *alarms*. A *CTMC with alarms (ACTMC)* over a finite set of alarms A is a tuple

$$\mathcal{A} \;=\; \big(S, Q, s_{\text{in}}, A, \langle S_a\rangle, \langle P_a\rangle, \langle F_a\rangle\big),$$

where (S, Q, s_{in}) is a CTMC and $\langle S_a\rangle, \langle P_a\rangle$, and $\langle F_a\rangle$ are tuples defined as follows: $\langle S_a\rangle = (S_a)_{a\in A}$ where S_a is the set of states where an alarm $a \in A$ is enabled; $\langle P_a\rangle = (P_a)_{a\in A}$ where P_a is a probability matrix of some alarm $a \in A$ for which $P_a(s,s) = 1$ if $s \in S \backslash S_a$; and $\langle F_a\rangle = (F_a)_{a\in A}$ where F_a is the CDF according to which the ringing time of an alarm $a \in A$ is distributed. We assume that each alarm distribution has a finite mean and $F_a(0) = 0$, i.e., a positive ringing time is chosen almost surely. Furthermore, we require $S_a \cap S_{a'} = \emptyset$ for $a \neq a'$, i.e., in each state at most one alarm is enabled. The set of states where some alarm is enabled is denoted by S_{on}, and we also use S_{off} to denote the set $S\backslash S_{\text{on}}$. The pairs $(s,s') \in S\times S$ with $Q(s,s') > 0$ and $P_a(s,s') > 0$ are referred to as *delay transitions* and a-*alarm transitions*, respectively.

Operational behavior. Since in every state only one alarm is active, the semantics of an ACTMC can be seen as an infinite CTMC amended with a timer that runs backwards and is set whenever a new alarm is set or the alarm gets disabled. A *run* of the ACTMC $\mathcal{A}$ is an infinite sequence $(s_0,\eta_0)t_0(s_1,\eta_1)t_1\ldots$ where $s_0 = s_{\text{in}}$, s_i is the current state, η_i is the current value of the timer, and t_i is the time spent in s_i. If $s_0 \in S_{\text{off}}$, then $\eta_0 \overset{\text{def}}{=} \infty$. Otherwise, $s_0 \in S_a$ for some $a \in A$ and the value of η_0 is selected randomly according to F_a. In a current configuration (s_i,η_i), a random delay t is chosen according to the exponential distribution with rate λ_{s_i}. Then, the time t_i and the next configuration (s_{i+1},η_{i+1}) are determined as follows:

- If $\eta_i \leq t$ (thus $s_i \in S_a$ for some $a \in A$), then $t_i \overset{\text{def}}{=} \eta_i$ and s_{i+1} is selected randomly according to $P_a(s_i,\cdot)$. The value of η_{i+1} is either set to ∞ or selected randomly according to F_b for some $b \in A$, depending on whether the chosen s_{i+1} belongs to S_{off} or S_b, respectively (note that it may happen that $b = a$).

- If $\eta_i > t$, then $t_i \overset{\text{def}}{=} t$ and s_{i+1} is selected randomly according to $Q(s_i,\cdot)/\lambda_{s_i}$. Clearly, if $s_{i+1} \in S_{\text{off}}$, then $\eta_{i+1} \overset{\text{def}}{=} \infty$. Further, if $s_i, s_{i+1} \in S_a$ for some $a \in A$, then $\eta_{i+1} \overset{\text{def}}{=} \eta_i - t$. Otherwise, $s_i \notin S_b$ and $s_{i+1} \in S_b$ for some $b \in A$ and η_{i+1} is selected randomly according to F_b.

Example 2.2.1. *The model explained in Example 1.0.1 is example of an ACTMC that extends the CTMC of Example 2.1.1 with two Dirac-distributed alarms $sleep$ and $wakeup$ and additional states. The set S_{sleep} and S_{wakeup} equal to $\{active_0\}$ and $\{asleep_1, \ldots, asleep_N\}$, respectively. The probability matrices of sleep and $wakeup$ are given by the corresponding alarm transitions of probability 1 depicted by dashed arrows in Figure 1.1. The distributions $F_{sleep}(d)$ and $F_{wakeup}(d)$ are Dirac distributions with fixed delay parameters 2 and 4, respectively. Recall that the CDF F of Dirac distribution with delay $d \in \mathbb{R}_{>0}$ assigns $F(\tau) = 0$ for $\tau < d$ and $F(\tau) = 1$ otherwise.*

The run of the ACTMC starts in the initial state $active_0$ that belongs to S_{sleep}. Thus η_0 is set randomly according to the CDF F_{sleep}, i.e., in this case it is set to 2 with probability 1. Time t is also chosen randomly according to the exponential distribution with rate 1.39, e.g., $t = 3$. Since $\eta_0 \leq t$, time $t_0 \overset{def}{=} 2$ is spent in $active_0$, the next state is $asleep_0$ with probability 1, and $\eta_1 \overset{def}{=} \infty$ since $asleep_0 \in S_{\text{off}}$. Hence, time t is chosen the same way as before, e.g., $t = 20$, and since $\eta_1 > t$ the time t_1 is set to 20 and the next state s_2 is $asleep_1$. State $asleep_1$ is in S_{wakeup}, thus η_2 is set to 4 with probability 1 according to F_{wakeup}. Finally assume, that in the current configuration $(s_2, \eta_2) = (asleep_1, 4)$ time t is chosen to smaller value than 4, e.g., $t = 3$. Thus, the next configuration is $(asleep_2, 4-t) = (asleep_2, 1)$, since both $asleep_1$ and $asleep_2$ belong to S_{wakeup}. The run executed so far is $(active_0, 2)2(asleep_0, \infty)20(asleep_1, 4)3(asleep_2, 1)\ldots$

Formal Semantics. We will now define a probability space over all runs initiated in a given $s_{\text{in}} \in S$. First, we define necessary measures, that formalize the choice of timer in configurations. Let μ_0 be the probability measure of η_0, i.e., if $s_{\text{in}} \in S_a$ for some $a \in A$ then μ_0 is measure of F_a, otherwise ∞ is chosen with probability 1, i.e., $\mu_0(\{\infty\}) = 1$. Similarly, let $\mu_{\eta,t}^{s,s'}$ be the measure of the distribution of timer value η' in configuration (s', η') if the previous configuration was (s, η) and time t was spent in (s, η). More formally, $\mu_{\eta,t}^{s,s'}$ is defined as follows:

$$\begin{aligned}
\mu_{\eta,t}^{s,s'}(\infty) &\overset{\text{def}}{=} 1 && \text{if } s' \in S_{\text{off}} \text{ (timer is turned off),}\\
\mu_{\eta,t}^{s,s'}(\eta - t) &\overset{\text{def}}{=} 1 && \text{if } s, s' \in S_a \text{ and } \eta > t \text{ (timer is kept), or}\\
\mu_{\eta,t}^{s,s'} &\text{ is measure of } F_b && \text{if } s' \in S_b \text{ for some } b \in A \text{ and}\\
& && \eta = t \text{ or } s \notin S_b \text{ (timer is newly set).}
\end{aligned}$$

Now we define a probability space $(\Pr_{\mathcal{A}}, \Omega_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}})$: $\Omega_{\mathcal{A}}$ is the set of all runs initiated in s_{in} and $\mathcal{F}_{\mathcal{A}}$ is a σ-field over $\Omega_{\mathcal{A}}$ generated by the set of all *cylinder sets* of the form $R^{s_0,I_0,\dots,s_n,I_n}_{J_0,\dots,J_{n-1}} = \{(s'_0,\eta_0)t_0\cdots \in \Omega_{\mathcal{A}} \colon \forall i \le n \;.\; s'_i = s_i \text{ and } \eta_i \in I_i, \text{ and } \forall i < n \;.\; t_i \in J_i\}$, here $s_0,\dots,s_n \in S$, $J_0,\dots,J_{n-1} \subseteq \mathbb{R}_{>0}$ are intervals, and each I of $\{I_0,\dots,I_n\}$ is either an interval on $\mathbb{R}_{>0}$ (i.e., $I \subseteq \mathbb{R}_{>0}$) or $I = \{\infty\}$. Given such a cylinder set $R^{s_0,I_0,\dots,s_n,I_n}_{J_0,\dots,J_{n-1}}$, we define the probability that a run of $\mathcal{A}$ belongs to $R^{s_0,I_0,\dots,s_n,I_n}_{J_0,\dots,J_{n-1}}$ by

$$\mathrm{Pr}_{\mathcal{A}}\left[R^{s_0,I_0,\dots,s_n,I_n}_{J_0,\dots,J_{n-1}}\right] \overset{\text{def}}{=}$$

$$\overset{\text{def}}{=} \mathbf{1}_{s_{\text{in}}}(s_0) \cdot \int\limits_{\eta_0\in I_0} \int\limits_{t_0\in J_0} \cdots \int\limits_{\eta_n\in I_n} \mu_0(\mathrm{d}\eta_0) \cdot \prod_{i=0}^{n-1} \mathrm{Pr}^{s_i,s_{i+1}}_{\eta_i}(\mathrm{d}t_i) \cdot \mu^{s_i,s_{i+1}}_{\eta_i,t_i}(\mathrm{d}\eta_{i+1}),$$

where the probability measure $\mathrm{Pr}^{s,s'}_{\eta}$ stands for the probability of moving to s' in a given time interval when staying in a configuration (s,η), i.e., for all $s,s' \in S$, a timer value $\eta \in \mathbb{R}_{>0} \cup \{\infty\}$, and an interval $[l,u] \subseteq \mathbb{R}_{\ge 0}$

$$\begin{aligned}\mathrm{Pr}^{s,s'}_{\eta}([l,u]) \overset{\text{def}}{=} & \underbrace{\mathbf{1}_{[0,\eta]}(l) \cdot (e^{-\lambda_s l} - e^{-\lambda_s \min\{u,\eta\}}) \cdot Q(s,s')/\lambda_s}_{\text{exp-delay transition occurred in the interval } [l,\min\{u,\eta\}]} \\ & + \underbrace{\mathbf{1}_{[l,u]}(\eta) \cdot e^{-\lambda_s \eta} \cdot \sum_{a\in A} \mathbf{1}_{S_a}(s) \cdot P_a(s,s')}_{a\text{-alarm transition occurred after time } \eta \in [l,u]}.\end{aligned}$$

The probability measure $\mathrm{Pr}_{\mathcal{A}}$ then extends uniquely to all sets of $\mathcal{F}_{\mathcal{A}}$.

Let $s,s' \in S$ be states of an ACTMC $\mathcal{A}$. We say that s' is *reachable* from s in $\mathcal{A}$ if there is a finite sequence $s_0,\dots,s_n$ of states such that $s = s_0$, $s' = s_n$, and $Q(s_i,s_{i+1}) > 0$ or $P_a(s_i,s_{i+1}) > 0$ (for some $a \in A$) for all $0 \le i \le n$. A state s is *unreachable* if it is not reachable from the initial state s_{in}. We say that $\mathcal{A}$ is *strongly connected* if its underlying graph is, i.e., for all $s,s' \in S$ it holds that s' is reachable from s.

Note that the timer is set to a new value in a state s only if $s \in S_a$ for some $a \in A$, and the previous state either does not belong to S_a or the transition used to enter s was an alarm transition[1]. Formally, we say that $s \in S_a$ is an *a-setting state* if there exists $s' \in S$ such that either $P_b(s',s) > 0$ for some $b \in A$, or $s' \notin S_a$ and $Q(s',s) > 0$. The set of all alarm-setting states is denoted by S_{set}. If $S_{\text{set}} \cap S_a$ is a singleton for each $a \in A$, we say that the alarms in $\mathcal{A}$ are *localized*. A configuration of an ACTMC run is called *regenerative* if its timer is set to ∞, or it was just set using a CDF of an active

1. In fact, another possibility (which does not require any special attention) is that s is the initial state of an ACTMC.

alarm. In the regenerative configuration the ACTMC run looses memory, i.e., the ACTMC from the state of a regenerative configuration behaves the same way as it was started in that state, regardless of the previous behavior. Observe, that one state can occur both in configurations that are and are not regenerative. Moreover, all reachable states of S_{set} appear in some regenerative configuration.

Reward Structure. To allow formalization of performance properties we enrich the model in a standard way with costs or rewards (see, e.g., [130]). For an ACTMC $\mathcal{A}$ with a state space S we additionally define a set of goal states G, reward rates $\mathcal{R}$, and impulse rewards $\mathcal{I}$, $\mathcal{I}_A$. Formally, the goal states G is a nonempty subset of S such that $s_{\text{in}} \notin G$ and $G \cap S_a = \emptyset$ for each $a \in A$, function $\mathcal{R} \colon S \to \mathbb{Q}_{\geq 0}$ assigns a reward rate $\mathcal{R}(s)$ to every state s such that the reward $\mathcal{R}(s)$ is given for every time unit spent in s, and functions $\mathcal{I}, \mathcal{I}_A \colon S \times S \to \mathbb{Q}_{\geq 0}$ assign to each delay transition and each alarm-setting transition, respectively, an instant execution reward. Note that the main difference between costs and rewards is that, costs are generally considered to be "bad" and the aim is to minimize the costs but rewards are considered to be "good" and the aim is to maximize the rewards. Thus our reward structure can be used to model also costs.

Performance Measures. We denote by MP the random variable assigning to each run $\omega = (s_0, \eta_0)t_0(s_1, \eta_1)t_1 \ldots$ the associated *mean payoff*, given by

$$\mathrm{MP}(\omega) \quad \stackrel{\text{def}}{=} \quad \limsup_{n\to\infty} \frac{\sum_{i=0}^{n} \big(\mathcal{R}(s_i) \cdot t_i + \mathcal{J}(s_i, s_{i+1})\big)}{\sum_{i=0}^{n} t_i} \,.$$

Here, $\mathcal{J}(s_i, s_{i+1})$ is either $\mathcal{I}(s_i, s_{i+1})$ or $\mathcal{I}_A(s_i, s_{i+1})$ depending on whether $t_i < \eta_i$ or not, respectively. We use $\mathbb{E}_{\mathcal{A}}[\mathrm{MP}]$ to denote the expectation of MP according to the probability measure $\mathrm{Pr}_{\mathcal{A}}$. In general, MP may take more than one value with positive probability. However, if the graph of the underlying ACTMC is strongly connected, almost all runs yield the same mean-payoff value independent of the initial state [48].

Similarly, we denote by TR the random variable assigning to each run $\omega = (s_0, \eta_0)t_0(s_1, \eta_1)t_1 \ldots$ the *total accumulated reward before reaching G* (in at least one transition), i.e.,

$$\mathrm{TR}(\omega) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} \big(t_i \cdot \mathcal{R}(s_i) + \mathcal{J}(s_i, s_{i+1})\big),$$

where $\mathcal{J}(s_i, s_{i+1})$ is either $\mathcal{I}(s_i, s_{i+1})$ or $\mathcal{I}_A(s_i, s_{i+1})$ depending on whether $t_i < \eta_i$ or not, respectively, and $n > 0$ is the minimal index such that $s_n \in G$. We set $\mathrm{TR}(\omega)$ to infinity whenever there is no such n. By $\mathbb{E}_{\mathcal{A}}[\mathrm{TR}]$ we denote the expected value of TR according to the probability measure $\mathrm{Pr}_{\mathcal{A}}$.

Example 2.2.2. *Assume the ACTMC $\mathcal{A}$ of Figure 1.1 and explained in Example 1.0.1. Thus the rate reward $\mathcal{R}(active_i) = 4$ and $\mathcal{R}(asleep_i) = 2$ for $0 \leq i \leq n$. The impulse reward $\mathcal{I}(s, s')$ equals to 6 if $s = s' = active_N$ or $s = s' = asleep_N$ and to 1 otherwise. The impulse reward $\mathcal{I}_A(asleep_i, active_i) = 4$ for $0 \leq i \leq n$. The function $\mathcal{I}_A$ returns 1 to all other pairs of states. Let*

$$\omega = (active_0, 2)2(asleep_0, \infty)20(asleep_1, 4)3(asleep_2, 1)1(active_2, \infty)\ldots$$

be a run of $\mathcal{A}$. Assume the set of goal states is $G = \{active_2\}$. The rate reward accumulated until reaching G is

$$2 \cdot \mathcal{R}(active_0) + 20 \cdot \mathcal{R}(asleep_0) + 3 \cdot \mathcal{R}(asleep_1) + 1 \cdot \mathcal{R}(asleep_2) = \\ = 2 \cdot 4 + 20 \cdot 2 + 3 \cdot 2 + 1 \cdot 2 = 56$$

The impulse reward accumulated until reaching G is $3 \cdot 1 + 4$ since $N \geq 2$ and the run ω moved from $asleep_2$ to $active_2$ what generates reward 4. Thus $\mathrm{TR}(\omega) = 56 + 7 = 63$.

Remark 2.2.3. *Observe that according to the definition of ACTMC we require $\lambda_s = \sum_{s' \in S} Q(s, s') > 0$ for each $s \in S$. This requirement simplifies the definition of the formal semantics and does not limit the modeling power of ACTMCs when using our performance measures. I.e., every model that does not satisfy this requirement can be translated to an ACTMC satisfying the requirement without changing the value of $\mathbb{E}_{\mathcal{A}}[\mathrm{MP}]$ and $\mathbb{E}_{\mathcal{A}}[\mathrm{TR}]$: Assume an ACTMC such that it contains a state s with $\lambda_s = 0$. If some run reaches state s it stays there until an alarm transition is executed. Assume that ACTMC was redefined such that $Q(s, s) > 0$ and $\mathcal{I}(s, s) = 0$. Now the runs stay again in s until the alarm transition is executed with possibly infinitely many interruptions by the newly created delay transition that does not affect the timer (since the alarm is still enabled in s), $\mathbb{E}_{\mathcal{A}}[\mathrm{MP}]$, or $\mathbb{E}_{\mathcal{A}}[\mathrm{TR}]$ (since the added delay transition incurs zero reward).*

Remark 2.2.4. *We do not study both performance measures at once. Thus the requirements on G do not limit the expressive power: If we study mean payoff, the choice of G is irrelevant. If we study the total reward and there is some state s in $G \cap S_a$ for some alarm $a \in A$, we can redefine S_a to not contain s and this change does not affect the expected total reward. The other requirements on G are natural.*

2.3 Parametric ACTMC

In ACTMCs, the distribution functions for the alarms are already fixed. For example, if alarm a is a timeout, it is set to some concrete value d, i.e., the associated F_a is a Dirac distribution such that $F_a(\tau) = 1$ for all $\tau \geq d$ and $F_a(\tau) = 0$ for all $\tau < d$. Similarly, if a is a random delay selected uniformly in the interval $[0.01, d]$, then $F_a(\tau) = 0$ for all $\tau < 0.01$ and $F_a(\tau) = \min\{1, (\tau - 0.01)/(d - 0.01)\}$ for all $\tau \geq 0.01$. We also consider alarms with Weibull distributions, where $F_a(\tau) = 0$ for all $\tau \leq 0$ and $F_a(\tau) = 1 - e^{-(\tau/d)^k}$ for all $\tau > 0$, where $k \in \mathbb{N}$ is a fixed constant.[2]

In the above examples, we can interpret d as a *parameter* and ask, e.g., what parameter values minimize the expected mean payoff. For simplicity, we restrict our attention to distributions with only *one* parameter.[3] A *parametric ACTMC* is defined similarly as an ACTMC, but instead of the concrete distribution function F_a, we specify a *parameterized* distribution function $F_a[x]$ together with the interval $[\ell_a, u_a]$ of eligible parameter values for every $a \in A$. We require $\ell_a, u_a \in \mathbb{Q}$ for each $a \in A$. For every $d \in [\ell_a, u_a]$, we use $F_a[d]$ to denote the distribution obtained by instantiating the parameter x with d. Formally, a parametric ACTMC is a tuple

$$\mathcal{N} \;=\; \big(S, Q, s_{\text{in}}, A, \langle S_a\rangle, \langle P_a\rangle, \langle F_a[x]\rangle, \langle \ell_a\rangle, \langle u_a\rangle\big)$$

where all components are defined in the same way as for ACTMCs except for the tuples $\langle F_a[x]\rangle$, $\langle \ell_a\rangle$, and $\langle u_a\rangle$ of all $F_a[x]$, ℓ_a, and u_a discussed above. Reachability, strong-connectedness, localized alarms, regenerative configuration, and reward structures are defined as for (non-parametric) ACTMCs.

A *parameter function* for $\mathcal{N}$ is a function $\mathbf{d}\colon A \to \mathbb{R}$ such that $\mathbf{d}(a) \in [\ell_a, u_a]$ for every $a \in A$. For every parameter function $\mathbf{d}$, we use $\mathcal{N}^{\mathbf{d}}$ to denote the ACTMC obtained from $\mathcal{N}$ by replacing each $F_a[x]$ with the distribution function $F_a[\mathbf{d}(a)]$. We allow only parametric ACTMCs that for each parametric function yield an ACTMC. When reward structures are defined on $\mathcal{N}$, we use $\mathrm{Pr}^{\mathbf{d}}_{\mathcal{N}}$, $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$, and $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$ to denote the probability measure, the expected mean payoff, and total reward in $\mathcal{N}^{\mathbf{d}}$, respectively. For a given

2. Note that a Weibull distribution with $k = 1$ is an exponential distribution.

3. In our current setting, distribution functions with several parameters can be accommodated by choosing a parameter to optimize and fixing the others. In some cases we can also use simple extensions to synthesize, e.g., both d_1 and d_2 for the uniform distribution in $[d_1, d_2]$ (see 6.5).

$\varepsilon > 0$ and performance measure $\mathrm{PM} \in \{\mathrm{MP}, \mathrm{TR}\}$, we say that a parameter function $\mathbf{d}$ is *ε-minimal* or *ε-maximal* in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}]$ if

$$\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}] \;\leq\; \inf_{\mathbf{d}'} \mathbb{E}^{\mathbf{d}'}_{\mathcal{N}}[\mathrm{PM}] + \varepsilon \quad \text{or} \quad \mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{PM}] \;\geq\; \sup_{\mathbf{d}'} \mathbb{E}^{\mathbf{d}'}_{\mathcal{N}}[\mathrm{PM}] - \varepsilon,$$

respectively, where $\mathbf{d}'$ ranges over all parameter functions for $\mathcal{N}$. In cases where we want to refer both minimal and maximal cases without stating it explicitly, we use the word *optimal*.

Synthesis Problems. In the following, we fix a strongly connected parametric ACTMC $\mathcal{N} = (S, Q, s_{\mathrm{in}}, A, \langle S_a\rangle, \langle P_a\rangle, \langle F_a[x]\rangle, \langle \ell_a\rangle, \langle u_a\rangle)$ and a reward structure $\mathrm{RS} = (G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$. We aim towards an algorithm synthesizing an ε-minimal or ε-maximal parameter function for $\mathcal{N}$ and expected total reward or mean payoff. That is, we deal with the following computational problems:

ε-optimal total-reward (parameter) synthesis problem (TRSP).

Input:	$\varepsilon \in \mathbb{Q}_{>0}$, a parametric ACTMC $\mathcal{N}$ and a reward structure RS, where $\mathcal{N}$ is without unreachable states and has rational Q, transition probabilities, and finitely representable $\langle F_a[x]\rangle$.
Output:	An ε-optimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$.

We can define the synthesis problem also for the expected mean payoff, but we require the ACTMC $\mathcal{N}$ to be strongly connected, i.e.,

ε-optimal mean-payoff (parameter) synthesis problem (MPSP).

Input:	$\varepsilon \in \mathbb{Q}_{>0}$, a strongly connected parametric ACTMC $\mathcal{N}$, and a reward structure RS, where $\mathcal{N}$ has rational Q, transition probabilities, and finitely representable $\langle F_a[x]\rangle$.
Output:	An ε-optimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$.

Remark 2.3.1. *Observe that each strongly connected parametric ACTMC is without unreachable states. Moreover, the restriction to parametric ACTMCs without unreachable states is not a limitation, since the unreachable states can be easily removed by a simple polynomial-time graphical algorithm (in size of the ACTMC) without changing the value of the expected total reward or mean payoff.*

2.4 Partially Observable Markov decision process

We will heavily use partially observable Markov decision processes in this thesis. Thus we will provide here necessary definitions.

Definition 2.4.1. *A* partially observable Markov decision process *(POMDP) is a tuple $\mathcal{M} = (V, Obs, O, Act, T, t, c, v_{\text{in}}, V')$, where*

- *V is a finite set of* vertices*,*
- *Obs is a finite set of observations,*
- *$O \colon V \to Obs$ returns an unique observation to each vertex,*
- *$Act = \bigcup_{o \in Obs} Act_o$ is a (possibly uncountable) set of* actions*, where $Act_o \neq \emptyset$ for each $o \in Obs$,*
- *$T \colon V \times Act \to \mathcal{D}(V)$ is a* transition function*,*
- *$t, c \colon V \times Act \to \mathbb{R}_{>0}$ provide the* time *and* reward *when executing an action from a vertex, and*
- *$v_{\text{in}} \in V$ is an initial vertex,*
- *$V' \subseteq V$ is a set of* goal vertices*.*

A (memoryless deterministic) *policy* is a function $\sigma \colon Obs \to Act$ which assigns to every observation o an action from Act_o. The behavior of a POMDP $\mathcal{M}$ with a fixed policy σ can be intuitively described as follows: An execution starts in the initial vertex v_{in}. In every step, assuming that the current vertex is v, the process plays action $b = \sigma(O(v))$ according to the policy σ on obtained observation $O(v)$, waits time $t(v, b)$ in v, generates reward $c(v, b)$, and moves to a new vertex v' with probability $T(v, b)(v')$. This way the execution forms a *run*, i.e., an alternating sequence of vertices and actions $v_0 b_0 v_1 b_1 v_2 \cdots \in (V \cdot Act)^\omega$.

Formal Semantics. Every policy σ uniquely determines a probability space $(\Omega_{\mathcal{M}}, \mathcal{F}_{\mathcal{M}}, \Pr^{\sigma}_{\mathcal{M}})$, where $\Omega_{\mathcal{M}}$ is the set of all *runs* $(V \cdot Act)^\omega$; $\mathcal{F}_{\mathcal{M}}$ is the sigma-field generated by all sets of runs $\rho(w) = \{\rho \colon \rho \text{ has prefix } w\}$ for all *finite paths* $w \in (V \cdot Act)^* \cdot V$, i.e., prefixes of runs ending with a vertex; and $\Pr^{\sigma}_{\mathcal{M}}$ is the unique probability measure such that for every finite path

$w = v_0 b_0 \dots b_{n-1} v_n$ it holds that

$$\Pr^\sigma_{\mathcal{M}}(\rho(w)) = \begin{cases} 0 & \text{if } \sigma(O(v_i)) \neq b_i \\ & \text{for some } 0 \leq i < n, \\ \mathbf{1}_{v_{\text{in}}}(v_0) \prod_{i=0}^{n-1} T(v_i, b_i)(v_{i+1}) & \text{otherwise.} \end{cases}$$

The probability measure $\Pr^\sigma_{\mathcal{M}}$ then extends uniquely to all sets of $\mathcal{F}_{\mathcal{M}}$.

A Markov decision process (MDP) is a POMDP where O is an injection. The subclass of MDPs is well studied in literature since it is much more analytically tractable than general POMDPs, see Section 3.3.5 for more details. For any $v \in V$, let us denote by $\mathcal{M}(v)$ the POMDP obtained from $\mathcal{M}$ by replacing the initial state by v. Similarly, for any $v \in V$, $\mathcal{M}[v]$ is the POMDP obtained from $\mathcal{M}$ by replacing the set of goal states by $\{v\}$.

Let $v, v' \in V$ be vertices of a POMDP $\mathcal{M}$. We say that v' is *reachable* from v in $\mathcal{M}$ if there is a finite sequence $v_0, \dots, v_n$ of vertices such that $v = v_0$, $v' = v_n$, and there is an action $b \in Act_{O(v_i)}$ such that $T(v_i, b)(v_{i+1}) > 0$ for all $0 \leq i < n$. A vertex s is *unreachable* if it is not reachable from the initial vertex v_{in}. We say that $\mathcal{M}$ is *strongly connected* if its underlying graph is, i.e., for all $v, v' \in V$ it holds that v' is reachable from v.

Performance Measures. We overload the definition of random variables TR and MP. Analogously to ACTMCs we denote by TR the random variable assigning to each run $v_0 b_0 v_1 b_1 \dots$ the *total accumulated reward before reaching* V' (in at least one transition), i.e., $\sum_{i=0}^{n-1} c(v_i, b_i)$, if there is a minimal $n > 0$ such that $v_n \in V'$, and as ∞ otherwise. By $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{TR}]$ we denote the expected value of TR according to the probability measure $\Pr^\sigma_{\mathcal{M}}$.

Similarly, we denote by MP the random variable assigning to each run $w = v_0 b_0 v_1 b_1 \dots$ the *mean-payoff*, given by

$$\mathrm{MP}(w) \stackrel{\text{def}}{=} \limsup_{n \to \infty} \frac{\sum_{i=0}^{n} c(v_i, b_i)}{\sum_{i=0}^{n} t(v_i, b_i)}.$$

By $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{MP}]$ we denote the expected value of MP according to the probability measure $\Pr^\sigma_{\mathcal{M}}$.

Given $\varepsilon \geq 0$ and a random variable $\mathrm{PM} \in \{\mathrm{TR}, \mathrm{MP}\}$ we say that a policy σ is *ε-minimal* or *ε-maximal* for $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{PM}]$ if $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{PM}] \leq \inf_{\sigma'} \mathbb{E}^{\sigma'}_{\mathcal{M}}[\mathrm{PM}] + \varepsilon$ or $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{PM}] \geq \sup_{\sigma'} \mathbb{E}^{\sigma'}_{\mathcal{M}}[\mathrm{PM}] - \varepsilon$, respectively, where σ' ranges over all policies of $\mathcal{M}$. We call σ *minimal* for $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{PM}]$ if it is 0-minimal for $\mathbb{E}^\sigma_{\mathcal{M}}[\mathrm{PM}]$. We call a policy *globally (ε-)minimal* for $\mathbb{E}^{\sigma}_{\mathcal{M}}[\mathrm{PM}]$ if it is (ε-)minimal for $\mathbb{E}^{\sigma}_{\mathcal{M}}[\mathrm{PM}]$ in $\mathcal{M}(v)$ for every $v \in V$. Similarly we can define the above notions for maximizations by replacing the word minimal by maximal. In cases where we want to denote both minimal and maximal and its derivatives we use the word optimal and its derivatives.

κ-Approximations of POMDPs. Let $\mathcal{M} = (V, Obs, O, Act, T, t, c, v_{\text{in}}, V')$ be a POMDP, and $\kappa \in \mathbb{Q}_{>0}$. We say that functions $T^{\kappa} \colon Act \to \mathcal{D}(V)$ and $t^{\kappa}, c^{\kappa} \colon Act \to \mathbb{R}_{\geq 0}$ are *κ-approximations* of T, t, c, if for all $v, v' \in V$, $o \in O(v)$, and $b \in Act_o$ it holds that $T(v,b)$ and $T^{\kappa}(v,b)$ have the same support, $|T(v,b)(v') - T^{\kappa}(v,b)(v')| \leq \kappa$, $|t(v,b) - t^{\kappa}(v,b)| \leq \kappa$, and $|c(v,b) - c^{\kappa}(v,b)| \leq \kappa$. A *$\kappa$-approximation* of $\mathcal{M}$ is a POMDP $(V, Obs, O, Act, T^{\kappa}, t^{\kappa}, c^{\kappa}, v_{\text{in}}, V')$ where T^{κ}, t^{κ}, c^{κ} are κ-approximations of T, t, c. We denote by $[\mathcal{M}]_{\kappa}$ the set of all κ-approximations of $\mathcal{M}$.

General POMDP. We use restricted notions of POMDPs and policies that are sufficient for problems that we study in this thesis. However, in literature there are more general definitions of POMDPs, where observations, rewards, and times may be given randomly according to given distributions. Thus it might be much harder (or impossible) to detect the current state and to choose the appropriate action. Moreover, there are more general definitions of policies, where an action may be chosen with some probability and the choice can depend on the whole history of the execution. More formally, let *history* be a finite sequence $o_0(b_0, c_0, t_0)o_1(b_1, c_1, t_1)\ldots(b_n, c_n, t_n)o_{n+1}$, where for each $i \leq n$ we have that o_i is observation, $b_i \in Act_{o_i}$ is the played action, c_i is the obtained reward, and t_i is the time spent in i-th state. A *history-dependent randomized* policy is a function that assigns to a history $o_0(b_0, c_0, t_0)o_1(b_1, c_1, t_1)\ldots(b_n, c_n, t_n)o_{n+1}$ a distribution $\mathcal{D}(Act_{o_{n+1}})$ on actions $Act_{o_{n+1}}$. We say that policy is *deterministic* if some action is always chosen with probability one. A policy is *memoryless* if it returns the same distribution for histories with the same last observation. We denote by HR, HD, MR, and MD the sets of history-dependent randomized, history-dependent deterministic, memoryless randomized, and memoryless deterministic policies, respectively.

Chapter 3

State of the Art

First we recall the related modeling formalisms to ACTMC. Then we define problems and performance measures, which are usually studied on the specified formalisms or are relevant to synthesis. Finally we list current most used methods for computing the values of performance measures or for the parameter synthesis.

3.1 Related Modeling Formalisms

We are interested in probabilistic models with continuous-time stochastic distributions. The section is divided into three parts. The first part consists of introduction to generalized semi-Markov process (GSMP) and some other GSMP based formalisms. The second part of this section contains a definition of stochastic Petri net (SPN) and recalls types of SPNs. The last part provides a short review of other related modeling formalisms.

3.1.1 Generalized Semi-Markov Process

Generalized semi-Markov processes (GSMPs) introduced by Matthes [118] is more general formalism than ACTMCs, because it allows multiple non-exponentially distributed alarms in a state. In the GSMP terminology, the alarms are called events. GSMPs were extensively studied in the scientific literature, since they have high modeling power and intuitive semantics [36, 71, 101].

Definition 3.1.1. *[36, 71, 101] A generalized semi-Markov process (GSMP) is a tuple $\mathcal{G} = (S, \mathcal{E}, \alpha, F, Succ, s_{\text{in}})$ where*

- *S is a finite set of states,*
- *$\mathcal{E}$ is a finite set of* events,

- $\alpha \colon S \to 2^{\mathcal{E}}$ *assigns to each state* s *a nonempty set of active events* $\alpha(s) \neq \emptyset$ *in* s,
- $Succ : S \times 2^{\mathcal{E}} \to \mathcal{D}(S)$ *is a* successor function *that assigns a probability distribution* $Succ(s, \mathcal{E}')$ *specifying a successor state to a state* s *and set of events* $\mathcal{E}'$ *that occurred simultaneously in* s,
- $F \colon \mathcal{E} \times \mathbb{R} \to [0,1]$ *provides for each event* $e \in \mathcal{E}$ *a CDF* $F(e, \cdot)$ *such that* $F(e, 0) = 0$, *and*
- $s_{\text{in}} \in S$ *is an* initial state.

The dynamics of GSMPs is similar to the one of ACTMCs. The execution of a GSMP starts in the initial state s_{in}. In every state each active event keeps one timer, thus we set a timer for each event e in $\alpha(s_{\text{in}})$ to a random value according to CDF $F(e, \cdot)$. The set of events $\mathcal{E}'$, which have the minimal timer value, first wait until their timers go off then they occur simultaneously and cause a change of state. The next state s' is chosen randomly according to the probability distribution $Succ(s, \mathcal{E}')$. In the new sate s' the occurred active events $\alpha(s') \cap \mathcal{E}'$ and the newly active events $\alpha(s') \setminus \alpha(s_{\text{in}})$ do not have a valid timer value, thus their timer is *newly set* the same way as above. The inherited events $\alpha(s') \cap \alpha(s_{\text{in}}) \setminus \mathcal{E}'$ keep their old timer value, that is decreased by the time spent in s_{in}. Then the execution proceeds in the same manner.

Similarly to ACTMCs we can define configuration as a pair of state and function $\mathbf{o} \colon \mathcal{E} \to \mathbb{R}_{>0} \cup \{\infty\}$ that assigns a timer value to each active event in the given state and ∞ to the other events. A *run* is an infinite sequence of configurations.[1] The formal semantics of a GSMP is usually defined using general state space Markov chains (see, e.g., [36]). In our setting it suffices to know that there is a well defined probability space $(\Omega_{\mathcal{G}}, \mathcal{F}_{\mathcal{G}}, \Pr_{\mathcal{G}})$ where $\Omega_{\mathcal{G}}$ is set of all runs of a GSMP, $\mathcal{F}_{\mathcal{G}}$ is a σ-field on $\Omega_{\mathcal{G}}$ and $\Pr_{\mathcal{G}}$ is a unique probability measure [17, 36]. Thus we can measure a set of runs that satisfy some property, e.g., $\Pr_{\mathcal{G}}(\text{runs that hit } s \in S \text{ in less than } 4 \text{ seconds})$.

Example 3.1.2. *We illustrate the definition of a GSMP on the graphical representation of the GSMP in Figure 3.1. The execution starts in the initial state* $idle$*. Immediately a timer is set for all active events:* new *and* $fault$*. The timer for the event* new *is set according to the exponential distribution with rate* 0.5,

1. Note that in definition of GSMP we require that there is at least one active event in each state. If this requirement is a problem when modeling a system by a GSMP we can easily overcome it: We introduce a new dummy event e such that for each state s with no active events and we set $\alpha(s) \overset{\text{def}}{=} \{e\}$, $Succ(s, e)(s) \overset{\text{def}}{=} 1$, and $F(e, \cdot)$ to an exponential CDF.

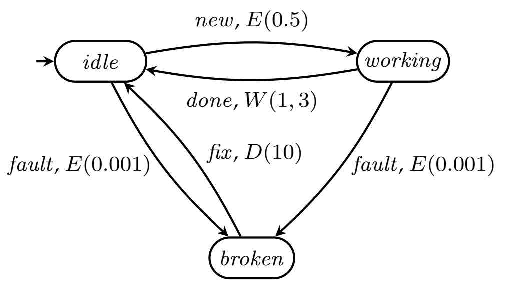


Figure 3.1: This figure depicts a GSMP model of a printer. If the printer is in the state *idle*, it either breaks in time distributed according to the exponential distribution with rate 0.001, or a new job arrives in time distributed exponentially with rate 0.5, what is denoted as $E(0.001)$ or $E(0.5)$, respectively. In the state *working* the printer either breaks or the job is successfully finished what is modeled by the event *done* with the Weibull distribution with scale $\lambda = 1$ and shape $k = 3$ denoted as $W(1,3)$. In the state *broken* the Dirac-distributed event *fix* can occur with delay 10.

say that it is 1.1. *Similarly, a timer value for the fault event is set, assume that it is* 85. *Then time passes and the event new with the minimal timer occurs. Thus the printer moves to the state working, where the only event which does not have a timer is the event done. A timer value for this event is set using the Weibull distribution with scale* 1 *and shape* 3, *say that it is* 84. *The event fault is still active, but already* 1.1 *second has passed since it was scheduled, thus it will happen in* 83.9 *seconds. Time passes again and the event fault with the minimal timer value is chosen. The GSMP moves to the state broken where the events done and fault are not active anymore, their timer values are forgotten, i.e., they are set to* ∞. *Again the active events that do not have a timer value yet get it and this process goes on infinitely. The run which was executed so far is* $(idle, (1.1, \infty, \infty, 85))(working, (\infty, 84, \infty, 83.9))(broken, (\infty, \infty, 10, \infty))\ldots$

The GSMP formalism is one of the most general formalisms that we review in this thesis. To demonstrate its relationship to other stochastic formalisms we need to define one additional notion. A configuration of a GSMP run is called *regenerative* if each of its timers is newly set, or it is ∞, or it belongs to an event with an exponential CDF. Now, we explain the relationship of GSMPs to common classes of stochastic formalisms:

- A CTMC can be equivalently defined as a GSMP where each event has an exponential CDF, i.e., both definitions of a CTMC provide the same modeling power.

- Similarly, an ACTMC can be equivalently defined as a GSMP where in each state, there is at most one active event with a non-exponential CDF.
- A *discrete time Markov chain* (DTMC) [125] is an ACTMC where all events in $\mathcal{E}$ are active only in one state and have a Dirac CDF with the same delay, i.e., there exists a constant c, such that for each $e \in \mathcal{E}$ and $\tau \in \mathbb{R}$ it holds that $F(e, \tau) = 0$ if $\tau < c$ and $F(e, \tau) = 1$ otherwise.
- A *fixed-delay CTMC* (fdCTMC) [100] is a GSMP where every event has an exponential or a Dirac CDF.
- A *Markov regenerative process* (MRP) is a GSMP where from each state (reachable from the initial state) a regenerative configuration is reached with probability 1.
- A *semi-Markov process* (SMP) [114, 118] is a GSMP where each run contains only regenerative configurations.

Reward Structure. Similarly to ACTMCs we define a *reward structure* of a GSMP as $(G, \mathcal{R}, \mathcal{I})$, where the set of goal states G and the reward rates $\mathcal{R}$ are defined the same way as for ACTMCs and a impulse rewards function $\mathcal{I} \colon S \times 2^{\mathcal{E}} \times S \to \mathbb{Q}_{\geq 0}$ assigns reward $\mathcal{I}(s, \mathcal{E}', s')$ to the change of state from s to s' triggered by occurrence of set of events $\mathcal{E}'$.

