Mean-Payoff Optimization in Continuous-Time Markov Chains with Parametric Alarms

Christel Baier[1], Clemens Dubslaff[1], Ľuboš Korenčiak[2(✉)], Antonín Kučera[2], and Vojtěch Řehák[2]

[1] TU Dresden, Dresden, Germany
{christel.baier,clemens.dubslaff}@tu-dresden.de
[2] Masaryk University, Brno, Czech Republic
{korenciak,kucera,rehak}@fi.muni.cz

Abstract. Continuous-time Markov chains with alarms (ACTMCs) allow for alarm events that can be non-exponentially distributed. Within *parametric* ACTMCs, the parameters of alarm-event distributions are not given explicitly and can be subject of parameter synthesis. An algorithm solving the ε-optimal parameter synthesis problem for parametric ACTMCs with long-run average optimization objectives is presented. Our approach is based on reduction of the problem to finding long-run average optimal strategies in semi-Markov decision processes (semi-MDPs) and sufficient discretization of parameter (i.e., action) space. Since the set of actions in the discretized semi-MDP can be very large, a straightforward approach based on explicit action-space construction fails to solve even simple instances of the problem. The presented algorithm uses an enhanced policy iteration on symbolic representations of the action space. The soundness of the algorithm is established for parametric ACTMCs with alarm-event distributions satisfying four 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 technique substantially improves the efficiency of the synthesis algorithm and allows to solve instances of realistic size.

1 Introduction

Mean-payoff is widely accepted as an appropriate concept for measuring long-run average performance of systems with rewards or costs. In this paper, we study the problem of synthesizing parameters for (possibly *non-exponentially* distributed) events in a given stochastic system to achieve an ε-optimal mean-payoff. One simple example of such events are *timeouts* widely used, e.g., to prevent deadlocks or to ensure some sort of progress in distributed systems. In practice, timeout

The authors are partly supported by the Czech Science Foundation, grant No. 15-17564S, by the DFG through the Collaborative Research Center SFB 912 – HAEC, the Excellence Initiative by the German Federal and State Governments (cluster of excellence cfAED), and the DFG-projects BA-1679/11-1 and BA-1679/12-1.

N. Bertrand and L. Bortolussi (Eds.): QEST 2017, LNCS 10503, pp. 190–206, 2017.
DOI: 10.1007/978-3-319-66335-7_12

durations are usually determined in an ad-hoc manner, requiring a considerable amount of expertise and experimental effort. This naturally raises the question of automating this design step, i.e., is there an algorithm synthesizing *optimal* timeouts?

The underlying stochastic model this paper relies on is provided by *continuous-time Markov chains with alarms (ACTMCs)*. Intuitively, ACTMCs extend continuous-time Markov chains by generally distributed *alarm events*, where at most one alarm is active during a system execution and non-alarm events can disable the alarm. In *parametric ACTMCs*, every alarm distribution depends on one single parameter ranging over a given interval of eligible values. For example, a timeout is a Dirac-distributed alarm event where the parameter specifies its duration. A *parameter function* assigning to every alarm a parameter value within the allowed interval yields a (non-parametric) ACTMC. We aim towards an algorithm that synthesizes a parameter function for an arbitrarily small $\varepsilon > 0$ achieving ε-optimal mean-payoff.

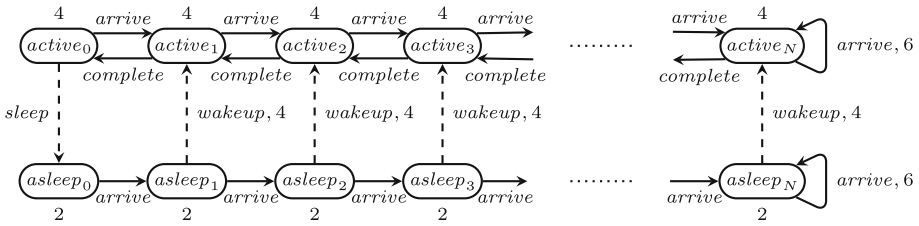

Fig. 1. Dynamic power manager of a disk drive.

Motivating Example. To get some intuition about the described task, consider a dynamic power management of a disk drive inspired by [28]. The behavior of the disk drive can be described as follows (see Fig. 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 then 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*. We annotate costs in terms of energy per time unit or instantaneous energy costs for events. 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, penalized by costs of 6. All other transitions incur with cost 1. 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 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 more power consuming *active* state. 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 paper:

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

Contribution. The main result of our paper is a *symbolic* algorithm for ε-optimal parameter synthesis that is *generic* in the sense that it is applicable to all systems where the optimized alarm events satisfy four abstractly formulated criteria. We show that these criteria are fulfilled, e.g., for timeout events modeled by Dirac distributions, uniformly distributed alarms (used in, e.g., in variants of the CSMA/CD protocol [5]), and Weibull distributions (used to model hardware failures [25]). For a given $\varepsilon > 0$, our algorithm first computes a sufficiently small discretization step such that an ε-optimal parameter function exists even when its range is restricted to the discretized parameter values. Since the discretization step is typically very small, an *explicit* construction of all discretized parameter values and their effects is computationally infeasible. Instead, our algorithm employs a symbolic variant of the standard policy iteration technique for optimizing the mean-payoff. It starts with *some* parameter function which is gradually improved until a fixed point is reached. In each improvement step, our algorithm computes a small *candidate subset* of the discretized parameter values such that a possible improvement is realizable by one of these candidate values. This is achieved by designing a suitable *ranking* function for each of the optimized events, such that an optimal parameter value is the minimal value of the ranking function in the interval of eligible parameter values. Then, the algorithm approximates the roots of the symbolic derivative of the ranking function, and constructs the candidate subset by collecting all discretized parameter values close to the approximated roots. This leads to a drastic efficiency improvement, which makes the resulting algorithm applicable to problems of realistic size.