3.1.2 Stochastic Petri Net

Petri net (PN) is a very popular formalism because it is simple to explain to people unfamiliar with it thanks to an appealing graphical representation and it has high modeling power, e.g., one can specify infinite state space by a finite description. A PN consists of disjoint finite sets of *places* and *transitions*. Each transition has specified sets of *input*, *inhibitor input*, and *output* places. A *marking* is a vector assigning to each place a nonnegative number of tokens that the place contains. A transition is *enabled* (in a marking) if each of its input places contains at least one token and each of its inhibitor places contains no tokens. Each enabled transition may *fire*, i.e., change the current marking, by removing one token from each of its input places and by adding one token to each of its output places. The transition that fires is chosen nondeterministically from the enabled transitions.

There are extensions of Petri net formalism that resolve this nondeterminism using stochasticity. In a stochastic Petri net (SPN) the event competition idea from ACTMCs or GSMPs is reused, random times are assigned to enabled transitions, and the transitions with currently minimal times are fired. There are numerous definitions of SPNs, the one presented here is based on [71, 101].

Definition 3.1.3. *[71, 101] A stochastic Petri net (SPN) is a tuple* $\mathcal{S} = (P, E, E', \mathrm{In}, \mathrm{Inh}, \mathrm{Out}, F, \mathbf{p}, \mathbf{m}_0)$ *where*

- *P is a finite set of places, E is a finite set of transitions, and $E' \subseteq E$ are the* immediate *transitions (the transitions in $E \setminus E'$ are called* timed*),*
- $\mathrm{In}(e), \mathrm{Inh}(e)$, *and* $\mathrm{Out}(e)$ *are the sets of input, inhibitor input, and output places, respectively, for each $e \in E$,*
- *$F(e, \cdot)$ is a clock-setting CDF for each timed transition e,*
- $\mathbf{p}(\mathbf{m}, E'', \cdot)$ *is a distribution on resulting markings for each original marking* $\mathbf{m} \colon P \to \mathbb{N}_0$ *and each set $E'' \subseteq E$ of transitions that fired simultaneously,*
- $\mathbf{m}_0 \colon P \to \mathbb{N}_0$ *is an initial marking.*

We assume that if $\mathbf{p}(\mathbf{m}, E'', \mathbf{m}') > 0$ then for any $p \in P$ it holds that

$$\max\left\{0,\ \mathbf{m}(p) - \sum_{e \in E''} \mathbf{1}_{\mathrm{In}(e)}(p)\right\} \ \leq\ \mathbf{m}'(p)\ \leq\ \mathbf{m}(p) + \sum_{e \in E''} \mathbf{1}_{\mathrm{Out}(e)}(p).$$

Intuitively, by firing a set of transitions E'' the number of tokens in any place cannot decrease below zero or more than the number of transitions in E'' that have an input arc from this place and similarly for increasing the number of tokens. We denote the set of enabled transitions in a marking $\mathbf{m}$ as $En(\mathbf{m})$. A *configuration* of an SPN is a pair $(\mathbf{m}, \mathbf{o})$ where $\mathbf{m}$ is the current marking and $\mathbf{o} : E \to \mathbb{R}_{>0} \cup \{\infty\}$ is a vector of remaining times before transitions fire (TBTF). We assume that $\mathbf{o}(e) = \infty$ iff transition e is not enabled in the marking $\mathbf{m}$ of configuration $(\mathbf{m}, \mathbf{o})$. An execution of SPN $\mathcal{N}$ starts in the configuration $(\mathbf{m}_0, \mathbf{o}_0)$ where $\mathbf{o}_0(e)$ equals to 0 for each enabled immediate transition e and $\mathbf{o}_0(e)$ is chosen randomly according to $F(e, \cdot)$ for each enabled timed transition e. In every step, assuming that the current configuration is $(\mathbf{m}_i, \mathbf{o}_i)$, the SPN waits for time $t_i \overset{\text{def}}{=} \min_{e \in E} \mathbf{o}(e)$, fires the set of transitions $E'' \overset{\text{def}}{=} \operatorname{argmin}_{e \in E} \mathbf{o}(e)$, and then moves to a next configuration $(\mathbf{m}_{i+1}, \mathbf{o}_{i+1})$ determined as follows:

- The marking $\mathbf{m}_{i+1}$ is chosen randomly according to the probability distribution $\mathbf{p}(\mathbf{m}_i, E'', \cdot)$.
- We set the $\mathbf{o}_{i+1}$ to infinity for each transition that is not enabled in $\mathbf{m}_{i+1}$. If e is immediate we set $\mathbf{o}_{i+1}(e)$ to zero. We say that e is *newly enabled* if it is enabled in $\mathbf{m}_{i+1}$, and was not enabled in $\mathbf{m}_i$ or it just fired, i.e., $e \in En(\mathbf{m}_{i+1}) \setminus (En(\mathbf{m}_i) \setminus E'')$. If e is timed and newly enabled we set $\mathbf{o}_{i+1}(e)$ randomly according to $F(E, \cdot)$. If e is timed and enabled but not newly enabled then it is *inherited* transition and we set $\mathbf{o}_{i+1}(e) = \mathbf{o}_i(e) - t_i$.

This way, the execution of an SPN forms a *run*, an infinite sequence of configurations $(\mathbf{m}_0, \mathbf{o}_0)(\mathbf{m}_1, \mathbf{o}_1)\ldots$[2] We forbid runs in which there is from some point an infinite sequence of immediate transitions, i.e., from some point j for each $i > j$ time t_i equals to zero. We skip the definition of SPN semantics and the corresponding probability space, that can be found in [71] employing general state-space Markov chains. Here it suffices to know, that there is a well defined probability space $(\Omega_{\mathcal{S}}, \mathcal{F}_{\mathcal{S}}, \mathrm{Pr}_{\mathcal{S}})$ where $\Omega_{\mathcal{S}}$ is set of all runs of SPN, $\mathcal{F}_{\mathcal{S}}$ is a σ-field on $\Omega_{\mathcal{S}}$, and $\mathrm{Pr}_{\mathcal{S}}$ is a unique probability measure [71].[3] We say that marking $\mathbf{m}$ is *reachable* in an SPN $\mathcal{S}$ if there is a positive probability of runs $(\mathbf{m}_0, \mathbf{o}_0)(\mathbf{m}_1, \mathbf{o}_1)\ldots$ of $\mathcal{S}$ where $\mathbf{m} = \mathbf{m}_i$ for some $i \in \mathbb{N}_0$. An SPN is called *bounded* if the number of reachable markings is finite. Similarly to GSMPs, a configuration of an SPN run is called *regenerative* if each its TBTF is newly set, or it is ∞, or it belongs to a transition with an exponential CDF.

Example 3.1.4. *We illustrate the SPN definition on the SPN in Figure 3.2. We will denote the current marking as a triple where the first, second, and third position corresponds to the number of tokens in the place idle, broken, and working, respectively. The execution starts in the initial marking* $(2, 0, 0)$. *There are two transitions enabled: new and fault. Both of them do not have a TBTF yet, thus we*

2. Similarly to ACTMC, we assume that there is at least one enabled transition in each marking. We can easily transform an arbitrary SPN to an SPN that satisfies this assumption by introducing a new dummy transition e with $\mathrm{In}(e) = \mathrm{Inh}(e) = \mathrm{Out}(e) = \emptyset$ (i.e., it is always enabled and causes no change when it fires alone) and if it fires at the same time with some other set of transitions then it has no effect on the original marking change (i.e., $\mathbf{p}(\mathbf{m}, E'' \cup \{e\}, \mathbf{m}') = \mathbf{p}(\mathbf{m}, E'', \mathbf{m}')$ for arbitrary markings $\mathbf{m}, \mathbf{m}'$ and set of transitions E''). All the performance measures listed later can be easily adapted or don't have to be changed at all to return the same values on both SPNs.

3. Our definition of SPN does not allow arc multiplicities. However, each SPN with m places, n transitions, and each arch of multiplicity at most k can be easily transformed (by using immediate transitions) to an equivalent SPN which has at most $(2k + 1) \cdot m$ places, $2km + n$ transitions, and multiplicity 1 of each arc.

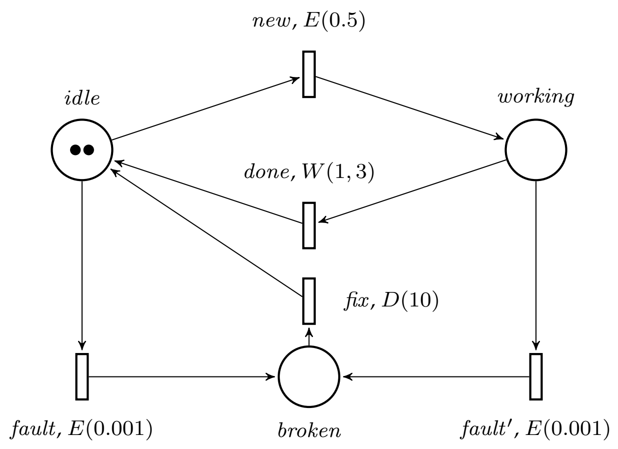


Figure 3.2: This is an SPN variation of the GSMP in Figure 3.1. We assume that there are two parallel printers available thus it is natural to model such system as an SPN. We assume that the initial marking is assigning 2 tokens to the state $idle$. The notation of distribution types of transitions is the same as for the GSMP in Figure 3.1.

set it according to their distributions. Assume that TBTF for new *is* 5 *and for fault is* 3*. The time passes and the transition fault with the minimal TBTF fires first, hence we end up in marking* $(1,1,0)$*. There are two newly enabled transitions fix and fault in marking* $(1,1,0)$*, thus a new TBTF is assigned to each, say* 10 *and* 5*. The transition* new *is inherited, thus its TBTF is* $5-3=2$*. The time passes and the transition* new *with the minimal time to fire (that is* 2*) fires after* 2 *seconds. The net moves to the marking* $(0,1,1)$ *and then it behaves in the same manner. The run that the net has executed so far is*

$$\big((2,0,0),(5,\infty,\infty,3,\infty)\big)\big((1,1,0),(2,\infty,10,5,\infty)\big)\big((0,1,1),\cdot\big)\ldots$$

The class of bounded SPNs has the same modeling power as the class of GSMPs [71]. There are several restricted classes of SPNs defined in the literature:

- A *Markov regenerative SPN* (MRSPN) [48] is an SPN such that from each configuration (reachable from the initial marking) a regenerative configuration is reached with probability 1. Bounded MRSPNs have the same modeling power as MRPs [48].
- *Deterministic and stochastic PN* (DSPN)[115] is an SPN where each timed transition e has exponential or Dirac CDF (a Dirac-distributed

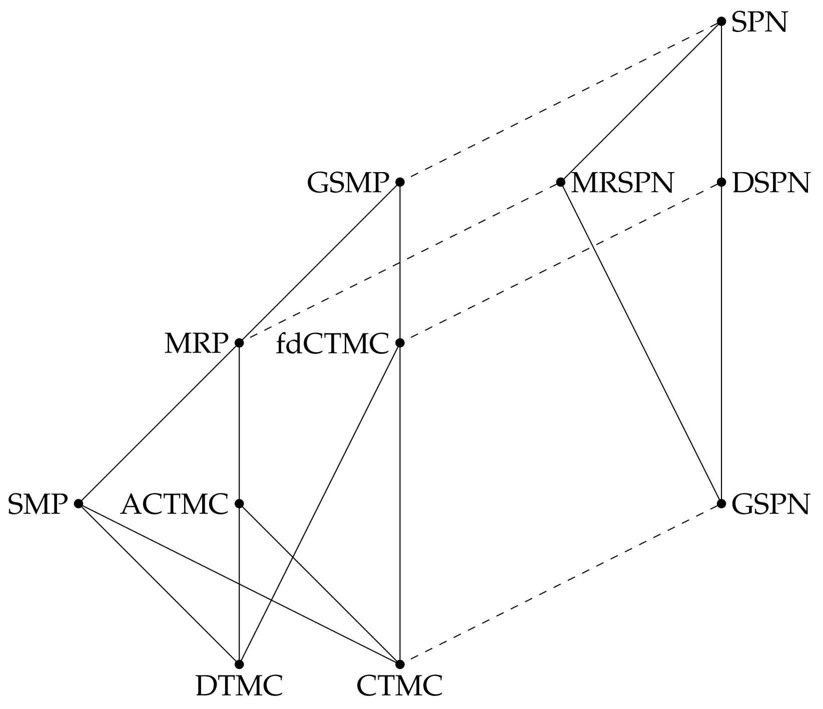


Figure 3.3: Hierarchy of formalisms according to their expressive power.

transition is called *deterministic* in the DSPN terminology). Bounded DSPNs have the same modeling power as fdCTMCs [100].

- *Generalized stochastic PN* (GSPN)[116] is an SPN where each timed transition e has an exponentially distributed CDF $F(\cdot, e)$. Bounded GSPNs have the same modeling power as CTMCs [48].

Reward Structure. Similarly to GSMP one can easily define a reward structure $(G, \mathcal{R}, \mathcal{I})$ for an SPN consisting of a set of goal markings G, a function $\mathcal{R}$ assigning nonnegative reward rate $\mathcal{R}(\mathbf{m}) \in \mathbb{Q}_{>0}$ to each marking $\mathbf{m}$, and a function $\mathcal{I}$ assigning to each subset of transitions E'' and all markings $\mathbf{m}$ and $\mathbf{m}'$ a nonnegative transition reward $\mathcal{I}(\mathbf{m}, E'', \mathbf{m}') \in \mathbb{Q}_{\geq 0}$.

The formalisms specified until now are tightly related to ACTMCs. We illustrate their hierarchy according to their modeling power in Figure 3.3. The order depicted by a dashed line means that the connected classes are equal for bounded nets.

3.1.3 Other Continuous-Time Stochastic Formalisms

There are various other formalisms that include continuous-time stochastic distributions. They are less related to our synthesis problems than the previously listed formalisms, thus we just briefly introduce them.

Stochastic Timed Automata. There are various stochastic extensions of the well-studied *timed automaton* formalism [3] by continuous-time distributions [5, 6, 20, 105]. To the best of our knowledge the most significant among such formalisms is a *stochastic timed automaton* (STA) [20, 29]. An STA behaves as a timed automaton, except the fact that the edges and occupation times are not chosen non-deterministically, but stochastically. When a new location is entered the occupation time t is randomly chosen according to a probability distribution (e.g., uniform or exponential distribution) and then the edge is chosen according to a discrete probability distribution on enabled edges after time t (i.e., all clocks are increased by t time units and edges with satisfied guards are collected). The STA formalism has higher expressive power than GSMPs [29] and hence the verification approaches restrict to subclasses of stochastic timed automata (STAa), e.g., one clock STAa. To the best of our knowledge the verification algorithms are not much related to our synthesis approaches, hence we do not list them.

Timed Automaton Observer. Another approach, how to apply the timed automata is to use them as observers of other stochastic models such as CTMCs [47, 61] or GSMPs with no Dirac-distributed events [6, 101]. The stochastic model actually generates timed words, i.e., alternating sequences of state labels and times spent in them. These words can be easily read by a deterministic timed automaton (DTA) that behaves synchronously with the stochastic model, i.e., it observes the state changes of the stochastic model, and the new state of the DTA is uniquely determined by the amount of time spent in the old state and by the observed state label. Using DTA observers it is possible to tackle behaviors what were not expressible in the original stochastic models, e.g., it is possible to express hard real-time bounds (such as timeouts) in a CTMC observed by a DTA. The class of CTMCs observed by DTAa is closely related to the class of fdCTMCs [32]. To the best of our knowledge all problems expressed using a CTMC observed by a DTA can be reduced to problems using a fdCTMC.

Queuing Theory. The *queuing theory* (QT) [69] was founded during the development of telecommunication networks and since then it was successfully applied in many other engineering contexts including traffic and industrial engineering. The basic building block of the formalism is one or more queues that are equipped with one or more servers and with an incoming flow of requests. The requests are stored in the queues and are served by the servers. The time between the request arrivals and the service time of a server can have various distributions. Moreover, the queues (with servers) may be organized in a network, where the served requests may be routed to an incoming request flow of other queues with certain probability. The problems and solution techniques in QT and for GSMPs are very related. In fact, the usual way how to compute or decide problems in QT is to use solution techniques for CTMC or GSMP problems.

Stochastic Process Algebras. A *process algebra* (PA) is a modeling formalism for specification of concurrent systems [51]. The key feature of PAs is compositionality that simplifies modeling by allowing specification of complex systems by first specifying simple processes of well separated functionalities, then composing the processes into more complicated ones using operators (such as choice or parallel composition) and synchronization on process actions. There are many stochastic extensions of PAs that incorporate continuous-time distributions on action delays [51, 93]. The most used stochastic algebra is the *performance evaluation PA* (PEPA) [51], that allows exponential distributions of action delays. Each PEPA process can be translated to an equivalent CTMC, hence the efficient algorithms for verification of CTMCs can be used for PEPA models. There are also other PAs that generate processes that correspond CTMCs or related models, see reviews in [51, 93]. Moreover, there are extensions of PAs that allow for generally-distributed action delays [25, 30, 31, 56, 68]. These extensions generate processes that correspond to GSMPs or their generalizations, e.g., allow non-determinism. Hence, the reviewed process algebras can be considered as high-level specification formalisms for CTMCs, GSMPs, and their extensions [93]. Recently, a parallel composition of STAa was defined [29].

3.2 Problems and Performance Measures

In this section we will define problems and properties usually checked on continuous-time stochastic systems. Let us fix such system $\mathcal{L}$ with state space S (for Petri nets the state space is the set of all reachable markings) and correctly defined probability measure $\mathrm{Pr}_{\mathcal{L}}$ on sets of runs of $\mathcal{L}$.

Let H_s be the random variable which for a state $s \in S$ and a run σ of $\mathcal{L}$ returns the first visiting time of s. A *time-bounded reachability* for given state $s \in S$ and time $t \in \mathbb{R}_{\geq 0}$, denoted by $tbr(s,t)$, is the probability that a run reaches the state s in time t, i.e., $tbr(s,t) \overset{\text{def}}{=} \mathrm{Pr}_{\mathcal{L}}(H_s \leq t)$. A *time-bounded reachability problem* (TBRP) is deciding whether for a state $s \in S$, a time $t \in \mathbb{Q}_{\geq 0}$ and a probability $p \in [0,1] \cap \mathbb{Q}$ it holds that $tbr(s,t) \leq p$.

A *transient probability* for a time $t \in \mathbb{R}_{\geq 0}$ is a discrete distribution $\pi(t)$ on S, which assigns to a state $s \in S$ a probability that $\mathcal{L}$ is in the state s in time t, i.e., $\pi(t)(s) \overset{\text{def}}{=} \mathrm{Pr}_{\mathcal{L}}(\text{runs that are in state } s \text{ at time } t)$. A *transient-probability problem* (TPP) is deciding whether for given state $s \in S$, time $t \in \mathbb{Q}_{\geq 0}$, and probability $p \in [0,1] \cap \mathbb{Q}$ it holds that $\pi(t)(s) \leq p$. Please observe that it is possible to reduce TBRP to TPP easily by making the state s absorbing (we redefine all active events in s such that the probability of moving from s to s is one), then $tbr(s,t) = \pi(t)(s)$. In literature the computation of transient probability is often referred to as *transient analysis*.

An *unbounded reachability* of a state $s \in S$, denoted as $ur(s)$, is a probability that s is reached by a run of $\mathcal{L}$ at some point of time, i.e., $ur(s) \overset{\text{def}}{=} \lim_{t \to \infty} tbr(s,t)$. Please observe that the limit always exists because $tbr(s,t)$ is non-decreasing and bounded by 1. An *unbounded-reachability problem* (URP) is deciding for $s \in S$ and probability $p \in [0,1] \cap \mathbb{Q}$, whether $ur(s) \leq p$. Note that TPP reduces to URP for formalisms where it is possible to add one Dirac-distributed event, i.e., for fdCTMC, GSMP, DSPN, and SPN of tightly related formalisms. We show that we can change the model $\mathcal{L}$ such that $\pi(t)(s)$ from the original model equals $ur(s')$ for some new state s' in the new model. First, we make a copy s' of each state s and in $\mathcal{L}$. Then, we define a new Dirac-distributed event e such that it is active everywhere, its delay is set to t, and the successor of s when e occurs is the copy of s with probability 1. Finally, we make the newly added states absorbing, i.e., for each new state s' and each subset of active events in s' there is probability 1 that the successor state is s'. It is easy to see that $\pi(t)(s)$ from the original model equals to $ur(s')$ in the new model.

We extend the definition of random variable TR from Section 2.2 to (GSMP or SPN) $\mathcal{L}$ with reward structure $(G, \mathcal{R}, \mathcal{I})$. The random variable TR assigns to each run $\omega = (s_0, \mathbf{o}_0)(s_1, \mathbf{o}_1)\ldots$ the *total accumulated reward before reaching G* (in at least one transition), i.e., we set $\mathrm{TR}(\omega) \overset{\text{def}}{=}$

$\sum_{i=0}^{n-1}(\mathbf{o}_i(e_i)\cdot\mathcal{R}(s_i)+\mathcal{I}(s_i,\mathcal{E}_i,s_{i+1}))$, where $\mathcal{E}_i \overset{\text{def}}{=} \operatorname{argmin}_e \mathbf{o}_i(e)$, $e_i \in \mathcal{E}_i$, and $n > 0$ is the minimal index such that $s_n \in G$. We set TR to infinity whenever there is no such n. By $\mathbb{E}_{\mathcal{L}}[\mathrm{TR}]$ we denote the expected value of TR according to the probability measure $\Pr_{\mathcal{L}}$.

A *total-reward problem* (TRP) is deciding whether for a reward structure $(G,\mathcal{R},\mathcal{I})$ and a reward threshold $r \in \mathbb{Q}_{\geq 0}$ it holds that $\mathbb{E}_{\mathcal{L}}[\mathrm{TR}] \leq r$. We can easily reduce URP to TRP. Assume we want to decide $ur(s) \leq p$. We introduce a reward structure $(G,\mathcal{R},\mathcal{I})$ where $\mathcal{R}$ is zero for each state, $\mathcal{I}$ assigns 1 for transitions leading to s and 0 for the other transitions, and $G \overset{\text{def}}{=} \{s\}$. It is easy to see, that $ur(s) = \mathbb{E}_{\mathcal{L}}[\mathrm{TR}]$.

A *limit of transient probabilities*, denoted $ltp \in \mathcal{D}(S)$, is the distribution over S that is a limit of transient probabilities, i.e., $ltp(s) \overset{\text{def}}{=} \lim_{t\to\infty}\pi(t)(s)$ if it exists. A *limit of transient probabilities problem* (LTPP) is deciding whether for a given state $s \in S$ and a probability $p \in [0,1]\cap\mathbb{Q}$ it holds that $ltp(s)$ exists and $ltp(s) \leq p$. The limit always exists for finite GSPNs and CTMCs [102], however, it does not have to exist for other discussed formalisms (see,. e.g., [114]). Instead a time frequency (see below) is studied [1, 101, 114]. The URP reduces to LTPP using similar construction as in the reduction of TBRP to TPP.

For state $s \in S$ and run $\omega = (s_0,\mathbf{o}_0)(s_1,\mathbf{o}_1)\ldots$ we define the random variable *time frequency* [36] as

$$\mathrm{TF}_s(\omega) \overset{\text{def}}{=} \lim_{n\to\infty}\frac{\sum_{i=0}^{n}\mathbf{1}_s(s_i)\cdot t_i}{\sum_{i=0}^{n} t_i},$$

where $t_i \overset{\text{def}}{=} \min_e \mathbf{o}_i(e)$. The variable TF_s returns the ratio of time spent in s. In [36] authors showed that the limit does not have to exist almost surely for fdCTMC, GSMP, DSPN, and SPN and they formulated sufficient conditions such that TF_s is *well defined*, i.e., that the limit exists for almost every run.

A *time-frequency problem* (TFP) for a given state $s \in S$ and $r,p \in [0,1]\cap\mathbb{Q}$ is deciding whether $\mathrm{TF}_s(\omega)$ exists for almost every run ω and $\Pr_{\mathcal{L}}(\mathrm{TF}_s \leq r) \leq p$. Again the URP can be reduced to TFP employing similar construction as in the reduction of TBRP to TPP. In literature the TFP is often referred to as *stationary analysis*. A *time-frequency distribution* is a function $\psi \in \mathcal{D}(S)$ that for each $s \in S$ satisfies $\psi(s) = \mathbb{E}_{\mathcal{L}}(\mathrm{TF}_s)$. The time-frequency distribution is in literature also called stationary, steady-state, or long-run-average distribution.

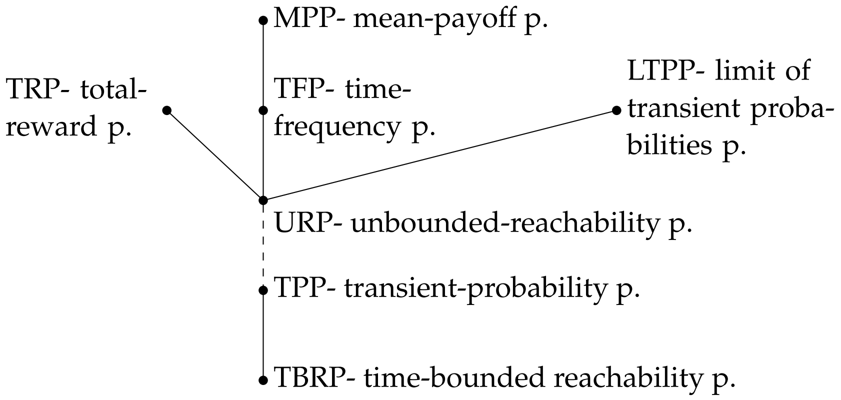

MPP- mean-payoff p.
TRP- total-reward p.
TFP- time-frequency p.
LTPP- limit of transient probabilities p.
URP- unbounded-reachability p.
TPP- transient-probability p.
TBRP- time-bounded reachability p.

Figure 3.4: Hierarchy of problems.

We extend the definition of the random variable MP from Section 2.2 to (GSMP or SPN) $\mathcal{L}$ with reward structure $(G, \mathcal{R}, \mathcal{I})$. The random variable MP assigns to each run $\omega = (s_0, \mathbf{o}_0)t_0(s_1, \mathbf{o}_1)t_1 \ldots$ the *mean payoff*, given by

$$\mathrm{MP}(\omega) \overset{\text{def}}{=} \lim_{n\to\infty} \frac{\sum_{i=0}^{n} (\mathbf{o}_i(e_i) \cdot \mathcal{R}(s_i) + \mathcal{I}(s_i, \mathcal{E}_i, s_{i+1}))}{\sum_{i=0}^{n} \mathbf{o}_i(e_i)},$$

where $\mathcal{E}_i \overset{\text{def}}{=} \mathrm{argmin}_e\, \mathbf{o}_i(e)$ and $e_i \in \mathcal{E}$. By $\mathbb{E}_{\mathcal{L}}[\mathrm{MP}]$ we denote the expected value of MP according to the probability measure $\mathrm{Pr}_{\mathcal{L}}$.

A *mean-payoff problem* (MPP) for a given reward structure $(G, \mathcal{R}, \mathcal{I})$ and a reward threshold $r \in \mathbb{Q}$ is deciding whether $\mathrm{MP}(\omega)$ exists for almost every run ω and $\mathbb{E}_{\mathcal{L}}[\mathrm{MP}] \leq r$. A TFP for a state s and $r, p \in [0,1] \cap \mathbb{Q}$ can be easily reduced to an MPP by introducing the reward structure $(\emptyset, \mathcal{R}, \mathcal{I})$, where $\mathcal{R}$ assigns 1 to s and 0 to the other states and $\mathcal{I}$ assigns always zero. It is easy to see that $\mathrm{Pr}_{\mathcal{L}}(\mathrm{TF}_s \leq r) = \mathbb{E}_{\mathcal{L}}[\mathrm{MP}]$. Conversely, a MPP can be reduced to a TFP for DTMC, CTMC, and SMP. These models are in a regenerative configuration after each transition, thus we can compute the frequency of each transition from the time frequency of corresponding output state and use this information to compute $\mathbb{E}_{\mathcal{L}}[\mathrm{MP}]$. Since TFP can be reduced to MPP, the results in [36] about the existence of TF apply also to MP.

We illustrate the problem hierarchy in Figure 3.4. The order is done according to the problem difficulty (a lower problem reduces to an upper one). The dashed line denotes that the order is only valid for formalisms where we can add one Dirac-distributed event.

One can define synthesis problems from all the above problems in the same manner as the expected mean-payoff or total-reward synthesis problems in Section 2.3. It is also possible to synthesize other parameters than the distribution parameters, e.g., synthesis of ε-optimal time $t \in \mathbb{Q}_{\geq 0}$ in order to maximize the transient probability $\pi(t)(s)$ of some state s.

There are extensions of well-known logics that include probabilistic operators for model checking of stochastic systems. This is natural, since the usual model-checking approaches may identify errors that happen with probability zero in a stochastic system and hence they are spurious. The new operators are usually derived from the above-defined properties. For instance, a *probabilistic computation tree logic* (PCTL) or PCTL* is an extension of CTL or CTL*, respectively, by the operator for measuring probability of runs satisfying a logical formula and by the time-bounded until operator that resembles the time-bounded reachability [14, 77]. Similarly, the *continuous stochastic logic* (CSL) is an extension of CTL for continuous-time models, that allows an operator for reasoning about time frequencies and time-bounded (within some interval) next and until [13, 17, 114]. Hence, the methods for performing model checking for one of the probabilistic logics depend on efficient solution methods for the above mentioned problems. Our synthesis approaches do not include logic objectives. Thus, we concentrate on solution techniques for the above-specified problems in Section 3.3.

3.3 Solution Methods

We first concentrate on clarification of decidability of the specified problems on the considered formalisms. Then we summarize the solution methods for the problems and for computation of the specified properties. They can be divided in *exact* and *approximation* methods depending on whether there are hard bounds on the error produced by the method. Then we discuss the related work concerning parameter synthesis in the stochastic formalisms, using knowledge of all the previous sections. Finally, we review solution techniques for computing optimal memoryless policies in MDPs and POMDPs. Sections 3.3.2 and 3.3.3 were inspired by [101], where there is a detailed report of relevant solution techniques.

3.3.1 Decidability

All specified problems are decidable for CTMCs and DTMCs, what follows from decidability of continuous stochastic logic (CSL) [13]. The URP and harder problems (see Figure 3.4) are undecidable on unbounded DSPNs and SPNs what can be shown by direct application of undecidability proof for reachability on timed PNs [129]. To our best knowledge it is not known whether TBRP (and thus also all other problems) is decidable for fdCTMCs and more expressive formalisms, see Figure 3.3. Decidability does not make sense for the whole classes of SMPs, ACTMCs, MRPs, GSMPs, and SPNs since the involved distribution functions might not be finitely representable in computers. However, there are many restricted classes, where decidability holds, e.g., for MRSPNs with expolynomial distributions all specified problems are decidable [79, 117].

To prove that the specified problems are decidable for some of our formalisms has not attracted much attention because with high probability the developed decidability algorithm would be too complicated for practical use. On the other hand, a lot of work was devoted to produce effective techniques, which would perform well at least to approximate the given properties on subclasses of stochastic models that have good properties. For example TBRP is decidable for CTMCs [13] but actually only an approximation algorithm based on uniformization [85] is used in practice to compute close enough value of $tbr(s, t)$. The whole CSL logic is efficiently approximately verifiable in practice on CTMCs [17, 94, 103].

3.3.2 Exact Methods

We provide a list of methods that give correct approximation of result with respect to some a priori given error bound. There are many restrictions introduced on the models to guarantee that the algorithms work correctly or effectively.

Basic Solution Methods

Employing the Markov property one can solve all the problems for DTMCs, CTMCs, and SMPs except for the transient-probability problem (TPP) and the time-bounded reachability problem (TBRP) by solving a set of linear

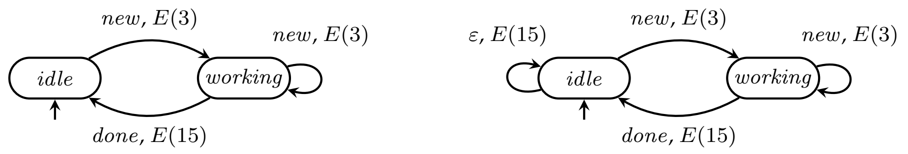


Figure 3.5: En example of uniformization.

equations [125]. The TPP and TBRP for DTMCs can be solved simply by a vector-matrix multiplication [125].

Uniformization

Uniformization [85] is also known as Jensen's method or randomization method. This method can be used in models where exponential distributions are allowed. It is mostly used on CTMCs, but with some modifications it can be used also on other formalisms, e.g., ACTMCs. Let $\mathcal{C} = (S, Q, s_{\text{in}})$ be a CTMC. The main idea of uniformization is an introduction of self-loops such that the expected time spent in each state is the same and other performance characteristics remain unchanged (e.g., the transition probabilities of reaching some state in particular time). Thus one can view the new CTMC as a DTMC that spends in each state the same (expected) time and on can use simple solution methods for DTMCs to compute properties for the CTMC. Recall, that the transition probabilities of moving from a state s to s' are $Q(s,s')/\sum_{s''} Q(s,s'')$ and the expected time spent in a state s is $1/\sum_{s''} Q(s,s'')$. By making the rate $Q(s,s)$ larger, one visit of state s is in average shorter, but there is larger probability that it is immediately visited again. In fact the rate of self loop has no effect on probability of reaching other states from s and the total (expected) time spent in s, what is illustrated by the next example.

Example 3.3.1. *Figure 3.5 shows the original CTMC (left) and the uniformized CTMC (right). Observe that the transition new from state $idle$ to state $working$ performs in average 3 times per second (the rate of transition new is 3) in both CTMCs. In average a transition occurs 18 times per second from state $idle$ in the right model, but in average every sixth transition is new.*

In similar fashion, one can also recompute the reward structure to keep the same value of the relevant performance measures. All these observations are direct consequences of the memoryless property.

The same exit rate from every state can simplify computations of many properties. We illustrate the use of uniformization in the approximation of the value of $\pi(t)(s)$. We can compute it as an infinite sum

$$\pi(t)(s) = \sum_{n=1}^{\infty} \Pr(Poiss(t \cdot \lambda) = n) \cdot \Pr(Y_n = s), \tag{3.1}$$

where $P(Poiss(t \cdot \lambda) = n)$ is the probability that exactly n exponential events with rate λ fire in sequence until time t (i.e., the probability that Poisson distributed random variable with mean $t \cdot \lambda$ equals n) and $\Pr(Y_n = s)$ is probability that s is reached in exactly n steps (note that this is efficiently computable by multiplying the initial distribution by the transition kernel of the embedded discrete time Markov chain [125]). Since $\Pr(Poiss(t \cdot \lambda) = n)$ decreases exponentially and $\Pr(Y_n = s)$ is bounded, for each allowed error $\varepsilon > 0$ one can easily find a sufficiently high N where the above infinite sum can be truncated to generate at most ε of error in $\pi(t)(s)$. Uniformization is a very efficient approach to approximate $\pi(t)(s)$ (hence also $tbr(s, t)$) [17, 94, 103].

Subordinated Markov Chain

The *subordinated Markov chain* (SMC) method applies uniformization to approximate $\Pr_{\mathcal{S}}(\mathrm{TF}_s \leq r)$ [8, 109] and $\pi(t)$ [74] in a bounded DSPN $\mathcal{S}$ with at most one Dirac-distributed (i.e, deterministic) transition enabled in each reachable marking. This class of DSPNs is equivalent to the class of fd-CTMCs with at most one concurrent non-exponential event, or to the class of ACTMCs with only Dirac-distributed alarms. We illustrate it on computation of $\Pr_{\mathcal{A}}(\mathrm{TF}_s \leq r)$ for a ACTMC $\mathcal{A}$ with Dirac-distributed alarms. First, a DTMC is built, such that its states correspond to states S' where either (a) no Dirac-distributed events are enabled or (b) a Dirac-distributed event is newly enabled. The DTMC transition probabilities for state $s \in S'$ are computed as follows:

- if s is of type (a), the probabilities are computed as ratio of rates of corresponding exponentially-distributed events from s and the exit rate of s,
- if s is of type (b), then there is a Dirac-distributed alarm a that became active in s. Recall that S_a denotes the set of states where a is active and let d be the delay of the Dirac distribution of e. The ACTMC $\mathcal{A}$

can reach S' either by staying in S_a (while performing exp-delay transitions) for time d and executing the alarm a, or by performing a sequence of exp-delay transitions that reach a state in $S' \setminus \{s\}$ before time d. Thus, $\mathcal{A}$ behaves as a CTMC until time d or until leaving S_a and then, if it did not leave S_a until time d, it performs an a-alarm transition from some $s' \in S_a$. Hence, we consider $\mathcal{A}$ as a CTMC, set the initial state to s, make all states in $S \setminus \{s\}$ absorbing, and compute $\pi(d)$ (i.e., the transient probability distribution at time d) using uniformization. Now the value $\pi(d)(s')$ for a state $s' \in S' \setminus S_a$ is the probability that s' was reached by exp-delay transitions. The probability $\pi(d)(s')$ for state $s' \in S_a$ is the probability that a-alarm transition occurred in s' and hence it is distributed to states of S' using P_a, i.e., the probability matrix of the alarm a. Note that for the remaining states $\pi(d)$ equals to zero and that $P_a(s', s') = 1$ for $s' \notin S_a$. Thus the transition probability distribution of the DTMC for state s equals to $\pi(d) \cdot P_a$. The CTMCs used for computation of $\pi(d)$ are in literature called *subordinated Markov chains*.

The created DTMC is denoted as *embedded* Markov chain (EMC) in literature. Using the EMC we can apply standard techniques to obtain the discrete-time frequencies of visits of every state which are used to compute the time frequencies employing the expected occupation times for states corresponding to S'. The SMC method can be applied to approximate $\mathbb{E}_{\mathcal{A}}[\mathrm{MP}]$ and $\mathbb{E}_{\mathcal{A}}[\mathrm{TR}]$ for an ACTMC $\mathcal{A}$ [48] where alarm densities satisfy mild assumptions. That will be heavily used in Chapter 6. The SMC method is closely related to a quantitative reachability in a CTMC observed by an one-clock time automaton [47].

Symbolic Integration

The subordinated Markov chain method works well because of two main ideas: the fact that it deals with regenerative models and the efficient computation of transition probabilities of EMC. Observe that the states of the EMC in SMC method are exactly those that occur in regenerative configurations, i.e., we do not need to remember any times to occur for non-exponential events. In those configurations the process looses memory what allows us to represent it as an EMC. This approach can be applied to larger classes of SMPs and MRPs if we still can efficiently compute the necessary quantities for the EMC: the transition probabilities, expected times and accumulated rewards between regenerations [48].

The obvious extension is application of an EMC to computation of $\pi(t)$ or $\mathbb{E}_{\mathcal{G}}[\mathrm{TF}_s]$ in an SMP $\mathcal{G}$ [48, 50] what was implemented in [148]. Here all times to occur are reset after each transition. Thus to compute the transition probabilities of an EMC, one just needs to evaluate the probability that some event occurs earlier than others. Usually this leads to a computation of integrals using distribution functions, what is feasible for commonly used distribution functions.

The method of *stochastic state classes* (SSCs) can be used to approximate $\pi(t)$ [4, 79, 80] and $\mathbb{E}_{\mathcal{S}}[\mathrm{TF}_s]$ [117] in MRSPN $\mathcal{S}$ with truncated expolynomial distributions. The method is based on the fact that for expolynomial distributions it is possible to compute integrals in a symbolical way [50] what is much faster than computing them numerically [48, 114]. The density of an truncated expolynomial distribution $f(x)$ is 0 if x is outside a given domain $[\ell, u]$, otherwise $f(x)$ is of the form $\sum_{j=1}^{m} c_j x^{n_j} e^{-\lambda_j x}$, where $c_j \in \mathbb{R}$, $m, n_j \in \mathbb{N}$, and $\lambda_j \in \mathbb{R}_{\geq 0}$. The method extends the idea of zones (denoted as SSCs) which are known from timed automata [60, 134]. The SSCs keep information about densities over event occurrence times. To compute the quantities for a state s of the an EMC one starts in a SSC corresponding to s and enumerates all SSCs that are reachable from s by occurrence of an transition enabled in s. This is repeated until some regenerative configuration is reached. The probabilities of moving from one SSC to the other and SSCs themselves can be computed efficiently thanks to the symbolical integration. One may compute all necessary quantities for the EMC from the SSCs, since they contain all the needed information in the densities. It may happen that there may be an infinite sequence of transitions that would never lead to regenerative configuration. However, such runs have zero measure in a MRSPN and the enumeration of SSCs can be truncated by an introduction of some approximation error. Algorithms for computing $\pi(t)$ and $\mathbb{E}_{\mathcal{S}}[\mathrm{TF}_s]$ were implemented in tool Oris [38].

Discretization

The method is usually used for approximation of $\pi(t)$ for non-Markovian SPNs [149] and for game extensions of discrete event systems such as continuous-time Markov decision processes. First of all the discretization step needs to be set. Then the changes of state occur in multiples of discretization step. Thus the process has only finitely many configurations and thus is easily analyzable using methods for DTMCs. Under the assumption that at most one transition occurs in one step an error bound was derived in [123].

3.3.3 Approximation Methods

This subsection refers to the methods for which the exact error is usually not bounded by any formula. Often the usability of these methods is given by empirical testing or comparison to other methods. In some cases the authors provide statistical guarantees for the generated error.