Some proofs are omitted due to space constraints. Full details can be found in the accompanied technical report [3].

Related Work. Synthesis of optimal timeouts guaranteeing quantitative properties in timed systems was considered in [11]. There are various parametric formalisms for timed systems that deal with some sort of synthesis, such as parametric timed automata [1,18,19], parametric one-counter automata [14], parametric timed Petri nets [29], or parametric Markov models [15]. However, all works referenced above do not consider models with continuous-time

distributions, thus they synthesize different parameters than we do. Contrary, the synthesis of appropriate rates in CTMCs was efficiently solved in [8,10,16,17]. A special variant of ACTMC, where only alarms with Dirac distributions are allowed, has been considered in [6,20,21]. Their algorithms synthesize ε-optimal alarm parameters towards an expected reachability objective. Using a simulation-based approach, the optimization environment of the tool TimeNET is able to approximate locally optimal distribution parameters in stochastic Petri nets, e.g., using methods as simulated annealing, hill climbing or genetic algorithms. To the best of our knowledge, we present the first algorithm that approximates globally mean-payoff optimal parameters of non-exponential distributions in continuous-time models.

The (non-parametric) ACTMCs form a subclass of Markov regenerative processes (MRP) [2,9,24]. Alternatively, ACTMCs can be also understood as a generalized semi-Markov processes (GSMPs) with at most one non-exponential event enabled in each state or as bounded stochastic Petri nets (SPNs) [13] with at most one non-exponential transition enabled in any reachable marking [9]. Note that ACTMCs are analytically tractable thanks to methods for subordinated Markov-chain (SMC) that allow for efficient computation of transient and steady-state distributions [9,22]. Recently, methods for computing steady-state distributions in larger classes of regenerative GSMPs or SPNs have been presented in [23]. We did not incorporate this method into our approach as our methods to compute sufficiently small discretization and approximation precisions to guarantee ε-optimal mean-payoffs are not directly applicable for this class of systems. To the best of our knowledge there are no efficient algorithms with a guaranteed error for computation of steady-state distribution for a general GSMP (or SPN). For some cases it is even known that the steady-state distribution does not exist [7].

2 Preliminaries

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 a countable set A, we denote by $\mathcal{D}(A)$ the set of discrete probability distributions over A, i.e., functions $\mu\colon A \to \mathbb{R}_{\geq 0}$ where $\sum_{a \in A} \mu(a) = 1$. 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 $a \in A$.

2.1 Continuous-Time Markov Chains with Alarms

A *continuous-time Markov chain (CTMC)* is a triple $\mathcal{C} = (S, \lambda, P)$, where S is a finite set of states, $\lambda \in \mathbb{R}_{>0}$ is a common exit rate[1], and P is a probability

[1] We can assume without restrictions that the parameter λ is the same for all states of S since every CTMC can be effectively transformed into an equivalent CTMC satisfying this property by the standard uniformization method (see, e.g., [26]).

matrix over S. Transitions in $\mathcal{C}$ are exponentially distributed over the time, i.e., the probability of moving from s to s' within time τ is $P(s,s') \cdot (1 - e^{-\lambda \cdot \tau})$.

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

$$\mathcal{A} \;=\; \big(S, \lambda, P, A, \langle S_a \rangle, \langle P_a \rangle, \langle F_a \rangle\big),$$

where (S, λ, P) 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 cumulative distribution function (CDF) according to which the ringing time of an alarm $a \in A$ is distributed. We assume that each distribution has 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 $P(s,s') > 0$ 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 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 $\eta_0 = \infty$. 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. Then, the time t_i and the next configuration (s_{i+1}, η_{i+1}) are determined as follows:

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

Similarly as for standard CTMCs, we define a probability space over all runs initiated in a given $s_0 \in S$. We say that $\mathcal{A}$ is *strongly connected* if its underlying graph is, i.e., for all $s, s' \in S$, where $s \neq s'$, there is a finite sequence $s_0, \ldots, s_n$ of states such that $s = s_0$, $s' = s_n$, and $P(s_i, s_{i+1}) > 0$ or $P_a(s_i, s_{i+1}) > 0$ (for some $a \in A$) for all $0 \leq i < n$.

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[2]. 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 $P(s', s) > 0$. 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*.

Cost Structures and Mean-Payoff for ACTMCs. We use the standard cost structures that assign non-negative cost values to both states and transitions (see, e.g., [27]). More precisely, we consider the following cost functions: $\mathcal{R}\colon S \to \mathbb{R}_{\geq 0}$, which assigns a cost rate $\mathcal{R}(s)$ to every state s such that the cost $\mathcal{R}(s)$ is paid for every time unit spent in s, and functions $\mathcal{I}, \mathcal{I}_A\colon S \times S \to \mathbb{R}_{\geq 0}$ that assign to each delay transition and each alarm-setting transition, respectively, an instant execution cost. For every run $\omega = (s_0, \eta_0), t_0, (s_1, \eta_1), t_1, \ldots$ of $\mathcal{N}$ we define the associated *mean-payoff* by

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

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 $\mathbb{E}[\text{MP}]$ to denote the expectation of MP. 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 [9].