Phase-Type Approximation

There are two main types of *phase type distributions*: discrete and continuous. A *discrete or continuous phase-type distribution* is distribution which is representable by distribution of time to absorption of an absorbing DTMC or CTMC, respectively. The number of states of the DTMC or CTMC is the number of *phases*. *Phase type approximation* (PH) is a process of finding for a given distribution a phase-type distribution, which will give the smallest error according to some measure. There are various measures such as minimum difference in probability density function, CDF, relative entropy, etc. There are also various phase-type approximating algorithms [11, 23, 127, 137] and these algorithms are implemented in many tools such as Ph-Fit [81], EMPht [11], G-FIT [137]. The key properties, why the phase-type approximations were extensively studied are the following:

- any distribution can be approximated arbitrarily closely by a phase type distributions,
- when a distribution is fitted by a phase-type gadget (i.e., by a CTMC or DTMC), one can take the gadget and paste it to the model with slight modifications, thus one will get a CTMC or DTMC approximation of the whole model,
- analysis of CTMCs and DTMCs is much simpler than analysis of models with more sophisticated distributions.

Thus the phase-type approximation can be used to help to approximate any measure defined in Subsection 3.2. However, there are also drawbacks of the phase-type approximation:

- the size of the model is exponential to the number of concurrent non-exponential events,
- some distributions are extremely difficult to fit (e.g., the continuous distributions with zero density on some interval when using contin-

uous PH approximation) and to get sufficiently good approximation one needs to use exponential number of phases.

Thus the phase-type approximation is efficient when it is used on models where the distributions are easy to fit and the parallelism is small [74]. Observe that the phase-type approximation may be incorrect. Assume that we are approximating a GSMP model for which $\mathbb{E}_{\mathcal{G}}[\mathrm{TF}_s]$ is not well defined by continuous phase-type approximation. We will obtain a CTMC, but for every CTMC the $\mathbb{E}_{\mathcal{C}}[\mathrm{TF}_s]$ is well defined.

There are proposed hybrid approaches of phase-type approximation, where the created model is similar to a DSPN or an fdCTMC [89, 98]. The discretization method is similar to the discrete phase-type approximation. The phase type approximation may produce smaller models because it can use cycles. Discretization usually creates gadgets in the form of chain.

Supplementary Variables

This method was first introduced in [52, 66, 110] for approximation of $ltp(s)$ for a DSPN. For each enabled transition in a reachable marking a supplementary variable that keeps its occurrence time is introduced. Then the set of partial differential equations is defined according to the rules of the DSPN. After that it is needed to solve the system of equations. Thus, the accuracy and performance is limited by the differential equation solver and the number of equations. The equations can be solved by discretization. Later the method was extended to approximation of $\pi(t)$ [112] and implemented in the tool DSPNExpress 2.0 [111] what is an ancestor of the TimeNET tool [148].

Approximation by a DSPN with Single Deterministic Transition

This method [73] is for approximating $\pi(t)(s)$ and $ltp(s)$ in a DSPN. First a time τ is chosen to be sufficiently small with respect to the delays of Dirac-distributed (i.e., deterministic) transitions. Each Dirac-distributed transition e with delay d fires after $\lfloor \frac{d}{\tau} \rfloor$ steps after being enabled. There is one central clock which ticks and all the Dirac-distributed transitions fire at an instant of a tick. However, they can become enabled any-time. Thus, this method generates some error in every firing of a Dirac-distributed transition. The correctness of the approximation is not proven. The method was reinvented in the biology setting and the Sabre tool [59] was extended to support this method [70].

Simulation

This method is also called the *Monte Carlo simulation* or the *statistical model checking* in the verification community. It is used for approximation of $\pi(t)(s)$, $ur(s)$, and $\mathrm{Pr}_{\mathcal{G}}(\mathrm{TF}_s \leq r)$ in a GSMP. The approximation is done by experimental executions of given models, where time of occurrence of events is chosen randomly according to the distributions of events. The final result is done by averaging the results of individual executions. There is a vast amount of literature coping with error bounds and confidence intervals of the simulation results [67, 71, 87, 88, 147]. There are also various tools which support simulation on SPNs such as Oris [38], TimeNET [148], and model checking tools for DTMCs, CTMCs, MDPs, and GSMPs [94, 103, 144]. Currently this method attracts a lot of attention in the area of verification [107, 120, 128, 145, 146] and in the computational biology [78, 87].

3.3.4 Related Work to Synthesis

We review the related work concerning the our ε-optimal synthesis problems (the TRSP and MPSP). First we briefly explain results that are loosely connected to our problems, then we explain strongly related works in detail.

Synthesis of optimal timeouts guaranteeing quantitative properties in timed systems was considered in [58]. Using an approach based on stochastic games, a parameter synthesis algorithm for timing parameters in probabilistic timed automata optimizing reachability *probabilities* has been presented in [91]. There are various parametric formalisms for timed systems that deal with some sort of parameter synthesis, such as parametric timed automata [7, 9, 82, 90, 119], parametric one-counter automata [72], parametric timed Petri nets [139], parametric DTMCs [57, 75, 84], parametric MDPs [46, 53], parametric Kripke structures [37], or timed branching processes [140]. However, all works referenced above do not consider models with continuous-time distributions, thus they synthesize different parameters than we do.

The problem of synthesizing *continuous* parameters has been studied in variety of engineering contexts such as in vehicle communication systems [97] and in avionic subsystems [12, 138]. These works do not provide a generic framework for synthesis of distribution parameters in continuous-time stochastic systems. Nevertheless, simpler cases have already been addressed. For instance [42, 141] consider a finite test case, a sequence of input and output actions, and synthesize times for input actions that maximize the probability of executing this acyclic sequence. Allowing cycles in ACTMCs makes the parameter synthesis problem much more demanding, e.g., due to potentially unbounded number of stochastic events between alarms. In [86, 106] authors generalize probabilistic timed automata with clock resets made according to probability distributions, that include Dirac distributions. In each state there is a finite set of allowed "actions" that can be used to simulate different lengths of delays. A scheduler that picks from the allowed actions the optimal ones to maximize probability of a sequence of state transitions can be computed using discretization of the region graph. In our approach we want to synthesize parameters from continuous domain what is significantly harder. Optimal control of continuous-time (semi)-Markov decision processes [33, 35, 39, 122] can be viewed as synthesis of *discrete* parameters in continuous-time systems. However, the schedulers are only allowed to choose actions from finite domains while parameter synthesis calls for continuous domains.

Alternatively to the previous works, in [142] authors compute the optimal value of a web server timeout using impulse and rate rewards. The implementation can dynamically update the optimal value of the timeout based on the current inter-arrival times of requests. The method works on the fixed fdCTMC model and cannot be directly applied to more general fdCTMC models.

The synthesis of rates in a CTMC to obtain globally optimal long-run measure similar to mean payoff is addressed in [2]. It is possible to synthesize multiple rates in each state. The synthesis problem is solved by a reduction to finding a memoryless randomized policy in an MDP that optimizes the long-run measure. The optimal values of the rates are then obtained from the probabilities of choosing actions given by the randomized policy. The correctness of the reduction heavily depends on the memoryless property of the exponential distribution. Thus the reduction does not work for more general distributions allowed in ACTMCs.

There is one contribution of [136] that is related to our synthesis algorithms. It is an approach that identifies subset of rates in a given CTMC that are scaled by a common reduction factor in order to satisfy an unbounded reachability property. To the best of our knowledge the used ideas cannot be utilized to solve our problems.

There is a vast amount of literature from biological and chemistry setting that is strongly related to the synthesis of rates in a CTMC. In [54] authors get a set of chemical traces for which they synthesize GSPNs with the highest probability to produce given traces. Their approach uses satisfiability modulo theories to get the structure of a GSPN and Monte Carlo based approaches to tune the rates in the GSPN to increase the required probability. In [88] authors deal with a CTMC where some rates are not given exactly (since they could not be precisely measured experimentally) but they are parameters of the model. The aim is either to find all parameter values that enforce satisfaction of some time bounded property with a given probability, or to find a concrete vector of parameters that maximizes the probability that the property holds. Authors use discretization, abstraction refinement, and gradient descent techniques. In [26] authors also deal with a CTMC where the rates depend on model parameters. Using statistical model checking the authors synthesize analytical function that for each vector of parameters returns the satisfaction probability of a given temporal logic property with time-bounded until operators. This can be used to reason about the whole space of parameters at once instead of performing analysis for each instantiation of parameters. In [28] authors present an algorithm that for a given parametric CTMC, n time bounded properties, and N observations of their joint satisfaction probabilities finds the values of parameters that make these observations most probable using learning techniques. Finally, in [18] authors provide algorithm that for a given parametric CTMC and a temporal logic property (with time-bounded until operators) identifies parameters that maximize the expected robustness (satisfaction score) of the property. The approaches of [18, 26, 28] were implemented in the tool U-Check [27]. All the woks referenced in this paragraph are based on statistical model checking methods, thus they do not provide hard guarantees but only statistical guaranties (i.e., that the obtained value is in some confidence interval with some probability). Since we aim on ε-optimal synthesis (with probability 1), we do not see how to incorporate these techniques in solution of our problems.

In [76] authors assume a parametric CTMC, where rates are polynomial expressions of parameters. They aim to synthesize set of parameters that enforce satisfaction of given time bounded reachability property. Their approach is based on discretization, analytical evaluation of the reachability probabilities and refinement techniques. Although, [76] uses same methods as we do in this thesis (namely discretization, computing common uniformization rate to correctly bound the error, root finding for polynomials, etc.), their results are not applicable in our setting due to the difference in objectives and possible cycles in ACTMCs.

In [44, 45] authors present a very efficient approach to synthesize rate parameters in a CTMC that optimize a given time bounded probability or reward formula up to an arbitrary small error. In our solution of TRSP and MPSP we will deal with an orthogonal subproblem: given a CTMC we need to find the time bound that optimizes with high precision some ranking function, that combines time bounded probability and reward. Our subproblem can be easily reduced to the problem solved in [44, 45] by introducing only one parameter, that will multiply all rates of the given CTMC. After obtaining the optimal parameter value p for some fixed time bound t, one can easily compute the optimal time bound from p and t. However, the method of [44, 45] would have to deal with more complex property to be optimized and according to Milan Češka (one of the authors of [44, 45]) this would significantly slow down the method. Thus we decided to use different approach based on isolating roots of univariate polynomials, see Chapter 5.

In [49, 62] authors present an extension of probabilistic model checker PRISM that enables an efficient execution of many computations in parallel thanks to the enhanced work with symbolic data structures of PRISM. After sufficient discretization of the parameter space, one can easily run one computation for each choice of parameters and find the optimal one. This would solve the TRSP and MPSP. However, the needed discretization is extremely dense and the amount of all choices of parameters is exponential in number of parameters, thus this approach is not feasible.

The last and the most related work to TRSP and MPSP is the TimeNET optimization environment (TOE) [24]. TOE is an extension of the TimeNET tool that uses the simulation-based analyses of SPNs in TimeNET. One of the uses of TOE is to specify some optimization measure (it supports also expected mean payoff and total accumulated reward) and search for model parameters that optimize the measure. For the searching TOE employs various heuristic algorithms, such as simulated annealing, hill climbing or genetic algorithms. Thus the obtained solution can have two sources of errors: the computed vector of parameters can be only locally optimal thanks to heuristic approaches and there are only statistical guarantees for accuracy thanks to the use of simulation-based techniques for evaluation of the optimization measure.

3.3.5 Solution Methods for (PO)MDPs

There was a lot of research devoted to solution techniques for finding optimal policies in MDPs and POMDPs, since there are many practically oriented problems that include them. E.g., navigation problem of a robot in a maze can be naturally modeled as a POMDP. We concentrate on finding optimal memoryless deterministic (MD) policies because we need only MD policies in our synthesis algorithms. In the following we first review solution methods for MDPs then for POMDPs.

MDP Solution Methods

There are three main solution methods for MDPs for optimization of both expected total reward and mean payoff: *linear programming*, *policy iteration*, and *value iteration*. Most facts described in this subsection are taken from [130]. All the methods return an optimal policy for an MDP $\mathcal{M}$ with finite action space provided that some assumptions are satisfied. For our purposes the sufficient assumption for minimization or maximization of $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ is strong-connectedness of $\mathcal{M}$ under each policy [130]. For minimization or maximization of $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ the sufficient condition in our setting is that under each policy there is probability one to reach a goal vertex from an arbitrary vertex [92, 130].

Linear Programming. The optimal policy for a finite MDP $\mathcal{M}$ (and both performance measures $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ or $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ and maximization or minimization) can be obtained by construction and solution of certain linear program. The optimal policy can be constructed in polynomial time in the size of $\mathcal{M}$ from the solution of the linear program. The linear program itself has size polynomial in the size of the MDP $\mathcal{M}$, thus the optimal solution can be computed in polynomial time.

Policy Iteration. Policy iteration (PI) is an example of a strategy improvement algorithm. It starts with an arbitrarily chosen policy. In each iteration it first computes necessary quantities for a current policy (usually by solving linear equations) and then it improves the current policy separately for each state using Bellman equations and the computed quantities. The policy iteration finishes when a fixed point is reached, i.e., when an improvement of a policy yields the same policy. The PI always terminates and returns an optimal policy [130] (to obtain termination of PI for the minimization of $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ one can use Theorem 7.3.2 and Proposition 7.3.4 of [130] and our assumption that there is probability one to reach a goal vertex from any vertex under arbitrary policy). Recently, it was shown that PI can make an exponential number (in number of states of the analyzed MDP) of iterations [64] what with combination with an exponential upper bound on the number of iterations gives an exponential algorithm (in each iteration, there is a polynomial number of instructions, each policy is strictly improved except for the last iteration and the number of memoryless deterministic policies is exponential in the number of states of the MDP) [130]. We will study PI algorithms in more detail in Chapter 5.

Value Iteration. Value iteration (VI) is also based on Bellman equations. It is a simple iterative algorithm that starts with some appropriately chosen solution vector. In each iteration it improves the current solution vector using the Bellman equations. VI terminates usually if the difference between two successively computed solution vectors (for each state) is smaller than some threshold. The threshold must be precisely computed to obtain the optimal policy from the last solution vector. Recently, new termination criteria have been developed in [10], where over- and under-approximations of the desired property are computed by VI in each step. Then the VI is stopped, when the difference in the over- and under-approximations is smaller than some precisely computed threshold (if just the ε-approximation of the desired property is needed and not the optimal policy, then one can terminate VI when the over- and under approximations are at most ε far from each other). VI can make an exponential number (in size of MDP) of iterations. Usually it makes more iterations than PI but each iteration is much simpler since there is no need to solve linear equations.

POMDP Solution Methods

There is a vast amount of literature for minimization or maximization of $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ or $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ in general POMDPs. The problem is hard to solve and the only efficient algorithms are based on heuristics. The optimal policies might be HR, i.e., they require both memory and randomization. However, we do not deal with the general problem, i.e., we aim to find the optimal memoryless deterministic (MD) policy. It was shown that finding an MD policy for a POMDP that minimizes the total number of steps to reach a given goal vertex is NP-complete [113]. Applying easy modifications to the proof of NP-hardness from [113] it can be used also for showing that finding MD policies that minimize or maximize $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ and $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ in a POMDP is NP-hard. In [113] authors also propose simple but efficient branch and bound algorithm that always finds an optimal MD policy for their problem. This algorithm can be easily adapted to finding an MD policy that optimizes $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ or $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ in POMDPs.

There are a few more algorithms to find an MD policy that optimizes $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ for POMDPs. The first algorithm is called *gradient-based policy iteration* (GBPI) and is an extension of PI to POMDPs [40, 41]. The GBPI algorithm may not return an optimal policy among the MD policies, it returns good policies for real world examples [41]. In [108] authors show that GBPI may not terminate (i.e., it can infinitely many times switch between two policies) thus they suggest a new algorithm, that resembles a gradient descent and always converges to some locally optimal MD policy. Similarly to GBPI there is an extension of value iteration algorithm for finding optimal MD policies in POMDPs [55]. The algorithm is called modified value iteration and it was shown that it can return better policies in POMDPs where the observation is randomly assigned depending on state [143]. However, both algorithms behave similarly for POMDPs that we use in this thesis [143], i.e., where the observation is obtained deterministically for given state. To the best of our knowledge, there are no other algorithms for finding optimal MD policies that minimize or maximize $\mathbb{E}_{\mathcal{M}}[\mathrm{TR}]$ or $\mathbb{E}_{\mathcal{M}}[\mathrm{MP}]$ for POMDPs.

Chapter 4

Explicit Parameter Synthesis

In this chapter we will present the first approach towards an efficient parameter synthesis in parametric ACTMCs. The approach is performed in two steps:

- We first use the subordinated Markov chain method explained in Section 3.3.2 to define certain POMDP for a given ACTMC with a reward structure. Each POMDP vertex has either one action or uncountably many actions – one for each eligible value of exactly one parameter. In fact, there is one to one correspondence between parameter functions in the ACTMC and policies in the POMDP and they induce the same associated expected mean payoff and total reward. Hence, we reduce each of the original synthesis problems into problem of finding ε-optimal policy in POMDP with uncountable number of actions. We prove also some properties of the obtained POMDP, that will be used later in the thesis.
- Then we provide an *explicit parameter synthesis algorithms* that are based on standard discretization of the created POMDP with uncountable action space to obtain finite POMDP such that at least one ε-optimal policy is kept in the finite POMDP. Then, for the finite POMDP one can apply standard algorithms (see Section 3.3.5) for finding expected mean-payoff or total-reward optimal policies that in turn give us ε-optimal policies for the uncountable POMDP and hence ε-optimal parameter functions in the original ACTMC. Special care must be taken for the expected total-reward parameter synthesis, since the algorithms might not work when the expected total reward may be infinite for some policies. We show that if this is the case, then the expected total reward is infinite for all policies. Thus we simply choose an arbitrary policy, compute the expected total reward. If it is infinite we return the chosen policy (since any policy is optimal), otherwise we perform some appropriate algorithm for finding an optimal policy in POMDP.

The rest of the chapter is structured as follows. We first reduce our parameter synthesis problems to problems of finding an optimal policy in POMDP and we discuss the properties of created POMDPs. Then, we provide the *explicit parameter synthesis algorithms* solving our problems and 2 assumptions that render the correctness of the algorithms. Finally, we provide a section with full proofs, that were just sketched in the previous sections.

In the the rest of the thesis, we fix a reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$ and a parametric ACTMC $\mathcal{N} = (S, Q, s_{\text{in}}, A, \langle S_a \rangle, \langle P_a \rangle, \langle F_a[x] \rangle, \langle \ell_a \rangle, \langle u_a \rangle)$ without unreachable states, rational Q, transition probabilities, and finitely representable $\langle F_a[x] \rangle$. When we deal with expected mean payoff synthesis problem we moreover assume $\mathcal{N}$ to be strongly connected.

4.1 From ACTMC to POMDP

Our approach to solve our synthesis problems (MPSP and TRSP) is based on their reduction to the problem of synthesizing expected mean-payoff and total-reward optimal policies in POMDPs. We will use the idea of the subordinated Markov chain method and regenerative analysis, i.e., we identify regenerative configurations and compute transition probabilities, expected rewards, and expected times between the regenerative configurations.

Let $a \in A$, and let $s \in S_a \cap S_{\text{set}}$. Then, for every $d \in [\ell_a, u_a]$, consider runs initiated in a configuration (s, η), where η is chosen randomly according to $F_a[d]$, i.e., assume we are working with runs of ACTMC $\mathcal{N}(s)^{\mathbf{d}}$ where $\mathbf{d}(a) = d$. Almost all such runs eventually visit a regenerative configuration (s', η') where either $s' \in S_{\text{off}}$ or η' is chosen randomly in $s' \in S_{\text{set}}$, i.e., either all alarms are disabled or one is newly set. We use $\Pi_s(d)$ to denote the associated probability distribution over $S_{\text{set}} \cup S_{\text{off}}$, i.e., $\Pi_s(d)(s')$ is the probability of visiting a regenerative configuration of the form (s', η') from s without previously visiting another regenerative configuration. Further, we use $€_s(d)$ and $\Theta_s(d)$ to denote the expected accumulated rewards and the expected time elapsed until visiting a regenerative configuration, respectively. We generalize the functions also on $s \in S_{\text{off}}$, where $\Pi_s(d) = Q(s, \cdot)/\lambda_s$, $€_s(d) = \mathcal{R}(s)/\lambda_s + Q(s, \cdot) \cdot \mathcal{I}_P/\lambda_s$, and $\Theta_s(d) = 1/\lambda_s$ for $d \in \mathbb{R}_{\geq 0}$. Note that here the functions are constant, i.e., they are independent of d. Moreover, observe that states $s \in S_{\text{off}}$ always appear in regenerative configuration and reach a regenerative configuration in one transition.

Now we define the POMDP for given ACTMC with a reward structure. The POMDP will be used in reduction of our synthesis problems to identify the ε-optimal parameter functions. Vertices of the POMDP will be precisely the states that occur in regenerative configurations. Actions of the POMDP will carry all the necessary effects of path sections between the regenerative configurations, i.e., the quantities given by functions Π_s, $€_s$, and Θ_s. Observe that each regenerative configuration can be uniquely encoded by its state, since each reachable state occurs in one or more configurations but only in one regenerative configuration.

Definition 4.1.1. *Let $\mathcal{M}_{\mathcal{N}}$ be a POMDP*

$$(S_{\mathrm{set}} \cup S_{\mathrm{off}}, A \cup S_{\mathrm{off}}, O, Act, T, t, c, s_{\mathrm{in}}, G)$$

such that:

- *The observation function O satisfies $O(s) = a$ for each state $s \in S_{\mathrm{set}} \cap S_a$ and $O(s) = s$ for each state $s \in S_{\mathrm{off}}$.*
- *The sets of actions are defined as*

  $$Act_a \overset{\mathit{def}}{=} \{\langle\!\langle a, d\rangle\!\rangle : d \in [\ell_a, u_a]\}$$

  for $a \in A$ and

  $$Act_s \overset{\mathit{def}}{=} \{\langle\!\langle s, 0\rangle\!\rangle\}$$

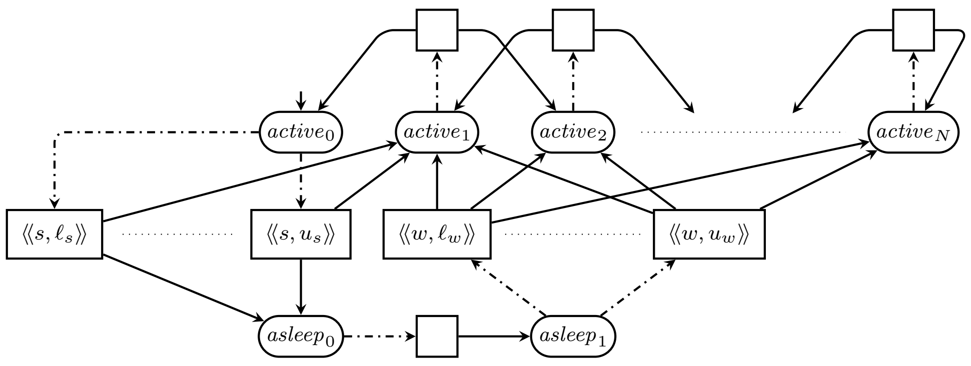
  for $s \in S_{\mathrm{off}}$.
- *The transition, reward, and time functions satisfy*

  $$\begin{aligned} T(s, \langle\!\langle a, d\rangle\!\rangle) &= \Pi_s(d) \\ c(s, \langle\!\langle a, d\rangle\!\rangle) &= €_s(d) \\ t(s, \langle\!\langle a, d\rangle\!\rangle) &= \Theta_s(d) \end{aligned}$$

  for all $a \in A$, $s \in S_{\mathrm{set}} \cap S_a$, and $\langle\!\langle a, d\rangle\!\rangle \in Act_a$ and

  $$\begin{aligned} T(s, \langle\!\langle s, 0\rangle\!\rangle) &= \Pi_s(0) \\ c(s, \langle\!\langle s, 0\rangle\!\rangle) &= €_s(0) \\ t(s, \langle\!\langle s, 0\rangle\!\rangle) &= \Theta_s(0) \end{aligned}$$

  for $s \in S_{\mathrm{off}}$.



$active_0$
$active_1$
$active_2$
$active_N$
$\langle\langle s, \ell_s \rangle\rangle$
$\langle\langle s, u_s \rangle\rangle$
$\langle\langle w, \ell_w \rangle\rangle$
$\langle\langle w, u_w \rangle\rangle$
$asleep_0$
$asleep_1$

Figure 4.1: The POMDP $\mathcal{M}_\mathcal{N}$ for parametric ACTMC $\mathcal{N}$ in Figure 1.1.

For an illustration of the previous definition we provide the POMDP $\mathcal{M}_\mathcal{N}$ for parametric ACTMC $\mathcal{N}$ of Figure 1.1 in Figure 4.1. The actions are denoted as rectangles. Each action without label should have a label $\langle\langle v, 0 \rangle\rangle$ where v should be substituted by the vertex label that the action is originating from.

Every MD policy σ of $\mathcal{M}_\mathcal{N}$ yields a unique parameter function $\mathbf{d}^\sigma$ for $\mathcal{N}$ defined by $\mathbf{d}^\sigma(a) = d$ where $a \in A$ and $\sigma(a) = \langle\langle a, d \rangle\rangle$. Hence, each of our synthesis problems can be solved by the reduction to finding ε-optimal policies in POMDPs as follows:

- obtain the above (infinite but finitely representable) POMDP $\mathcal{M}_\mathcal{N}$,
- identify an ε-optimal policy σ of $\mathcal{M}_\mathcal{N}$ w.r.t. appropriate measure, and
- translate σ to $\mathbf{d}^\sigma$.

The correctness of the reduction is based on the following proposition.

Proposition 4.1.2. *For any policy σ of $\mathcal{M}_\mathcal{N}$ it holds that*

$$\mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}[\mathrm{TR}] = \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}[\mathrm{TR}] \quad \textit{and} \quad \mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}[\mathrm{MP}] = \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}[\mathrm{MP}].$$

Sketch. Informally, the proposition follows from the fact, that our performance measures are defined using sums that sum the reward and time of each step of the path. In the POMDP $\mathcal{M}_\mathcal{N}$ we omit some states, thus we make larger steps and we count the correct effect (i.e., expected reward, expected time, and probability) of each larger step. The desired result is formally shown in Section 4.3. □

Now we will state and prove some properties of POMDP $\mathcal{M}_{\mathcal{N}}$. The next proposition follows directly from the construction of $\mathcal{M}_{\mathcal{N}}$.

Proposition 4.1.3. *If ACTMC $\mathcal{N}$ has localized alarms, then $\mathcal{M}_{\mathcal{N}}$ is an MDP.*

The previous proposition is very useful, since there are efficient polynomial time algorithms for finding optimal strategies in MDPs. The next two propositions are handy for showing that solution methods for finding optimal strategies in MDPs and POMDPs are applicable for $\mathcal{M}_{\mathcal{N}}$.

Proposition 4.1.4. *Let $s, s' \in S_{\text{set}} \cup S_{\text{off}}$ be such that $s \neq s'$. If s' is reachable from s in $\mathcal{N}$ then s' is reachable from s in $\mathcal{M}_{\mathcal{N}}$ under arbitrary policy σ.*

Sketch. Informally, every transition leaving some reachable state has a nonzero probability that it occurs. This follows from the fact that PDF of an exponential distribution is positive on all nonnegative real numbers. Hence, for each state s' reachable from s in ACTMC $\mathcal{N}$, it holds that there is positive probability that some run reaches s' from s. From the construction of $\mathcal{M}_{\mathcal{N}}$, the positive probability of runs that reach s' from s carries from $\mathcal{N}$ also to $\mathcal{M}_{\mathcal{N}}$. The full proof is provided in Section 4.3. □

The proof of the next result is omitted as it is similar to the proof of Proposition 4.1.4.

Proposition 4.1.5. *Let $a \in A$, $s \in S_{\text{set}} \cap S_a$, and $d, d' \in [\ell_a, u_a]$. Then distributions $\Pi_s(d)$ and $\Pi_s(d')$ have the same support.*

4.2 Explicit Parameter Synthesis Algorithms

Note that the action space of $\mathcal{M}_{\mathcal{N}}$ is dense and that $\Pi_s(d)$, $\Theta_s(d)$, and $€_s(d)$ might be irrational. Our algorithms depend on the solution methods explained in Section 3.3.5 (e.g., linear programming for MDPs), hence we have to ensure a finite action space and rational probability and expectation values. We thus define the δ-discretization of $\mathcal{M}_{\mathcal{N}}$ as an MDP $\mathcal{M}_{\mathcal{N}}\langle\delta\rangle = (S_{\text{set}} \cup S_{\text{off}}, A \cup S_{\text{off}}, O, Act^{\delta}, T^{\delta}, t^{\delta}, c^{\delta}, s_{\text{in}}, G)$ for a given discretization function $\delta\colon A \to \mathbb{Q}_{>0}$. $\mathcal{M}_{\mathcal{N}}\langle\delta\rangle$, which is defined as $\mathcal{M}_{\mathcal{N}}$ above but over the action space $Act^{\delta} \overset{\text{def}}{=} \bigcup_{a \in A} Act^{\delta}_a \cup \bigcup_{s \in S_{\text{off}}} Act_s$ with

$$Act^{\delta}_a \overset{\text{def}}{=} \big\{ \langle\!\langle a, d \rangle\!\rangle : d = \ell_a + i \cdot \delta(a) < u_a, i \in \mathbb{N}_0 \big\} \cup \big\{ \langle\!\langle a, u_a \rangle\!\rangle \big\}.$$

To ensure rational values of $\Pi_s(d)$, $\Theta_s(d)$, and $€_s(d)$, we consider the set of κ-approximations $[\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ of $\mathcal{M}_{\mathcal{N}}\langle\delta\rangle$ for any $\kappa \in \mathbb{Q}_{>0}$. From Proposition 4.1.4 it follows that if $\mathcal{N}$ is strongly connected, every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ is also strongly connected.

A straightforward approach towards solving MPSP for $\mathcal{N}$ is to compute a sufficiently small discretization function δ, approximation constant κ, and some $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ such that synthesizing an optimal policy in $\mathcal{M}$ yields an ε-optimal parameter function for $\mathcal{N}$. As $\mathcal{M}$ is finite and contains only rational probability and expectation values, the synthesis of an optimal policy for $\mathcal{M}$ can then be carried out using standard algorithms for POMDPs and MDPs (see, Section 3.3.5). This approach is applicable under the following mild assumptions:

Assumption 1. *For every $\varepsilon \in \mathbb{Q}_{>0}$, there are computable $\delta\colon A \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$ such that for every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ and every optimal policy σ for $\mathcal{M}$, the associated parameter function $\mathbf{d}^\sigma$ is ε-optimal for $\mathcal{N}$.*[1]

Assumption 2. *For all $\kappa \in \mathbb{Q}_{>0}$ and $s \in S_{\text{set}}$, there are computable rational κ-approximations Π_s^κ, Θ_s^κ, $€_s^\kappa$ of Π_s, Θ_s, $€_s$.*

Assumption 1 usually follows from perturbation bounds on the expected mean payoff and total reward using a straightforward error-propagation analysis. Assumption 2 can be obtained, e.g., by first computing $\kappa/2$-approximations of Π_s, Θ_s, and $€_s$ for $s \in S_{\text{set}} \cap S_a$, considering a as alarm with Dirac distribution, and then integrate the obtained functions over the probability measure determined by $F_a[x]$ to get the resulting κ-approximation (see also [32, 48]). Hence, Assumptions 1 and 2 rule out only those types of distributions that are rarely used in practice. The assumptions are satisfied for uniform, Dirac, and Weibull distributions in particular, but also for many other distributions as well. Note that Assumption 2 implies that for all $\delta\colon S_{\text{set}} \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$, there is a computable $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$. The pseudo-code of the explicit algorithm for minimization of $\mathbb{E}_{\mathcal{N}}^{\mathbf{d}}[\text{MP}]$ is provided in Algorithm 1. The correctness of Algorithm 1 is trivial, thus we just state the following.

1. This assumption is formulated in a general way and can be easily adapted for specific case. E.g., if we want to find an ε-minimal parameter function in $\mathbb{E}_{\mathcal{N}}[\text{MP}]$ the statement of the assumption would look like: For every $\varepsilon \in \mathbb{Q}_{>0}$, there are computable $\delta\colon A \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$ such that for every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ and every minimal policy σ in $\mathbb{E}_{\mathcal{M}}^{\sigma}[\text{MP}]$ for $\mathcal{M}$, the associated parameter function $\mathbf{d}^\sigma$ is ε-minimal in $\mathbb{E}_{\mathcal{N}}^{\mathbf{d}}[\text{MP}]$ for $\mathcal{N}$.

Algorithm 1: Explicit algorithm for minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$

```
input : A strongly connected parametric ACTMC $\mathcal{N}$ with rational $Q$,
        transition probabilities, and finitely representable $\langle F_a[x] \rangle$;
        reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$; and $\varepsilon \in \mathbb{Q}_{>0}$ such that
        Assumptions 1–2 are fulfilled.
output: An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$.
1 compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
2 compute $\delta$ and $\kappa$ of Assumption 1
3 fix the functions $\Pi^{\kappa}_{s}, \Theta^{\kappa}_{s}, €^{\kappa}_{s}$ of Assumption 2 determining
    $\mathcal{M}_{\kappa} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$
4 if $\mathcal{M}_{\kappa}$ is an MDP then construct and solve linear program for $\mathcal{M}_{\kappa}$
    obtaining policy $\sigma$ with minimal $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{MP}]$
5 else find policy $\sigma$ with minimal $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{MP}]$ by using the branch and
    bound algorithm
6 return $\mathbf{d}^{\sigma}$
```

Theorem 4.2.1. *The Algorithm 1 effectively solves the ε-minimal mean-payoff parameter synthesis problem for parametric ACTMCs and reward structures that fulfill Assumptions 1–2.*

The algorithm for maximization of the expected mean payoff and its correctness statement are dual and thus omitted.

From Section 3.3.5 we know that for applying the same approach as above for TRSP we need to assure that under each policy there is probability 1 to reach a goal state from an arbitrary state. Observe that from definition of the expected total reward, if $\mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\mathrm{TR}] < \infty$ then probability of reaching G from s_{in} and all states reachable from s_{in} is 1 for a fixed policy σ. From restriction to parametric ACTMC $\mathcal{N}$ without unreachable states and from Proposition 4.1.4, we have that all states in $\mathcal{M}_{\mathcal{N}}$ are reachable. Moreover, since $\mathcal{M}_{\mathcal{N}}$ has a finite state space we have the following corollary of Proposition 4.1.5.

Corollary 4.2.2. *Let σ, σ' be policies of $\mathcal{M}_{\mathcal{N}}$. Then $\mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\mathrm{TR}] = \infty$ if and only if $\mathbb{E}^{\sigma'}_{\mathcal{M}_{\mathcal{N}}}[\mathrm{TR}] = \infty$.*

Hence it suffices to pick an arbitrary policy σ of $\mathcal{M}_{\mathcal{N}}$ and check whether $\mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\mathrm{TR}] = \infty$. This can be done in polynomial time in the size of $\mathcal{M}_{\mathcal{N}}$ using DTMC solution techniques such as solving of linear equations. If

$\mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\mathrm{TR}] \neq \infty$ we can use solution techniques for MDPs and POMDPs. The pseudo-code of the explicit algorithm for minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$ is provided in Algorithm 2.

Algorithm 2: Explicit algorithm for minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$

```
input : A parametric ACTMC $\mathcal{N}$ without unreachable states, with
        rational $Q$, transition probabilities, and finitely representable
        $\langle F_a[x] \rangle$; reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$; and $\varepsilon \in \mathbb{Q}_{>0}$ such that
        Assumptions 1–2 are fulfilled.
output: An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$.
1 compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
2 compute $\delta$ and $\kappa$ of Assumption 1
3 fix the functions $\Pi^{\kappa}_{s}, \Theta^{\kappa}_{s}, €^{\kappa}_{s}$ of Assumption 2 determining
    $\mathcal{M}_{\kappa} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$
4 choose an arbitrary policy $\sigma$ of $\mathcal{M}_{\kappa}$ and compute $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{TR}]$
5 if $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{TR}] = \infty$ then return $\mathbf{d}^{\sigma}$
6 if $\mathcal{M}_{\kappa}$ is an MDP then construct and solve linear program for $\mathcal{M}_{\kappa}$
    obtaining policy $\sigma$ with minimal $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{TR}]$
7 else find policy $\sigma$ with minimal $\mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{TR}]$ by using the branch and
    bound algorithm
8 return $\mathbf{d}^{\sigma}$
```

Theorem 4.2.3. *The Algorithm 2 effectively solves the ε-minimal total-reward parameter synthesis problem for parametric ACTMCs and reward structures that fulfill Assumptions 1–2.*

We omit a trivial proof of Theorem 4.2.3. Moreover, we omit the dual algorithm, theorem, and proof for maximization of the expected total reward.

Complexity of Explicit Algorithms. It is hard to provide the worst case complexity of the explicit algorithms for ACTMC with arbitrary distributions satisfying Assumptions 1 and 2, because we do not know the complexity of computation of κ-approximations Π^{κ}_{s}, Θ^{κ}_{s}, $€^{\kappa}_{s}$ of Assumptions 2. Thus we evaluate complexity for ACTMCs with only Dirac-distributed alarms as it was done in [32] for minimization of expected total reward: The bounds κ and δ are in the worst case exponential in encoding size of $\mathcal{N}$ if Q, ℓ_a, and u_a are given in unary encoding for each $a \in A$ (see Chapter 6 where formulas for δ and κ are provided). For each $a \in A$, $s \in S_{\text{set}} \cap S_a$, and $s \in [\ell_a, u_a] \cap \mathbb{Q}$ the κ-approximations $\Pi^{\kappa}_s(d)$, $\Theta^{\kappa}_s(d)$, $€^{\kappa}_s(d)$ of Assumptions 2 can be computed in polynomial time to the encoding size of $\mathcal{N}$ provided that Q, ℓ_a, and u_a are given in unary encoding [32]. Thus, the running time of the explicit algorithm for minimization of expected total reward is exponential in the encoding size of $\mathcal{N}$ if Q, ℓ_a, and u_a are given in unary encoding for each $a \in A$ and is double exponential in $||\mathcal{N}||$. One cannot hope for polynomial complexity as the corresponding threshold problem for the expected total reward is NP-complete, even if we restrict to instances where Q, ℓ_a, and u_a are of magnitude polynomial in $||\mathcal{N}||$ [32]. These results can be easily extended to the maximization of expected total reward and minimization and maximization of mean payoff.

4.3 Formal Proofs

In this section we provide the proofs of propositions that were just sketched in this chapter. We first formally prove that ACTMC $\mathcal{N}$ with its reward structure and POMDP $\mathcal{M}_{\mathcal{N}}$ have the same expected mean payoff and total reward for corresponding parameter functions and policies.

Proposition 4.1.2. *For any policy σ of $\mathcal{M}_{\mathcal{N}}$ it holds that*

$$\mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\text{TR}] = \mathbb{E}^{\mathbf{d}^{\sigma}}_{\mathcal{N}}[\text{TR}] \quad \textit{and} \quad \mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}}[\text{MP}] = \mathbb{E}^{\mathbf{d}^{\sigma}}_{\mathcal{N}}[\text{MP}].$$

Proof. Let us fix a policy σ and let R denote the set of all runs in $\mathcal{M}_{\mathcal{N}}$ for σ. Similarly, let $\hat{R}$ denote the set of all runs in $\mathcal{N}$ for $\mathbf{d}^{\sigma}$. Note that for each $n \in \mathbb{N}$, both R and $\hat{R}$ can be partitioned into countable collections of sets $\mathcal{R}_n = \{R_{s_0 \cdots s_n} \colon s_0, \ldots, s_n \in S_{\text{set}} \cup S_{\text{off}}\}$ and $\hat{\mathcal{R}}_n = \{\hat{R}_{s_0 \cdots s_n} \mid s_0, \ldots, s_n \in S\}$ where

$$\begin{aligned} R_{s_0 \cdots s_n} &\overset{\text{def}}{=} \left\{ s'_0 b_0 s'_1 b_1 \cdots \mid \forall i \le n : s'_i = s_i \text{ and } b_i = \sigma(O(s_i)) \right\} \\ \hat{R}_{s_0 \cdots s_n} &\overset{\text{def}}{=} \{ (s'_0, \eta_0) t_0 \cdots \mid s'_0 = s_0, \exists i_1 < \cdots < i_n : \\ &\qquad \forall j \le n \colon s'_{i_j} = s_j \text{ and } (s'_{i_j}, \eta_{i_j}) \text{ is regenerative}, \\ &\qquad \forall i < i_n \text{ such that } i \neq i_j \text{ for any } j \colon (s'_i, \eta_i) \text{ is not regenerative} \}. \end{aligned}$$

Intuitively, the indices $s_0, \cdots, s_n$ in the latter definition are the states that correspond to the regenerative configurations, i.e., states where the "big steps" that are simulated by single steps in $\mathcal{M}_{\mathcal{N}}$ start. For each $n \in \mathbb{N}$, let W_n and $\hat{W}_n$ denote the reward accumulated in first n steps of $\mathcal{M}_{\mathcal{N}}$ and first n "big steps" of $\mathcal{N}$, respectively, such that reward stops being accumulated when G is reached. Moreover, for each $n \in \mathbb{N}$, let W_n^r and $\hat{W}_n^r$ denote the reward accumulated in first n steps of $\mathcal{M}_\mathcal{N}$ and first n "big steps" of $\mathcal{N}$, respectively. Similarly, we define W_n^t and $\hat{W}_n^t$ to be the time passed up to n-th step of $\mathcal{M}_\mathcal{N}$ and n-th "big step" of $\mathcal{N}$, respectively.