2.2 Parametric ACTMCs

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 $0 \leq \tau < d$. 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.[3]

In the above examples, we can interprete d as a *parameter* and ask what parameter values minimize the expected long-run average costs. For simplicity, we restrict our attention to distributions with only *one* parameter.[4] 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 $a \in A$. For every

[2] In fact, another possibility (which does not require any special attention) is that s is the initial state of a run.

[3] Note that a Weibull distribution with $k = 1$ is an exponential distribution.

[4] In our current setting, distribution functions with several parameters can be accommodated by choosing the 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 Appendix B in [3]).

$d \in [\ell_a, u_a]$, we use $F_a[d]$ to denote the distribution obtained by instantiating the parameter x with d. Formally, a parametric ACTMC is a tuple

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

where all components are defined in the same way as for ACTMC 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. Strong connectedness, localized alarms, and cost 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 ACTMC. When cost structures are defined on $\mathcal{N}$, we use $\mathbb{E}[\mathrm{MP}^{\mathbf{d}}]$ to denote the expected mean-payoff in $\mathcal{N}^{\mathbf{d}}$. For a given $\varepsilon > 0$, we say that a parameter function $\mathbf{d}$ is *ε-optimal* if

$$\mathbb{E}[\mathrm{MP}^{\mathbf{d}}] \;\leq\; \inf_{\mathbf{d}'} \mathbb{E}[\mathrm{MP}^{\mathbf{d}'}] \;+\; \varepsilon,$$

where $\mathbf{d}'$ ranges over all parameter functions for $\mathcal{N}$.

2.3 Semi-Markov Decision Processes

A *semi-Markov decision process (semi-MDP)* is a tuple $\mathcal{M} = (M, Act, Q, t, c)$, where M is a finite set of states, $Act = \biguplus_{m \in M} Act_m$ is a set of actions where $Act_m \neq \emptyset$ is a subset of actions enabled in a state m, $Q\colon Act \to \mathcal{D}(M)$ is a function assigning the probability $Q(b)(m')$ to move from $m \in M$ to $m' \in M$ executing $b \in Act_m$, and functions $t, c\colon Act \to \mathbb{R}_{\geq 0}$ provide the expected time and costs when executing an action, respectively.[5] A *run* in $\mathcal{M}$ is an infinite sequence $\omega = m_0, b_0, m_1, b_1, \ldots$ where $b_i \in Act_{m_i}$ for every $i \geq 0$. The mean-payoff of ω is

$$\mathrm{MP}(\omega) \;=\; \limsup_{n \to \infty} \Big(\sum\nolimits_{i=0}^{n} c(b_i)\Big) \,/\, \Big(\sum\nolimits_{i=0}^{n} t(b_i)\Big).$$

A (stationary and deterministic) *strategy* for $\mathcal{M}$ is a function $\sigma\colon M \to Act$ such that $\sigma(m) \in Act_m$ for all $m \in M$. *Applying* σ to $\mathcal{M}$ yields the standard probability measure $\Pr^{\sigma}$ over all runs initiated in a given initial state m_{in}. The expected mean-payoff achieved by σ is denoted by $\mathbb{E}[\mathrm{MP}^{\sigma}_{\mathcal{M}}]$. An *optimal*[6] strategy achieving the *minimal* expected mean-payoff is guaranteed to exist, and it is computable by a simple *policy iteration algorithm* (see, e.g., [27]).

κ-Approximations of Semi-MDPs. Let $\mathcal{M} = (M, Act, Q, t, c)$ be a semi-MDP, and $\kappa \in \mathbb{Q}_{>0}$. We say that $Q^{\kappa}\colon Act \to \mathcal{D}(M)$ and $t^{\kappa}, c^{\kappa}\colon Act \to \mathbb{R}_{\geq 0}$

[5] For our purposes, the actual distribution of the time and costs spent before executing some action is irrelevant, only their expectations matter, see Sect. 11.4 in [27].

[6] This strategy is optimal not only among stationary and deterministic strategies, but even among all randomized and history-dependent strategies.

are κ-*approximations* of Q, t, c, if for all $m, m' \in M$ and $b \in Act_m$ it holds that $Q(b)$ and $Q^\kappa(b)$ have the same support, $|Q(b)(m') - Q^\kappa(b)(m')| \leq \kappa$, $|t(b) - t^\kappa(b)| \leq \kappa$, and $|c(b) - c^\kappa(b)| \leq \kappa$. A κ-*approximation* of $\mathcal{M}$ is a semi-MDP $(M, Act, Q^\kappa, t^\kappa, c^\kappa)$ where Q^κ, t^κ, c^κ are κ-approximations of Q, t, c. We denote by $[\mathcal{M}]_\kappa$ the set of all κ-approximations of $\mathcal{M}$.

3 Synthesizing ε-optimal Parameter Functions

In the following, we fix a strongly connected parametric ACTMC $\mathcal{N} = (S, \lambda, P, A, \langle S_a \rangle, \langle P_a \rangle, \langle F_a[x] \rangle, \langle \ell_a \rangle, \langle u_a \rangle)$ with localized alarms and cost functions $\mathcal{R}$, $\mathcal{I}$, and $\mathcal{I}_A$, and aim towards an algorithm synthesizing an ε-optimal parameter function for $\mathcal{N}$. Here, ε-optimality is understood with respect to the expected mean-payoff. That is, we deal with the following computational problem:

ε-optimal parameter synthesis for parametric ACTMCs with localized alarms.

Input: $\varepsilon \in \mathbb{Q}_{>0}$, a strongly connected parametric ACTMC $\mathcal{N}$ with localized alarms, rational transition probabilities, rate λ, bounds $\langle \ell_a \rangle$,$\langle u_a \rangle$, and cost functions $\mathcal{R}$,$\mathcal{I}$, and $\mathcal{I}_A$.

Output: An ε-optimal parameter function $\mathbf{d}$.

3.1 The Set of Semi-Markov Decision Processes $[\mathcal{M}_\mathcal{N}\langle\delta\rangle]_\kappa$

Our approach to solve the above problem is based on a reduction to the problem of synthesizing expected mean-payoff optimal strategies in semi-MDPs. Let $a \in A$, and let $s \in S_a \cap S_{\text{set}}$. Recall that $\mathcal{N}$ is localized and thus, s is the uniquely defined a-setting state. Then, for every $d \in [\ell_a, u_a]$, consider runs initiated in a configuration (s, η), where η is chosen randomly according to $F_a[d]$. 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 $\mathit{\Pi}_s(d)$ to denote the associated probability distribution over $S_{\text{set}} \cup S_{\text{off}}$, i.e., $\mathit{\Pi}_s(d)(s')$ is the probability of visiting a regenerative configuration of the form (s', η') from s without previously visiting another regenerative configuration. Further, we use $\mathbb{E}_{s(d)}$ and $\mathit{\Theta}_s(d)$ to denote the expected accumulated costs and the expected time elapsed until visiting a regenerative configuration, respectively. We use the same notation also for $s \in S_{\text{off}}$, where $\mathit{\Pi}_s(d) = P(s, \cdot)$, $\mathbb{E}_s(d) = \mathcal{R}(s)/\lambda + P(s, \cdot) \cdot \mathcal{I}_P$, and $\mathit{\Theta}_s(d) = 1/\lambda$ are independent of d. The semi-MDP $\mathcal{M}_\mathcal{N} = (S_{\text{set}} \cup S_{\text{off}}, Act, Q, t, c)$ is defined over actions

$$Act \;=\; \big\{ \langle\!\langle s, d \rangle\!\rangle : d \in [\ell_a, u_a], s \in S_{\text{set}} \cap S_a, a \in A \big\} \cup \big\{ \langle\!\langle s, 0 \rangle\!\rangle : s \in S_{\text{off}} \big\},$$

where for all $\langle\!\langle s, d \rangle\!\rangle \in Act$ we have $Q(\langle\!\langle s, d \rangle\!\rangle) = \mathit{\Pi}_s(d)$, $t(\langle\!\langle s, d \rangle\!\rangle) = \mathit{\Theta}_s(d)$, and $c(\langle\!\langle s, d \rangle\!\rangle) = \mathbb{E}_s(d)$. Note that the action space of $\mathcal{M}_\mathcal{N}$ is dense and that $\mathit{\Pi}_s(d)$, $\mathit{\Theta}_s(d)$, and $\mathbb{E}_s(d)$ might be irrational. For our algorithms, 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 $\mathcal{M}_\mathcal{N}\langle\delta\rangle = (S_{\text{set}} \cup S_{\text{off}}, Act^\delta, Q^\delta, t^\delta, c^\delta)$ for

a given discretization function $\delta\colon S_{\text{set}} \to \mathbb{Q}_{>0}$. $\mathcal{M}_{\mathcal{N}}\langle\delta\rangle$ is defined as $\mathcal{M}_{\mathcal{N}}$ above, but over the action space $Act^{\delta} = \bigcup_{s \in S_{\text{set}} \cup S_{\text{off}}} Act^{\delta}_{s}$ with

$$Act^{\delta}_{s} \;=\; \big\{\langle\!\langle s, d\rangle\!\rangle : d = \ell_a + i \cdot \delta(s) < u_a, i \in \mathbb{N}_0\big\} \cup \big\{\langle\!\langle s, u_a\rangle\!\rangle\big\}$$

for $s \in S_{\text{set}} \cap S_a$ and $Act^{\delta}_{s} = \{\langle\!\langle s, 0\rangle\!\rangle\}$ otherwise.

To ensure rational values of $\mathit{\Pi}_s(d)$, $\Theta_s(d)$, and $\mathbb{E}_s(d)$, 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 $\kappa \in \mathbb{Q}_{>0}$. Note that, as $\mathcal{N}$ is strongly connected, every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ is also strongly connected.

3.2 An Explicit Parameter Synthesis Algorithm

Every strategy σ minimizing the expected mean-payoff in $\mathcal{M}_{\mathcal{N}}$ yields an optimal parameter function $\mathbf{d}^{\sigma}$ for $\mathcal{N}$ defined by $\mathbf{d}^{\sigma}(a) = d$ where $\sigma(s) = \langle\!\langle s, d\rangle\!\rangle$ for the unique a-setting state s. A naive approach towards an ε-optimal parameter function minimizing the expected mean-payoff in $\mathcal{N}$ 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 strategy 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 strategy for $\mathcal{M}$ can then be carried out using standard algorithms for semi-MDP (see, e.g., [27]). This approach is applicable under the following mild assumptions:

1. For every $\varepsilon \in \mathbb{Q}_{>0}$, there are computable $\delta\colon S_{\text{set}} \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$ such that for every $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ and every optimal strategy σ for $\mathcal{M}$, the associated parameter function $\mathbf{d}^{\sigma}$ is ε-optimal for $\mathcal{N}$.
2. For all $\kappa \in \mathbb{Q}_{>0}$ and $s \in S_{\text{set}}$, there are computable rational κ-approximations $\mathit{\Pi}^{\kappa}_{s}(d)$, $\Theta^{\kappa}_{s}(d)$, $\mathbb{E}^{\kappa}_{s}(d)$ of $\mathit{\Pi}_s$, Θ_s, $\mathbb{E}_s$.