By a straightforward induction on n, it is easy to show that for each $n \in \mathbb{N}$ it holds that

$$\begin{aligned}
\mathrm{Pr}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[R_{s_0\cdots s_n}\right] &= \mathrm{Pr}^{\mathbf{d}^\sigma}_{\mathcal{N}}\left[\hat{R}_{s_0\cdots s_n}\right] \qquad \text{for all sequences of states } s_0\cdots s_n,\\
\mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[W_n\right] &= \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}\left[\hat{W}_n\right], \text{ and}\\
\mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[\frac{W_n^r}{W_n^t}\right] &= \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}\left[\frac{\hat{W}_n^r}{\hat{W}_n^t}\right].
\end{aligned}$$

This implies that, denoting by N and $\hat{N}$ the set of all runs of $\mathcal{M}_\mathcal{N}$ and $\mathcal{N}$ that do not reach a goal state,

$$\mathrm{Pr}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[N\right] > 0 \qquad \iff \qquad \mathrm{Pr}^{\mathbf{d}^\sigma}_{\mathcal{N}}\left[\hat{N}\right] > 0.$$

Furthermore, we get the required results as

$$\begin{aligned}
\mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}[\mathrm{TR}] &= \lim_{n\to\infty} \mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[W_n\right]\\
&= \lim_{n\to\infty} \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}\left[\hat{W}_n\right]\\
&= \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}[\mathrm{TR}]
\end{aligned}$$

and since $\mathcal{N}$ and $\mathcal{M}_\mathcal{N}$ are strongly connected

$$\begin{aligned}
\mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}[\mathrm{MP}] &= \lim_{n\to\infty} \mathbb{E}^{\sigma}_{\mathcal{M}_\mathcal{N}}\left[\frac{W_n^r}{W_n^t}\right]\\
&= \lim_{n\to\infty} \mathbb{E}^{\sigma}_{\mathcal{N}}\left[\frac{\hat{W}_n^r}{\hat{W}_n^t}\right]\\
&= \mathbb{E}^{\mathbf{d}^\sigma}_{\mathcal{N}}[\mathrm{MP}].
\end{aligned}$$

□

Now we show that reachability in an ACTMC $\mathcal{N}$ carries over to the POMDP $\mathcal{M}_\mathcal{N}$.

Proposition 4.1.4. *Let $s, s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$ be such that $s \neq s'$. If s' is reachable from s in $\mathcal{N}$ then s' is reachable from s in $\mathcal{M}_\mathcal{N}$ under arbitrary policy σ.*

Proof. Let us fix a policy σ of $\mathcal{M}_{\mathcal{N}}$ and states $s, s' \in S_{\text{set}} \cup S_{\text{off}}$ such that $s \neq s'$. Since s' is reachable from s in $\mathcal{N}$, there is a sequence of states $s_0, \dots, s_n$ such that $s = s_0$, $s' = s_n$, and $Q(s_i, s_{i+1}) > 0$ or $P_a(s_i, s_{i+1}) > 0$ (for some $a \in A$) for all $0 \leq i < n$. We define a set of runs $R^{s_0, I_0, \dots, s_n, I_n}_{J_0, \dots, J_{n-1}}$ where

$$I_0 \overset{\text{def}}{=} \begin{cases} \text{support of } F_a[\mathbf{d}^\sigma(a)] & \text{if } s_0 \in S_a \text{ for some } a \in A \text{, or} \\ \infty & \text{otherwise} \end{cases}$$

and for each $i < n$ we set

$$J_i \overset{\text{def}}{=} \begin{cases} (0, \max I_i) & \text{if } I_i \neq \infty \text{, or} \\ [0, 5] & \text{otherwise} \end{cases}$$

and

$$I_{i+1} \overset{\text{def}}{=} \begin{cases} I_i & \text{if } \exists a \in A \colon s_i, s_{i+1} \in S_a \text{ and } Q(s_i, s_{i+1}) > 0, \\ \infty & \text{if } s_{i+1} \in S_{\text{off}} \text{, or} \\ \text{support of } F_a[\mathbf{d}^\sigma(a)] & \text{otherwise (i.e., } s_{i+1} \in S_a \text{ for some } a \in A \text{).} \end{cases}$$

Clearly $\Pr^{\mathbf{d}^\sigma}_{\mathcal{N}(s_0)}\big(R^{s_0, I_0, \dots, s_n, I_n}_{J_0, \dots, J_{n-1}}\big) > 0$. Let $s'_0, \dots, s'_k$ be the minimal subsequence of $s_0, \dots, s_n$ that contains all states s_i such that I_i was set to a support of some $F_a[\mathbf{d}^\sigma(a)]$ or $s_i \in S_{\text{off}}$. Observe that states $s'_0, \dots, s'_k$ are exactly states of sequence of regenerative configurations of each run in $R^{s_0, I_0, \dots, s_n, I_n}_{J_0, \dots, J_{n-1}}$. From $\Pr^{\mathbf{d}^\sigma}_{\mathcal{N}(s_0)}\big(R^{s_0, I_0, \dots, s_n, I_n}_{J_0, \dots, J_{n-1}}\big) > 0$ from the construction of $\mathcal{M}_{\mathcal{N}}$ we have that

$$\Pr^{\sigma}_{\mathcal{M}_{\mathcal{N}}(s_0)}\big(s_0 \sigma(O(s_0)) s_1 \sigma(O(s_1)) \dots\big) > 0.$$

□

Chapter 5

Symbolic Parameter Synthesis

Often, the explicit algorithms are computationally infeasible due the large number of actions in $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$. We developed new *symbolic algorithms* that are more effective than the explicit algorithms. The algorithms are based on policy iteration for MDPs, thus they are correct for ACTMCs with localized alarms. We will first describe the symbolic algorithms for optimization of the expected mean payoff and then for the expected total reward in ACTMCs with localized alarms. We also describe extensions of the symbolic algorithms for optimization of the expected mean payoff for ACTMCs with non-localized alarms. In this chapter we assume that the previously fixed $\mathcal{N}$ has all alarms localized, unless we explicitly state differently.

5.1 ACTMCs with Localized Alarms

The discretization steps returned by function δ of Assumption 1 are typically small numbers. Note that the MDP $\mathcal{M}$ of Assumption 2 has (at most) the same number of states as $\mathcal{N}$, however the number of actions can be very large. When $\delta(a)$ is small and $s \in S_{\text{set}} \cap S_a$, we have to evaluate κ-approximations $\Pi^{\kappa}_{s}(d)$, $\Theta^{\kappa}_{s}(d)$, $€^{\kappa}_{s}(d)$ for a very large number of discretized parameter values d. E.g., for the disk drive model of Example 1.0.1 we get more than 10^{19} actions even for good choice of constants: $N = 2$ and $\varepsilon = 0.1$. Our symbolic parameter synthesis algorithms compute the set of states of some $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ (see Assumption 1) but avoid computing the set of all actions of $\mathcal{M}$ and their effects. The algorithms are obtained by modifying the standard policy iteration technique [130] for MDPs. We now describe the symbolic approach for minimization of expected mean payoff, the maximization is dual.

5.1.1 Expected Mean Payoff

Standard Policy Iteration Algorithm. When applied to $\mathcal{M}$, the standard policy iteration (see Algorithm 3) starts by picking an arbitrary policy σ, which is then repeatedly improved until a fixed point is reached. In each iteration, the current policy σ is first evaluated by computing the associated *gain* g and *bias* $\mathbf{h}$.[1] Then, for each state $s \in S_{\text{set}} \cap S_a$, every outgoing action $\langle\!\langle a, d \rangle\!\rangle$ is ranked by the function

$$F_s^\kappa[g, \mathbf{h}](d) \stackrel{\text{def}}{=} €_s^\kappa(d) - g \cdot \Theta_s^\kappa(d) + \Pi_s^\kappa(d) \cdot \mathbf{h} \qquad (\times)$$

where $€_s^\kappa$, Θ_s^κ, and Π_s^κ are the determining functions of $\mathcal{M}$. If the action chosen by σ at a does not have the best (minimal) rank, it is improved by redefining $\sigma(a)$ to some best-ranked action. The new policy is then evaluated by computing its gain and bias and possibly improved again. The standard algorithm terminates when the same policy is obtained after an improvement, i.e., a fixed-point was reached. The termination guaranteed by the fact that in each iteration except the last one the associated gain and bias are strictly lexicographically improved [130]. Since there is a finite number of distinct policies, the policy iteration algorithm must terminate.

Symbolic κ-approximations. In many cases, $\Pi_s(d)$, $\Theta_s(d)$, and $€_s(d)$ for $s \in S_{\text{set}}$ are expressible as infinite sums where the summands comprise elementary functions such as polynomials or $\exp(\cdot)$. Given κ, one may effectively truncate these infinite sums into finitely many initial summands such that the obtained expressions are differentiable in the interval $[\ell_a, u_a]$ and yield the *analytical κ-approximations* $\boldsymbol{\Pi}_s^\kappa(d)$, $\boldsymbol{\Theta}_s^\kappa(d)$, and $\boldsymbol{€}_s^\kappa(d)$, respectively. Now we can analytically approximate $F_s^\kappa[g, \mathbf{h}](d)$ by the value $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$ obtained from $(\times)$ by using the analytical κ-approximations:

$$\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d) \stackrel{\text{def}}{=} \boldsymbol{€}_s^\kappa(d) - g \cdot \boldsymbol{\Theta}_s^\kappa(d) + \boldsymbol{\Pi}_s^\kappa(d) \cdot \mathbf{h}. \qquad (\star)$$

This function is differentiable for $d \in [\ell_a, u_a]$ when g and $\mathbf{h}$ are constant. Note that the discretized parameters minimizing $F_s^\kappa[g, \mathbf{h}](d)$ are either close to ℓ_a, u_a, or roots of the derivative of $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$. Using the roots and bounds ℓ_a and u_a, we can identify a small set of candidate actions and explicitly evaluate only those (instead of all actions). Note, that $\boldsymbol{\Pi}_s^\kappa(d)$, $\boldsymbol{\Theta}_s^\kappa(d)$, $\boldsymbol{€}_s^\kappa(d)$

1. Here, it suffices to know that g is a scalar and $\mathbf{h}$ is a vector assigning numbers to states and that they can be computed in polynomial time in the number of states of an MDP by solving linear equations. For more details, see Sections 8.2.1 and 8.6.1 in [130].

Algorithm 3: Policy Iteration Algorithm for Minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$

```
**input :** A strongly connected parametric ACTMC $\mathcal{N}$ with localized
         alarms, rational $Q$, transition probabilities, and finitely
         representable $\langle F_a[x] \rangle$; reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$; and
         $\varepsilon \in \mathbb{Q}_{>0}$ such that Assumptions 1–2 are fulfilled.
**output:** An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$.
1  compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
2  compute $\delta$ and $\kappa$ of Assumption 1
3  fix the functions $\Pi^{\kappa}_{s}, \Theta^{\kappa}_{s}, €^{\kappa}_{s}$ of Assumption 2 determining
   $\mathcal{M}_{\kappa} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$
4  choose an arbitrary state $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$ and a policy $\sigma'$ for $\mathcal{M}_{\kappa}$
5  **repeat**
6      $\sigma := \sigma'$
       // policy evaluation
7      compute the *gain*, i.e., the scalar $g := \mathbb{E}^{\sigma}_{\mathcal{M}_{\kappa}}[\mathrm{MP}]$
8      compute the *bias*, i.e., the vector $\mathbf{h}\colon S_{\mathrm{set}} \cup S_{\mathrm{off}} \to \mathbb{Q}$ satisfying
        $\mathbf{h}(s') = 0$ and $\mathbf{h}(s) = €^{\kappa}_{s}(d) - g \cdot \Theta^{\kappa}_{s}(d) + \Pi^{\kappa}_{s}(d) \cdot \mathbf{h}$ for each
        $s \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$, where $\sigma(O(s)) = \langle\langle \cdot, d \rangle\rangle$
9      **foreach** $a \in A$ *and* $s \in S_{\mathrm{set}} \cap S_a$ **do**
           // policy improvement
10         $B := \operatorname{argmin}_{\langle\langle a,d \rangle\rangle \in Act^{\delta}_{a}} F^{\kappa}_{s}[g, \mathbf{h}](d)$
11         **if** $\sigma(a) \in B$ **then** $\sigma'(a) := \sigma(a)$
12         **else** $\sigma'(a) := \langle\langle a, d \rangle\rangle$ where $\langle\langle a, d \rangle\rangle \in B$
13 **until** $\sigma = \sigma'$
14 **return** $\mathbf{d}^{\sigma}$
```

may return *irrational* values for rational arguments. Hence, they cannot be evaluated precisely even for the discretized parameter values. However, when Assumption 2 is fulfilled, it is safe to use *rational* κ-approximations $\Pi_s^\kappa(d)$, $\Theta_s^\kappa(d)$, $€_s^\kappa(d)$ for this purpose. Before we provide our symbolic algorithm, we formally state the additional assumptions required to guarantee its soundness:

Assumption 3. *For all $a \in A$, $s \in S_{\mathrm{set}} \cap S_a$, $\delta\colon A \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$, there are analytical κ-approximations $\boldsymbol{\Pi}_s^\kappa$, $\boldsymbol{\Theta}_s^\kappa$, $\boldsymbol{€}_s^\kappa$ of Π_s, Θ_s, $€_s$, respectively, such that the function $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$, where $g \in \mathbb{Q}$ and $\mathbf{h}\colon S_{\mathrm{set}} \cup S_{\mathrm{off}} \to \mathbb{Q}$ are constant, is differentiable for $d \in [\ell_a, u_a]$. Further, there is an algorithm approximating the roots of the derivative of $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$ in the interval $[\ell_a, u_a]$ up to the absolute error $\delta(a)/2$.*

Assumption 4. *For each $s \in S_{\mathrm{set}}$ (let a be the alarm of s) there is a computable constant $\Pi_s^{\min} \in \mathbb{Q}_{>0}$ such that for all $d \in [\ell_a, u_a]$ and $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$ we have that $\Pi_s(d)(s') > 0$ implies $\Pi_s(d)(s') \geq \Pi_s^{\min}$.*

Note that compared to Assumption 2, the κ-approximations of Assumption 3 are harder to construct: we require closed forms for $\boldsymbol{\Pi}_s^\kappa$, $\boldsymbol{\Theta}_s^\kappa$, and $\boldsymbol{€}_s^\kappa$ making the symbolic derivative of $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$ computable and suitable for effective root approximation.

Symbolic policy iteration algorithm. Algorithm 4 closely mimics the standard policy iteration algorithm (see Algorithm 3) except for two major modifications. At line 3, we compute a new precision ξ that is smaller than κ in order to overcome all errors associated with the symbolic algorithm. The second major modification is in the policy improvement, where we do the following for each alarm a and each a-setting state s: First, the local extrema points of $\boldsymbol{F}_s^\xi[g, \mathbf{h}](d)$ (cf. Equation ($\star$)) in the interval $[\ell_a, u_a]$ are identified by computing roots of its symbolic derivative (line 11) with error $\delta(a)/2$. Then, we construct a small set C of *candidate actions* that are close to these roots and the bounds ℓ_a, u_a (line 12). Observe that we include all discretized actions that are at most $3 \cdot \delta(a)/2$ far from the roots or the bounds ℓ_a, u_a, to overcome errors when isolating the roots. Each given candidate action is then evaluated using the function $F_s^\xi[g, \mathbf{h}](d) = €_s^\xi(d) - g \cdot \Theta_s^\xi(d) + \Pi_s^\xi(d) \cdot \mathbf{h}$ (cf. Equation ($\times$)). An improving candidate action is chosen based on the computed values in set B (lines 14–15).

Algorithm 4: Symbolic Algorithm for Minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$

```
   input : A strongly connected parametric ACTMC $\mathcal{N}$ with localized
           alarms, rational $Q$, transition probabilities, and finitely
           representable $\langle F_a[x] \rangle$; reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$; and
           $\varepsilon \in \mathbb{Q}_{>0}$ such that Assumptions 1–4 are fulfilled.
   output: An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$.
 1 compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
 2 compute $\delta$, $\kappa$, and $\Pi^{\min}_s$ of Assumptions 1 and 4
 3 let $\xi = \min\{\kappa/3, \Pi^{\min}_s/4 \colon \text{ where } s \in S_{\mathrm{set}}\}$
 4 fix the functions $\Pi^{\xi}_s, \Theta^{\xi}_s, €^{\xi}_s$ of Assumption 2 determining
     $\mathcal{M}_{\xi} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\xi}$
 5 choose an arbitrary state $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$ and a policy $\sigma'$ for $\mathcal{M}_{\xi}$
 6 repeat
 7     $\sigma := \sigma'$
       // policy evaluation
 8     compute the gain, i.e., the scalar $g := \mathbb{E}^{\sigma}_{\mathcal{M}_{\xi}}[\mathrm{MP}]$
 9     compute the bias, i.e., the vector $\mathbf{h} \colon S_{\mathrm{set}} \cup S_{\mathrm{off}} \to \mathbb{Q}$ satisfying
        $\mathbf{h}(s') = 0$ and $\mathbf{h}(s) = €^{\xi}_s(d) - g \cdot \Theta^{\xi}_s(d) + \Pi^{\xi}_s(d) \cdot \mathbf{h}$ for each
        $s \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$, where $\sigma(O(s)) = \langle\!\langle \cdot, d \rangle\!\rangle$
10     foreach $a \in A$ and $s \in S_{\mathrm{set}} \cap S_a$ do
           // policy improvement
11         compute the set $R$ of $\delta(a)/2$-approximations of the roots of the
             derivative of $\boldsymbol{F}^{\xi}_s[g, \mathbf{h}](d)$ in $[\ell_a, u_a]$ using Assumption 3
12         $C := \{\sigma(a)\} \cup$
             $\big\{\langle\!\langle a, d \rangle\!\rangle \in Act^{\delta}_a \colon |d - r| \leq 3 \cdot \delta(a)/2, \text{ for } r \in R \cup \{\ell_a, u_a\}\big\}$
13         $B := \operatorname{argmin}_{\langle\!\langle a,d \rangle\!\rangle \in C} F^{\xi}_s[g, \mathbf{h}](d)$
14         if $\sigma(a) \in B$ then $\sigma'(a) := \sigma(a)$
15         else $\sigma'(a) := \langle\!\langle a, d \rangle\!\rangle$ where $\langle\!\langle a, d \rangle\!\rangle \in B$
16 until $\sigma = \sigma'$
17 return $\mathbf{d}^{\sigma}$
```

Theorem 5.1.1 (Correctness of Algorithm 4)**.** *Algorithm 4 effectively solves the ε-minimal parameter synthesis problem for parametric ACTMCs with localized alarms and reward structures that fulfill Assumptions 1–4.*

Sketch. Observe that we keep in the candidate actions also the action chosen by the currently improved policy (see, line 12). Hence, the currently improved policy can be chosen in each iteration and the gain and bias would be maintained. From the properties of the policy iteration [130], if the policy is improved, then gain and bias strictly lexicographically improve. Thus, Algorithm 4 strictly lexicographically improves the gain and bias except the last iteration. Since the number of policies of $\mathcal{M}_\xi$ is finite, Algorithm 4 terminates.

A challenging point is that we compute only approximate minima of the function $\boldsymbol{F}^\xi_s[g,\mathbf{h}](d)$, which is *different* from the function $F^\xi_s[g,\mathbf{h}](d)$ used to evaluate the candidate actions. There may exist an action that is not in the candidate set C even if it has minimal $F^\xi_s[g,\mathbf{h}](d)$. Hence, the policy computed by Algorithm 4 is not necessarily minimal for $\mathcal{M}_\xi$. Fortunately, due to Assumption 1, the policy induces ε-minimal parameters for the parametric ACTMC $\mathcal{N}$ if it is minimal for *some* $\mathcal{M}' \in [\mathcal{M}_\mathcal{N}\langle\delta\rangle]_\kappa$. Therefore, for each $s \in S_{\text{set}}$ we construct Π'_s, Θ'_s, and $€'_s$ determining such $\mathcal{M}'$. Omitting the details, the functions Π'_s, Θ'_s, $€'_s$ are constructed from Π^ξ_s, $€^\xi_s$, Θ^ξ_s and slightly (by at most 2ξ) shifted $\boldsymbol{\Pi}^\xi_s$, $\boldsymbol{€}^\xi_s$, $\boldsymbol{\Theta}^\xi_s$. The constant ξ was chosen sufficiently small such that the shifted $\boldsymbol{\Pi}^\xi_s$, $\boldsymbol{€}^\xi_s$, $\boldsymbol{\Theta}^\xi_s$ are still κ-approximations of Π_s, Θ_s, $€_s$ and the shifted $\boldsymbol{\Pi}^\xi_s(d)(\cdot)$ is a correct distribution for each $d \in [\ell_a, u_a]$. The technical details of the construction are provided in Section 5.4. □

The algorithm, theorem, and proof for maximization of the expected mean payoff are dual to the minimization. In the algorithm it is sufficient to exchange argmin with argmax and update the header. In the corresponding theorem and proof one needs to use dual words, functions and operators to those highlighted in red.

5.1.2 Expected Total Reward

As discussed in Section 3.3.5 there is also policy iteration algorithm for identifying policy that optimizes the expected total reward in an MDP. The algorithm works the same way as the policy iteration algorithm for optimization of expected mean payoff, except for different ranking function in the policy

Algorithm 5: Symbolic Algorithm for Minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$

```
input : A parametric ACTMC $\mathcal{N}$ without unreachable states, with
        localized alarms, rational $Q$, transition probabilities, and
        finitely representable $\langle F_a[x] \rangle$; reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$;
        and $\varepsilon \in \mathbb{Q}_{>0}$ such that Assumptions 1–4 are fulfilled.
output: An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}]$.

1  compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
2  compute $\delta$, $\kappa$, and $\Pi^{\min}_s$ of Assumptions 1 and 4
3  let $\xi = \min\{\kappa/3, \Pi^{\min}_s/4\colon \text{ where } s \in S_{\mathrm{set}}\}$
4  fix the functions $\Pi^{\xi}_s, \Theta^{\xi}_s, \in^{\xi}_s$ of Assumption 2 determining
     $\mathcal{M}_\xi \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\xi$
5  choose an arbitrary policy $\sigma'$ of $\mathcal{M}_\xi$ and compute $\mathbb{E}^{\sigma'}_{\mathcal{M}_\xi}[\mathrm{TR}]$
6  if $\mathbb{E}^{\sigma'}_{\mathcal{M}_\xi}[\mathrm{TR}] = \infty$ then return $\mathbf{d}^{\sigma'}$
7  set scalar $g := 0$
8  repeat
9      $\sigma := \sigma'$
       // policy evaluation
10     compute the vector $\mathbf{h}\colon S_{\mathrm{set}} \cup S_{\mathrm{off}} \to \mathbb{Q}$ satisfying $\mathbf{h}(s) = 0$ for each
         $s \in G$ and $\mathbf{h}(s) = \mathbb{E}^{\sigma}_{\mathcal{M}_\xi(s)}[\mathrm{TR}]$ for each $s \in (S_{\mathrm{set}} \cup S_{\mathrm{off}}) \setminus G$
11     foreach $a \in A$ and $s \in S_{\mathrm{set}} \cap S_a$ do
           // policy improvement
12         compute the set $R$ of $\delta(a)/2$-approximations of the roots of the
             derivative of $\boldsymbol{F}^{\xi}_s[g, \mathbf{h}](d)$ in $[\ell_a, u_a]$ using Assumption 3
13         $C := \{\sigma(a)\} \cup$
             $\big\{\langle\!\langle a, d\rangle\!\rangle \in Act^{\delta}_a\colon |d - r| \leq 3 \cdot \delta(a)/2, \text{ for } r \in R \cup \{\ell_a, u_a\}\big\}$
14         $B := \operatorname{argmin}_{\langle\!\langle a,d\rangle\!\rangle \in C} F^{\xi}_s[g, \mathbf{h}](d)$
15         if $\sigma(a) \in B$ then $\sigma'(a) := \sigma(a)$
16         else $\sigma'(a) := \langle\!\langle a, d\rangle\!\rangle$ where $\langle\!\langle a, d\rangle\!\rangle \in B$
17 until $\sigma = \sigma'$
18 return $\mathbf{d}^{\sigma}$
```

improvement step and thus a slightly different policy evaluation step. We illustrate the differences on an MDP $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$. In the policy evaluation step a solution vector $\mathbf{h}\colon S_{\text{set}} \cup S_{\text{off}} \to \mathbb{Q}$ is computed for current policy σ, which returns zero for goal states and for each non-goal state s it returns the expected total reward from s to a goal state in $\mathcal{M}$ using σ, i.e., $\mathbf{h}(s) = 0$ for $s \in G$ and $\mathbf{h}(s) = \mathbb{E}^{\sigma}_{\mathcal{M}(s)}[\mathrm{TR}]$ for $s \in (S_{\text{set}} \cup S_{\text{off}}) \setminus G$. Then for each alarm $a \in A$ and state $s \in S_{\text{set}} \cap S_a$ every outgoing action $\langle\!\langle a, d\rangle\!\rangle$ is ranked by a function

$$F^{\kappa}_s[\mathbf{h}](d) \quad = \quad €^{\kappa}_s(d) + \Pi^{\kappa}_s(d) \cdot \mathbf{h}$$

where $€^{\kappa}_s$ and Π^{κ}_s are the determining functions of $\mathcal{M}$. Note that, one can use the same ranking functions as for optimization of the expected mean payoff, where $g = 0$ and vector $\mathbf{h}\colon S_{\text{set}} \cup S_{\text{off}} \to \mathbb{Q}$ has different semantics. As a consequence we use the same assumptions also for symbolic algorithms for optimization of expected total reward. The symbolic algorithm for minimization of expected total reward is provided in Algorithm 5. The symbolic algorithm for maximization of the expected total reward is dual to Algorithm 5, it suffices to exchange argmin with argmax and update the header. Similarly as Theorem 5.1.1 and its proof one can provide theorems and their proofs also for minimization and maximization of expected total reward. They are very similar to the expected mean-payoff case, thus they are omitted. The only significant change is that the termination follows from the fact that in the standard policy iteration the vector $\mathbf{h}$ is strictly improved in each iteration except the last one [130] (for minimization $\mathbf{h}$ monotonically decreases and for maximization it monotonically increases). This carries over to our symbolic algorithms for optimization of expected total reward using the same arguments as for the expected mean-payoff case.

5.2 Expected Mean Payoff in General ACTMCs

From now on we allow non-localized alarms in $\mathcal{N}$. Hence, the POMDPs in $[\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ might no longer be MDPs and we cannot use the MDP algorithms for finding optimal policies. However, as we stated in the related work section there is an algorithm called gradient based policy iteration (GBPI) for finding memoryless policies in POMDP that optimize the expected mean payoff. The algorithm is an extension of the basic policy iteration for optimization of mean payoff. The main complication in the basic policy iteration when executed on POMDP is that there might be an alarm $a \in A$ with multiple a-setting states $S_{\text{set}} \cap S_a$. I.e., the states in $S_{\text{set}} \cap S_a$ have the same observation and we have to choose the same action for them. This might cause a problem in the policy improvement step, since the optimal choices of parameters (i.e., actions) given by the ranking functions $F_s^\kappa[g, \mathbf{h}]$ might be different for each $s \in S_{\text{set}} \cap S_a$. One can overcome this problem by summing the effects of ranking function for all a-setting states and choose the improving action according to the sum. However, some states can be visited much more often than the others and their ranking function should have a higher priority. Thus the most natural solution is to average the ranking functions for a-setting states by the time-frequency distribution of the POMDP $\mathcal{M}$ for current policy σ. This approach is used also in the gradient based policy iteration.

We describe the GBPI as an extension of to the standard policy iteration when executed on POMDP $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$. In the policy improvement step, in addition to the gain and bias we compute also the time-frequency distribution $\psi\colon S_{\text{set}} \cup S_{\text{off}} \to \mathbb{Q}_{\geq 0}$ of $\mathcal{M}$ for current policy σ. Observe that we can represent $\mathcal{M}$ for a fixed policy as a CTMC (or DTMC with non-uniform steps) for which we can compute the invariant distribution (that coincides with the time-frequency distribution for CTMCs) using the standard methods that run in polynomial time in the size of the CTMC, e.g., solving of linear equations. Then in the policy improvement step for every alarm $a \in A$ we rank each action $\langle\!\langle a, d\rangle\!\rangle$ by a function

$$\sum_{s \in S_{\text{set}} \cap S_a} \psi(s) \cdot F_s^\kappa[g, \mathbf{h}](d)$$

where $F_s^\kappa[g, \mathbf{h}](d) = €_s^\kappa(d) - g \cdot \Theta_s^\kappa(d) + \Pi_s^\kappa(d) \cdot \mathbf{h}$ and $€_s^\kappa$, Θ_s^κ, and Π_s^κ are the determining functions of $\mathcal{M}$. The remaining parts of GBPI are the same as for the standard policy iteration. Note that GBPI may not return the optimal policy and it may not even terminate for certain POMDPs [108]. Hence GBPI is a heuristic algorithm. However, the use of GBPI may result in satisfactory policies for realistic models, what was demonstrated in [41, 108]. Since the problem of finding an MD policy that optimizes the expected mean payoff in a POMDP is NP-hard, one cannot hope for an efficient exact algorithm.

There is an interesting question whether GBPI terminates for arbitrary POMDP generated by our synthesis algorithms. Unfortunately the answer is negative, what is illustrated in Example 5.2.1. Thus we add a simple mechanism to GBPI that stops the algorithm if some previously used pol-

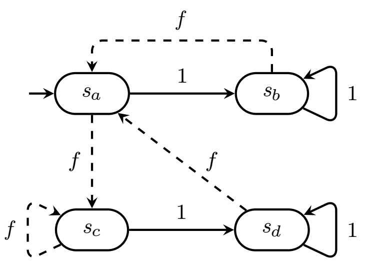


Figure 5.1: ACTMC model where GBPI may not terminate and Algorithm 6 may not return ε-optimal parameters.

icy is obtained (what indicates that GBPI will not terminate) and stops the algorithm, see lines 6, 10, and 18-19 of Algorithm 6.

By using the same approach as in the previous section we obtain a symbolic algorithm for minimization of expected mean payoff in ACTMC with non-localized alarms from the gradient based policy iteration, see Algorithm 6. The algorithm depends on one additional assumption that is a stronger version of Assumption 3:

Assumption 5. *In addition to Assumption 3, for each alarm $a \in A$ and rational $\psi \in \mathcal{D}(S_{\text{set}} \cup S_{\text{off}})$ there is an algorithm approximating the roots of the derivative of $\sum_{s \in S_{\text{set}} \cap S_a} \psi(s) \cdot \boldsymbol{F}^{\kappa}_s[g, \mathbf{h}](d)$ in the interval $[\ell_a, u_a]$ up to the absolute error $\delta(a)/2$.*

We do not provide any theorem, since the Algorithm 6 may not terminate with the ε-minimal policy, see Example 5.2.1. Note that Algorithms 4 and 6 choose the same improving action in the policy improvement when they are executed on an ACTMC with localized alarms. Since the gain (i.e., expected mean payoff) is never decreased by Algorithm 4 (see, proof of Theorem 5.1.1) the Algorithms 4 and 6 behave the same way on an ACTMC with localized alarms. One can obtain a symbolic algorithm for maximization of the expected mean payoff in an ACTMC with non-localized alarms by modifying the header and changing argmin with argmax in Algorithm 6.

Example 5.2.1. *Assume the parametric ACTMC $\mathcal{N}$ depicted in Figure 5.1. There is only one alarm f that has a Dirac-distributed CDF with eligible delay parameter values given by interval $[0.1, 5]$. The impulse rewards are zero everywhere and the rate reward is 0 in state s_d and 10 in the other states. We will show that Algorithm 6 may not return an ε-minimal parameter function for $\varepsilon = 0.5$.*

First, the sets $S_{\text{set}} = \{s_a, s_c\}$ and $S_{\text{off}} = \emptyset$ are computed. Observe that there are two f-setting states s_a and s_c. Hence, the POMDP vertices s_a and s_c have

Algorithm 6: General Symbolic Algorithm for Minimization of $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$

```
input : A strongly connected parametric ACTMC $\mathcal{N}$ with rational $Q$,
        transition probabilities, and finitely representable $\langle F_a[x]\rangle$;
        reward structure $(G, \mathcal{R}, \mathcal{I}, \mathcal{I}_A)$; and $\varepsilon \in \mathbb{Q}_{>0}$ such that
        Assumptions 1–5 are fulfilled.
output: An $\varepsilon$-minimal parameter function $\mathbf{d}$ in $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{MP}]$.
1  compute the sets $S_{\mathrm{set}}$ and $S_{\mathrm{off}}$
2  compute $\delta$, $\kappa$, and $\Pi^{\min}_s$ of Assumptions 1 and 4
3  let $\xi = \min\{\kappa/3, \Pi^{\min}_s/4\colon \text{ where } s \in S_{\mathrm{set}}\}$
4  fix the functions $\Pi^{\xi}_s, \Theta^{\xi}_s, €^{\xi}_s$ of Assumption 2 determining
   $\mathcal{M}_\xi \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\xi$
5  choose an arbitrary state $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$ and a policy $\sigma$ for $\mathcal{M}_\xi$
6  set $\mathfrak{S} := \{\sigma\}$, $\hat{g} := \infty$, and $\hat{\sigma} := \sigma$
7  while true do
       // policy evaluation
8      compute the stationary distribution $\psi$ of $\mathcal{M}_\xi$ for $\sigma$
9      compute the gain, i.e., the scalar $g := \mathbb{E}^{\sigma}_{\mathcal{M}_\xi}[\mathrm{MP}]$
10     if $\hat{g} > g$ then $\hat{g} := g$ and $\hat{\sigma} := \sigma$
11     compute the bias, i.e., the vector $\mathbf{h}\colon S \to \mathbb{Q}$ satisfying $\mathbf{h}(s') = 0$ and
         for each $s \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$, $\mathbf{h}(s) = €^{\xi}_s(d) - g \cdot \Theta^{\xi}_s(d) + \Pi^{\xi}_s(d) \cdot \mathbf{h}$, where
         $\sigma(O(s)) = \langle\!\langle a, d\rangle\!\rangle$
12     foreach $a \in A$ do
           // policy improvement
13         compute the set $R$ of $\delta(a)/2$-approximations of the roots of the
             derivative of $\sum_{s \in S_{\mathrm{set}} \cap S_a} \psi(s) \cdot \boldsymbol{F}^{\xi}_s[g, \mathbf{h}](d)$ in $[\ell_a, u_a]$ using
             Assumption 5
14         $C := \{\sigma(a)\} \cup$
             $\left\{\langle\!\langle a, d\rangle\!\rangle \in Act^{\delta}_a\colon |d - r| \le 3 \cdot \delta(a)/2, \text{ for } r \in R \cup \{\ell_a, u_a\}\right\}$
15         $B := \operatorname{argmin}_{\langle\!\langle a,d\rangle\!\rangle \in C} \sum_{s \in S_{\mathrm{set}} \cap S_a} \psi(s) \cdot F^{\xi}_s[g, \mathbf{h}](d)$
16         if $\sigma(a) \in B$ then $\sigma(a) := \sigma(a)$
17         else $\sigma(a) := \langle\!\langle a, d\rangle\!\rangle$ where $\langle\!\langle a, d\rangle\!\rangle \in B$
18         if $\sigma \in \mathfrak{S}$ then return $\mathbf{d}^{\hat{\sigma}}$
19         $\mathfrak{S} := \mathfrak{S} \cup \{\sigma\}$
```

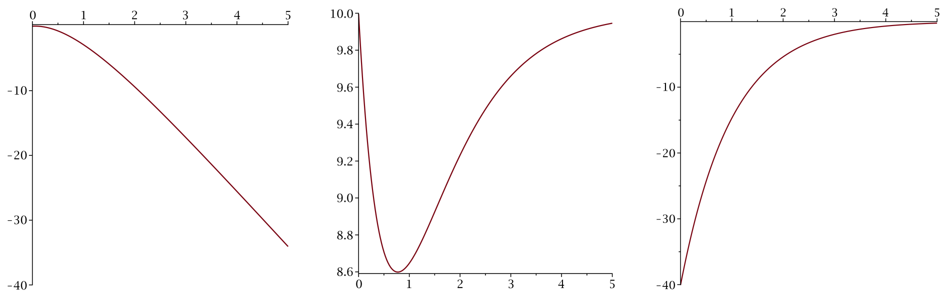


Figure 5.2: Graphs for the symbolic algorithm for minimization of the expected mean payoff in ACTMC of Figure 5.1. The horizontal axes give all eligible parameter values of the event f. The middle graph shows the corresponding expected mean-payoff. The left and right graphs show the values of ranking functions when the parameter of the event f is set to 0.1 and 5, respectively.

the same observation and always perform the same action. Moreover, observe that actions $\langle\langle f, 0.1\rangle\rangle$ and $\langle\langle f, 5\rangle\rangle$ belong to Act_f^δ for arbitrary choice of δ. Assume that the initial policy σ assigns to the alarm f an action $\langle\langle f, 0.1\rangle\rangle$. Here, the action $\langle\langle f, 5\rangle\rangle$ has the minimal rank (see, Figure 5.2 left). Intuitively, this is because of the following reasons:

- *much more time is spent in the vertex s_c than in s_a, hence the ranking function of s_a has a very weak influence on the overall ranking function (for each delay parameter $d \in \mathbb{R}_{>0}$ of f the time frequency of s_a and s_c is $1 - e^{-d}$ and e^{-d}, respectively) and*

- *the ranking function of s_c has minimum in 5 since by increasing the delay less reward is obtained in s_c and the increased probability of moving to s_a does not matter much because it has relatively small bias compared to s_c (the bias reflects that using the current policy a very short time was spent in s_a thus reaching it more often has a small impact on the expected mean payoff).*

Hence, the policy σ is redefined such that $\sigma(f) = \langle\langle f, 5\rangle\rangle$. Now, the action $\langle\langle f, 0.1\rangle\rangle$ has the minimal rank (see, Figure 5.2 right) because of the following reasons:

- *much more time is spent in the vertex s_a than in s_c, hence the ranking function of s_c has a very weak influence on the overall ranking function and*

- *the ranking function of s_a has minimum in 0.1 since by decreasing the delay a shorter time is spent s_a and more time is spent in more optimal (according to the current policy) vertex s_c.*

Hence, the policy σ is redefined such that $\sigma(f) = \langle\langle f, 0.1\rangle\rangle$ and Algorithm 6 terminates. Note that the GBPI would infinitely long switch between the first and second policy. The expected mean payoff generated by the first and second policy is approximately 9.56 and 9.94 ,respectively. Thus Algorithm 6 returns parameter function assigning 0.1 to the alarm f. However the minimal policy generates the expected mean payoff of approximately 8.6 (see, the middle graph of Figure 5.2). Thus Algorithm 6 may not return an ε-minimal parameter function. One can use almost the same example also to show that the dual of Algorithm 6 may not return an ε-maximal parameter function. It is sufficient to set the rate reward to 10 for state s_d and to 0 for the other states.

5.3 Complexity of Symbolic Algorithms

Performing the worst-case complexity analysis of our symbolic algorithms is problematic. The main difficulty is to bound the time to obtain the roots of symbolic derivative of ranking function and to estimate the actual number of different roots. Both quantities depend on the properties of the ranking function whose analytical form is determined by the alarm CDF and the chosen ε in a non-trivial way, which makes the analysis cumbersome. Therefore, we decided to evaluate the efficiency of our algorithm mainly experimentally (see Chapter 7) and provide a short discussion about complexity of symbolic algorithms just for ACTMC with Dirac distributed alarms:

As noted in Section 3.3.5, the policy iteration can make at most exponential number (in number of vertices with more than one action, thus in $|S_{\text{set}}|$) of iterations [64]. In each iteration we do the following:

- In the policy evaluation we solve linear equations to obtain π, g, and $\mathbf{h}$ what runs in polynomial time in the number of vertices, i.e., in $|S_{\text{set}} \cup S_{\text{off}}|$. The numbers that occur in the linear equations are obtained from determining functions $\Pi^{\xi}_{s}, \Theta^{\xi}_{s}, \text{€}^{\xi}_{s}$ of $\mathcal{M}_{\xi}$ evaluated on parameters given by the current policy. The encoding size of numbers occurring in the equations and the computation time needed to obtain them are polynomial in $||\mathcal{N}||$ if for each $a \in A$ the Q, ℓ_a, and u_a are given in unary encoding [32]. Thus the policy evaluation runs in polynomial time in $||\mathcal{N}||$ if for each $a \in A$ the Q, ℓ_a, and u_a are given in unary.
- As we will see in the next chapter, for Dirac-distributed alarms the functions Π^{ξ}_{s}, €^{ξ}_{s}, Θ^{ξ}_{s} equal to $\boldsymbol{\Pi}^{\xi}_{s}$, $\boldsymbol{\text{€}}^{\xi}_{s}$, $\boldsymbol{\Theta}^{\xi}_{s}$, respectively and they have

structure $P(d) \cdot e^{c \cdot d}$, where $P(d)$ is an univariate polynomial of a variable d and $c \in \mathbb{Q}$ is a constant. Thus also the ranking function $\boldsymbol{F}^{\xi}_{s}[g, \mathbf{h}]$ has the same form. Let $n \in \mathbb{N}_0$ be the degree of the polynomial in $\boldsymbol{F}^{\xi}_{s}[g, \mathbf{h}]$. After differentiation of $\boldsymbol{F}^{\xi}_{s}[g, \mathbf{h}]$ we obtain again function of the same form and the degree of the polynomial remains n. Since, e^x is always positive for arbitrary $x \in \mathbb{R}$, we need to isolate roots in the obtained univariate polynomial of degree n. Hence, the maximal number of the roots is n. From the first bullet (the polynomial running time to compute the actual values of functions Π^{ξ}_{s}, €^{ξ}_{s}, Θ^{ξ}_{s}) we get that n is polynomial in $||\mathcal{N}||$ if for each $a \in A$ the Q, ℓ_a, and u_a are given in unary. The root finding itself runs in logarithmic time in the encoding size of precision and in polynomial time in the encoding size of the univariate polynomial where the roots are to be isolated [96, 135]. Thus, the policy improvement takes polynomial time in $||\mathcal{N}||$ if for each $a \in A$ the Q, ℓ_a, and u_a are given in unary.