Assumption 1 usually follows from perturbation bounds on the expected mean-payoff using a straightforward error-propagation analysis. Assumption 2 can be obtained, e.g., by first computing $\kappa/2$-approximations of $\mathit{\Pi}_s$, Θ_s, and $\mathbb{E}_s$ 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 [6,9]). Hence, Assumptions 1 and 2 rule out only those types of distributions that are rarely used in practice. In particular, the assumptions are satisfied for uniform, Dirac, and Weibull distributions. 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 $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$. Usually, this naive *explicit approach* to parameter synthesis is computationally infeasible due the large number of actions in $\mathcal{M}$.

3.3 A Symbolic Parameter Synthesis Algorithm

Our symbolic parameter synthesis algorithm computes the set of states of some $\mathcal{M} \in [\mathcal{M}_{\mathcal{N}}\langle\delta\rangle]_{\kappa}$ (see Assumption 1) but avoids computing the set of all actions

of $\mathcal{M}$ and their effects. The algorithm is obtained by modifying the standard policy iteration [27] for semi-MDPs.

Standard Policy Iteration Algorithm. When applied to $\mathcal{M}$, standard policy iteration starts by picking an arbitrary strategy σ, which is then repeatedly improved until a fixed point is reached. In each iteration, the current strategy σ is first evaluated by computing the associated *gain* g and *bias* $\mathbf{h}$.[7] Then, for each state $s \in S_{\text{set}}$, every outgoing action $\langle\langle s, d \rangle\rangle$ is ranked by the function

$$F_s^\kappa[g, \mathbf{h}](d) \quad = \quad \mathbb{E}_s^\kappa(d) - g \cdot \Theta_s^\kappa(d) + \Pi_s^\kappa(d) \cdot \mathbf{h} \qquad (\times)$$

where $\mathbb{E}_s^\kappa$, Θ_s^κ, and Π_s^κ are the determining functions of $\mathcal{M}$. If the action chosen by σ at s does not have the best (minimal) rank, it is improved by redefining $\sigma(s)$ to some best-ranked action. The new strategy is then evaluated by computing its gain and bias and possibly improved again. The standard algorithm terminates when for all states the current strategy σ is no improvement to the previous.

Symbolic κ-approximations. In many cases, $\Pi_s(d)$, $\Theta_s(d)$, and $\mathbb{E}_s(d)$ 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 $\boldsymbol{\mathbb{E}}_s^\kappa(d)$, 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:

$$\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d) \quad = \quad \boldsymbol{\mathbb{E}}_s^\kappa(d) - g \cdot \boldsymbol{\Theta}_s^\kappa(d) + \boldsymbol{\Pi}_s^\kappa(d) \cdot \mathbf{h}. \qquad (\star)$$

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 isolated roots and bounds ℓ_a and u_a, we identify a small set of candidate actions and explicitly evaluate only those instead of all actions. Note, that $\boldsymbol{\Pi}_s^\kappa(d)$, $\boldsymbol{\Theta}_s^\kappa(d)$, $\boldsymbol{\mathbb{E}}_s^\kappa(d)$ 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 $\Pi_s^\kappa(d)$, $\Theta_s^\kappa(d)$, $\mathbb{E}_s^\kappa(d)$ for this purpose. Before we provide our symbolic algorithm, we formally state the additional assumptions required to guarantee its soundness:

3. For all $a \in A$, $s \in S_{\text{set}} \cap S_a$, $\delta \colon S_{\text{set}} \to \mathbb{Q}_{>0}$ and $\kappa \in \mathbb{Q}_{>0}$, there are analytical κ-approximations $\boldsymbol{\Pi}_s^\kappa$, $\boldsymbol{\Theta}_s^\kappa$, $\boldsymbol{\mathbb{E}}_s^\kappa$ of Π_s, Θ_s, $\mathbb{E}_s$, respectively, such that the function $\boldsymbol{F}_s^\kappa[g, \mathbf{h}](d)$, where $g \in \mathbb{Q}$ and $\mathbf{h} \colon S_{\text{set}} \cup S_{\text{off}} \to \mathbb{Q}$ 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 $\delta(s)$.

[7] Here, it suffices to know that g is a scalar and $\mathbf{h}$ is a vector assigning numbers to states; for more details, see Sects. 8.2.1 and 8.6.1 in [27].

Algorithm 1. Symbolic policy iteration

input : A strongly connected parametric ACTMC $\mathcal{N}$ with localized alarms, rational cost functions $\mathcal{R}$, $\mathcal{I}_P$, $\mathcal{I}_{P_a}$, and $\varepsilon \in \mathbb{Q}_{>0}$ such that Assumptions 1–4 are fulfilled.
output: An ε-optimal parameter function $\mathbf{d}$.

1 compute the sets S_{set} and S_{off}
2 compute δ, κ, and $\Pi_s^{\min}$ of Assumptions 1 and 4
3 let $\xi = \min\{\kappa/4, \Pi_s^{\min}/3 : \text{where } s \in S_{\text{set}}\}$
4 fix the functions $\Pi_s^\xi, \Theta_s^\xi, \boldsymbol{\epsilon}_s^\xi$ of Assumption 2 determining $\mathcal{M}_\xi \in [\mathcal{M}_\mathcal{N}\langle\delta\rangle]_\xi$
5 choose an arbitrary state $s' \in S_{\text{set}} \cup S_{\text{off}}$ and a strategy σ' for $\mathcal{M}_\xi$
6 **repeat**
7 $\quad \sigma := \sigma'$
$\quad$ `// policy evaluation`
8 $\quad$ compute the *gain*, i.e., the scalar $g := \mathbb{E}[\text{MP}^\sigma]$
9 $\quad$ compute the *bias*, i.e., the vector $\mathbf{h}\colon S \to \mathbb{Q}$ satisfying $\mathbf{h}(s') = 0$ and for each $s \in S_{\text{set}} \cup S_{\text{off}}$, $\mathbf{h}(s) = \boldsymbol{\epsilon}_s^\xi(d) - g \cdot \Theta_s^\xi(d) + \Pi_s^\xi(d) \cdot \mathbf{h}$, where $\sigma(s) = \langle\!\langle s, d\rangle\!\rangle$
10 $\quad$ **foreach** $a \in A$ *and* $s \in S_{\text{set}} \cap S_a$ **do**
$\quad\quad$ `// policy improvement`
11 $\quad\quad$ compute the set R of $\delta(s)/2$-approximations of the roots of the derivative of $\boldsymbol{F}_s^\xi[g, \mathbf{h}](d)$ in $[\ell_a, u_a]$ using Assumption 3
12 $\quad\quad$ $C := \{\sigma(s)\} \cup \{\langle\!\langle s, d\rangle\!\rangle \in Act_s^\delta\colon |d - r| \le 3 \cdot \delta(s)/2, \text{ for } r \in R \cup \{\ell_a, u_a\}\}$
13 $\quad\quad$ $B := \operatorname{argmin}_{\langle\!\langle s,d\rangle\!\rangle \in C} F_s^\xi[g, \mathbf{h}](d)$
14 $\quad\quad$ **if** $\sigma(s) \in B$ **then** $\sigma'(s) := \sigma(s)$
15 $\quad\quad$ **else** $\sigma'(s) := \langle\!\langle s, d\rangle\!\rangle$ where $\langle\!\langle s, d\rangle\!\rangle \in B$
16 **until** $\sigma = \sigma'$
17 **return** $\mathbf{d}^\sigma$

4. For each $s \in S_{\text{set}}$ (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 $s' \in S_{\text{set}} \cup S_{\text{off}}$ we have that $\Pi_s(d)(s') > 0$ implies $\Pi_s(d)(s') \ge \Pi_s^{\min}$.

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 $\boldsymbol{\epsilon}_s^\kappa$ 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 1 closely mimics the standard policy iteration algorithm except for the definition of new precision ξ at line 3 and the policy improvement part. The local extrema points of $\boldsymbol{F}_s^\xi[g, \mathbf{h}](d)$ (cf. Eq. $(\star)$) in the interval $[\ell_a, u_a]$ are identified by computing roots of its symbolic derivative (line 11). 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). Each given candidate action is then evaluated using the function $F_s^\xi[g, \mathbf{h}](d) =$

$\mathbb{E}_s^\xi(d) - g \cdot \Theta_s^\xi(d) + \Pi_s^\xi(d) \cdot \mathbf{h}$ (cf. Eq. (×)). An improving candidate action is chosen based on the computed values (lines 14–15).

Theorem 1 (Correctness of Algorithm 1). *The symbolic policy iteration algorithm effectively solves the ε-optimal parameter synthesis problem for parametric ACTMCs and cost functions that fulfill Assumptions 1–4.*

Proof (Sketch). Since the number of actions of $\mathcal{M}_\xi$ is finite, Algorithm 1 terminates. A challenging point is that we compute only approximate minima of the function $\boldsymbol{F}_s^\xi[g, \mathbf{h}](d)$, which is *different* from the function $F_s^\xi[g, \mathbf{h}](d)$ used to evaluate the candidate actions. There may exist an action that is not in the candidate set C even if it has minimal $F_s^\xi[g, \mathbf{h}](d)$. Hence, the strategy computed by Algorithm 1 is not necessarily optimal for $\mathcal{M}_\xi$. Fortunately, due to Assumption 1, the strategy induces ε-optimal parameters for any parametric ACTMC if it is optimal 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 $\mathbb{E}'_s$ determining such $\mathcal{M}'$. Omitting the details, the functions Π'_s, Θ'_s, $\mathbb{E}'_s$ are constructed from Π_s^ξ, $\mathbb{E}_s^\xi$, Θ_s^ξ and slightly (by at most 2ξ) shifted $\boldsymbol{\Pi}_s^\xi$, $\boldsymbol{\mathbb{E}}_s^\xi$, $\boldsymbol{\Theta}_s^\xi$. The constant ξ was chosen sufficiently small such that the shifted $\boldsymbol{\Pi}_s^\xi$, $\boldsymbol{\mathbb{E}}_s^\xi$, $\boldsymbol{\Theta}_s^\xi$ are still κ-approximations of Π_s, Θ_s, $\mathbb{E}_s$ and the shifted $\boldsymbol{\Pi}_s^\xi(d)(\cdot)$ is a correct distribution for each $d \in [\ell_a, u_a]$. The technical details of the construction are provided in Appendix A of [3]. □

The following theorem implies that the explicit and symbolic algorithms are applicable to parametric ACTMCs with uniform, Dirac, exponential, or Weibull distributions. The proof is technical, see Appendix B of [3].