5.4 Formal Proofs

Here we provide the technical proof of Theorem 5.1.1 that was just sketched in the previous text for the readability purposes. The dual versions of the theorem and proof below can be obtained by using the dual functions and words to those highlighted in red .

Theorem 5.1.1 (Correctness of Algorithm 4)**.** *Algorithm 4 effectively solves the ε-minimal parameter synthesis problem for parametric ACTMCs with localized alarms and reward structures that fulfill Assumptions 1–4.*

Proof. We fix the parametric ACTMC $\mathcal{N}$, MDP $\mathcal{M}_\xi \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\xi$, the policy σ returned by Algorithm 4, all functions Π^{ξ}_{s}, €^{ξ}_{s}, Θ^{ξ}_{s}, $\boldsymbol{\Pi}^{\xi}_{s}$, $\boldsymbol{\text{€}}^{\xi}_{s}$, $\boldsymbol{\Theta}^{\xi}_{s}$, and gain g and bias $\mathbf{h}$ used during the last iteration of the outer cycle of Algorithm 4.

Observe that we keep in the candidate actions also the action chosen by the currently improved policy (see, line 12). Hence, the currently improved policy can be chosen, in each iteration and the gain and bias would be maintained. From the properties of the policy iteration [130], if the policy is changed, then the gain and bias strictly lexicographically improve. Thus, Algorithm 4 strictly lexicographically improves the gain and bias except the last iteration. Since the number of policies of $\mathcal{M}_\xi$ is finite, Algorithm 4 terminates.

A challenging point is that we compute only approximate minima of the function $\boldsymbol{F}^{\xi}_{s}[g,\mathbf{h}](d)$, which might be *different* from the function $F^{\xi}_{s}[g,\mathbf{h}](d)$ used to evaluate the candidate actions. There may exist an action that is not in the candidate set C even if it has minimal $F^{\xi}_{s}[g,\mathbf{h}](d)$. Hence, the policy computed by Algorithm 4 is not necessarily minimal for $\mathcal{M}_{\xi}$. Fortunately, due to Assumption 1, the policy induces ε-minimal parameters of $\mathcal{N}$ if it is minimal for *some* $\mathcal{M}' \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$. Therefore, it is sufficient to construct Π'_s, Θ'_s, and $€'_s$ determining such an MDP $\mathcal{M}'$ that is possibly different from $\mathcal{M}_{\xi}$.

Now, we construct an MDP $\mathcal{M}' \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ such that σ is minimal for $\mathcal{M}'$. For each $a \in A$ and $s \in S_{\text{set}} \cap S_a$ we define κ-approximations Π'_s, Θ'_s, $€'_s$ (of Π_s, Θ_s, $€_s$ on $\{d' \colon \langle\!\langle a, d'\rangle\!\rangle \in Act^{\delta}_{a}\}$) and $F'_s[g,\mathbf{h}](\cdot) \overset{\text{def}}{=} €'_s(\cdot) - g \cdot \Theta'_s(\cdot) + \Pi'_s(\cdot) \cdot \mathbf{h}$ such that

$$F'_s[g,\mathbf{h}](d) \leq F'_s[g,\mathbf{h}](d'),$$

for d satisfying $\sigma(a) = \langle\!\langle a, d\rangle\!\rangle$ and every $d' \in \{d'' \colon \langle\!\langle a, d''\rangle\!\rangle \in Act^{\delta}_{a}\}$.

We will construct the functions Π'_s, Θ'_s, $€'_s$ separately for each state $s \in S_{\text{set}}$. Let a be the unique alarm such that $s \in S_a$, d be a rational number such that $\sigma(a) = \langle\!\langle a, d\rangle\!\rangle$, and $D \overset{\text{def}}{=} \{d'' : \langle\!\langle a, d''\rangle\!\rangle \in Act^{\delta}_{a}\}$, i.e., $D = \{d'' : d'' = \ell_a + i \cdot \delta(a) < u_a, i \in \mathbb{N}_0\} \cup \{u_a\}$. Furthermore, let C be the candidate set used to improve the policy σ for s during the last iteration of the outer cycle in Algorithm 4. We define $C' \overset{\text{def}}{=} \{d'' : \langle\!\langle a, d''\rangle\!\rangle \in C\}$.

First, we will construct Π'_s, Θ'_s, $€'_s$ and prove the theorem assuming that we have in hand certain "shifted" κ-approximations $\boldsymbol{\Pi}'_s$, $\boldsymbol{€}'_s$, $\boldsymbol{\Theta}'_s$ that have some good properties. Then, we will construct the κ-approximations $\boldsymbol{\Pi}'_s$, $\boldsymbol{€}'_s$, $\boldsymbol{\Theta}'_s$ and show that they have the needed properties.

Assume we can define "shifted" κ-approximations $\boldsymbol{\Pi}'_s$, $\boldsymbol{€}'_s$, $\boldsymbol{\Theta}'_s$ (of Π_s, $€_s$, Θ_s on $[\ell_a, u_a]$) and $\boldsymbol{F}'_s[g,\mathbf{h}](d') \overset{\text{def}}{=} \boldsymbol{€}'_s(d') - g \cdot \boldsymbol{\Theta}'_s(d') + \boldsymbol{\Pi}'_s(d') \cdot \mathbf{h}$ for each $d' \in [\ell_a, u_a]$ such that

$$\exists c \geq 0 \colon \forall d' \in [\ell_a, u_a] \colon \boldsymbol{F}^{\xi}_{s}[g,\mathbf{h}](d') + c = \boldsymbol{F}'_s[g,\mathbf{h}](d') \tag{5.1}$$

and

$$\forall d' \in C' \colon F^{\xi}_{s}[g,\mathbf{h}](d') \leq \boldsymbol{F}'_s[g,\mathbf{h}](d'). \tag{5.2}$$

From Assumption 3 and definition of $\boldsymbol{F}'_s[g,\mathbf{h}]$ it follows that $\boldsymbol{F}'_s[g,\mathbf{h}]$ is continuous and it has the same arguments of local extrema as $\boldsymbol{F}^{\xi}_{s}[g,\mathbf{h}]$ on

$[\ell_a, u_a]$. From the definition of the candidate set C (line 12 of Algorithm 4) and C', it follows that $C' \cap \operatorname{argmin}_{d' \in D} \boldsymbol{F}'_s[g, \mathbf{h}](d')$ is nonempty.

For each $d' \in D$ we set

- $\Pi'_s(d') \overset{\text{def}}{=} \Pi^\xi_s(d')$, $€'_s(d') \overset{\text{def}}{=} €^\xi_s(d')$, $\Theta'_s(d') \overset{\text{def}}{=} \Theta^\xi_s(d')$ if $d' = d$,
- $\Pi'_s(d') \overset{\text{def}}{=} \boldsymbol{\Pi}'_s(d')$, $€'_s(d') \overset{\text{def}}{=} \boldsymbol{€}'_s(d')$, $\Theta'_s(d') \overset{\text{def}}{=} \boldsymbol{\Theta}'_s(d')$ if $d' \neq d$, and
- $F'_s[g, \mathbf{h}](d') \overset{\text{def}}{=} €'_s(d') - g \cdot \Theta'_s(d') + \Pi'_s(d') \cdot \mathbf{h}$.

Let $\underline{d}$ be an arbitrary candidate of $C' \cap \operatorname{argmin}_{d' \in D} \boldsymbol{F}'_s[g, \mathbf{h}](d')$. Then for each $d' \in D$ we have

$$F'_s[g, \mathbf{h}](d) = F^\xi_s[g, \mathbf{h}](d) \leq \boldsymbol{F}'_s[g, \mathbf{h}](\underline{d}) \leq \boldsymbol{F}'_s[g, \mathbf{h}](d') = F'_s[g, \mathbf{h}](d').$$

The left inequality follows from the definition of $\boldsymbol{F}'_s[g, \mathbf{h}]$ (see equation (5.2)), the right inequality follows from the definition of $\underline{d}$.

To complete the proof, it remains to define κ-approximations $\boldsymbol{\Pi}'_s$, $\boldsymbol{€}'_s$, $\boldsymbol{\Theta}'_s$ (of Π_s, $€_s$, Θ_s on $[\ell_a, u_a]$) such that they satisfy equations (5.1) and (5.2). Let $\overline{d} \overset{\text{def}}{=} \operatorname{argmax}_{d' \in C'} F^\xi_s[g, \mathbf{h}](d') - \boldsymbol{F}^\xi_s[g, \mathbf{h}](d')$. If $F^\xi_s[g, \mathbf{h}](\overline{d}) - \boldsymbol{F}^\xi_s[g, \mathbf{h}](\overline{d}) \leq 0$ we set $\Delta\Pi \overset{\text{def}}{=} \mathbf{0}_{S_{\text{set}} \cup S_{\text{off}}}$, $\Delta\Theta \overset{\text{def}}{=} 0$, and $\Delta€ \overset{\text{def}}{=} 0$. Otherwise, we set

- $\Delta\Pi \overset{\text{def}}{=} \Pi^\xi_s(\overline{d}) - \boldsymbol{\Pi}^\xi_s(\overline{d})$,
- $\Delta\Theta \overset{\text{def}}{=} \Theta^\xi_s(\overline{d}) - \boldsymbol{\Theta}^\xi_s(\overline{d})$, and
- $\Delta€ \overset{\text{def}}{=} €^\xi_s(\overline{d}) - \boldsymbol{€}^\xi_s(\overline{d})$.

Now, for all $d' \in D$, we put

- $\boldsymbol{\Pi}'_s(d') \overset{\text{def}}{=} \boldsymbol{\Pi}^\xi_s(d') + \Delta\Pi$,
- $\boldsymbol{\Theta}'_s(d') \overset{\text{def}}{=} \boldsymbol{\Theta}^\xi_s(d') + \Delta\Theta$, and
- $\boldsymbol{€}'_s(d') \overset{\text{def}}{=} \boldsymbol{€}^\xi_s(d') + \Delta€$.

We show that (5.1) and (5.2) hold: For each $d' \in [\ell_a, u_a]$ we have that

$$\begin{aligned} \boldsymbol{F}'_s[g, \mathbf{h}](d') &= \boldsymbol{€}'_s(d') - g \cdot \boldsymbol{\Theta}'_s(d') + \boldsymbol{\Pi}'_s(d') \cdot \mathbf{h} \\ &= (\boldsymbol{€}^\xi_s(d') + \Delta€) - g \cdot (\boldsymbol{\Theta}^\xi_s(d') + \Delta\Theta) + (\boldsymbol{\Pi}^\xi_s(d') + \Delta\Pi) \cdot \mathbf{h} \\ &= \boldsymbol{F}^\xi_s[g, \mathbf{h}](d') + c, \end{aligned}$$

where $c = 0$ if $F_s^\xi[g,\mathbf{h}](\overline{d}) - \boldsymbol{F}_s^\xi[g,\mathbf{h}](\overline{d}) \leq 0$ and $c = F_s^\xi[g,\mathbf{h}](\overline{d}) - \boldsymbol{F}_s^\xi[g,\mathbf{h}](\overline{d}) >$ 0 otherwise. Thus (5.1) holds. Since $c = \max\{0, F_s^\xi[g,\mathbf{h}](\overline{d}) - \boldsymbol{F}_s^\xi[g,\mathbf{h}](\overline{d})\}$ and $\overline{d} \overset{\text{def}}{=} \operatorname{argmax}_{d' \in C'} F_s^\xi[g,\mathbf{h}](d') - \boldsymbol{F}_s^\xi[g,\mathbf{h}](d')$, for each $d' \in C'$ it holds that

$$F_s^\xi[g,\mathbf{h}](d') \leq \boldsymbol{F}_s^\xi[g,\mathbf{h}](d') + c = \boldsymbol{F}'_s[g,\mathbf{h}](d'),$$

what implies (5.2).

It remains to show that $\boldsymbol{\Pi}'_s$, $\boldsymbol{€}'_s$, $\boldsymbol{\Theta}'_s$ are κ-approximations of Π_s, $€_s$, Θ_s on $[\ell_a, u_a]$: Note, that for each $s' \in S_{\text{set}} \cup S_{\text{off}}$ it holds that $\Delta\Pi(s') \leq 2\xi$, $\Delta\Theta \leq 2\xi$, and $\Delta€ \leq 2\xi$. Since $\xi \leq \kappa/3$, $\boldsymbol{\Theta}'_s$ and $\boldsymbol{€}'_s$ are κ-approximations of Θ_s and $€_s$ on $[\ell_a, u_a]$, respectively, and for each $d' \in [\ell_a, u_a]$ and $s' \in S_{\text{set}} \cup S_{\text{off}}$ it holds that $|\boldsymbol{\Pi}'_s(d')(s') - \Pi_s(d')(s')| \leq 3\xi \leq \kappa$.

Furthermore, for each $d' \in [\ell_a, u_a]$, $\boldsymbol{\Pi}'_s(d')$ is a distribution and it has the same support as $\Pi_s(d')$: Let us fix $d' \in [\ell_a, u_a]$. Since $\boldsymbol{\Pi}_s^\xi$ and Π_s^ξ are ξ-approximations of Π_s, then for each state $s' \in S_{\text{set}} \cup S_{\text{off}}$ it holds that if $\Pi_s(d')(s') = 0$ then $\boldsymbol{\Pi}_s^\xi(d')(s') = 0$ and $\Pi_s^\xi(d')(s') = 0$ (if defined) and thus also $\boldsymbol{\Pi}'_s(d')(s') = 0$. Moreover, since $\xi \leq \Pi_s^{\min}/4$ and $|\boldsymbol{\Pi}'_s(d')(s') - \Pi_s(d')(s')| \leq 3\xi$ we have that if $\Pi_s(d')(s') > 0$ then $\boldsymbol{\Pi}'_s(d')(s') > 0$ for each $s' \in S_{\text{set}} \cap S_{\text{off}}$. Thus, $\boldsymbol{\Pi}'_s(d')$ has the same support as $\Pi_s(d')$. Finally, observe that $\sum_{s' \in S_{\text{set}} \cup S_{\text{off}}} \Delta\Pi(s') = 0$. Thus, it holds that $\sum_{s' \in S_{\text{set}} \cup S_{\text{off}}} \boldsymbol{\Pi}'_s(d')(s') = 1$. All the previous statements imply that $\boldsymbol{\Pi}'_s(d')$ is a distribution, thus $\boldsymbol{\Pi}'_s$ is κ-approximation of Π'_s on $[\ell_a, u_a]$. $\square$

Chapter 6

Distributions Satisfying Our Assumptions

In this technical chapter we show that the Assumptions 1–5 formulated in Section 4.2 are satisfied for the ε-optimal mean-payoff parameter synthesis problem where some concrete types of distributions $F_a[d]$ are used. The list is by no means exhaustive and the studied distributions should be seen just as examples demonstrating the practical applicability of our symbolic algorithms. Moreover we provide formulas for computing function δ and constant κ of Assumption 1. Some of the input arguments in the formulas take into account the worst case transition structure of the underlying model and they are too loose for practical models. Thus we also provide a few heuristics that usually give much better values of these parameters and hence we get much better δ and κ.

The following theorem implies that the explicit and symbolic algorithms are applicable to parametric ACTMCs with uniform, Dirac, exponential, or Weibull distributions.

Theorem 6.0.1. *Assumptions 1–5 are fulfilled for parametric ACTMCs with rational cost functions where, for all $a \in A$, the function $F_a[x]$ is CDF of either a uniform, Dirac, exponential, or Weibull distribution.*

First, we define new Assumptions A and B that formalize important discretization bounds on certain quantities of a parametric ACTMC. In Sections 6.1 and 6.2, we show that these two new assumptions imply Assumptions 1 and 4. This we successfully use in Sections 6.3, 6.5, and 6.6 where we separately show that Assumptions 2, 3, 5, A, and B are fulfilled for Dirac, uniform, and Weibull distributions, respectively. Note that the exponential distribution is a special case of the Weibull distribution, where the fixed shape constant k is 1.

Recall that from the previous chapters we have fixed a reward structure and a finitely representable ACTMC without unreachable states. We formalize the following two **assumptions**:

Assumption A. *For every $s \in S_{\text{set}}$, there are effectively computable positive rational bounds*

$$\Pi_s^{\min}, \Theta_s^{\min}, \Theta_s^{\max}, \text{ and } €_s^{\max}$$

such that for all $d \in [\ell_a, u_a]$, where a is the alarm of s, i.e., $s \in S_a$, we have

- $\Pi_s^{\min} \leq \Pi_s(d)(s')$ *for all* $s' \in S_{\text{set}} \cup S_{\text{off}}$ *where* $\Pi_s(d)(s') > 0$
- $\Theta_s^{\min} \leq \Theta_s(d) \leq \Theta_s^{\max}$
- $€_s(d) \leq €_s^{\max}$.

Assumption B. *For every $s \in S_{\text{set}}$ and $\kappa \in \mathbb{Q}_{>0}$, there is a discretization bound $\delta_{(s,\kappa)} \in \mathbb{Q}_{>0}$ such that for every $d, d' \in [\ell_a, u_a]$, where $|d - d'| \leq \delta_{(s,\kappa)}$, it holds that*

- $|\Pi_s(d)(s') - \Pi_s(d')(s')| \leq \kappa$ *for all* $s' \in S_{\text{set}} \cup S_{\text{off}}$,
- $|\Theta_s(d) - \Theta_s(d')| \leq \kappa$, *and*
- $|€_s(d) - €_s(d')| \leq \kappa$

where a is the alarm of s, i.e., $s \in S_a$.

Intuitively, Assumption B connects κ-approximation and δ-discretization. We will show the connection more formally. To simplify the reasoning we first define an auxiliary assumption:

Assumption C. *For every $\varepsilon \in \mathbb{Q}_{>0}$ we can compute sufficiently small $\kappa \in \mathbb{Q}_{>0}$ such that for every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}]_\kappa$ and every optimal policy σ for $\mathcal{M}$, the associated parameter function $\mathbf{d}^\sigma$ is ε-optimal for $\mathcal{N}$.*

Now we will show that if Assumptions B and C are fulfilled then also Assumption 1 is fulfilled, i.e., for every $\varepsilon \in \mathbb{Q}_{>0}$, there are computable $\delta\colon A \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$ such that for every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_\kappa$ and every optimal policy σ for $\mathcal{M}$, the associated parameter function $\mathbf{d}^\sigma$ is ε-optimal for $\mathcal{N}$.

Let us fix an $\varepsilon \in \mathbb{Q}_{>0}$ and a $\kappa \in \mathbb{Q}_{>0}$ from Assumption C. Thus we can make κ error when computing the action effects. We will compute the action effects with higher precision $\kappa/2$ (i.e., we will allow only $\kappa/2$-approximation for Assumption 1) and dedicate $\kappa/2$ of precision for the errors caused by discretization. Let $\delta\colon A \to \mathbb{Q}_{>0}$ be a function obtained from Assumption B by setting $\delta(a) \overset{\text{def}}{=} \min\{\delta_{(s,\kappa/2)} \colon s \in S_a \cap S_{\text{set}}\}$ for each $a \in A$. We show that an optimal policy σ of $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa/2}$ induces an ε-optimal policy $\mathbf{d}^\sigma$ for

$\mathcal{N}$ (what induces validity of Assumption 1): Let Π_s, Θ_s, $€_s$ be the determining functions of $\mathcal{M}$. We define the determining functions Π'_s, Θ'_s, $€'_s$ of $\mathcal{M}' \in [\mathcal{M}_\mathcal{N}]_{2\kappa/2}$ as follows: for each $d \in [\ell_a, u_a)$ we define $\Pi'_s(d) \overset{\text{def}}{=} \Pi_s(\delta'(d))$, $\Theta'_s(d) \overset{\text{def}}{=} \Theta_s(\delta'(d))$, and $€'_s(d) \overset{\text{def}}{=} €_s(\delta'(d))$, where $\delta'(d) = \ell_a + i \cdot \delta(a)$ for $d \in \big[\ell_a + i \cdot \delta(a), \min\{u_a, \ell_a + (i+1) \cdot \delta(a)\}\big)$ and $i \in \mathbb{N}_0$, and $\Pi'_s(u_a) \overset{\text{def}}{=} \Pi_s(u_a)$, $\Theta'_s(u_a) \overset{\text{def}}{=} \Theta_s(u_a)$, and $€'_s(u_a) \overset{\text{def}}{=} €_s(u_a)$. Clearly, $\mathcal{M}' \in [\mathcal{M}_\mathcal{N}]_{2\kappa/2} = [\mathcal{M}_\mathcal{N}]_\kappa$, σ is an optimal policy of $\mathcal{M}'$, and $\mathbf{d}^\sigma$ is an ε-optimal policy of $\mathcal{N}$, what follows from our choice of κ from Assumption C.

Thus we concentrate on a computation of a sufficiently small κ for each $\varepsilon > 0$, what is addressed separately for expected total reward and mean payoff in the next two sections.

6.1 Computation of κ for Expected Total Reward

Using standard techniques of numerical analysis, we express the bound on κ in order to have bounded change in expected total accumulated reward caused by κ-approximation as a function of the worst-case (among all possible initial states) expected total reward and expected number of steps before reaching goal vertices. Let $\mathcal{M}$ be a POMDP with set of vertices V and set of goal vertices V', σ be a policy of $\mathcal{M}$, and $\kappa' > 0$. We denote by $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\text{TR}]_{\max}$ and $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\#]_{\max}$ the maximal expected total reward and maximal expected number of steps until reaching goal vertices from any non-goal vertex under policy σ in any κ'-approximation of $\mathcal{M}$, respectively. More formally, we define

- $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\text{TR}]_{\max} \overset{\text{def}}{=} \sup_{\mathcal{M}', v \in V \setminus V'} \mathbb{E}^\sigma_{\mathcal{M}'(v)}[\text{TR}]$ and
- $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\#]_{\max} \overset{\text{def}}{=} \sup_{\mathcal{M}'', v \in V \setminus V'} \mathbb{E}^\sigma_{\mathcal{M}''(v)}[\text{TR}]$

where $\mathcal{M}'$ ranges among all κ'-approximations of $\mathcal{M}$ and $\mathcal{M}''$ ranges among all κ'-approximations of $\mathcal{M}$ where reward 1 is assigned to each vertex-action combination. Note that it might be impossible to compute exact values of $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\text{TR}]_{\max}$ and $\mathbb{E}^{\sigma,\kappa'}_\mathcal{M}[\#]_{\max}$. Actually, we use these quantities only in the next lemma where it is sufficient to provide just upper bounds on these quantities. Hence, we first obtain bound on κ as a function of these quantities, then we bound the quantities form above using two approaches: by functions of constants from Assumption A and heuristics.

Lemma 6.1.1. *Let $\mathcal{M}$ be a POMDP with vertices V and goal vertices V' and σ be a policy of $\mathcal{M}$ that almost surely reaches a goal vertex from each vertex. Moreover, let $\mathcal{M}'$ be a κ-approximation of $\mathcal{M}$ and $\kappa' > 0$. For every $\varepsilon > 0$ and vertex $v \in V \setminus V'$ it holds that $\left|\mathbb{E}^{\sigma}_{\mathcal{M}(v)}[\mathrm{TR}] - \mathbb{E}^{\sigma}_{\mathcal{M}'(v)}[\mathrm{TR}]\right| \leq \varepsilon$ if*

$$\kappa \leq \min\left\{\frac{1}{2\cdot \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}\cdot n}, \frac{\varepsilon}{2\cdot \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}\cdot(1+\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}\cdot n)}, \kappa'\right\}$$

where $n = |V \setminus V'|$ is the number of non-goal vertices of $\mathcal{M}$.

Proof. In the following we denote by $||\mathbf{x}||_\infty$ a *uniform norm* of a vector $\mathbf{x}$, i.e., the maximal absolute value of a component of $\mathbf{x}$. This norm induces a natural matrix norm $||\cdot||_\infty$, where for a real matrix X the number $||X||_\infty$ represents the maximal absolute row sum of X.

Since the vertex space is finite, under σ a goal vertex is reached with probability 1 from any vertex, and since κ-approximation maintains the transition structure, both $\mathbb{E}^{\sigma}_{\mathcal{M}(v)}[\mathrm{TR}]$ and $\mathbb{E}^{\sigma}_{\mathcal{M}'(v)}[\mathrm{TR}]$ are finite for each vertex $v \in V \setminus V'$. We start the proof by expressing the values $\mathbb{E}^{\sigma}_{\mathcal{M}(v)}[\mathrm{TR}]$ and $\mathbb{E}^{\sigma}_{\mathcal{M}'(v)}[\mathrm{TR}]$ as solutions of certain systems of linear equations. To achieve this, we consider discrete time absorbing Markov chains $\mathcal{A}$ and $\mathcal{A}'$ obtained by fixing policy σ in $\mathcal{M}$ and $\mathcal{M}'$, respectively, and replacing all transitions outgoing from states in V' with self-loops on these states. Note that the set V' is exactly the set of absorbing states of both $\mathcal{A}$ and $\mathcal{A}'$.

Let $V'' = V \setminus V'$ be the set of transient states of $\mathcal{A}$ (and hence also of $\mathcal{A}'$) and let $P, P' \colon V'' \times V'' \to [0,1]$ be the substochastic matrices encoding transitions in transient parts of $\mathcal{A}$ and $\mathcal{A}'$, respectively. That is, P is an $|V''| \times |V''|$ substochastic matrix with $P(v,w) = T(v,\sigma(O(v)))(w)$, for all $v, w \in V''$, and similarly for P'. Next, we denote by $\mathbf{f}, \mathbf{f}' \colon V'' \to \mathbb{R}_{>0}$ the $|V''|$-dimensional vectors of one-step rewards obtained with policy σ in $\mathcal{M}$ and $\mathcal{M}'$, respectively. That is, $\mathbf{f}(v) = c(v,\sigma(O(v)))$ for all $v \in V''$, and similarly for $\mathbf{f}'$.

From standard results on Markov models with the total accumulated reward criterion (see, e.g.,Theorem 7.1.3 in [130]) we have that the $|V''|$-dimensional vector $(\mathbb{E}^{\sigma}_{\mathcal{M}(v)}[\mathrm{TR}])_{v\in V''}$ is equal to the unique solution $\mathbf{x}$ of the following system of linear equations:

$$(I-P)\cdot \mathbf{x} = \mathbf{f}.$$

(The uniqueness of the solution comes from $\mathcal{A}$ being absorbing, which implies that $(I-P)$ is invertible [95]). Similarly, the vector $(\mathbb{E}^{\sigma}_{\mathcal{M}'(s)}[\mathrm{TR}])_{v\in V''}$ is the unique solution $\mathbf{x}'$ of the following system of equations:

$$(I-P')\cdot\mathbf{x}'=\mathbf{f}'.$$

By our assumptions, we have $P'=P+\Delta P$, where ΔP is a matrix whose each entry has absolute value bounded by κ, and $\mathbf{f}'=\mathbf{f}+\Delta\mathbf{f}$, where each entry of vector $\Delta\mathbf{f}$ has absolute value bounded by κ, hence

$$(I-P-\Delta P)\cdot\mathbf{x}'=\mathbf{f}+\Delta\mathbf{f}.$$

It suffices to obtain a bound on the maximal component of $|\mathbf{x}-\mathbf{x}'|$. We can assume that $||\mathbf{x}-\mathbf{x}'||_\infty=\max_{v\in V''}|\mathbf{x}'(v)-\mathbf{x}(v)|$. If this assumption does not hold, we swap $\mathcal{M}$ and $\mathcal{M}'$. Now we replace $\Delta\mathbf{f}$ with $|\Delta\mathbf{f}|$. We claim that the unique solution $\hat{\mathbf{x}}'$ to the system

$$(I-P-\Delta P)\cdot\hat{\mathbf{x}}'=\mathbf{f}+|\Delta\mathbf{f}|.$$

satisfies $\hat{\mathbf{x}}'\geq\mathbf{x}'$. Indeed, this is because $\hat{\mathbf{x}}'=(I-P-\Delta P)^{-1}\cdot(\mathbf{f}+|\Delta\mathbf{f}|)\geq(I-P-\Delta P)^{-1}\cdot(\mathbf{f}+\Delta\mathbf{f})=\mathbf{x}'$, the middle inequality following from the fact that the matrix $(I-P-\Delta P)^{-1}$, so called fundamental matrix of a Markov chain $\mathcal{A}'$, is always non-negative. So to prove the lemma it suffices to give a bound on $||\mathbf{x}-\hat{\mathbf{x}}'||_\infty$. Note that $||\Delta\mathbf{f}||_\infty\leq\kappa$.

We use standard results on stability of perturbed systems of linear equations. From, e.g., proof of Theorem 3 in [83] it follows that

$$||\mathbf{x}-\hat{\mathbf{x}}'||_\infty\leq\frac{||(I-P)^{-1}||_\infty}{1-||(I-P)^{-1}||_\infty\cdot||\Delta P||_\infty}\cdot\left(||\Delta\mathbf{f}||_\infty+||\mathbf{x}||_\infty\cdot||\Delta P||_\infty\right).\tag{6.1}$$

Note that if $\mathcal{M}$ and $\mathcal{M}'$ are swapped, then we replace $||(I-P)^{-1}||_\infty$ and $||\mathbf{x}||_\infty$ in inequality (6.1) by $||(I-P')^{-1}||_\infty$ and $||\mathbf{x}'||_\infty$, respectively. From the definition of κ-approximation we know that $||\Delta P||_\infty\leq\kappa\cdot|V''|=\kappa\cdot n$ and $||\Delta\mathbf{f}||_\infty\leq\kappa$, and from the definition of $\mathbf{x}$ and $||\mathbf{x}'||_\infty$ we have that $||\mathbf{x}||_\infty,||\mathbf{x}'||_\infty\leq\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}$ if we require $\kappa\leq\kappa'$. Moreover, as noted above, $(I-P)^{-1}$ is the fundamental matrix of the absorbing Markov chain $\mathcal{A}$, i.e., a matrix such that the entry $(I-P)^{-1}(v,w)$ is equal to the expected number of visits to a transient vertex w when starting in a transient vertex v of $\mathcal{A}$. Similarly, $(I-P')^{-1}(v,w)$ is equal to the expected number of visits to a transient vertex w when starting in a transient vertex v of $\mathcal{A}'$. In particular, if $\kappa \leq \kappa'$ sum of each row of $(I-P)^{-1}$ and $(I-P')^{-1}$ is bounded by $\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}$, thus $||(I-P)^{-1}||_\infty, ||(I-P')^{-1}||_\infty \leq \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}$. Plugging these (in)equalities into (6.1) we get

$$\begin{aligned}||\mathbf{x}-\hat{\mathbf{x}}'||_\infty &\leq \frac{\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}}{1-\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}\cdot\kappa\cdot n}\cdot\left(\kappa+\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}\cdot\kappa\cdot n\right)\\ &\leq 2\cdot\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}\cdot(\kappa+\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}\cdot\kappa\cdot n),\end{aligned}$$

the last inequality following from the fact that in the lemma statement we require $\kappa \leq 1/(2\cdot\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}\cdot n)$. The lemma easily follows. $\square$

We can obtain bounds on $\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max}$ and $\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}$ under arbitrary policy σ of POMDP $\mathcal{M}$ using bounds on transition probabilities and rewards of $\mathcal{M}$. Note that in the next lemma, we use only the bounds specified in Assumption A for $s \in S_{\mathrm{set}}$. Bounds for other states can be computed accordingly.

Lemma 6.1.2. *Let $\mathcal{M}$ be a POMDP, where there is probability 1 to reach a goal vertex from each vertex under arbitrary policy of $\mathcal{M}$. If $\kappa' \leq \min\{T_{\min}/2, c_{\max}\}$, then*

$$\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\#]_{\max} \leq \frac{n}{(T_{\min}/2)^n} \quad \textit{and} \quad \sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max} \leq \frac{2n\cdot c_{\max}}{(T_{\min}/2)^n},$$

where σ ranges over all policies of $\mathcal{M}$, n is the number of non-goal vertices of $\mathcal{M}$, and $T_{\min}, c_{\max} \in \mathbb{Q}_{>0}$ are bounds on the minimal probability and maximal rewards occurring in $\mathcal{M}$, respectively.

Proof. Let us fix a policy σ, a non-goal vertex v of $\mathcal{M}$, and let $\mathcal{M}'$ be a κ'-approximation of $\mathcal{M}$. The runs of $\mathcal{M}'$ that reach a goal vertex from v before reaching some non-goal vertex twice (including the first occurrence of v) have probability at least $(T_{\min}/2)^n$ (the lower bound on minimal probability is $T_{\min}/2$ because of κ'-approximation) and they make at most n steps before reaching the goal vertex. The remaining runs of $\mathcal{M}'$ make at most n steps before reaching the first non-goal vertex second time and have probability at most $(1-(T_{\min}/2)^n)$. We can use the same arguments again for every vertex that was the first reached second time from v before reaching

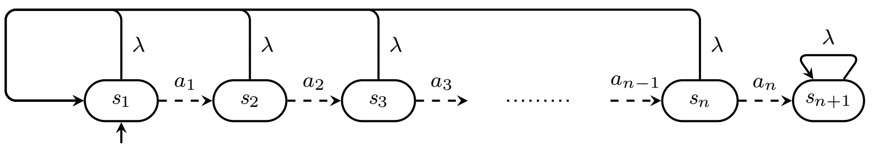


Figure 6.1: CTMC model of an M/M/1/N queue.

a goal vertex. Thus the expected number of steps until reaching a goal from v can be bounded from above by the following infinite sum

$$\sum_{i=1}^{\infty} i \cdot n \cdot \left(1 - (T_{\min}/2)^n\right)^{i-1} \cdot (T_{\min}/2)^n = \frac{n}{(T_{\min}/2)^n}.$$

The second part of the lemma follows from the fact, that the runs generate at most $2 \cdot c_{\max}$ of reward in each step due to κ'-approximation. $\square$

6.1.1 Heuristics for Larger κ

We will discuss bounds of Lemma 6.1.2 applied to POMDP $\mathcal{M}_{\mathcal{N}}$ for given ACTMC $\mathcal{N}$. The theoretical bound of Lemma 6.1.2 is exponential in the number of vertices of $\mathcal{M}_{\mathcal{N}}$. Observe that the bound on $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_{\mathcal{N}}}[\#]_{\max}$ linearly influences $1/\kappa$. For some distributions the use of theoretical upper bound of Lemma 6.1.2 may cause that we get an exponential number of actions in $\mathcal{M}_{\mathcal{N}}$ (in the number of states of $\mathcal{M}_{\mathcal{N}}$). This cannot be significantly improved, since we can provide a model, that asymptotically matches this upper bound and perturbation of transition probabilities makes a large error in the expected number of steps until reaching goal vertices (what is a special case of expected total reward), see the following example.

Example 6.1.3. *Assume the ACTMC $\mathcal{N}$ in Figure 6.1, where there are n Dirac distributed alarms, each alarm a_i is active in the state s_i only, all the rates are λ, and all delays of Dirac distributions are d. Moreover, assume that $G = \{s_{n+1}\}$, rate rewards are always zero and impulse rewards return 1 for each transition. To reach the goal state, there must be n successful alarm transitions, that have probability $p \overset{\text{def}}{=} e^{-\lambda d}$. If a delay transition is executed, the process is restarted. Thus, by partitioning the runs according to the number of times they reached state s_1, we can express the expected number of steps from s_1 to the goal state as the following infinite sum*

$$\sum_{i=1}^{\infty} l_i \cdot (1-p^n)^{i-1} \cdot p^n,$$

where l_i is the conditional expected number of steps from s_1 to the goal state if s_1 was visited i-times. The number l_i can be bounded from below by i, thus we get

$$\sum_{i=1}^{\infty} l_i \cdot (1-p^n)^{i-1} \cdot p^n \geq \sum_{i=1}^{\infty} i \cdot (1-p^n)^{i-1} \cdot p^n = \frac{1}{p^n}.$$

Note that even for very small perturbations of Dirac delays, the change in expected number of steps can be enormous. I.e., let $n = 4$ and $\lambda = 1$, if $\mathbf{d}(a_i) = 4.0$ for each $i \in \{0, \dots, n\}$ then $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}] \approx 9.05 \cdot 10^6$ and if $\mathbf{d}(a_i) = 4.1$ for each $i \in \{0, \dots, n\}$ then $\mathbb{E}^{\mathbf{d}}_{\mathcal{N}}[\mathrm{TR}] \approx 1.35 \cdot 10^7$.

However, most of the real world ACTMCs do not have such a bad transition structure and much better bounds on $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\#]_{\max}$ can be provided. In particular, one can directly observe the transition structure of $\mathcal{M}_\mathcal{N}$ and provide a more accurate bound for $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\#]_{\max}$ by hand or one can use the following heuristic:

1. set $\varepsilon \overset{\text{def}}{=} 1$,
2. obtain $\hat{\mathcal{N}}$ by making all alarms in $\mathcal{N}$ localized,
3. compute an upper bound on $\sup_\sigma \mathbb{E}^{\sigma,\hat{\kappa}'}_{\mathcal{M}_{\hat{\mathcal{N}}}}[\#]_{\max}$ and $\sup_\sigma \mathbb{E}^{\sigma,\hat{\kappa}'}_{\mathcal{M}_{\hat{\mathcal{N}}}}[\mathrm{TR}]_{\max}$ using Lemma 6.1.2 or by observing the transition structure of $\mathcal{M}_{\hat{\mathcal{N}}}$,
4. compute $\hat{\kappa}$ and $\hat{\delta}$, fix $\hat{\mathcal{M}} \in \left[\mathcal{M}_{\hat{\mathcal{N}}}\langle\hat{\delta}\rangle\right]_{\hat{\kappa}}$, and change $\hat{\mathcal{M}}$ to generate reward 1 in each action,
5. find the policy $\hat{\sigma}$ that maximizes $\mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}}[\mathrm{TR}]$ using the symbolic algorithm,
6. compute $\mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}(s)}[\mathrm{TR}]$ for each $s \in S_{\mathrm{set}} \cup S_{\mathrm{off}} \setminus G$ (usual methods perform this in one computation for fixed policy, moreover it is a side product of linear programming and policy iteration methods for MDPs),
7. now, if $\kappa' \leq \hat{\kappa}$ then $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\#]_{\max} \leq \max_{s \in S_{\mathrm{set}} \cup S_{\mathrm{off}} \setminus G} \mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}(s)}[\mathrm{TR}] + 1$.

The heuristic works thanks to the following observations: By making the alarms localized in ACTMCs (i.e., creating MDPs instead of POMDPs), we can only improve the maximal expected total reward, since there are fewer restrictions on the policies. Recall that from our assumptions it follows that all vertices in $\hat{\mathcal{M}}$ are reachable from the initial vertex. Hence the policy $\hat{\sigma}$ that maximizes $\mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}}[\mathrm{TR}]$ maximizes also $\mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}(s)}[\mathrm{TR}]$ for each $s \in (S_{\mathrm{set}} \cup S_{\mathrm{off}}) \setminus G$, otherwise we would be able to improve it.

Usually $\max_{s \in S_{\text{set}} \cup S_{\text{off}} \setminus G} \mathbb{E}^{\hat{\sigma}}_{\hat{\mathcal{M}}(s)}[\text{TR}] + 1$ is much tighter upper bound on $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\#]_{\max}$ than the theoretical bound of Lemma 6.1.2. The usage of this heuristic can significantly speed up the parameter synthesis: we have to run the parameter synthesis twice, but in both runs we have better bounds on $\hat{\kappa}$ and κ (in the first run thanks to relatively large ε and in the second run thanks to the better bound on $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\#]_{\max}$) what can help a lot. Similar heuristic as above can be applied also for bounding $\sup_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\text{TR}]_{\max}$.

Moreover, when we aim for ε-minimal policy in parametric ACTMC $\mathcal{N}$ with localized alarms, we can use the following observations: Every POMDP $\mathcal{M}_\mathcal{N}$ created for $\mathcal{N}$ is in fact an MDP since each vertex $\mathcal{M}_\mathcal{N}$ has unique observation. From our assumptions it holds that all vertices in $\mathcal{M}_\mathcal{N}$ are reachable from the initial vertex. Observe that the minimal policy for $\mathcal{M}_\mathcal{N}$ is minimal also for $\mathcal{M}_\mathcal{N}(s)$ for each $s \in (S_{\text{set}} \cup S_{\text{off}}) \setminus G$, otherwise we would be able to improve the policy. Thus arbitrary policy σ' of $\mathcal{M}_\mathcal{N}$ forms an upper bound on the minimal solution, thus $\inf_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\text{TR}]_{\max} \leq \max_{s \in (S_{\text{set}} \cup S_{\text{off}}) \setminus G} \mathbb{E}^{\sigma'}_{\mathcal{M}_\mathcal{N}(s)}[\text{TR}]$ holds for small enough κ'. Observe that, we want to show that Assumption C holds, where we do not need to provide proper κ that would bound the perturbation error for *every* policy, but it is sufficient to provide a good κ in order not to miss the ε-minimal policy. I.e., we aim to provide a good κ for policies that are near ε-minimal policies and for that purpose we can use arbitrary upper bound on $\inf_\sigma \mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\text{TR}]_{\max} + \varepsilon$ to substitute the term $\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\text{TR}]_{\max}$ in Lemma 6.1.1. Thus one can choose an arbitrary policy σ' of $\mathcal{M}_\mathcal{N}$, compute $\max_{s \in (S_{\text{set}} \cup S_{\text{off}}) \setminus G} \mathbb{E}^{\sigma'}_{\mathcal{M}_\mathcal{N}(s)}[TR]$ (errors in computation must be added) and use it for computation of κ substituting the term $\mathbb{E}^{\sigma,\kappa'}_{\mathcal{M}_\mathcal{N}}[\text{TR}]_{\max}$ in Lemma 6.1.1. One can even use the minimum from series of such computations, because they are relatively fast.

6.2 Computation of κ for Expected Mean Payoff

Since the mean payoff is defined as a fraction, we first connect the error bound of a fraction with perturbation bounds of its numerator and denominator. Intuitively, in order to guarantee an error φ of a fraction, the numerator and the denominator have to be computed with a certain precision.

Lemma 6.2.1. *For every $a, b, a', b', \overline{a}, \overline{b}, \underline{b}, \varphi \in \mathbb{R}_{>0}$ such that $a \leq \overline{a}$, and $\underline{b} \leq b \leq \overline{b}$, we have that*

$$\text{if both } |a-a'| \text{ and } |b-b'| \text{ are } \leq \frac{\underline{b}^2 \cdot \varphi}{\overline{a} + \overline{b} + \overline{b} \cdot \varphi} \text{ then } \left| \frac{a}{b} - \frac{a'}{b'} \right| \leq \varphi.$$

Proof. Let φ' be $(\underline{b}^2 \cdot \varphi)/(\overline{a} + \overline{b} + \overline{b} \cdot \varphi)$. We divide the proof into the following sub-cases:

(i) Let $a \geq a'$ and $b \leq b'$.
Then $a/b \geq a'/b'$. Due to $|a-a'| \leq \varphi'$, we have that $a' \geq a - \varphi'$ and, due to $|b - b'| \leq \varphi'$, we have that $b' \leq b + \varphi'$. Also note that

$$\varphi' = \frac{\underline{b}^2 \cdot \varphi}{\overline{a} + \overline{b} + \overline{b} \cdot \varphi} \leq \frac{b^2 \cdot \varphi}{a+b}.$$

Hence,

$$\left| \frac{a}{b} - \frac{a'}{b'} \right| = \frac{a}{b} - \frac{a'}{b'} \leq \frac{a}{b} - \frac{a-\varphi'}{b+\varphi'} = \frac{ab + a\varphi' - ba + b\varphi'}{b^2 + b\varphi'} =$$
$$= \frac{a\varphi' + b\varphi'}{b^2 + b\varphi'} = \frac{a+b}{b^2/\varphi' + b} \leq \frac{a+b}{b^2/\varphi'} = \frac{\varphi' \cdot (a+b)}{b^2} \leq \varphi.$$

(ii) Let $a \leq a'$ and $b \geq b'$.
Then $a/b \leq a'/b'$. Due to $|a-a'| \leq \varphi'$, we have that $a' \leq a + \varphi'$ and, due to $|b - b'| \leq \varphi'$, we have that $b' \geq b - \varphi'$. Also note that

$$\varphi' = \frac{\underline{b}^2 \cdot \varphi}{\overline{a} + \overline{b} + \overline{b} \cdot \varphi} \leq \frac{b^2 \cdot \varphi}{a + b + b \cdot \varphi}$$

Hence,

$$\left| \frac{a}{b} - \frac{a'}{b'} \right| = \frac{a'}{b'} - \frac{a}{b} \leq \frac{a+\varphi'}{b-\varphi'} - \frac{a}{b} = \frac{ab + \varphi' b - ba + \varphi' a}{b^2 - b\varphi'} =$$
$$= \frac{\varphi' b + \varphi' a}{b^2 - b\varphi'} = \frac{b+a}{b^2/\varphi' - b} \leq \frac{b+a}{b^2/((b^2 \cdot \varphi)/(a+b+b\cdot\varphi)) - b} =$$
$$= \frac{b+a}{(a+b)/\varphi + b - b} = \varphi.$$

(iii) Let $a < a'$ and $b < b'$.
If $a/b \geq a'/b'$ then $\left|\frac{a}{b} - \frac{a'}{b'}\right| = \frac{a}{b} - \frac{a'}{b'} \leq \frac{a'}{b} - \frac{a'}{b'} \leq \varphi$ due to (i).
If $a/b < a'/b'$ then $\left|\frac{a}{b} - \frac{a'}{b'}\right| = \frac{a'}{b'} - \frac{a}{b} \leq \frac{a'}{b} - \frac{a}{b} \leq \varphi$ due to (ii).

(iv) Let $a > a'$ and $b > b'$.
If $a/b \geq a'/b'$ then $\left|\frac{a}{b} - \frac{a'}{b'}\right| = \frac{a}{b} - \frac{a'}{b'} \leq \frac{a}{b'} - \frac{a'}{b'} \leq \varphi$ due to (i).
If $a/b < a'/b'$ then $\left|\frac{a}{b} - \frac{a'}{b'}\right| = \frac{a'}{b'} - \frac{a}{b} \leq \frac{a}{b'} - \frac{a}{b} \leq \varphi$ due to (ii).

□

The next lemma establishes a perturbation bound on κ such that the expected mean payoff achieved by a given policy changes among κ-approximations at most by a given $\varepsilon > 0$. First we define necessary notions. Let $\mathcal{M}$ be a POMDP with vertex space V, and σ be a policy such that $\mathcal{M}$ stays strongly connected when the set of actions is restricted to those selected by σ. We denote by $\mathcal{M}[v]$ for $v \in V$ a POMDP $\mathcal{M}$ where the goal vertices are set to $\{v\}$. For all $v, v' \in V$, let $\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v']}[\mathrm{TR}]$ and $\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v']}[\mathrm{Time}]$ be the expected total reward (in at least one transition) and time accumulated along a run initiated in v before visiting v' (note that every v' is eventually visited by a run initiated in v with probability one). For $\kappa' > 0$ we set

- $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max} \overset{\text{def}}{=} \sup_{v,v' \in V, \sigma, \mathcal{M}'} \mathbb{E}^{\sigma}_{\mathcal{M}'(v)[v']}[\mathrm{TR}]$,
- $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max} \overset{\text{def}}{=} \sup_{v,v' \in V, \sigma, \mathcal{M}'} \mathbb{E}^{\sigma}_{\mathcal{M}'(v)[v']}[\mathrm{Time}]$, and
- $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min} \overset{\text{def}}{=} \inf_{v,v' \in V, \sigma, \mathcal{M}'} \mathbb{E}^{\sigma}_{\mathcal{M}'(v)[v']}[\mathrm{Time}]$,

where σ ranges among all policies of $\mathcal{M}$ and $\mathcal{M}'$ ranges among all κ'-approximations of $\mathcal{M}$.

Lemma 6.2.2. *Let $\mathcal{M}$ be a strongly connected POMDP, σ be a policy such that $\mathcal{M}$ stays strongly connected when the set of actions is restricted to those selected by σ. Let $\mathcal{M}'$ be a κ-approximation of $\mathcal{M}$ and $\kappa' > 0$. For every error $\varepsilon > 0$, it holds that $\left|\mathbb{E}^{\sigma}_{\mathcal{M}}[\mathrm{MP}] - \mathbb{E}^{\sigma}_{\mathcal{M}'}[\mathrm{MP}]\right| \leq \varepsilon$ if*

$$\kappa \leq \min\left\{\frac{(\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min})^2 \cdot \frac{\varepsilon}{n}}{w_{max} \cdot (2 + \frac{\varepsilon}{n}) \cdot (1 + n \cdot w_{max})}, \kappa'\right\},$$

where $w_{\max} = \max\left\{\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}, \mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max}\right\}$, and n is the number of vertices of $\mathcal{M}$.

Proof. Let $\mathcal{M} = (V, Obs, O, Act, T, t, c, v_{\mathrm{in}}, V')$ be a POMDP where $V = \{v_1, \ldots, v_n\}$ and σ be some policy for $\mathcal{M}$. Note that every κ-approximation $\mathcal{M}'$ of $\mathcal{M}$ can be obtained via a sequence of POMDPs $\mathcal{M}_1, \ldots, \mathcal{M}_{n+1}$ where $\mathcal{M}_1 = \mathcal{M}$, $\mathcal{M}_{n+1} = \mathcal{M}'$, and every $\mathcal{M}_{i+1}$ is obtained from $\mathcal{M}_i$ by modifying only the values of $T(v_i, b)$, $t(v_i, b)$, and $c(v_i, b)$ with $b \in Act$. Note that, the bounds $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}, \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}, \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min} \in \mathbb{Q}_{>0}$ for $\kappa' = 0$ may not hold for all $\mathcal{M}_i$ due to the changes in the previous POMDPs. Hence, we require $\kappa \leq \kappa'$ and then $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$, and $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min}$ correctly bound the corresponding quantities of each POMDP $\mathcal{M}_i$ for $1 \leq i \leq n+1$. Thus, it suffices to construct $\kappa \in \mathbb{Q}_{>0}$ such that $\kappa \leq \kappa'$ and the expected mean payoff obtained by applying σ to $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ differs by at most ε/n.

Now, we discuss one of the changes, say in $v_i = v$, i.e., we omit the indexes for the sake of simplicity. As $\mathcal{M}$ is strongly connected, the mean-payoff value $\mathbb{E}^{\sigma}_{\mathcal{M}}[\text{MP}]$ achieved by σ in $\mathcal{M}$ can be expressed as

$$\mathbb{E}^{\sigma}_{\mathcal{M}}[\text{MP}] = \frac{\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v]}[\text{TR}]}{\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v]}[\text{Time}]}. \tag{6.2}$$

To use Lemma 6.2.1, we need to provide an upper bound on the numerator and upper and lower bounds on the denominator. From definition of $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min}$, and κ' it holds that

- $\mathbb{E}^{\sigma}_{\mathcal{M}(v')[v'']}[\text{TR}] \;\leq\; \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$, and
- $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min} \;\leq\; \mathbb{E}^{\sigma}_{\mathcal{M}(v')[v'']}[\text{Time}] \;\leq\; \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$.

Moreover, according to Lemma 6.2.1 we need to bound the changes in the numerator and denominator. For this purpose, observe that

$$\begin{aligned}\mathbb{E}^{\sigma}_{\mathcal{M}}[\text{MP}] &= \frac{\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v]}[\text{TR}]}{\mathbb{E}^{\sigma}_{\mathcal{M}(v)[v]}[\text{Time}]} = \\ &= \frac{c(v,b) + \sum_{v' \in V\setminus\{v\}} T(v,b)(v') \cdot \mathbb{E}^{\sigma}_{\mathcal{M}(v')[v]}[\text{TR}]}{t(v,b) + \sum_{v' \in V\setminus\{v\}} T(v,b)(v') \cdot \mathbb{E}^{\sigma}_{\mathcal{M}(v')[v]}[\text{Time}]}\end{aligned} \tag{6.3}$$

where $b = \sigma(O(v))$. Note that for $v' \neq v$, $\mathbb{E}^{\sigma}_{\mathcal{M}(v')[v]}[\text{TR}]$ and $\mathbb{E}^{\sigma}_{\mathcal{M}(v')[v]}[\text{Time}]$ do not depend on the $T(v,\cdot)$, $t(v,\cdot)$, and $c(v,\cdot)$. Hence, these values do not change when modifying $T(v,b)$, $t(v,b)$, and $c(v,b)$ for $b \in Act$, and we can treat them as constants. If $T(v,b)(v')$, $t(v,b)$, and $c(v,b)$ change at most by κ for $b = \sigma(O(v))$, then the numerator and the denominator of the above fraction (6.3) change at most by $\kappa + n \cdot \kappa \cdot \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$ and $\kappa + n \cdot \kappa \cdot \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$, respectively. I.e., we use maximum to bound both of them. By Lemma 6.2.1, the error of the fraction is bounded by ε/n if

$$\kappa + n \cdot \kappa \cdot w_{max} \;\leq\; \frac{(\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min})^2 \cdot \frac{\varepsilon}{n}}{\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max} + \mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max} \cdot (1 + \frac{\varepsilon}{n})},$$

where $w_{max} = \max\left\{\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}, \mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max}\right\}$, what we can strengthen to

$$\kappa + n \cdot \kappa \cdot w_{max} \leq \frac{(\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min})^2 \cdot \frac{\varepsilon}{n}}{w_{max} \cdot (2 + \frac{\varepsilon}{n})},$$

i.e.,

$$\kappa \leq \frac{(\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min})^2 \cdot \frac{\varepsilon}{n}}{w_{max} \cdot (2 + \frac{\varepsilon}{n}) \cdot (1 + n \cdot w_{max})}.$$

Using the bounds of the original POMDP and the above-mentioned restriction on κ, we obtain

$$\kappa \leq \min\left\{\frac{(\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min})^2 \cdot \frac{\varepsilon}{n}}{w_{max} \cdot (2 + \frac{\varepsilon}{n}) \cdot (1 + n \cdot w_{max})}, \kappa'\right\}$$

where $w_{max} = \max\left\{\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}, \mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max}\right\}$. $\square$

Now, it remains to correctly bound $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max}$, and $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min}$, what is done in the next lemma. Note that in this lemma, we use only the bounds specified in Assumption A for $s \in S_{\mathrm{set}}$. Bounds for other states can be computed accordingly. Hence, Lemma 6.2.3 finishes the proof that Assumption 1 holds if Assumptions A and B are satisfied.

Lemma 6.2.3. *Let $\mathcal{M}$ be a POMDP that stays strongly connected when using arbitrary policy. If $\kappa' \leq \min\{T_{\min}/2, t_{\min}/2, c_{\max}\}$, then*

$$\begin{aligned}\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min} &\geq t_{\min}/2,\\ \mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max} &\leq 2 \cdot t_{\max} \cdot n \cdot (2/T_{\min})^n, \text{ and}\\ \mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max} &\leq 2 \cdot c_{\max} \cdot n \cdot (2/T_{\min})^n,\end{aligned}$$

where n is the number of vertices of $\mathcal{M}$ and $T_{\min}, t_{\min}, t_{\max}, c_{\max} \in \mathbb{Q}_{>0}$ are bounds on the minimal probability, minimal and maximal time step, and maximal rewards occurring in $\mathcal{M}$, respectively.

Proof. Note that for each POMDP $\mathcal{M}' = (V, Obs, O, Act, T, t, c, v_{\mathrm{in}}, V')$ that is κ'-approximation of $\mathcal{M}$, it holds that

- $T(v,b)(v') > 0$ implies $T(v,b)(v') \geq T_{\min}/2$,
- $t_{\min}/2 \leq t(v,b) \leq 2t_{\max}$,
- $c(v,b) \leq 2c_{\max}$

for all $v, v' \in V$, $o \in O(v)$ and $b \in Act_o$.

The first inequality of the Lemma follows from the fact that for each POMDP $\mathcal{M}'$, its vertices $v, v' \in V$, and policy σ of $\mathcal{M}'$, all the paths from v to v' make at least one step. This step has minimal expected time $t_{\min}/2$. Using Lemma 6.1.2 we get the other two inequalities. □

Hence, we can conclude with the following corollary.

Corollary 6.2.4. *If Assumptions A and B are fulfilled, then also Assumption 1 is fulfilled.*

6.2.1 Heuristic for Larger κ

By combining Lemma 6.2.2 and Lemma 6.2.3 we get that $1/\kappa$ is exponential in the number of states $n = |S_{\text{set}} \cup S_{\text{off}}|$. This comes from the fact that we used the worst-case transition structure for computing the bounds in Lemma 6.2.3. However, we can use the algorithm for synthesis of ε-optimal policies for expected total reward to compute much better bounds on $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$, and $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min}$ that takes into account the transition structure of $\mathcal{N}$ instead of the worst-case one. We now provide the heuristic with justification of its correctness for computation of bound on $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{TR}]_{\max}$ using synthesis algorithm for expected total reward (the approach can be easily adapted also for obtaining bounds on $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\max}$ and $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\text{Time}]_{\min}$):

Let us fix a parametric ACTMC $\mathcal{N}$ with a reward structure and set $\varepsilon = 1$. We will always maximize the expected total reward, thus we will not state it explicitly in the following text. First, we create $\mathcal{N}'$ by making all alarms in $\mathcal{N}$ localized. Observe that for each $s \in S_{\text{set}} \cup S_{\text{off}}$ the MDP $\mathcal{M}_{\mathcal{N}'}(s)[s]$ is without unreachable states, thus the ε-maximal policy σ_s (in expected total reward) for $\mathcal{M}_{\mathcal{N}'}(s)[s]$ is ε-maximal for $\mathcal{M}_{\mathcal{N}'}(s')[s]$ and each $s' \in S_{\text{set}} \cup S_{\text{off}}$ (using the same ideas as in Section 6.1.1). Now we compute $\mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s)[s]}[\text{TR}]$ and obtain $\mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s')[s]}[\text{TR}]$ for each $s' \in S_{\text{set}} \cup S_{\text{off}}$ since the usual polynomial-time methods return correct answer for an arbitrary initial state. We can perform the same computation for each possible goal state. It follows that $\sup_{s',s \in S_{\text{set}} \cup S_{\text{off}}, \sigma} \mathbb{E}^{\sigma}_{\mathcal{M}_{\mathcal{N}}(s')[s]}[\text{TR}] \leq \max_{s',s \in S_{\text{set}} \cup S_{\text{off}}} \mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s')[s]}[\text{TR}] + \varepsilon$, where σ ranges among all polices of $\mathcal{M}_{\mathcal{N}}$. Thus we have a bound on expected total reward for $\mathcal{M}_{\mathcal{N}}$ but we need a bound for an arbitrary $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}]_{\kappa'}$. Observe that in each computation of ε-maximal policy for $\mathcal{M}_{\mathcal{N}'}(s)[s]$ where $s \in S_{\text{set}} \cup S_{\text{off}}$, we computed some δ_s and κ_s and looked for maximal policy for some $\mathcal{M} \in \left[\mathcal{M}_{\mathcal{N}'(s)[s]}\langle\delta_s\rangle\right]_{\kappa_s}$. From construction of δ_s and κ_s it holds that $\left|\mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s')[s]}[\mathrm{TR}] - \mathbb{E}^{\sigma}_{\mathcal{M}(s')}[\mathrm{TR}]\right| \leq \varepsilon$, where σ is the maximal policy of $\mathcal{M}$ and $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$. Moreover, from the way, how we obtained κ_s (see Section 6.1) it holds that $\left|\mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s')[s]}[\mathrm{TR}] - \mathbb{E}^{\sigma'}_{\mathcal{M}'(s')}[\mathrm{TR}]\right| \leq \varepsilon$, where $\mathcal{M}' \in [\mathcal{M}_{\mathcal{N}'}(s)[s]]_{\kappa_s}$, σ' is the maximal policy of $\mathcal{M}'$, and $s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}$. Hence, if κ' of Lemma 6.2.2 is less than $\min_{s \in S_{\mathrm{set}} \cup S_{\mathrm{off}}} \kappa_s$ then

$$\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max} \leq \max_{s,s' \in S_{\mathrm{set}} \cup S_{\mathrm{off}}} \mathbb{E}^{\sigma_s}_{\mathcal{M}_{\mathcal{N}'}(s')[s]}[\mathrm{TR}] + 2 \cdot \varepsilon.$$

Observe, that we need to run the above heuristic only for states in S_{set}, because following the proof of Lemma 6.2.2 for states in S_{off} we do not have to make any discretization or κ-approximation (the associated effects of $\mathcal{M}_{\mathcal{N}}$ are rational for $s \in S_{\mathrm{off}}$). Thus we need to run $3 \cdot |S_{\mathrm{set}}|$ times the synthesis algorithm for the expected total reward in localized ACTMC to compute better bounds on $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{TR}]_{\max}$, $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\max}$, and $\mathbb{E}^{\kappa'}_{\mathcal{M}}[\mathrm{Time}]_{\min}$. All the computations can be performed using the symbolic algorithm, which is very efficient. Thus this heuristic is very useful in practice, especially for ACTMC with non-localized alarms, where the explicit algorithm (and building large POMDP) is necessary to get a correct result.

6.3 Dirac Distributions

We start with showing that the Assumptions 2, 3, 5, A, and B are fulfilled for Dirac distributions. The results will then be used in the analysis of other distributions as well. We assume a fixed $s \in S_{\mathrm{set}}$ such that $s \in S_a$ where $F_a[d](\tau) = 1$ for all $\tau \geq d$ and $F_a[d](\tau) = 0$ for all $\tau < d$, and where $d \in [\ell_a, u_a] \subset (0, \infty)$.[1]

Note that for Dirac distributions, we can easily obtain Π_s, $€_s$, and Θ_s by employing uniformization. We define the set of states S_s that contains s and all states from S_a that are reachable from s using delay transitions and without passing through $S \setminus S_a$. Now we define the uniformization rate Λ_s as the maximal exit rate of states in S_s, i.e., $\Lambda_s \overset{\mathrm{def}}{=} \max_{s' \in S_s} \lambda_{s'}$. Using the uniformization rate, we can define the probability matrix $P_s \colon S \times S \to [0, 1]$

1. Note that we need to restrict ℓ_a and u_a to work with correct parametric ACTMC.

as

$$P_s(s', s'') \overset{\text{def}}{=} \begin{cases} Q(s', s'')/\Lambda_s & \text{if } s' \in S_s,\, s'' \in S \text{ and } s' \neq s'', \\ 1 - \sum_{s''' \in S} Q(s', s''')/\Lambda_s & \text{if } s', s'' \in S_s \text{ and } s' = s'', \\ 1 & \text{if } s', s'' \in S \setminus S_s \text{ and } s' = s'', \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, all states in S_{off} are absorbing in P_s. Then, using uniformization approach we easily get

$$\Pi_s(d) \quad = \quad \sum_{i=0}^{\infty} e^{-\Lambda_s d} \frac{(\Lambda_s d)^i}{i!} \cdot \left(\mathbf{1}_s \cdot P_s^i\right) \cdot P_a$$

where i represents the number of delay transitions fired before the alarm timer rings (i.e., time d), $e^{-\Lambda_s d} \frac{(\Lambda_s d)^i}{i!}$ is the corresponding probability according to the Poisson distribution, $\mathbf{1}_s \cdot P_s^i$ is the probability distribution over states after i delay transitions, and P_a is the probability matrix of the alarm a.

Similarly, to compute $€_s$ using uniformization approach, we first have to define some additional notions. Let $\boldsymbol{\mathcal{R}}_s \colon S \to \mathbb{Q}_{\geq 0}$ be a vector such that

$$\boldsymbol{\mathcal{R}}_s(s') \overset{\text{def}}{=} \begin{cases} \mathcal{R}(s') + \mathcal{I}(s', s') \cdot Q(s', s') & \text{if } s' \in S_s \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, after applying uniformization, the impulse rewards on self loops may need to be recomputed, since the self loop may get a higher rate. To avoid complicated recomputation we add the proportional amount of self-loop reward to the rate reward. More formally, the self loop in state $s' \in S_s$ has rate $Q(s', s')$, thus the self loop occurs $Q(s', s')$ times per time unit. Since in each occurrence the self loop generates reward $\mathcal{I}(s', s')$ we need to add $\mathcal{I}(s', s') \cdot Q(s', s')$ to the rate reward of state $s' \in S_s$. Moreover, we define vectors $\boldsymbol{\mathcal{I}}_s, \boldsymbol{\mathcal{I}}_a \colon S \to \mathbb{Q}_{\geq 0}$ as follows

$$\boldsymbol{\mathcal{I}}_s(s') \overset{\text{def}}{=} \begin{cases} \sum_{s'' \in S, s'' \neq s'} \mathcal{I}(s', s'') \cdot Q(s', s'')/\Lambda_s & \text{if } s' \in S_s \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\boldsymbol{\mathcal{I}}_a(s') \overset{\text{def}}{=} \begin{cases} \sum_{s'' \in S} \mathcal{I}_A(s', s'') \cdot P_a(s', s'') & \text{if } s' \in S_s \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Using the above notions we get the following lemma.

Lemma 6.3.1. *For any state $s \in S_{\text{set}}$ and $d \in \mathbb{R}_{>0}$ it holds that*

$$\text{€}_s(d) = \sum_{i=0}^{\infty} e^{-\Lambda_s d} \frac{(\Lambda_s d)^i}{i!} \bigg(\sum_{j=0}^{i-1} \big(\mathbf{1}_s \cdot P_s^j\big) \cdot \left(\frac{d \cdot \boldsymbol{\mathcal{R}}_s}{i+1} + \boldsymbol{\mathcal{I}}_s \right) + \big(\mathbf{1}_s \cdot P_s^i\big) \cdot \left(\frac{d \cdot \boldsymbol{\mathcal{R}}_s}{i+1} + \boldsymbol{\mathcal{I}}_a \right) \bigg).$$

Proof. Indeed, the distribution over states after i delay transitions is $\mathbf{1}_s \cdot P_s^i$, the probability that exactly i exponential transitions occur before time d is $e^{-\Lambda_s d}(\Lambda_s d)^i/i!$, and under the condition that i transitions occur, the expected time spent in each of the $i+1$ visited states is $d/(i+1)$ [109]. □

Note that Θ_s is a special case of €$_s$ where $\mathcal{R}(s) = 1$ for all $s \in S$, and $\mathcal{I}, \mathcal{I}_A$ return zero for all arguments.

Although the functions $\Pi_s(\hat{\tau})$, €$_s(\hat{\tau})$, and $\Theta_s(\hat{\tau})$ are defined as infinite sums, for every $\hat{\tau} \in \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$, a $k \in \mathbb{N}$ can be effectively computed such that for all $s \in S_{\text{set}}$ and $d \leq \hat{\tau}$, the truncated versions of Π_s, €$_s$, and Θ_s, obtained by taking the first k summands of the corresponding defining series under-approximate the values of $\Pi_s(d)$, €$_s(d)$, and $\Theta_s(d)$ up to an absolute error of at most κ (note that $e^{-\Lambda_s d}(\Lambda_s d)^i/i!$ decreases exponentially while the rewards increase linearly). We restrict k to be always larger than $|S|$, thus we detect all non-zero probabilities and all distributions will have the same support. We use

$$\Pi_s^{\hat{\tau},\kappa}, \text{€}_s^{\hat{\tau},\kappa}, \text{ and } \Theta_s^{\hat{\tau},\kappa}$$

to denote these truncated versions that are κ-approximations for all values d up to $\hat{\tau}$. Note that, due to the factor $e^{-\Lambda_s d}$, the values of $\Pi_s^{\hat{\tau},\kappa}$, €$_s^{\hat{\tau},\kappa}$, and $\Theta_s^{\hat{\tau},\kappa}$ are still irrational even for rational d's. Nevertheless, these values can be effectively approximated by rational numbers up to an arbitrarily small error. Also note that the components of $\Pi_s^{\hat{\tau},\kappa}$ may not sum up to 1. When we need to understand $\Pi_s^{\hat{\tau},\kappa}$ as a distribution, the remaining probability is split evenly among the states where $\Pi_s^{\hat{\tau},\kappa}$ is positive.

We show that Assumptions A and B, as well as Assumptions 2, 3, and 5 are satisfied.

Assumption A. Deriving the bounds $\Pi_s^{\min}$, $\Theta_s^{\min}$, $\Theta_s^{\max}$, and €$_s^{\max}$ for $s \in S_{\text{set}} \cap S_a$ for some $a \in A$ is easy. We put

- $\Pi_s^{\min} = (P_{\min})^{n_s} \cdot \min\left\{\frac{e^{-\Lambda_s d}(\Lambda_s d)^k}{k!} : 0 \le k \le n_s, d \in \{\ell_a, u_a\}\right\}$
- $\Theta_s^{\min} = \int_0^{\ell_a} x \cdot \Lambda_s \cdot e^{-\Lambda_s x}\,\mathrm{d}x + \ell_a \cdot e^{-\Lambda_s \ell_a} = \frac{1-e^{-\Lambda_s \cdot \ell_a}}{\Lambda_s}$,
- $\Theta_s^{\max} = u_a$,
- $€_s^{\max} = \mathcal{R}_{\max} \cdot u_a + \mathcal{I}_{\max} \cdot (\Lambda_s\, u_a + 1)$,

where $n_s = |S_s|$, $P_{\min} = \min\{P_s(s', s''), P_a(s', s'') \colon s' \in S_s, s'' \in S\}$, $\mathcal{R}_{\max} = \max\{\boldsymbol{\mathcal{R}}_s(s') \colon s' \in S_s\}$ and $\mathcal{I}_{\max} = \max\{\boldsymbol{\mathcal{I}}_s(s'), \boldsymbol{\mathcal{I}}_a(s') \colon s' \in S_s\}$. The first bullet specifies the minimal probability of runs in the uniformized CTMC from s to arbitrary state. The second bullet bounds the minimal expected time spent in the uniformized CTMC for choice $d = \ell_a$. The third bullet is easy to see, and the forth bullet uses the knowledge that the mean number of transitions made in the uniformized CTMC until time d is $\Lambda_s d$. Note that although some of the defining expressions involve the function e^x, we can still effectively under-approximate their values by positive rationals. Much tighter bounds, can be easily obtained in polynomial time to the size of uniformized CTMC for s by using transient solution techniques for CTMCs in times ℓ_a and u_a.

Assumption B. To define an appropriate $\delta_{(s,\kappa)} \in \mathbb{Q}_{>0}$, first note that for every $s \in S_{\text{set}}$, $d \in [\ell_a, u_a]$, and every $\delta \in \mathbb{Q}_{>0}$ such that $d - \delta \in [\ell_a, u_a]$, we have that

- $|\Theta_s(d) - \Theta_s(d{-}\delta)| \le \delta$,
- $|€_s(d) - €_s(d{-}\delta)| \le \delta\mathcal{R}_{\max} + 2\Lambda_s\delta\mathcal{I}_{\max}$,
- $|\Pi_s(d)(s') - \Pi_s(d{-}\delta)(s')| \le \Lambda_s\delta$,

where $\mathcal{R}_{\max}$ and $\mathcal{I}_{\max}$ are as defined above. Note also that $\delta\mathcal{R}_{\max}$ bounds the change of the rate reward caused by changing d by δ, $\Lambda_s\delta$ bounds the change of the number of delay transitions in time d vs. time $d - \delta$, and $\Lambda_s\delta$ can be also used as a bound on the change of probabilities in the transient probability distribution in CTMCs with rate Λ_s after time d vs. $d - \delta$ (see, e.g., Theorem 2.1.1 of [125]). Hence, for a given $\kappa > 0$, we can set

$$\delta_{(s,\kappa)} \;=\; \min\left\{\kappa, \frac{\kappa}{\mathcal{R}_{\max} + 2\Lambda_s\mathcal{I}_{\max}}\right\}.$$

Assumption 2. Let $\hat{\tau} = u_a$. The functions Π_s^κ, Θ_s^κ, and $€_s^\kappa$ are defined as rational $\kappa/2$-approximations of $\Pi_s^{\hat{\tau},\kappa/2}$, $€_s^{\hat{\tau},\kappa/2}$, and $\Theta_s^{\hat{\tau},\kappa/2}$ that are later approximated (for which we use the remaining $\kappa/2$ error) by rational functions using truncated Taylor series for $e^{(\cdot)}$.

Assumptions 3 and 5. For $\hat{\tau} = u_a$, we put

$$\mathbf{\Pi}_s^\kappa = \Pi_s^{\hat{\tau},\kappa}, \quad \boldsymbol{€}_s^\kappa = €_s^{\hat{\tau},\kappa}, \quad \mathbf{\Theta}_s^\kappa = \Theta_s^{\hat{\tau},\kappa}, and$$

$$\boldsymbol{F}_s^\kappa[g,\mathbf{h}](d) \;=\; \boldsymbol{€}_s^\kappa(d) - g\cdot\mathbf{\Theta}_s^\kappa(d) + \mathbf{\Pi}_s^\kappa(d)\cdot\mathbf{h},$$

where $g \in \mathbb{Q}$ and $\mathbf{h}\colon S_{\text{set}} \cup S_{\text{off}} \to \mathbb{Q}$ are constant (cf. Equation ($\star$)). Observe that $\boldsymbol{€}_s^\kappa(d)$, $\mathbf{\Theta}_s^\kappa(d)$, and $\mathbf{\Pi}_s^\kappa(d)\cdot\mathbf{h}$ are of the form $e^{-\Lambda_s d}\cdot Poly(d)$, where $Poly(d)$ is a polynomial. Hence, the first derivative of $\boldsymbol{F}_s^\kappa[g,\mathbf{h}](d)$ also has the special form $e^{-\Lambda_s d}\cdot V_{s,g,\mathbf{h}}$ where $V_{s,g,\mathbf{h}}$ is a polynomial of one free variable. Since $e^{-\Lambda d}$ is positive for $d \in \mathbb{R}_{\geq 0}$, the derivative $e^{-\Lambda_s d}\cdot V_{s,g,\mathbf{h}}$ is zero iff $V_{s,g,\mathbf{h}}$ is zero. Thus, the roots of the derivative in the interval $[\ell_a, u_a]$ can be approximated by approximating the roots of $V_{s,g,\mathbf{h}}$, which is computationally easy (e.g., using tools such as MAPLE [19]). The Assumption 5 follows similarly as Assumption 3, but for each alarm a we need to choose the same uniformization rate Λ_a in all a-setting states, i.e., for all $s \in S_{\text{set}} \cap S_a$ we require $\Lambda_a = \Lambda_s$. Then the $\sum_{s\in S_{\text{set}}\cap S_a} \pi(s)\cdot\boldsymbol{F}_s^\kappa[g,\mathbf{h}](d)$ will have the form $e^{-\Lambda_a d}\cdot Poly(d)$. The Λ_a can be obtained by taking maximum of exit rates of all states in S_a, i.e., $\Lambda_a \overset{\text{def}}{=} \max_{s\in S_a}\lambda_s$.

6.4 Functions for Alarms with non-Dirac Distributions

The main idea repeatedly applied in the following subsections is that, e.g., the reward function $\Theta_s(d)$ for a non-Dirac continuous distribution $F_a[d]$ with a known density function $f_a[d]\colon \mathbb{R} \to \mathbb{R}_{\geq 0}$ can be computed by integration of the reward function Dirac-$\Theta_s(\cdot)$ for some Dirac distribution applying the density function as a measure. That is,

$$\Theta_s(d) = \int_{min_a}^{max_a} \text{Dirac-}\Theta_s(\tau)\cdot f_a[d](\tau)\,\mathrm{d}\tau, \tag{6.4}$$

where τ stands for the randomly chosen time according to the non-Dirac distribution and min_a and max_a bound the support of $f_a[d]$. Note that, for every (even non-continuous) distributions we can obtain $\Theta_s(d)$ by employing Lebesgue integral, i.e., let μ_a be the measure of $F_a[d]$, then

$$\Theta_s(d) = \int_{[min_a,max_a]} \text{Dirac-}\Theta_s(\tau)\;\mu_a(\mathrm{d}\tau).$$

In following subsections, we denote the following values computed for the Dirac distribution (see last section) Π_s, Θ_s, $€_s$, $\Pi_s^{\hat{\tau},\kappa}$, $\Theta_s^{\hat{\tau},\kappa}$, $€_s^{\hat{\tau},\kappa}$, $\Pi_s^{\min}$, $\Theta_s^{\min}$, $\Theta_s^{\max}$, and $€_s^{\max}$ by Dirac-Π_s, Dirac-Θ_s, Dirac-$€_s$, Dirac-$\Pi_s^{\hat{\tau},\kappa}$, Dirac-$\Theta_s^{\hat{\tau},\kappa}$, Dirac-$€_s^{\hat{\tau},\kappa}$, Dirac-$\Pi_s^{\min}$, Dirac-$\Theta_s^{\min}$ Dirac-$\Theta_s^{\max}$, and Dirac-$€_s^{\max}$, respective-ly.

6.5 Uniform Distributions

In a (continuous) uniform distribution there are two parameters, the lower bound on possible random values, say α, and the upper bound, say β. Let us note that using a smart construction, we can easily synthesize both of the parameters by the explicit and symbolic algorithms. Observe that the first parameter α can be modeled as an alarm with Dirac distribution (parametrized by α that we can synthesize) that subsequently enables a uniformly distributed alarm with the first parameter fixed to 0 and the second parameter, say β', that is also subject of synthesis. Then, the required parameters of the original uniform distribution are α and $\beta = \alpha + \beta'$. Note that the newly created uniformly distributed alarm may not be localized, what can rule out the applicability of our synthesis algorithm.

Therefore, in what follows we consider alarms that are *uniformly* distributed on a time interval starting in 0 with a parametrized length. That is, we assume a fixed $s \in S_{\text{set}} \cap S_a$ for some $a \in A$ where $F_a[d](\tau) = \frac{\tau}{d}$ for all $0 \leq \tau \leq d$, $F_a[d](\tau) = 0$ for all $\tau < 0$, and $F_a[d](\tau) = 1$ for all $\tau > d$, where $d \in [\ell_a, u_a] \subset (0, \infty)$.[2]

Assumption A. Note that, for each $d \in [\ell_a, u_a]$ the uniform distribution has its support at most on $[0, u_a]$. Moreover, Dirac-Θ_s, Dirac-$€_s$ are bounded from above by Dirac-$\Theta_s^{\max}$, Dirac-$€_s^{\max}$ on $[0, u_a]$. Thus also the integral expressions defining Θ_s and $€_s$ are bounded and we have:

- $\Theta_s^{\max} \overset{\text{def}}{=} \text{Dirac-}\Theta_s^{\max} = u_a$ and
- $€_s^{\max} \overset{\text{def}}{=} \text{Dirac-}€_s^{\max} = \mathcal{R}_{\max} \cdot u_a + \mathcal{I}_{\max} \cdot (\Lambda_s \, u_a + 1)$,

where $\mathcal{R}_{\max}$, $\mathcal{I}_{\max}$, and Λ_s are the same as in Section 6.3. Note that, for each $d \in [\ell_a, u_a]$, the uniform distribution has at least $1/2$ of the probability on $[\ell_a/2, u_a]$. Thus, we can use half of Dirac-$\Pi_s^{\min}$ and Dirac-$\Theta_s^{\min}$ for Dirac distribution with delay in $[\ell_a/2, u_a]$:

2. Note that we need to restrict ℓ_a and u_a to work with a correct parametric ACTMC.

- $\Pi_s^{\min} = (P_{\min})^{n_s} \cdot \min\left\{\frac{e^{-\Lambda_s d}(\Lambda_s d)^k}{2\cdot k!} : 0{\leq}k{\leq}n_s, d \in \{\ell_a/2, u_a\}\right\}$ and
- $\Theta_s^{\min} = \frac{1-e^{-\Lambda_s \cdot \ell_a/2}}{2\Lambda_s}$,

where n_s and $P_{\min}$ are the same as in Section 6.3.