Theorem 2. *Assumptions 1–4 are fulfilled for parametric ACTMCs with rational cost functions where for all $a \in A$ we have that $F_a[x]$ is either a uniform, Dirac, exponential, or Weibull distribution.*

4 Experimental Evaluation

We demonstrate feasibility of the symbolic algorithm presented in Sect. 3 on the running example of Fig. 1 and on a preventive maintenance model inspired by [12]. The experiments were carried out[8] using our prototype implementation of the symbolic algorithm implemented in Maple [4]. Maple is appropriate as it supports the root isolation of univariate polynomials with arbitrary high precision due to its symbolic engine. The implementation currently supports Dirac and uniform distributions only, but could be easily extended by other distributions fulfilling Assumptions 1–4.

Disk Drive Model. In the running example of this paper (see Sect. 1 and Fig. 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-optimal. Let us describe the impact

[8] All the computations were run on a machine equipped with Intel Core™ i7-3770 CPU processor at 3.40 GHz and 8 GiB of DDR RAM.

of choosing delay values d_s and d_w on the expected mean-payoff in more detail. In Fig. 2 (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 is small, e.g., $N = 2$ (dashed curves), the expected mean-payoff is optimal 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.

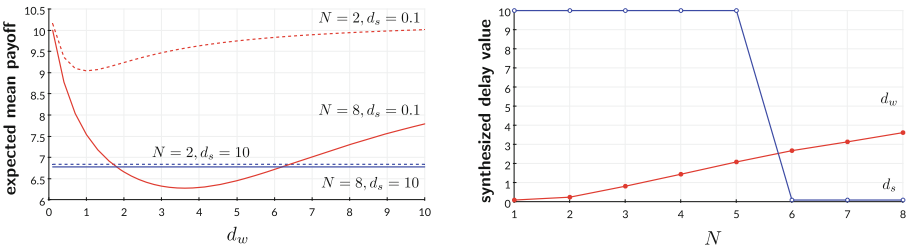

Fig. 2. Results for the disk drive example: optimal expected mean-payoff (left), and trade-off illustrated by the synthesized delay values (right)

The results of applying our synthesis algorithm for determining ε-optimal 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 Fig. 2 (right). From this figure we observe that for increasing queue sizes, also the synthesized value d_w increases, whereas the optimal value for d_s is u in case $N < 6$ and ℓ otherwise.

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

Fig. 3. Statistics of the symbolic algorithm applied to the disk drive example

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

Fig. 4. Results and statistics of the symbolic algorithm applied to the preventive maintenance example

The table in Fig. 3 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 ε-optimal parameter synthesis problem cannot be carried out using the explicit approach (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).

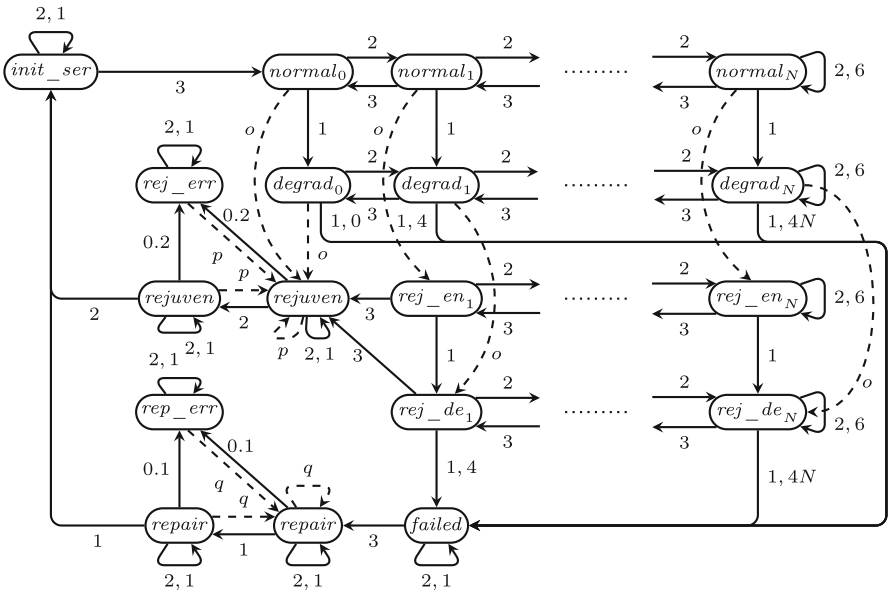

Fig. 5. Preventive maintenance of a server.

Preventive Maintenance. As depicted in Fig. 5, we consider a slightly modified model of a server that is susceptible to software faults [12]. A *rejuvenation* is the process of performing *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 Fig. 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 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 event 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 process 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. 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 Fig. 4. The CPU time of model creation grows (almost quadratically) 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).