Assumption B. We assume that $\delta < d$ for each $d \in [\ell_a, u_a]$. First, consider the bound on $|€_s(d) - €_s(d{-}\delta)|$. Note that $1/d$ is the density of $F_a[d]$ on interval $[0, d]$. We have that

$$
\begin{aligned}
&|\text{Dirac-}€_s(d) - \text{Dirac-}€_s(d{-}\delta)| \ = \\
&= \left| \int_0^d \frac{\text{Dirac-}€_s(\tau)}{d}\,\mathrm{d}\tau - \int_0^{d-\delta} \frac{\text{Dirac-}€_s(\tau)}{d-\delta}\,\mathrm{d}\tau \right| \\
&= \int_0^d \frac{\text{Dirac-}€_s(\tau)}{d}\,\mathrm{d}\tau - \int_0^{d-\delta} \frac{\text{Dirac-}€_s(\tau)}{d-\delta}\,\mathrm{d}\tau \\
&\qquad\qquad\qquad\qquad\qquad\qquad\text{as Dirac-}€_s \text{ is nondecreasing} \\
&= \int_{d-\delta}^d \frac{\text{Dirac-}€_s(\tau)}{d}\,\mathrm{d}\tau + \int_0^{d-\delta} \frac{\text{Dirac-}€_s(\tau)}{d} - \frac{\text{Dirac-}€_s(\tau)}{d-\delta}\,\mathrm{d}\tau \\
&\leq \int_{d-\delta}^d \frac{\text{Dirac-}€_s(\tau)}{d}\,\mathrm{d}\tau \qquad\qquad\qquad \text{as } \tfrac{1}{d} - \tfrac{1}{d-\delta} \text{ is negative} \\
&\leq \frac{1}{d}\int_{d-\delta}^d \mathcal{R}_{\max}\cdot u_a + \mathcal{I}_{\max}\cdot(\Lambda_s\, u_a + 1)\,\mathrm{d}\tau \quad \text{as Dirac-}€_s(\tau) \leq \text{Dirac-}€_s^{\max} \\
&\leq \delta\cdot(\mathcal{R}_{\max}\cdot u_a + \mathcal{I}_{\max}\cdot(\Lambda_s\, u_a + 1))/\ell_a \qquad\qquad\qquad \text{as } d \geq \ell_a
\end{aligned}
$$

Similarly we can get a bound on $|\Theta_s(d) - \Theta_s(d{-}\delta)| \leq \delta \cdot u_a/\ell_a$. Now we consider a bound on $|\Pi_s(d)(s') - \Pi_s(d{-}\delta)(s')|$ for $s' \in S_{\text{set}} \cup S_{\text{off}}$:

$$
\begin{aligned}
&|\Pi_s(d)(s') - \Pi_s(d{-}\delta)(s')| \ = \\
&= \left| \int_0^d \frac{\text{Dirac-}\Pi_s(\tau)(s')}{d}\,\mathrm{d}\tau - \int_0^{d-\delta} \frac{\text{Dirac-}\Pi_s(\tau)(s')}{d-\delta}\,\mathrm{d}\tau \right| \\
&= \left| \int_{d-\delta}^d \frac{\text{Dirac-}\Pi_s(\tau)(s')}{d}\,\mathrm{d}\tau + \left(\frac{1}{d} - \frac{1}{d-\delta}\right)\cdot \int_0^{d-\delta} \text{Dirac-}\Pi_s(\tau)(s')\,\mathrm{d}\tau \right| \\
&= \left| \int_{d-\delta}^d \frac{\text{Dirac-}\Pi_s(\tau)(s')}{d}\,\mathrm{d}\tau - \frac{\delta}{d^2 - d\delta}\cdot \int_0^{d-\delta} \text{Dirac-}\Pi_s(\tau)(s')\,\mathrm{d}\tau \right| \\
&= |G - H|,
\end{aligned}
$$

where $G \overset{\text{def}}{=} \int_{d-\delta}^{d} \frac{\text{Dirac-}\Pi_s(\tau)(s')}{d}\,\mathrm{d}\tau$ and $H \overset{\text{def}}{=} \frac{\delta}{d^2-d\delta} \cdot \int_0^{d-\delta} \text{Dirac-}\Pi_s(\tau)(s')\,\mathrm{d}\tau$. Observe that $G \geq 0$ and $H \geq 0$. If $G < H$ then

$$\begin{aligned}
|\Pi_s(d)(s') - \Pi_s(d-\delta)(s')| \;\leq\; H \;&=\; \frac{\delta}{d^2 - d\delta} \cdot \int_0^{d-\delta} \text{Dirac-}\Pi_s(\tau)(s')\,\mathrm{d}\tau \\
&\leq\; \frac{\delta}{d^2 - d\delta} \cdot \int_0^{d-\delta} 1\,\mathrm{d}\tau \\
&=\; \frac{\delta}{d^2 - d\delta} \cdot (d-\delta) = \frac{\delta}{d} \leq \frac{\delta}{\ell_a}.
\end{aligned}$$

If $G \geq H$ then

$$\begin{aligned}
|\Pi_s(d)(s') - \Pi_s(d-\delta)(s')| \;\leq\; G \;&=\; \frac{1}{d} \int_{d-\delta}^{d} \text{Dirac-}\Pi_s(\tau)(s')\,\mathrm{d}\tau \\
&\leq\; \frac{1}{d} \int_{d-\delta}^{d} 1\,\mathrm{d}\tau = \frac{\delta}{d} \leq \frac{\delta}{\ell_a}.
\end{aligned}$$

Hence, we need

$$\max\left\{\delta \cdot \frac{\mathcal{R}_{\max} \cdot u_a + \mathcal{I}_{\max} \cdot (\Lambda_s\, u_a + 1)}{\ell_a}, \delta \cdot \frac{u_a}{\ell_a}, \frac{\delta}{\ell_a}\right\} \leq \kappa.$$

Thus, for a given $\kappa > 0$, we can set

$$\delta_{(s,\kappa)} \;=\; \min\left\{\frac{\kappa \cdot \ell_a}{\mathcal{R}_{\max} \cdot u_a + \mathcal{I}_{\max} \cdot (\Lambda_s\, u_a + 1)}, \frac{\kappa \cdot \ell_a}{u_a}, \kappa \cdot \ell_a, \ell_a\right\},$$

where the term ℓ_a is from our assumption that $\delta < d$ for each $d \in [\ell_a, u_a]$.

Assumptions 2, 3, and 5. We show that $€_s$ can be κ-approximated by a function $\boldsymbol{€}_s^{\kappa}(d)$ of the form $V(d)/d$, where $V(d)$ is a computable polynomial of rational coefficients. By using the same technique, we can construct also $\boldsymbol{\Theta}_s^{\kappa}$ and $\boldsymbol{\Pi}_s^{\kappa}$, which are of the same form. Hence, the function $\boldsymbol{F}_s^{\kappa}[g, \mathbf{h}](d) = \boldsymbol{€}_s^{\kappa}(d) - g \cdot \boldsymbol{\Theta}_s^{\kappa}(d) + \boldsymbol{\Pi}_s^{\kappa}(d) \cdot \mathbf{h}$ has also the same form, since g and $\mathbf{h}$ are constant. After differentiation of $\boldsymbol{F}_s^{\kappa}[g, \mathbf{h}](d)$ we get a function of form $V'(d)/d^2$, where we isolate roots of polynomial $V'(d)$ to obtain the candidate actions. Note that the denominator d^2 may disable root $d = 0$ that is not within the eligible parameter values. Generally, the correctness of the symbolic algorithms is not harmed by including candidate actions computed from the "false positive" root, since these actions do not correspond to values that are near extrema of $\boldsymbol{F}_s^{\kappa}[g, \mathbf{h}](d)$ and some other action of the candidate set must outperform all of them. Thus, the function $\boldsymbol{F}^{\kappa}_{s}[g,\mathbf{h}](d)$ fulfills Assumption 2 and 3. It is easy to see that Assumption 5 is also fulfilled.

Now we show that $€_s$ can be κ-approximated by a function $\boldsymbol{€}^{\kappa}_{s}(d)$ of the form $V(d)/d$, where $V(d)$ is a computable polynomial of rational coefficients. From definition, $€_s(d)$ equals to $\int_0^d 1/d \cdot \text{Dirac-}€_s(\tau)\,\mathrm{d}\tau$. Instead of Dirac-$€_s(\tau)$ we can use its $\kappa\ell_a/2$ approximation Dirac-$€_s^{u_a,\kappa/2}(\tau) = P(\tau)\cdot e^{-\Lambda_s\tau}$, where $P(\tau)$ is an univariate polynomial with rational coefficients. The main difficulty is the integration of $P(\tau)\cdot e^{-\Lambda_s\tau}$. Fortunately, we can use the Taylor series representation for $e^{-\Lambda_s\tau}$, i.e.,

$$e^{-\Lambda_s\tau} = \sum_{n=0}^{\infty} \frac{1}{n!}\cdot(-\Lambda_s\tau)^n.$$

Then, we can truncate the infinite sum, such that we cause at most $\kappa\ell_a/2$ error in $P(\tau)\cdot e^{-\Lambda_s\tau}$ for each $\tau\in[0,u_a]$ (observe that we can compute the maximal and minimal values of polynomial $P(\tau)$ for $\tau\in[0,u_a]$ using differentiation and root isolation of polynomials). Thus we obtain polynomial $P(\tau)\cdot S(\tau)$ with rational coefficients, that $\kappa\ell_a$-approximates Dirac-$€_s(\tau)$. Hence we can set

$$\boldsymbol{€}^{\kappa}_{s}(d) \;=\; \frac{1}{d}\cdot\int_0^d P(\tau)\cdot S(\tau)\;\mathrm{d}\tau = \frac{1}{d}\cdot V(d),$$

what is a κ-approximation of $€_s$ since $d\le\ell_a$ and we are done.

6.6 Weibull and Other Continuous Distributions

As we already noted the abstract assumptions enabling the applicability of our explicit and symbolic algorithms are satisfied also by other distributions. Since the arguments are increasingly more involved (and not used in our experiments reported in Chapter 7), we just sketch the arguments that show how to handle distributions with polynomial approximations and illustrate them on the Weibull distribution. Observe that exponential distribution is a special case of the Weibull distribution, where the fixed shape constant $k=1$.

In what follows we assume a fixed $s\in S_{\text{set}}\cap S_a$ for some $a\in A$ and $F_a[d](\tau)$ such that for each $d\in[\ell_a,u_a]$ the CDF $F_a[d](\tau)$ is continuous and differentiable for all $\tau>0$ and $F_a[d](\tau)=0$ for all $\tau\le 0$.

Assumption A. Let $f_a[d](\tau)$ be the derivative of $F_a[d](\tau)$, i.e., a probability density function. Note that for uniform distribution, we first bounded the support of the density function. Here the support of the density function can be whole $\mathbb{R}_{\geq 0}$ even for d bounded by ℓ_a and u_a. Hence, we first provide bounds min'_s and max'_s satisfying

$$\int_0^{\infty} f_a[d](\tau) \cdot \langle\ .\ \rangle \,\mathrm{d}\tau - \int_{min'_s}^{max'_s} f_a[d](\tau) \cdot \langle\ .\ \rangle \,\mathrm{d}\tau \ \leq\ \frac{1}{2},$$

for all $d \in [\ell_a, u_a]$ and $\langle\ .\ \rangle \in \{\text{Dirac-}\Pi_s(\tau), \text{Dirac-}\Theta_s(\tau), \text{Dirac-}€_s(\tau)\}$. To find such bounds we strengthen the condition by requiring $\langle\ .\ \rangle \in \{1,$ Dirac-$\Theta_s^{\max}$, Dirac-$€_s^{\max}\}$. We use Dirac-$\Theta_s^{\max} = \tau$ and Dirac-$€_s^{\max} = \mathcal{R}_{\max} \cdot \tau + \mathcal{I}_{\max} \cdot (\lambda\tau{+}1)$, where $\mathcal{R}_{\max}$ and $\mathcal{I}_{\max}$ are the same as in Section 6.3. Note that the bounds min'_s and max'_s exist because the density function has to decrease in its extrema faster than at most linearly increasing (in τ) functions Dirac-$\Theta_s^{\max}$ and Dirac-$€_s^{\max}$.

Having such bounds min'_s and max'_s, we can safely set

- $\Theta_s^{\max} = \text{Dirac-}\Theta_s^{\max} + 1/2 = max'_s + 1/2$ and
- $€_s^{\max} = \text{Dirac-}€_s^{\max} + 1/2 = \mathcal{R}_{\max} \cdot max'_s + \mathcal{I}_{\max} \cdot (\Lambda_s\, max'_s + 1) + 1/2.$

Let $c_s \overset{\text{def}}{=} \min_{d \in [\ell_a, u_a]} \int_{min'_s}^{max'_s} f_a[d](\tau)\,\mathrm{d}\tau$. Intuitively, c_s is the lower bound on probability that is assigned to numbers in $[min'_s, max'_s]$. One can easily bound c_s from below by $1/2$ or more exactly by

$$(1 - 1/(2 \cdot \max\{1, \text{Dirac-}\Theta_s^{\max}, \text{Dirac-}€_s^{\max}\})).$$

Using c_s we can set

$$\begin{aligned}\Pi_s^{\min} &= c_s \cdot \text{Dirac-}\Pi_s^{\min}\\ &= c_s \cdot (P_{\min})^{n_s} \cdot \min\left\{ \frac{e^{-\Lambda_s d}(\Lambda_s d)^k}{k!} \ :\ 0{\leq}k{\leq}n_s, d \in \{min'_s, max'_s\} \right\}\end{aligned}$$

and

$$\Theta_s^{\min} = c_s \cdot \text{Dirac-}\Theta_s^{\min} = c_s \cdot \frac{1 - e^{-\Lambda_s \cdot min'_s}}{2\Lambda_s},$$

where n_s and $P_{\min}$ are the same as in Section 6.3.

Assumptions 2, 3, and 5. Employing Assumption A, we can compute sufficiently small κ. Now, we define max_s to be bound such that

$$1 - \int_0^{max_s} f_a[d](\tau)\,\mathrm{d}\tau \ \leq\ \frac{\kappa}{2 \cdot \langle\ .\ \rangle}$$

for all $d \in [\ell_a, u_a]$ and $\langle\ .\ \rangle \in \{1, max_s, \mathcal{R}_{\max} \cdot \max_s + \mathcal{I}_{\max} \cdot (\lambda \max_s + 1)\}$, i.e., $\langle\ .\ \rangle$ is the same as above, but we use max_s instead of τ.

Now, let $U(\tau)$ be a polynomial approximation of $f_a[d](\tau) \cdot \text{Dirac-}\text{€}_s(\tau)$ whose error is bounded by $\kappa/(2 \cdot max_s)$ on interval $[0, max_s]$ and for each $d \in [\ell_a, u_a]$. Then $\boldsymbol{€}_s^\kappa = \int_0^{max_s} U(\tau)\,\mathrm{d}\tau$ is a κ-approximation of $€_s$ since

$$\left| \int_0^{max_s} f_a[d](\tau) \cdot \text{Dirac-}€_s(\tau)\,\mathrm{d}\tau - \int_0^{\infty} f_a[d](\tau) \cdot \text{Dirac-}€_s(\tau)\,\mathrm{d}\tau \right| \leq \frac{\kappa}{2},$$

what follows from the setting of max_s and since

$$\begin{aligned} &\left| \int_0^{max_s} f_a[d](\tau) \cdot \text{Dirac-}€_s(\tau)\,\mathrm{d}\tau - \int_0^{max_s} U(\tau)\,\mathrm{d}\tau \right| = \\ &= \left| \int_0^{max_s} f_a[d](\tau) \cdot \text{Dirac-}€_s(\tau) - U(\tau)\,\mathrm{d}\tau \right| \\ &\leq \left| \int_0^{max_s} \frac{\kappa}{2 \cdot max_s}\,\mathrm{d}\tau \right| = \frac{\kappa}{2}. \end{aligned}$$

Note that $U(\tau)$ is a polynomial with variable τ, but might not be polynomial in the parameter d. Thus, there is a question whether $\int_0^{max_s} U(\tau)\,\mathrm{d}\tau$ is differentiable for every $d \in [\ell_a, u_a]$. As a concrete example, let us consider a Weibull distribution with a fixed shape parameter $k \in \mathbb{N}$ and the scale parameter $1/d$ (i.e., we aim at synthesizing an ε-optimal scale). The Weibull density function is

$$kd \cdot (\tau d)^{k-1} \cdot e^{-(\tau d)^k}.$$

As we know from Section 6.3, $\text{Dirac-}€_s^{max_s,\kappa}(\tau) = P(\tau) \cdot e^{-\Lambda_s \tau}$, where $P(\tau)$ is a polynomial function and $\kappa > 0$. Note that both $e^{-(\tau d)^k}$ and $e^{-\Lambda_s \tau}$ can be approximated to an arbitrary precision for $d \in [\ell_a, u_a]$ and $\tau \in [0, max_s]$ using a finite prefix of the Taylor series

$$\sum_{n=0}^{\infty} \frac{1}{n!} \cdot (-(\tau d)^k - \Lambda_s \tau)^n.$$

Then, for some sufficiently high bound $i \in \mathbb{N}$,

$$U(\tau) \;\; = \;\; kd \cdot (\tau d)^{k-1} \cdot P(\tau) \cdot \sum_{n=0}^{i} \frac{1}{n!} \cdot (-(\tau d)^k - \Lambda_s \tau)^n$$

is a polynomial function of τ and hence, $\boldsymbol{€}_s^\kappa = \int_0^{max_s} U(\tau)\,\mathrm{d}\tau$ is a polynomial function of d. Similarly we can provide polynomial κ-approximations $\boldsymbol{\Pi}_s^\kappa$ and $\boldsymbol{\Theta}_s^\kappa$, thus Assumption 3 holds. It is easy to see that Assumption 5 also holds. Moreover, $\boldsymbol{\Pi}_s^\kappa$, $\boldsymbol{\Theta}_s^\kappa$, and $\boldsymbol{€}_s^\kappa$ are computable for each $d \in [\ell_a, u_a]$, thus Assumption 2 is also fulfilled.

Assumption B. Let us fix $s \in S_{\text{set}} \cap S_a$ and $\kappa \in \mathbb{Q}_{>0}$. We will show how to compute a discretization bound $\delta_{(s,\kappa,€)} \in \mathbb{Q}_{>0}$ such that for every $d, d' \in [\ell_a, u_a]$ where $|d - d'| \le \delta_{(s,\kappa,€)}$ it holds that

$$|€_s(d) - €_s(d')| \le \kappa.$$

First, we set $\kappa' \overset{\text{def}}{=} \kappa/3$. Using κ', we κ'-approximate $€_s$ function by polynomial $\boldsymbol{€}_s^{\kappa'}$. Then we use $\boldsymbol{€}_s^{\kappa'}$ to obtain safe bounds on its first derivative using its second derivative and root isolation on interval $[\ell_a, u_a]$. We use these bounds to obtain sufficiently small $\delta_{(s,\kappa,€)}$ to cause at most κ' error. Then for every $d, d' \in [\ell_a, u_a]$ where $|d - d'| \le \delta_{(s,\kappa,€)}$ it holds that

$$|€_s(d) - €_s(d')| \le 2\kappa' + |\boldsymbol{€}_s^{\kappa'}(d) - \boldsymbol{€}_s^{\kappa'}(d')| \le 3\kappa' = \kappa.$$

We can similarly define and obtain bounds $\delta_{(s,\kappa,\Pi)}$ and $\delta_{(s,\kappa,\Theta)}$. Then $\delta_{(s,\kappa)} \overset{\text{def}}{=} \min\{\delta_{(s,\kappa,\Pi)}, \delta_{(s,\kappa,\Theta)}, \delta_{(s,\kappa,€)}\}$.

Chapter 7

Experimental Evaluation

In this chapter we deal with implementations of our synthesis algorithms and their evaluation on practical examples. The implementation was carried out in PRISM model checker and mathematical tool MAPLE, that is needed to isolate roots of polynomials of high degree with high precision. One of the products of the implementation is an extension of PRISM model checker that supports specification and verification of fdCTMC, and some of our synthesis algorithms (applicable to only for subclass of ACTMCs and fdCTMCs). The extension is still actively developed to support specification and verification of ACTMCs and larger classes of models.

We performed the experimental evaluation on four practical examples taken from the literature and PRISM case studies. All of the models are scalable, thus we can make several problem instances for differently scaled models and error bounds. The observations from the experimental evaluation can be summarized as follows. First, the explicit algorithms are feasible only on instances of our problems that result in relatively large discretization bounds, because otherwise a very large amount of memory is required to store all the discretized actions. Contrary, our symbolic algorithms are feasible for all problem instances listed in this thesis. Moreover, when evaluating our symbolic algorithms we found out that the amount of needed memory is very small, the symbolic algorithms always outperform the explicit ones (in some cases by three orders of magnitude), and their running time grows slowly in the size of our instances.

In the following we first dedicate a section to describe the experimental evaluation of the synthesis algorithms for the expected total reward, then we do the same for the expected mean payoff. Both sections contain information about the implementation, models used for the evaluation, results of the experiments, and discussion about the obtained results. Finally, we provide a section that introduces the currently developed tool that will support our synthesis algorithms among with verification algorithms for ACTMCs and larger classes of models.

7.1 Expected Total Reward

The implementation of synthesis algorithms for the expected total reward was carried out gradually as we were developing more efficient synthesis algorithms presented in our papers. We first introduced the explicit algorithm in [32], then we optimized and implemented it along some other features [100], and based on the facts discovered during the implementation and prototyping we developed and implemented the symbolic algorithm [99]. All of these results were for fdCTMC models with at most one concurrently active non-exponential event and for minimization of the expected total reward. This thesis extends the total-reward synthesis results to ACTMCs and to maximization of the expected total reward. The implementation of the optimized explicit algorithm was part of a larger extension of the PRISM probabilistic model checker [100]. Thus also the symbolic algorithm was implemented in PRISM to provide the most faithful comparison of both algorithms.

PRISM is currently the most well known and used tool for verification of discrete- and continuous-time Markov chains and Markov decision processes. We chose PRISM, mainly because it offers very efficient algorithms for performing transient analysis in CTMCs and algorithms to find total-reward optimal policies in MDPs that we need for the explicit algorithm. Moreover, the PRISM community frequently raises requests to incorporate the possibility to express delays with deterministic durations in a CTMC.[1] The standard PRISM recommendation is to approximate the deterministic durations using a phase-type technique [124] and thus obtaining a CTMC. This works for some models, however there are models for which such approximation can cause either a large error or a state space explosion (see, e.g., [63, 98]). The fdCTMC formalism [32, 70, 98] is the requested extension of CTMCs by Dirac-distributed events that model (with zero error) the deterministic transitions or timeouts. Thus we decided to extend PRISM by new model type fdCTMC and not just provide a stand-alone prototype implementation of our synthesis algorithms.

The features of the PRISM extension are as follows.

- We provided an extension of the PRISM language and of the internal data structures to support specification of fdCTMC with impulse and rate rewards (or equivalently costs). Hence, our version of PRISM

1. `http://www.prismmodelchecker.org/manual/FrequentlyAskedQuestions/PRISMModelling#det_delay`

is ready for other experiments with fdCTMC algorithms including the possibility to support model-checking options as for CTMCs and DTMCs.

- We added an evaluation of expected total reward until reaching a given set of goal states.
- We implemented the explicit and symbolic algorithm for minimization of expected total reward.[2]

For more details on extension of PRISM language and structures please refer to [100].

Explicit Algorithm. The implementation of the explicit synthesis algorithm works as pseudo-code in Algorithm 2 on ACTMCs with localized alarms. First, a finite discretized MDP is built using the optimizations reported later, and then this MDP is solved by the standard MDP algorithms of PRISM. Currently there are three solution methods available for computing an optimal policy for total reachability reward in a finite MDP: policy iteration, value iteration, and Gauss-Seidl value iteration. The policy iteration has been identified as the fastest one.

Symbolic Algorithm. The prototype implementation of the symbolic algorithm is implemented as part of our PRISM extension and works as Algorithm 5. We tested several libraries and tools for isolating real roots of polynomials (Apache Commons, Matlab, MAPLE, Sage, etc.). The best performance was achieved by MAPLE [19] since it was the only one that could isolate roots with good enough precision. We decided to use this software in our implementation. Currently, we call MAPLE directly from Java, providing the polynomial and the required precision for the roots. We measure the CPU time for all MAPLE calls and add it to the final CPU time. Because MAPLE is proprietary our implementation is not yet suited for public use (we will provide it on demand). We are implementing a new Java library for isolating roots of univariate polynomial with arbitrary precision (what is to the best of our knowledge available only in MAPLE) and appropriately re-implement the symbolic algorithm.

All the computations were run on platform HP DL980 G7 with 8 64-bit processors Intel Xeon X7560 2.26GHz (together 64 cores) and 448 GiB DDR3

2. Currently we allow only models with localized alarms.

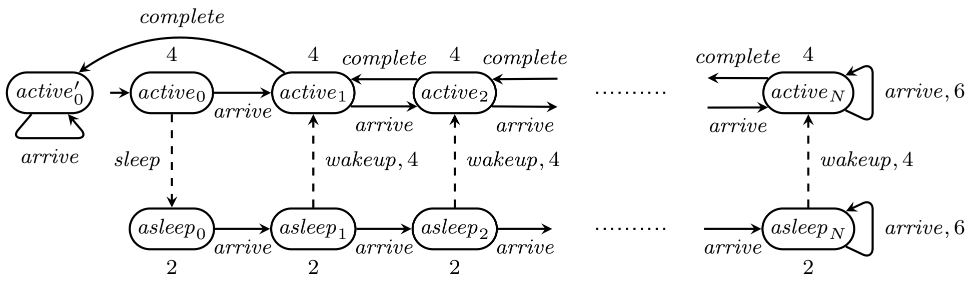


Figure 7.1: Dynamic power manager of a disk drive.

N	CPU time [s]						Exp. total costs
	ε :	0.005	0.0025	0.0016	0.00125	0.00100	
	$1/\varepsilon$:	200	400	600	800	1000	
2		17.29	36.46	58.05	86.73	98.63	0.3364
4		37.07	76.60	133.88	944.08	1189.89	0.3375
6		52.05	132.18	1100.78	1336.70	1519.26	0.3375
8		115.95	1252.82	2321.93	3129.16	3419.42	0.3375

Table 7.1: Running times of the explicit algorithm for the disk drive model.

RAM. The time and space was measured by the Linux command `time`. For all experiments in this section we used method to compute lower and upper bound of eligible parameter values for each parameter [32]. The method is applicable only for minimization of expected total reward and it is very complicated to extend it for maximization of expected total reward or for optimization of expected mean payoff.

7.1.1 Disc Drive Model

We tested the explicit and symbolic algorithms on ACTMC of Figure 7.1 that is slightly changed model from Example 1.0.1. All constants that do not appear in the Figure 7.1 are the same as in Example 1.0.1. We are interested in finding parameters that minimize the expected total cost until emptying the queue.

Table 7.1 shows the time needed to compute an ε-minimal parameter function by the explicit algorithm for the model of Figure 7.1 where $N = 2, 4, 6, 8$ and ε is progressively smaller. The obtained expected total costs

N	CPU time [s]						Exp. total costs
	ε :	0.005	0.0025	0.0016	0.00125	0.00100	
	$1/\varepsilon$:	200	400	600	800	1000	
2		2.22	2.34	2.34	2.39	2.42	0.3364
4		2.37	2.38	2.40	2.37	2.38	0.3375
6		2.39	2.39	2.43	2.39	2.42	0.3375
8		2.40	2.42	2.44	2.46	2.44	0.3375

Table 7.2: Running times of the symbolic algorithm for the disk drive model.

agree (on the required precision) among all instances with the same queue size and are provided in the last column. There is a visible gap separating CPU times smaller than 200 and higher than 900 in Table 7.1. This is caused by the garbage collector of Java. Despite our effort to eliminate the influences of garbage collector to our experiments, we could not entirely rule them out, since the generated MDPs are large. E.g., for the largest model (where $N = 8$ and error is 0.001) we needed 148GiB of memory to store the MDP.

Equivalent results for the symbolic algorithm are provided in Table 7.2. The obtained expected total reachability costs are the same as for explicit algorithm results reported in Table 7.1. The symbolic algorithm performs significantly better, especially for smaller ε where the action space of the associated MDP $\mathcal{M}$ is already quite large. This is because the solution of polynomials in MAPLE is extremely fast and that policy iteration performs small number of iterations (3 for all instances). Even the total number of isolated roots thorough all iterations is a small (3 for all instances). The largest degree of polynomials varies from 73 (for $N = 2$ and error 0.005) to 91 (for $N = 8$ and error 0.001).

We could not run the experiments for significantly smaller ε or larger N since the required precisions κ were stricter than the precision of double. The performance of both algorithms on large models is illustrated in the next subsection.

7.1.2 Communication Protocol Model

We first explain a practical non-trivial problem in Example 7.1.1 and then we will address a concrete model that will be used for the experimental evaluation.

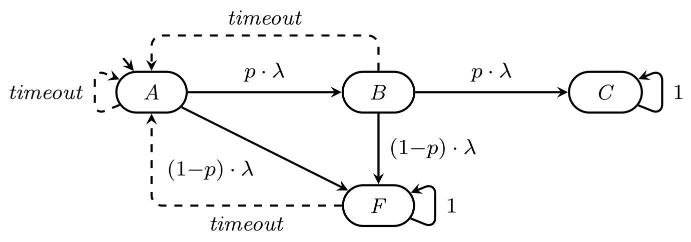


Figure 7.2: A simplified ACTMC model of a communication protocol.

Example 7.1.1. *Consider a simple communication protocol where Alice tries to establish a connection with Bob via an unreliable communication channel. Alice starts by sending an* Invite *message to Bob, and then she waits for Bob's* Ack *message. Since each of these messages can be lost, Alice sets a timeout after which she restarts the protocol and sends another* Invite *(the* Ack *messages confirming a successful receipt of a "previous"* Invite *are recognized and ignored). The protocol terminates when a connection is established, i.e., both messages are delivered successfully before the timeout. The behavior of the unreliable channel is stochastic; a message is successfully delivered with a (known) probability p, and the delivery time has a (known) distribution $Dtime$. A simplified ACTMC model of the protocol is given in Figure 7.2. The delay and alarm transitions of a Dirac-distributed alarm $timeout$ are indicated by solid and dashed arrows, respectively. A faithful modeling of the $Dtime$ distribution using the phase-type approximation requires extra auxiliary states which are omitted in Figure 7.2 for the sake of simplicity. Hence, the simplified model corresponds to the situation when $Dtime$ is the exponential distribution with rate λ. Observe that the transition depicted in Figure 7.2 from states A or B are equivalent to an alarm transition with exponential distribution with rate λ and probability $(1-p)$ to reach state F and probability p to reach state B or C, respectively.*

Note that the alarm is set in the initial state A, and it is disabled in the terminal state C. If the alarm occurs in any state except for C, the protocol is restarted and the alarm is reset. Now, the question is how to set the delay of the alarm so that the expected time needed to complete the protocol (i.e., to reach the state C from the state A) is minimized. If the delay is too large, a lot of time is wasted by waiting in the failure state F. If it is too small, there is not enough time to complete the communication and the protocol is restarted many times before it succeeds. In this particular case, one may still argue that an optimal delay can be computed by hand and no synthesis algorithm is needed. Now consider a more complicated scenario where Alice tries to establish a simultaneous connection with $Bob_1, \ldots, Bob_n$ via

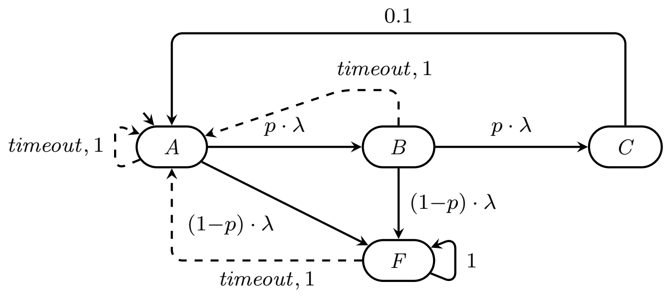


Figure 7.3: An ACTMC model of the communication with Bob$_i$.

different unreliable channels which are also unstable (i.e., an already established link with Bob$_i$ gets broken after a random time whose distribution is known). Alice needs to determine a suitable delays of alarms that achieve the optimal expected time of completing the protocol. Since the properties of the individual channels can be different and the probability of breaking an already established connection increases as more and more Bobs get connected, the delays chosen by Alice should actually depend on the subset of connections that remain to be established. The corresponding tuple of optimal delays is hard to compute manually.

We consider the model discussed in Example 7.1.1 where Alice is communicating with Bob$_1, \dots,$ Bob$_n$. The communication with Bob$_i$ is modeled as the ACTMC of Figure 7.3. So, the only difference from the ACTMC of Figure 7.2 is that now we also model the possibility of "breaking" an already established connection by delay transition with rate 0.1. We set $p = 0.9$ and $\lambda = 1.1$, the rate costs are equal to 1, all alarm transitions incur the impulse cost 1, and the delay transitions incur zero cost.

The whole protocol is modeled as a ACTMC obtained by constructing the "parallel composition" of n identical copies of the ACTMC of Figure 7.3 (i.e., we assume that all Bobs use the same type of communication channel). If some connection is broken, then all the unconnected communications are restarted. Clearly, the naïve parallel composition of multiple copies of our ACTMC model does not result in an ACTMC model since there are multiple concurrently active alarms. However, all the parallel alarms run synchronously (communication is restarted when connection breaks) and they should have the same delay of Dirac distribution, since the underlying parallel components are the same (all alarm delays depend only on the number of unconnected transitions). Thus every set of parallel alarms can be modeled by one alarm and we obtain an ACTMC model with reduced state

n	ε	Num. states	Num. roots	Max pol. degree	CPU time [s]	
					symbolic	explicit
1	10^{-2}	4	8	55	2.91	4.4
1	10^{-3}	4	8	60	2.94	11.84
1	10^{-4}	4	8	64	2.96	75.18
1	10^{-5}	4	10	69	3.01	3429.88
2	10^{-2}	32	16	122	3.65	33.00
2	10^{-3}	32	20	129	4.93	1265.45
2	10^{-4}	32	20	135	4.91	N/A
3	10^{-2}	192	30	202	6.02	1765.71
3	10^{-3}	192	31	210	7.16	N/A
3	10^{-4}	192	32	220	7.47	N/A
4	10^{-2}	1024	40	280	10.71	N/A
4	10^{-3}	1024	40	290	11.41	N/A
5	10^{-2}	5120	55	360	26.36	N/A
6	10^{-2}	24576	65	449	221.76	N/A

Table 7.3: Performance characteristics for the communication protocol model.

space due to merging of equivalent states. Note that, the number of states grows exponentially in n.

Table 7.3 shows the outcomes achieved by the explicit and the symbolic algorithm. The first column gives the number of Bobs involved in the protocol, the second column is the allowed error ε, the third column specifies the total number of states of the resulting ACTMC model, the fourth and the fifth columns specify the maximal number of roots and the maximal degree of the constructed polynomials in the symbolic algorithm, and the last two columns give the time needed to compute the results. The N/A result stands for out of memory exception. Note that the explicit algorithm cannot analyze a protocol with more than three Bobs, and tends to be significantly worse especially for small ε.

Let us note that the symbolic algorithm could handle even larger model instances, but we cannot provide such results with our current implementation because of the limitation of the double precision in floating types (we would need a higher precision).

Table 7.4 shows the alarm delays synthesized for the models. As we already mentioned , the delays should depend on the number of connections

n	T1	T2	T3	T4	T5	T6
1	3.779370					
2	3.737017	3.868655				
3	3.661561	3.784139	3.946357			
4	3.577685	3.684519	3.826398	4.014022		
5	3.498647	3.587113	3.705449	3.864535	4.073141	
6	3.430744	3.501000	3.596000	3.724862	3.899238	4.125076

Table 7.4: The synthesized timeouts for the communication protocol model.

that are yet to be established, so there are n delays for a model instance involving n Bobs.

7.1.3 Optimizations for the Explicit Algorithm

Given the discretization bounds one has to compute the transition probabilities and expected accumulated reward for each action in the MDP corresponding to the discretized parameter of an alarm. This can be done using the transient analysis of subordinated CTMCs [109].

Prototype Implementation. In the first implementation of our explicit algorithm we used a straightforward approach to call built-in methods of PRISM to compute the required quantities for each discretized alarm parameter separately. This is reasonable since there are various efficient and well debugged built-in methods for transient analysis in PRISM. However, we experienced that most of the time was spent computing the transient analysis rather than solving the created MDP, e.g., 520 seconds out of 540 seconds of total time. One of the reasons is that in each iteration a small portion of memory is allocated and freed by built-in PRISM methods. Since there is a large number of actions, the amount of reallocated memory was slowing down the computation. Thus we decided to reimplement the computation of transient probabilities applying principles of dynamic programming.

Iterative Computation of Transient Analysis. The transient probabilities can be very efficiently approximated up to an arbitrary small error using the uniformization technique. The problem is that we have to compute the transient probabilities for each value of a very large set $\{i{\cdot}\delta \mid i \in \mathbb{N}_0$ and $0 <$

$i \leq u_a/\delta\} \cup \{u_a\}$ and allow only a fixed error κ for each computation. The transient probability vector $\pi(\delta)$ of a CTMC $\mathcal{C}$ at time δ can be computed using uniformization by

$$\pi(\delta) = \sum_{j=0}^{J} \mathbf{1}_{s_{\text{in}}} \cdot P^j \cdot \frac{(\lambda \cdot \delta)^j}{j!} \cdot e^{-\lambda \cdot \delta}, \tag{7.1}$$

where $\mathbf{1}_{s_{\text{in}}}$ is the initial vector of $\mathcal{C}$, λ is the uniformization rate of $\mathcal{C}$, and P is the transition kernel of the uniformized C. The choice of number J influences the error of the formula. It is easy to compute the value of J such that the error is sufficiently small.

However, for time $i \cdot \delta$ we can use the previously computed transient probabilities as

$$\pi(i \cdot \delta) = \sum_{j=0}^{J} \pi((i-1) \cdot \delta) \cdot P^j \cdot \frac{(\lambda \cdot \delta)^j}{j!} \cdot e^{-\lambda \cdot \delta}. \tag{7.2}$$

It is again easy to compute J such that the overall allowed error is not exceeded. Instead of performing naïve computation for each number in $\{i \cdot \delta \mid i \in \mathbb{N}_0 \text{ and } 0 < i \leq u_a/\delta\} \cup \{u_a\}$ with according number of steps $J_1, \ldots, J_{\lceil u_a/\delta \rceil}$ to cause error bounded by κ in each computation, we compute the transient probabilities iteratively with sufficiently large J to cause a small error in all computations. For example, if we have $\delta = 0.1$, $u_a/\delta = 1000$, rate $\lambda = 1.0$ and $\kappa = 0.01$ using the naïve method we have to do $J_1 + \cdots + J_{\lceil u_a/\delta \rceil} = 66,265$ steps and using the iterative method $J \cdot \lceil u_a/\delta \rceil = 3,000$ steps. This is significant difference since a vector matrix multiplication is performed in each step. Thus we hard-programmed the iterative computation of transient probabilities and accumulated rewards in CTMC what caused a dramatic speedup thanks to a smaller number of arithmetic operations and better memory management.

Precomputation. Careful reader may have noticed that (7.2) can be further simplified to

$$\pi(i \cdot \delta) = \pi((i-1) \cdot \delta) \cdot e^{-\lambda \cdot \delta} \cdot \sum_{j=0}^{J} P^j \cdot \frac{(\lambda \cdot \delta)^j}{j!}. \tag{7.3}$$

Hence, the matrix $e^{-\lambda \cdot \delta} \cdot \sum_{j=0}^{J} P^j \cdot (\lambda \cdot \delta)^j / j!$ can be easily precomputed beforehand and used for computation of each $\pi(i \cdot \delta)$ to increase the savings even more. However, this is not true. J is small and the matrix P is sparse for the most reasonable models and error bounds. But $e^{-\lambda\cdot\delta}\cdot\sum_{j=0}^{J}P^j\cdot(\lambda\cdot\delta)^j/j!$ is not sparse for almost each error bound, P, and λ, what is known as "fill-in" phenomenon. Thus using (7.2) is typically more efficient than using (7.3). Similar observations were discussed in [73].

7.2 Expected Mean Payoff

Now we compare feasibility of the symbolic and explicit algorithms for minimization of expected mean payoff on the running example explained in Example 1.0.1 and on a preventive maintenance model inspired by [65]. The experiments were carried out[3] using our prototype implementation of the symbolic algorithm implemented in MAPLE [19]. MAPLE is appropriate as it supports the root isolation of univariate polynomials with arbitrary high precision due to its symbolic engine. Moreover, the code in MAPLE is easily readable and verifiable, since it is basically mathematical formulas that in many cases were directly taken from our paper [15]. Finally, we save the overhead caused when calling MAPLE from PRISM (as in our previous implementation). The implementation currently supports ACTMCs with localized alarms with Dirac and uniform distributions only, but could be easily extended to more general ACTMCs. The algorithm itself is divided into two steps. First, the input model is preprocessed and all the relevant polynomials are created. Then, the optimal policy is obtained by the symbolic policy iteration algorithm.

7.2.1 Disk Drive Model

In the running example of this thesis (see Chapter 1 and Example 1.0.1) we aimed towards synthesizing delays d_s and d_w such that the long-run average power consumption of the disk drive is ε-minimal. We set minimal delay $\ell = 0.1$ and maximal delay $u = 10$ for both alarms.

Let us describe the impact of choosing delay values d_s and d_w on the expected mean payoff in more detail. In Figure 7.4 (left), we illustrate the trade-off between choosing different delays d_w depending on delays $d_s \in \{0.1, 10\}$ and queue sizes $N \in \{2, 8\}$. When the queue size is small, e.g., $N =$

3. All the computations were run on a machine equipped with Intel Core$^{\text{TM}}$ i7-3770 CPU processor at 3.40 GHz and 8 GiB of DDR RAM. The CPU time was measured by MAPLE.

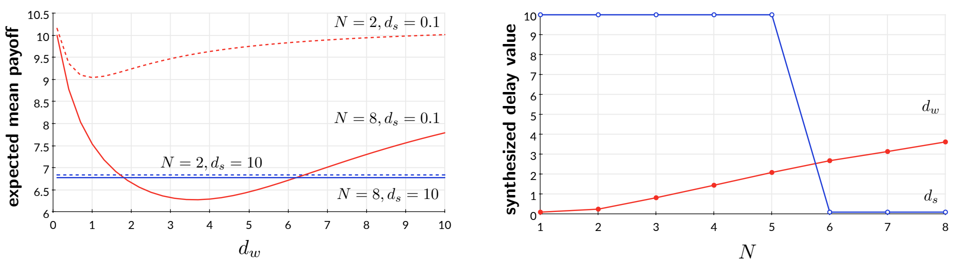


Figure 7.4: Results for the disk drive model: optimal expected mean payoff (left), and trade-off illustrated by the synthesized delay values (right).

2 (dashed curves), the expected mean payoff is minimal for large d_s (here, $d_s = 10$). Differently, when the queue size is large, e.g., $N = 8$ (solid curves), it is better to choose small d_s (here, $d_s = 0.1$) to minimize the expected mean payoff with d_w chosen at the minimum of the solid curve at around 3.6. This illustrates that the example is non-trivial.

The results of applying our synthesis algorithm for determining ε-minimal delays d_s and d_w depending on different queue sizes $N \in \{1, ..., 8\}$ with common delay bounds $\ell = 0.1$ and $u = 10$ are depicted in Figure 7.4 (right). From this figure we observe that for increasing queue sizes, also the synthesized value d_w increases, whereas the optimal value for $sleep$ is u in case $N < 6$ and ℓ otherwise.