Optimizations in the Implementation. For the sake of a clean presentation in this paper, we established *global* theoretical upper bounds on δ and κ sufficient to guarantee ε-optimal solutions, see Appendix B of [3]. The theoretical bounds assume the worst underlying transition structure of a given ACTMC. In the prototype implementation, we applied some optimizations mainly computing *local* upper bounds for each state in the constructed semi-MDP. Also, to achieve better perturbation bounds on the expected mean-payoff, i.e., to compute bounds on expected time and cost to reach some state from all other states, we rely on techniques presented in [6,20]. Using these optimizations, for instance in the experiment of disk drive model, some discretization bounds 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.

References

1. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: STOC, pp. 592–601. ACM (1993)
2. Amparore, E.G., Buchholz, P., Donatelli, S.: A structured solution approach for Markov regenerative processes. In: Norman, G., Sanders, W. (eds.) QEST 2014. LNCS, vol. 8657, pp. 9–24. Springer, Cham (2014). doi:10.1007/978-3-319-10696-0_3

3. Baier, C., Dubslaff, C., Korenčiak, Ľ., Kučera, A., Řehák, V.: Mean-payoff optimization in continuous-time Markov chains with parametric alarms. CoRR, abs/1706.06486 (2017)
4. Bernardin, L., et al.: Maple 16 Programming Guide (2012)
5. Bertsekas, D.P., Gallager, R.G.: Data Networks, 2nd edn. Prentice-Hall International, Upper Saddle River (1992)
6. Brázdil, T., Korenčiak, Ľ., Krčál, J., Novotný, P., Řehák, V.: Optimizing performance of continuous-time stochastic systems using timeout synthesis. In: Campos, J., Haverkort, B.R. (eds.) QEST 2015. LNCS, vol. 9259, pp. 141–159. Springer, Cham (2015). doi:10.1007/978-3-319-22264-6_10
7. Brázdil, T., Krčál, J., Křetínský, J., Řehák, V.: Fixed-delay events in generalized semi-Markov processes revisited. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 140–155. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23217-6_10
8. Češka, M., Dannenberg, F., Kwiatkowska, M., Paoletti, N.: Precise parameter synthesis for stochastic biochemical systems. In: Mendes, P., Dada, J.O., Smallbone, K. (eds.) CMSB 2014. LNCS, vol. 8859, pp. 86–98. Springer, Cham (2014). doi:10.1007/978-3-319-12982-2_7
9. Choi, H., Kulkarni, V.G., Trivedi, K.S.: Markov regenerative stochastic Petri nets. Perform. Eval. **20**(1–3), 337–357 (1994)
10. Alfaro, L.: Stochastic transition systems. In: Sangiorgi, D., Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 423–438. Springer, Heidelberg (1998). doi:10.1007/BFb0055639
11. Diciolla, M., Kim, C.H.P., Kwiatkowska, M., Mereacre, A.: Synthesising optimal timing delays for timed I/O automata. In: EMSOFT, pp. 1–10. ACM (2014)
12. German, R.: Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets. Wiley, Hoboken (2000)
13. Haas, P.J.: Stochastic Petri Nets: Modelling, Stability, Simulation. Springer, New York (2010)
14. Haase, C., Kreutzer, S., Ouaknine, J., Worrell, J.: Reachability in succinct and parametric one-counter automata. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 369–383. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04081-8_25
15. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. STTT **13**(1), 3–19 (2011)
16. Han, T., Katoen, J.P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: RTSS, pp. 173–182. IEEE (2008)
17. Jha, S.K., Langmead, C.J.: Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. TCS **412**(21), 2162–2187 (2011)
18. Jovanović, A., Kwiatkowska, M.: Parameter synthesis for probabilistic timed automata using stochastic game abstractions. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) RP 2014. LNCS, vol. 8762, pp. 176–189. Springer, Cham (2014). doi:10.1007/978-3-319-11439-2_14
19. Jovanovic, A., Lime, D., Roux, O.H.: Integer parameter synthesis for real-time systems. IEEE Trans. Softw. Eng. **41**(5), 445–461 (2015)
20. Korenčiak, Ľ., Kučera, A., Řehák, V.: Efficient timeout synthesis in fixed-delay CTMC using policy iteration. In: MASCOTS, pp. 367–372. IEEE (2016)
21. Korenčiak, Ľ., Řehák, V., Farmadin, A.: Extension of PRISM by synthesis of optimal timeouts in fixed-delay CTMC. In: iFM, pp. 130–138 (2016)

22. Lindemann, C.: An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models. Perform. Eval. **18**(1), 79–95 (1993)
23. Martina, S., Paolieri, M., Papini, T., Vicario, E.: Performance evaluation of Fischer's protocol through steady-state analysis of Markov regenerative processes. In: MASCOTS, pp. 355–360. IEEE (2016)
24. Minh, D.L.P., Minh, D.D.L., Nguyen, A.L.: Regenerative Markov chain Monte Carlo for any distribution. Commun. Stat.-Simul. C. **41**(9), 1745–1760 (2012)
25. Nelson, W.: Weibull analysis of reliability data with few or no failures. J. Qual. Technol. **17**, 140–146 (1985)
26. Norris, J.R.: Markov Chains. Cambridge University Press, Cambridge (1998)
27. Puterman, M.L.: Markov Decision Processes. Wiley, Hoboken (1994)
28. Qiu, Q., Wu, Q., Pedram, M.: Stochastic modeling of a power-managed system: construction and optimization. In: ISLPED, pp. 194–199. ACM Press (1999)
29. Traonouez, L.-M., Lime, D., Roux, O.H.: Parametric model-checking of time Petri nets with stopwatches using the state-class graph. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 280–294. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85778-5_20