Table 7.5 shows the running time of creation and solving of the Maple models, as well as the largest polynomial degrees for selected queue sizes $N = \{2, 4, 6, 8\}$ and error bounds $\varepsilon = \{0.1, 0.01, 0.001, 0.0005\}$. In all cases, discretization step sizes of $10^{-6} \cdot 10^{-19} < \delta(\cdot) < 10^{-19}$ were required to obtain results guaranteeing ε-optimal parameter functions. These small discretization constants underpin that the the explicit approach is not feasible for this case study (our implementation of the explicit algorithm runs out of memory for all of the listed instances). However, the symbolic algorithm evaluating roots of polynomials with high degree is capable to solve the problem within seconds in all cases. This can be explained through the small number of candidate actions we had to consider (always at most 200) and small number of iterations needed for policy iteration to converge. The running times are better than for minimization of expected total reward in almost the same model reported in Section 7.1.1. This is probably because we do not call multiple time times Maple from PRISM to isolate roots of polynomials.

N	ε	creating time [s]	solving time [s]	poly degree
2	0.1	0.15	0.24	46
	0.01	0.15	0.25	46
	0.001	0.16	0.28	53
	0.0005	0.16	0.33	53
4	0.1	0.14	0.25	46
	0.01	0.16	0.25	46
	0.001	0.16	0.28	53
	0.0005	0.16	0.33	53
6	0.1	0.16	0.35	46
	0.01	0.16	0.35	46
	0.001	0.17	0.40	53
	0.0005	0.18	0.43	53
8	0.1	0.19	0.35	46
	0.01	0.19	0.35	46
	0.001	0.20	0.43	53
	0.0005	0.22	0.44	53

Table 7.5: Statistics of the symbolic algorithm applied to the disk drive model.

7.2.2 Preventive Maintenance Model

As depicted in Figure 7.5, we consider a slightly modified model of a server that is susceptible to software faults [65]. A *rejuvenation* is the process of performing a *preventive maintenance* of the server after a fixed period of time (usually during night time) to prevent performance degradation or even failure of the server. The first row of states in Figure 7.5 represents the normal behavior of the server. Jobs arrive with rate 2 and are completed with rate 3. If job arrives and queue is full, it is rejected what is penalized by cost 6. Degradation of the server is modeled by delay transitions of rates 1 leading to $degrad$ states of the second row or eventually leading to the $failed$ state. The failure causes rejection of all jobs in the queue and incurs cost 4 for each rejected job. After the failure is reported (delay transition with rate 3), the repair process is initiated and completed after two exponentially distributed steps of rate 1. The repair can also fail with a certain probability (rate 0.1), thus after uniformly distributed time, the repair pro-

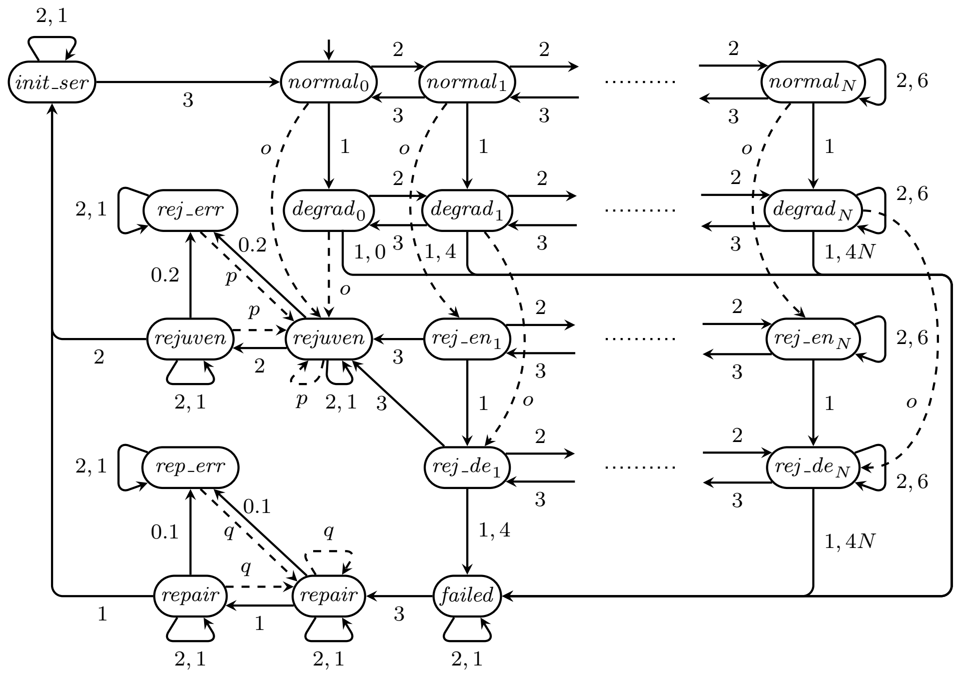


Figure 7.5: Preventive maintenance of a server.

cess is restarted. After each successful repair, the server is initialized by an exponential event with rate 3. The rejuvenation procedure is enabled after staying in *normal* or *degrad* states for time d_o. Then the rejuvenation itself is initiated after all jobs in the queue are completed. The rejuvenation procedure behaves similarly as the repair process, except that it is two times faster (all rates are multiplied by two).

First, we want to synthesize the value of the delay after which the rejuvenation is enabled, i.e., we aim towards the optimal schedule for rejuvenation to minimize cost. Furthermore, we synthesize the shifts d_p and d_q of the uniform distributions with length 2 associated with rejuvenation and repair, respectively, i.e., the corresponding uniform distribution function is $F_x[d_x](\tau) = \min\{1, \max\{0, (\tau - d_x)/2\}\}$, where $x \in \{p, q\}$. The interval of eligible values is $[0.1, 10]$ for all synthesized parameters.

Similarly as for previous example we show results of experiments for queue sizes $N = \{2, 4, 6, 8\}$ and error bounds $\varepsilon = \{0.1, 0.01, 0.001, 0.0001\}$ in Table 7.6. The CPU time of the model creation grows (almost quadrati-

N	ε	creating time [s]	solving time [s]	poly degree	results
2	0.1	0.15	1.80	86	$\mathbb{E}[MP]$ 0.85524
	0.01	0.15	2.57	92	d_o 1.82752
	0.001	0.15	2.97	96	d_p 0.66167
	0.0001	0.15	3.84	101	d_q 2.05189
4	0.1	0.83	1.92	86	$\mathbb{E}[MP]$ 0.46127
	0.01	0.92	2.40	92	d_o 1.92513
	0.001	1.04	3.06	97	d_p 0.66167
	0.0001	1.04	4.24	101	d_q 2.05189
6	0.1	2.25	2.18	87	$\mathbb{E}[MP]$ 0.33060
	0.01	2.36	2.53	92	d_o 1.95764
	0.001	2.37	3.83	97	d_p 0.66167
	0.0001	2.41	4.41	101	d_q 2.05189
8	0.1	17.08	2.22	87	$\mathbb{E}[MP]$ 0.29536
	0.01	17.60	2.81	93	d_o 1.96540
	0.001	17.78	3.33	97	d_p 0.66167
	0.0001	17.87	4.48	102	d_q 2.05189

Table 7.6: Results and statistics of the symbolic algorithm applied to the preventive maintenance model.

cally) to the number of states, caused by multiplication of large matrices in Maple. As within the disk drive example, we obtained the solutions very fast since we had to consider small number of candidate actions (always at most 500). We did not run the explicit algorithm for the preventive maintenance model because the created MDP would not fit in the memory for all considered model instances.

Optimizations in the implementation. To compute κ (and thus also δ) we used heuristics that were described in Section 6.2.1. Namely to compute upper bounds on expected time and cost until reaching some state from all other states, we employed the synthesis algorithm for maximization of expected total reward. Using these optimizations, for instance in the experiment of disk drive model, some discretization bounds of function δ were improved from $2.39 \cdot 10^{-239}$ to $7.03 \cdot 10^{-19}$. Note that even with these optimizations, the explicit algorithm for parameter synthesis would not be feasible as, more than 10^{18} actions would have to be considered for each state. This would clearly exceed the memory limit of state-of-the art computers.

7.3 New Implementation

Our prototype implementations of the synthesis algorithms are preliminary and we are in progress of optimizing them for public use. Optimally, the implementation of our synthesis algorithms should be in PRISM, as it is the most used tool by the community. However, there are two main issues why we have not provided a proper PRISM implementation of our synthesis algorithms yet:

- the need to call a proprietary software MAPLE to isolate roots of polynomials with sufficiently high precision and
- the need to extend the PRISM language, parser and internal data structures to support ACTMCs.

We currently work on both issues. We are developing a new extension of PRISM that will support specification of GSMPs (a more general formalism than ACTMCs), verification algorithms for GSMPs and ACTMCs, and our synthesis algorithms for ACTMCs. The PRISM extension will work as a standalone tool (similarly as our fdCTMC PRISM extension). There will be no need to call MAPLE since we are developing a library for root isolation of univariate polynomials with arbitrary precision. The synthesis algorithms will be reimplemented to use floating types with an arbitrary precision.

Chapter 8

Conclusions

In this thesis, we have studied continuous-time Markov chains with alarms (ACTMCs), where each alarm-setting distribution may depend on one parameter. We dealt with problems of identifying parameters that induce ε-optimal expected mean payoff or total reward. We presented selected results of our papers [15, 32, 99, 100] and we extended some of them to provide a complete and well-structured presentation of state-of-the-art results to our synthesis problems.

More concretely, we presented explicit algorithms that solve the synthesis problems by a reduction to finding ε-optimal policies in POMDPs and sufficient discretization of their action space. Even though the explicit algorithms were showed to have high time and space requirements due to large action space, they are important for two reasons: they form a theoretical background for subsequent symbolic algorithms and for a large class of ACTMCs with non-localized alarms there are no other correct algorithms known.

Additionally, we provided symbolic algorithms that represent the action space symbolically and perform an enhanced policy iteration, where only a small set of necessary candidate actions is explicitly constructed on demand. The candidate actions were identified by root isolation in certain univariate polynomials with high precision. All algorithms are applicable to ACTMCs satisfying abstractly formulated criteria, which were shown to hold for Dirac, uniform, exponential, Weibull, and many other distributions that are approximable by polynomials. Experimental evaluation on practical models of realistic size shows promising results of the symbolic algorithms.

8.1 Topics for Future Research

In the future research it would be interesting to test the feasibility of the stochastic state classes (SSCs) method [79]. Basically, the SSCs can be used instead of the subordinated Markov chain method to compute the explicit and symbolic action effects. The SSCs method is applicable to the class of Markov regenerative processes, where events have truncated expolynomial distributions that include exponential, uniform, Weibull, and Dirac distributions. This class allows for multiple concurrent non-exponentially-distributed events. Thus our explicit and symbolic algorithms would be applicable to a larger class of models. However, there are some additional challenges. Employing the SSCs method one will obtain general truncated expolynomial ranking functions. Hence, the answers to the following questions need to be found. How quickly can be the roots of truncated expolynomials isolated? Is it feasible to approximate the expolynomials by polynomials and isolate roots of the polynomials? Is the computation of the action effects using the SSCs method feasible for high precision?

Another potential research topic is the feasibility assessment of the smoothed model checking method [26] that was discussed as one of the statistical model checking synthesis methods for CTMCs in Section 3.3.4. We are confident that the method can be extended to ACTMCs. It is expected that it will not outperform the symbolic algorithms, but it can be very useful for ACTMCs with non-localized alarms as it can significantly outperform the explicit algorithms and it provides statistical guarantees.

Bibliography

[1] L. de Alfaro. “Formal Verification of Probabilistic Systems”. PhD thesis. Stanford University, 1997.

[2] L. de Alfaro. “Stochastic transition systems”. In: *Concurrency Theory (CONCUR)*. Vol. 1466. Lecture Notes in Computer Science. Springer, 1998, pp. 423–438.

[3] R. Alur and D. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (1994), pp. 183–235.

[4] R. Alur and M. Bernadsky. “Bounded model checking for GSMP models of stochastic real-time systems”. In: *Hybrid Systems: Computation and Control (HSCC)*. Vol. 3927. Lecture Notes in Computer Science. Springer, 2006, pp. 19–33.

[5] R. Alur, C. Courcoubetis, and D. L. Dill. “Model-checking for probabilistic real-time systems”. In: *Automata, Languages and Programming (ICALP)*. Vol. 510. Lecture Notes in Computer Science. Springer, 1991, pp. 115–126.

[6] R. Alur, C. Courcoubetis, and D. L. Dill. “Verifying automata specifications of probabilistic real-time systems”. In: *Real-Time: Theory in Practice*. Vol. 600. Lecture Notes in Computer Science. Springer, 1991, pp. 28–44.

[7] R. Alur, T. A. Henzinger, and M. Y. Vardi. “Parametric real-time reasoning”. In: *Symposium on Theory of Computing (STOC)*. ACM Press, 1993, pp. 592–601.

[8] E. G. Amparore, P. Buchholz, and S. Donatelli. “A structured solution approach for Markov regenerative processes”. In: *Quantitative Evaluation of Systems (QEST)*. Vol. 8657. Lecture Notes in Computer Science. Springer, 2014, pp. 9–24.

[9] É. André. “An Inverse Method for the Synthesis of Timing Parameters in Concurrent Systems”. PhD thesis. l’École Normale Supérieure de Cachan, 2010.

[10] P. Ashok, K. Chatterjee, P. Daca, J. Křetínský, and T. Meggendorfer. "Value iteration for long-run average reward in Markov decision processes". In: *Computer Aided Verification (CAV)*. Vol. 10426. Lecture Notes in Computer Science. Springer, 2017, pp. 201–221.

[11] S. Asmussen, O. Nerman, and M. Olsson. "Fitting phase-type distributions via the EM algorithm". In: *Scandinavian Journal of Statistics* 23.4 (1996), pp. 419–441.

[12] N. C. Audsley and A. Grigg. "Timing analysis of the ARINC 629 databus for real-time applications". In: *Microprocessors and Microsystems* 21.1 (1997), pp. 55–61.

[13] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. "Model-checking continous-time Markov chains". In: *ACM Transactions on Computational Logic* 1.1 (2000), pp. 162–170.

[14] C. Baier. *On algorithmic verification methods for probabilistic systems*. Habilitation thesis, Fakultät für Mathematik & Informatik, Universität Mannheim. 1998.

[15] C. Baier, C. Dubslaff, Ľ. Korenčiak, A. Kučera, and V. Řehák. "Mean-payoff optimization in continuous-time Markov chains with parametric alarms". In: *Quantitative Evaluation of Systems (QEST)*. Vol. 10503. Lecture Notes in Computer Science. Springer, 2017, pp. 190–206.

[16] C. Baier, C. Dubslaff, Ľ. Korenčiak, A. Kučera, and V. Řehák. "Synthesis of optimal resilient control strategies". In: *Automated Technology for Verification and Analysis (ATVA)*. Vol. 10482. Lecture Notes in Computer Science. Springer, 2017, pp. 417–434.

[17] C. Baier, B. R. Haverkort, H. Hermanns, and J. Katoen. "Model-checking algorithms for continuous-time Markov chains". In: *IEEE Transactions on Software Engineering* 29.6 (2003), pp. 524–541.

[18] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. "On the robustness of temporal properties for stochastic models". In: *Hybrid Systems Biology (HSB)*. Vol. 125. Electronic Proceedings in Theoretical Computer Science. 2013, pp. 3–19.

[19] L. Bernardin et al. *Maple 16 Programming Guide*. 2012.

[20] N. Bertrand, P. Bouyer, T. Brihaye, Q. Menet, C. Baier, M. Größer, and M. Jurdzinski. "Stochastic timed automata". In: *Logical Methods in Computer Science* 10.4 (2014).

[21] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Second Edition. Prentice-Hall International, 1992.

[22] M. Bezděka, O. Bouda, Ľ. Korenčiak, M. Madzin, and V. Řehák. “Sequence chart studio”. In: *Application of Concurrency to System Design (ACSD)*. IEEE Computer Society Press, 2012, pp. 148–153.

[23] A. Bobbio, A. Horváth, and M. Telek. “Matching three moments with minimal acyclic phase type distributions”. In: *Stochastic Models* 21.2-3 (2005), pp. 303–326.

[24] C. Bodenstein and A. Zimmermann. “TimeNET optimization environment”. In: *Performance Evaluation Methodologies and Tools (VALUETOOLS)*. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (ICST), 2014, pp. 129–133.

[25] H. C. Bohnenkamp, P. R. D’Argenio, H. Hermanns, and J.-P. Katoen. “MoDeST: A compositional modeling formalism for hard and softly timed systems”. In: *IEEE Transactions on Software Engineering* 32.10 (2006), pp. 812–830.

[26] L. Bortolussi, D. Milios, and G. Sanguinetti. “Smoothed model checking for uncertain continuous-time Markov chains”. In: *Information and Computation* 247 (2016), pp. 235–253.

[27] L. Bortolussi, D. Milios, and G. Sanguinetti. “U-check: Model checking and parameter synthesis under uncertainty”. In: *Quantitative Evaluation of Systems (QEST)*. Vol. 9259. Lecture Notes in Computer Science. Springer, 2015, pp. 89–104.

[28] L. Bortolussi and G. Sanguinetti. “Learning and designing stochastic processes from logical constraints”. In: *Quantitative Evaluation of Systems (QEST)*. Vol. 8054. Lecture Notes in Computer Science. Springer, 2013, pp. 89–105.

[29] P. Bouyer, T. Brihaye, P. Carlier, and Q. Menet. “Compositional design of stochastic timed automata”. In: *Computer Science - Theory and Applications, Computer Science Symposium in Russia (CSR)*. Vol. 9691. Lecture Notes in Computer Science. Springer, 2016, pp. 117–130.

[30] M. Bravetti, M. Bernardo, and R. Gorrieri. “Towards performance evaluation with general distributions in process algebras”. In: *Concurrency Theory (CONCUR)*. Vol. 1466. Lecture Notes in Computer Science. Springer, 1998, pp. 405–422.

[31] M. Bravetti and R. Gorrieri. “The theory of interactive generalized semi-Markov processes”. In: *Theoretical Computer Science* 282.1 (2002), pp. 5–32.

[32] T. Brázdil, Ľ. Korenčiak, J. Krčál, P. Novotný, and V. Řehák. “Optimizing performance of continuous-time stochastic systems using timeout synthesis”. In: *Quantitative Evaluation of Systems (QEST)*. Vol. 9259. Lecture Notes in Computer Science. Springer, 2015, 141–159.

[33] T. Brázdil, V. Forejt, J. Krčál, J. Křetínský, and A. Kučera. “Continuous-time stochastic games with time-bounded reachability”. In: *Information and Computation* 224 (2013), pp. 46–70.

[34] T. Brázdil, Ľ. Korenčiak, J. Krčál, J. Křetínský, and V. Řehák. “On time-average limits in deterministic and stochastic Petri nets”. In: *International Conference on Performance Engineering (ICPE)*. Poster paper. ACM Press, 2013, pp. 421–422.

[35] T. Brázdil, J. Krčál, J. Křetínský, A. Kučera, and V. Řehák. “Stochastic real-time games with qualitative timed automata objectives”. In: *Concurrency Theory (CONCUR)*. Vol. 6269. Lecture Notes in Computer Science. Springer, 2010, pp. 207–221.

[36] T. Brázdil, J. Krčál, J. Křetínský, and V. Řehák. “Fixed-delay events in generalized semi-Markov processes revisited”. In: *Concurrency Theory (CONCUR)*. Vol. 6901. Lecture Notes in Computer Science. Springer, 2011, pp. 140–155.

[37] L. Brim, M. Češka, M. Demko, S. Pastva, and D. Šafránek. “Parameter synthesis by parallel coloured CTL model checking”. In: *Computational Methods in Systems Biology (CMSB)*. Vol. 9308. Lecture Notes in Computer Science. Springer, 2015, pp. 251–263.

[38] G. Bucci, L. Carnevali, L. Ridi, and E. Vicario. “Oris: a tool for modeling, verification and evaluation of real-time systems”. In: *Software Tools for Technology Transfer* 12.5 (2010), pp. 391–403.

[39] P. Buchholz, E. M. Hahn, H. Hermanns, and L. Zhang. “Model checking algorithms for CTMDPs”. In: *Computer Aided Verification (CAV)*. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 225–242.

[40] X.-R. Cao. “From perturbation analysis to Markov decision processes and reinforcement learning”. In: *Discrete Event Dynamic Systems* 13.1-2 (2003), pp. 9–39.

[41] X.-R. Cao and H.-T. Fang. "Gradient-based policy iteration: an example". In: *Conference on Decision and Control (CDC)*. Vol. 3. IEEE Computer Society Press. 2002, pp. 3367–3371.

[42] L. Carnevali, L. Ridi, and E. Vicario. "A quantitative approach to input generation in real-time testing of stochastic systems". In: *IEEE Transactions on Software Engineering* 39.3 (2013), pp. 292–304.

[43] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.

[44] M. Češka, P. Pilař, N. Paoletti, L. Brim, and M. Z. Kwiatkowska. "PRISM-PSY: precise GPU-accelerated parameter synthesis for stochastic systems". In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Vol. 9636. Lecture Notes in Computer Science. Springer, 2016, pp. 367–384.

[45] M. Češka, F. Dannenberg, M. Z. Kwiatkowska, and N. Paoletti. "Precise parameter synthesis for stochastic biochemical systems". In: *Computational Methods in Systems Biology (CMSB)*. Vol. 8859. Lecture Notes in Computer Science. Springer, 2014, pp. 86–98.

[46] T. Chen, E. M. Hahn, T. Han, M. Z. Kwiatkowska, H. Qu, and L. Zhang. "Model repair for Markov decision processes". In: *Theoretical Aspects of Software Engineering (TASE)*. IEEE Computer Society Press, 2013, pp. 85–92.

[47] T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. "Quantitative model checking of continuous-time Markov chains against timed automata specifications". In: *Logic in Computer Science (LICS)*. IEEE Computer Society Press, 2009, pp. 309–318.

[48] H. Choi, V. G. Kulkarni, and K. S. Trivedi. "Markov regenerative stochastic Petri nets". In: *Performance Evaluation* 20.1 (1994), pp. 337–357.

[49] P. Chrszon, C. Dubslaff, S. Klüppelholz, and C. Baier. "Family-based modeling and analysis for probabilistic systems - featuring ProFeat". In: *Fundamental Approaches to Software Engineering (FASE)*. Vol. 9633. Lecture Notes in Computer Science. Springer, 2016, pp. 287–304.

[50] G. Ciardo, R. German, and C. Lindemann. "A characterization of the stochastic process underlying a stochastic Petri net". In: *IEEE Transactions on Software Engineering* 20.7 (1994), pp. 506–515.

[51] A. Clark, S. Gilmore, J. Hillston, and M. Tribastone. “Stochastic process algebras”. In: *Formal Methods for Performance Evaluation*. Vol. 4486. Lecture Notes in Computer Science. Springer, 2007, pp. 132–179.

[52] D. R. Cox. “The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 51 (1955), pp. 433–441.

[53] M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, I. Papusha, H. A. Poonawala, and U. Topcu. “Sequential convex programming for the efficient verification of parametric MDPs”. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Vol. 10206. Lecture Notes in Computer Science. Springer, 2017, pp. 133–150.

[54] N. Dalchau, N. Murphy, R. L. Petersen, and B. Yordanov. “Synthesizing and tuning chemical reaction networks with specified behaviours”. In: *CoRR* abs/1508.04403 (2015).

[55] J. Dankert, L. Yang, and J. Si. “An analysis of gradient-based policy iteration”. In: *International Joint Conference on Neural Networks (IJCNN)*. Vol. 5. IEEE Computer Society Press. 2005, pp. 2977–2982.

[56] P. R. D’Argenio and J.-P. Katoen. “A theory of stochastic systems. part II: process algebra”. In: *Information and Computation* 203.1 (2005), pp. 39–74.

[57] C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, and E. Ábrahám. “PROPhESY: A PRObabilistic ParamEter SYnthesis tool”. In: *Computer Aided Verification (CAV)*. Vol. 9206. Lecture Notes in Computer Science. Springer, 2015, pp. 214–231.

[58] M. Diciolla, C. H. P. Kim, M. Z. Kwiatkowska, and A. Mereacre. “Synthesising optimal timing delays for timed I/O automata”. In: *Embedded Software (EMSOFT)*. ACM Press, 2014, 16:1–16:10.

[59] F. Didier, T. A. Henzinger, M. Mateescu, and V. Wolf. “SABRE: A tool for stochastic analysis of biochemical reaction networks”. In: *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, 2010, pp. 193–194.

[60] D. L. Dill. "Timing assumptions and verification of finite-state concurrent systems". In: *Automatic Verification Methods for Finite State Systems (AVMFSS)*. Vol. 407. Lecture Notes in Computer Science. Springer, 1989, pp. 197–212.

[61] S. Donatelli, S. Haddad, and J. Sproston. "Model checking timed and stochastic properties with CSL^{TA}". In: *IEEE Transactions on Software Engineering* 35.2 (2009), pp. 224–240.

[62] C. Dubslaff, C. Baier, and S. Klüppelholz. "Probabilistic model checking for feature-oriented systems". In: *Transactions on Aspect-Oriented Software Development*. Lecture Notes in Computer Science 12 (2015), pp. 180–220.

[63] M. Fackrell. "Fitting with matrix-exponential distributions". In: *Stochastic Models* 21.2-3 (2005), pp. 377–400.

[64] J. Fearnley. "Exponential lower bounds for policy iteration". In: *Automata, Languages and Programming (ICALP)*. Vol. 6199. Lecture Notes in Computer Science. Springer, 2010, pp. 551–562.

[65] R. German. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. Wiley, 2000.

[66] R. German and C. Lindemann. "Analysis of stochastic Petri nets by the method of supplementary variables". In: *Performance Evaluation* 20.1-3 (1994), pp. 317–335.

[67] P. W. Glynn. "A GSMP formalism for discrete event systems". In: *Proceedings of the IEEE* 77.1 (1989), pp. 14–23.

[68] N. Götz, U. Herzog, and M. Rettelbach. "Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras". In: *Performance Evaluation of Computer and Communication Systems*. Vol. 729. Lecture Notes in Computer Science. Springer, 1993, pp. 121–146.

[69] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris. *Fundamentals of Queueing Theory*. Fourth Edition. Wiley Series in Probability and Statistics. Wiley, 2011.

[70] C. C. Guet, A. Gupta, T. A. Henzinger, M. Mateescu, and A. Sezgin. "Delayed continuous-time Markov chains for genetic regulatory circuits". In: *Computer Aided Verification (CAV)*. Vol. 7358. Lecture Notes in Computer Science. Springer, 2012, pp. 294–309.

[71] P. J. Haas. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer Series in Operations Research and Financial Engineering. Springer, 2010.

[72] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. "Reachability in succinct and parametric one-counter automata". In: *Concurrency Theory (CONCUR)*. Vol. 5710. Lecture Notes in Computer Science. Springer, 2009, pp. 369–383.

[73] S. Haddad, L. Mokdad, and P. Moreaux. "A new approach to the evaluation of non Markovian stochastic Petri nets". In: *Petri Nets and Other Models of Concurrency (ICATPN)*. Vol. 4024. Lecture Notes in Computer Science. Springer, 2006, pp. 221–240.

[74] S. Haddad, P. Moreaux, and G. Chiola. "Efficient handling of phase-type distributions in generalized stochastic Petri nets". In: *Petri Nets and Other Models of Concurrency (ICATPN)*. Vol. 1248. Lecture Notes in Computer Science. Springer, 1997, pp. 175–194.

[75] E. M. Hahn, H. Hermanns, and L. Zhang. "Probabilistic reachability for parametric Markov models". In: *Software Tools for Technology Transfer* 13.1 (2011), pp. 3–19.

[76] T. Han, J.-P. Katoen, and A. Mereacre. "Approximate parameter synthesis for probabilistic time-bounded reachability". In: *Real-Time Systems Symposium (RTSS)*. IEEE Computer Society Press, 2008, pp. 173–182.

[77] H. Hansson and B. Jonsson. "A logic for reasoning about time and reliability". In: *Formal Aspects of Computing* 6.5 (1994), pp. 512–535.

[78] M. Heiner, S. Lehrack, D. R. Gilbert, and W. Marwan. "Extended stochastic Petri nets for model-based design of wetlab experiments". In: *Transactions on Computational Systems Biology*. Lecture Notes in Computer Science 11 (2009), pp. 138–163.

[79] A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. "Transient analysis of non-Markovian models using stochastic state classes". In: *Performance Evaluation* 69.7 (2012), pp. 315–335.

[80] A. Horváth, L. Ridi, and E. Vicario. "Transient analysis of generalised semi-Markov processes using transient stochastic state classes". In: *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, 2010, pp. 231–240.

[81] A. Horváth and M. Telek. "PhFit: A general phase-type fitting tool". In: *Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS)*. Vol. 2324. Lecture Notes in Computer Science. Springer, 2002, pp. 82–91.

[82] T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager. "Linear parametric model checking of timed automata". In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Vol. 2031. Lecture Notes in Computer Science. Springer, 2001, pp. 189–203.

[83] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. Wiley, 1966.

[84] N. Jansen, F. Corzilius, M. Volk, R. Wimmer, E. Ábrahám, J.-P. Katoen, and B. Becker. "Accelerating parametric probabilistic verification". In: *Quantitative Evaluation of Systems (QEST)*. Vol. 8657. Lecture Notes in Computer Science. Springer, 2014, pp. 404–420.

[85] A. Jensen. "Markoff chains as an aid in the study of Markoff processes". In: *Scandinavian Actuarial Journal* 1953.1 (1953), pp. 87–91.

[86] P. G. Jensen and J. H. Taankvist. *Learning Optimal Scheduling for Time Uncertain Settings*. Student Project. Aalborg University, 2014.

[87] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. "A Bayesian approach to model checking biological systems". In: *Computational Methods in Systems Biology (CMSB)*. Vol. 5688. Lecture Notes in Computer Science. Springer, 2009, pp. 218–234.

[88] S. K. Jha and C. J. Langmead. "Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement". In: *Theoretical Computer Science* 412.21 (2011), pp. 2162–2187.

[89] R. Jones and G. Ciardo. "On phased delay stochastic Petri nets: definition and an application". In: *Petri Nets and Performance Models (PNPM)*. IEEE Computer Society Press, 2001, pp. 165–174.

[90] A. Jovanovic, D. Lime, and O. H. Roux. "Integer parameter synthesis for real-time systems". In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 445–461.

[91] A. Jovanovic and M. Z. Kwiatkowska. “Parameter synthesis for probabilistic timed automata using stochastic game abstractions”. In: *Reachability Problems (RP)*. Vol. 8762. Lecture Notes in Computer Science. Springer, 2014, pp. 176–189.

[92] L. C. M. Kallenberg. “Survey of linear programming for standard and nonstandard Markovian control problems. part I: theory”. In: *Mathematical Methods of Operations Research* 40.1 (1994), pp. 1–42.

[93] J.-P. Katoen and P. R. D’Argenio. “General distributions in process algebra”. In: *Lectures on Formal Methods and Performance Analysis*. Vol. 2090. Lecture Notes in Computer Science. Springer, 2000, pp. 375–430.

[94] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. “The ins and outs of the probabilistic model checker MRMC”. In: *Performance Evaluation* 68.2 (2011), pp. 90–104.

[95] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand Company, 1960.

[96] M. Kerber. “On the complexity of reliable root approximation”. In: *Computer Algebra in Scientific Computing (CASC)*. Vol. 5743. Lecture Notes in Computer Science. Springer, 2009, pp. 155–167.

[97] M. Khaksari and C. Fischione. “Performance analysis and optimization of the joining protocol for a platoon of vehicles”. In: *International Symposium on Communications, Control and Signal Processing (ISCCSP)*. IEEE Computer Society Press, 2012, pp. 1–6.

[98] Ľ. Korenčiak, J. Krčál, and V. Řehák. “Dealing with zero density using piecewise phase-type approximation”. In: *European Performance Engineering Workshop (EPEW)*. Vol. 8721. Lecture Notes in Computer Science. Springer, 2014, pp. 119–134.

[99] Ľ. Korenčiak, A. Kučera, and V. Řehák. “Efficient timeout synthesis in fixed-delay CTMC using policy iteration”. In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society Press, 2016, pp. 367–372.

[100] Ľ. Korenčiak, V. Řehák, and A. Farmadin. “Extension of PRISM by synthesis of optimal timeouts in fixed-delay CTMC”. In: *integrated Formal Methods (iFM)*. Vol. 9681. Lecture Notes in Computer Science. Springer, 2016, pp. 130–138.

[101] J. Krčál. "Formal Analysis of Discrete-Event Systems with Hard RealTime Bounds." PhD thesis. Faculty of Informatics, Masaryk University, Brno, 2014.

[102] V. G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.

[103] M. Z. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: verification of probabilistic real-time systems". In: *Computer Aided Verification (CAV)*. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591.

[104] M. Z. Kwiatkowska, G. Norman, and D. Parker. "Quantitative analysis with the probabilistic model checker PRISM". In: *Electronic Notes in Theoretical Computer Science* 153.2 (2006), pp. 5–31.

[105] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. "Verifying quantitative properties of continuous probabilistic timed automata". In: *Concurrency Theory (CONCUR)*. Vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 123–137.

[106] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. "Verifying quantitative properties of continuous probabilistic timed automata". In: *Concurrency Theory (CONCUR)*. Vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 123–137.

[107] A. Legay, B. Delahaye, and S. Bensalem. "Statistical model checking: An overview". In: *Runtime Verification (RV)*. Vol. 6418. Lecture Notes in Computer Science. Springer, 2010, pp. 122–135.

[108] Y. Li, B. Yin, and H. Xi. "Finding optimal memoryless policies of POMDPs under the expected average reward criterion". In: *European Journal of Operational Research* 211.3 (2011), pp. 556–567.

[109] C. Lindemann. "An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models". In: *Performance Evaluation* 18.1 (1993), pp. 79–95.

[110] C. Lindemann and G. S. Shedler. "Numerical analysis of deterministic and stochastic Petri nets with concurrent deterministic transitions". In: *Performance Evaluation* 27 (1996), pp. 565–582.

[111] C. Lindemann, A. Reuys, and A. Thümmler. "The DSPNexpress 2.000 performance and dependability modeling environment". In: *Fault-Tolerant Computing Symposium (FTCS)*. IEEE Computer Society Press, 1999, pp. 228–231.

[112] C. Lindemann and A. Thümmler. "Transient analysis of deterministic and stochastic Petri nets with concurrent deterministic transitions". In: *Performance Evaluation* 36-37.1-4 (1999), pp. 35–54.

[113] M. L. Littman. "Memoryless policies: theoretical limitations and practical results". In: *From Animals to Animats, Simulation of Adpative Behavior (SAB)*. Vol. 3. The MIT Press. 1994, pp. 238–245.

[114] G. G. I. López, H. Hermanns, and J.-P. Katoen. "Beyond memoryless distributions: model checking semi-Markov chains". In: *Process Algebra and Probabilistic Methods (PAPM)*. Vol. 2165. Lecture Notes in Computer Science. Springer, 2001, pp. 57–70.

[115] M. Marsan and G. Chiola. "On Petri nets with deterministic and exponentially distributed firing times". In: *Advances in Petri Nets 1987*. Vol. 266. Lecture Notes in Computer Science. Springer, 1987, pp. 132–145.

[116] M. A. Marsan, G. Conte, and G. Balbo. "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems". In: *A Transactions on Computer Systems* 2.2 (1984), pp. 93–122.

[117] S. Martina, M. Paolieri, T. Papini, and E. Vicario. "Performance evaluation of Fischer's protocol through steady-state analysis of Markov regenerative processes". In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society Press, 2016, pp. 355–360.

[118] K. Matthes. "Zur Theorie der Bedienungsprozesse". In: *Transactions of the Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*. Publishing House of the Czechoslovak Academy of Sciences, 1962, pp. 513–528.

[119] J. S. Miller. "Decidability and complexity results for timed automata and semi-linear hybrid automata". In: *Hybrid Systems: Computation and Control (HSCC)*. Vol. 1790. Lecture Notes in Computer Science. Springer, 2000, pp. 296–309.

[120] D. L. P. Minh, D. D. L. Minh, and A. L. Nguyen. "Regenerative Markov chain Monte Carlo for any distribution". In: *Communications in Statistics - Simulation and Computation* 41.9 (2012), pp. 1745–1760.

[121] W. Nelson. "Weibull analysis of reliability data with few or no failures". In: *Journal of Quality Technology* 17 (1985), pp. 140–146.

[122] M. R. Neuhäußer and L. Zhang. “Time-bounded reachability probabilities in continuous-time Markov decision processes”. In: *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, 2010, pp. 209–218.

[123] M. R. Neuhäußer. “Model Checking Nondeterministic and Randomly Timed Systems”. PhD thesis. RWTH Aachen University and University of Twente, 2010.

[124] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Courier Dover Publications, 1981.

[125] J. R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[126] R. Obermaisser. *Time-Triggered Communication*. CRC Press, 2011.

[127] T. Osogami and M. Harchol-Balter. “Closed form solutions for mapping general distributions to quasi-minimal PH distributions”. In: *Performance Evaluation* 63.6 (2006), pp. 524–552.

[128] M. Paolieri, A. Horváth, and E. Vicario. “Probabilistic model checking of regenerative concurrent systems”. In: *IEEE Transactions on Software Engineering* 42.2 (2016), pp. 153–169.

[129] L. Popova-Zeugmann. *Time and Petri Nets*. Springer, 2013.

[130] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[131] Q. Qiu, Q. Wu, and M. Pedram. “Stochastic modeling of a power-managed system: construction and optimization”. In: *International Symposium on Low Power Electronics and Design (ISLPED)*. ACM Press, 1999, pp. 194–199.

[132] K. Ramamritham and J. A. Stankovic. “Scheduling algorithms and operating systems support for real-time systems”. In: *Proceedings of the IEEE* 82.1 (1994), pp. 55–67.

[133] J. S. Rosenthal. *A First Look at Rigorous Probability Theory*. World Scientific Publishing, 2006.

[134] L. Sassoli and E. Vicario. “Close form derivation of state-density functions over DBM domains in the analysis of non-Markovian models”. In: *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, 2007, pp. 59–68.

[135] V. Sharma. “Complexity of real root isolation using continued fractions”. In: *Theoretical Computer Science* 409.2 (2008), pp. 292–310.

[136] B. S. K. Tati and M. Siegle. "Parameter and controller synthesis for Markov chains with actions and state labels". In: *Synthesis of Complex Parameters (SynCoP)*. Vol. 44. OpenAccess Series in Informatics. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015, pp. 63–76.

[137] A. Thümmler, P. Buchholz, and M. Telek. "A novel approach for phase-type fitting with the EM algorithm". In: *IEEE Transactions on Dependable and Secure Computing* 3.3 (2006), pp. 245–258.

[138] K. Tiassou, K. Kanoun, M. Kaâniche, C. Seguin, and C. Papadopoulos. "Aircraft operational reliability - a model-based approach and a case study". In: *Reliability Engineering & System Safety* 120 (2013), pp. 163–176.

[139] L.-M. Traonouez, D. Lime, and O. H. Roux. "Parametric model-checking of time Petri nets with stopwatches using the state-class graph". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Vol. 5215. Lecture Notes in Computer Science. Springer, 2008, pp. 280–294.

[140] A. Trivedi and D. Wojtczak. "Timed branching processes". In: *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, 2010, pp. 219–228.

[141] N. Wolovick, P. R. D'Argenio, and H. Qu. "Optimizing probabilities of real-time test case execution". In: *International Conference on Software Testing Verification and Validation (ICST)*. IEEE Computer Society Press, 2009, pp. 446–455.

[142] W. Xie, H. Sun, Y. Cao, and K. S. Trivedi. "Optimal webserver session timeout settings for web users". In: *Computer Measurement Group*. Computer Measurement Group, 2002, pp. 799–820.

[143] L. Yang, J. Dankert, and J. Si. "A performance gradient perspective on gradient-based policy iteration and a modified value iteration". In: *International Journal Intelligent Computing and Cybernetics* 1.4 (2008), pp. 509–520.

[144] H. L. S. Younes. "Ymer: A statistical model checker". In: *Computer Aided Verification (CAV)*. Vol. 3576. Lecture Notes in Computer Science. Springer, 2005, pp. 429–433.

[145] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. "Numerical vs. statistical probabilistic model checking". In: *Software Tools for Technology Transfer* 8.3 (2006), pp. 216–228.

[146] H. L. S. Younes and R. G. Simmons. “Probabilistic verification of discrete event systems using acceptance sampling”. In: *Computer Aided Verification (CAV)*. Vol. 2404. Lecture Notes in Computer Science. Springer, 2002, pp. 223–235.

[147] H. L. S. Younes and R. G. Simmons. “Statistical probabilistic model checking with a focus on time-bounded properties”. In: *Information and Computation* 204.9 (2006), pp. 1368–1409.

[148] A. Zimmermann, J. Freiheit, R. German, and G. Hommel. “Petri net modelling and performability evaluation with TimeNET 3.0”. In: *Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS)*. Vol. 1786. Lecture Notes in Computer Science. Springer, 2000, pp. 188–202.

[149] A. Zimmermann, J. Freiheit, and G. Hommel. “Discrete time stochastic Petri nets for the modeling and evaluation of real-time systems”. In: *International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press, 2001, p. 100.