An MM Algorithm to Estimate Parameters in Continuous-Time Markov Chains

Giovanni Bacci[1(✉)], Anna Ingólfsdóttir[2], Kim G. Larsen[1], and Raphaël Reynouard[2]

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark
{giovbacci,kgl}@cs.aau.dk
[2] Department of Computer Science, Reykjavík University, Reykjavík, Iceland
{annai,raphal20}@ru.is

Abstract. PRISM and STORM are popular model checking tools that provide a number of powerful analysis techniques for Continuous-time Markov chains (CTMCs). The outcome of the analysis is strongly dependent on the parameter values used in the model which govern the timing and probability of events of the resulting CTMC. However, for some applications, parameter values have to be empirically estimated from partially-observable executions.

In this work, we address the problem of estimating parameter values of CTMCs expressed as PRISM models from a number of partially-observable executions which might possibly miss some dwell time measurements. The semantics of the model is expressed as a parametric CTMC (pCTMC), i.e., CTMC where transition rates are polynomial functions over a set of parameters. Then, building on a theory of algorithms known by the initials MM, for minorization–maximization, we present an iterative maximum likelihood estimation algorithm for pCTMCs. We present an experimental evaluation of the proposed technique on a number of CTMCs from the quantitative verification benchmark set. We conclude by illustrating the use of our technique in a case study: the analysis of the spread of COVID-19 in presence of lockdown countermeasures.

Keywords: MM Algorithm · Continuous-time Markov chains · Maximum likelihood estimation

1 Introduction

A continuous-time Markov chain (CTMC) is a model of a dynamical system that, upon entering some state, remains in that state for a random real-valued amount of time—called the dwell time or sojourn time—and then transitions

K.G. Larsen and G. Bacci were supported by the S40S Villum Investigator Grant (37819) from Villum Fonden; R. Reynouard and A. Ingólfsdóttir were supported by the project *Learning and Applying Probabilistic Systems* (206574-051) of the Icelandic Research Fund.

N. Jansen and M. Tribastone (Eds.): QEST 2023, LNCS 14287, pp. 82–100, 2023.
https://doi.org/10.1007/978-3-031-43835-6_6

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 →
   beta * s * i * plock/SIZE : (s'=s - 1)&(i'=i + 1);
[] i>0 & r<SIZE →
   gamma * i * plock : (i'=i - 1)&(r'=r + 1);
endmodule
```

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module Susceptible
s : [0..SIZE] init 99936;
[infection] s>0 → s : (s'=s - 1);
endmodule

module Infected
i : [0..SIZE] init 48;
[infection] i>0 & i<SIZE → i : (i'=i + 1);
[recovery] i>0 → i : (i'=i - 1);
endmodule

module Recovered
r : [0..SIZE] init 16;
[recovery] r<SIZE → 1 : (r'=r + 1);
endmodule

module Rates
[infection] true → beta * plock/SIZE : true;
[recovery] true → gamma * plock : true;
endmodule
```

Fig. 1. (Left) SIR model with lockdown from [36], (Right) Semantically equivalent formulation of the model to the left where different individuals are modeled as distinct modules interacting with each other via synchronization.

probabilistically to another state. CTMCs are popular models in performance and dependability analysis. They have wide application and constitute the underlying semantics for real-time probabilistic systems such as queuing networks [33], stochastic process algebras [24], and calculi for systems biology [13,29].

Model checking tools such as PRISM [30] and STORM [14] provide a number of powerful analysis techniques for CTMCs. Both tools accept models written in the PRISM language, a state-based language based on [1] that supports compositional design via a uniform treatment of synchronous and asynchronous components.

For example, consider the PRISM model depicted in Fig. 1 (left) implementing a variant of the Susceptible-Infected-Recovered (SIR) model proposed in [36] to describe the spread of disease in presence of lockdown restrictions. The model distinguishes between three types of individuals: susceptible, infected, and recovered respectively associated with the state variables `s`, `i`, and `r`. Susceptible individuals become infected through contact with another infected person and can recover without outside interference. The SIR model is parametric in `beta`, `gamma`, and `plock`. `beta` is the *infection coefficient*, describing the probability of infection after the contact of a susceptible individual with an infected one; `gamma` is the *recovery coefficient*, describing the rate of recovery of an infected individual (in other words, 1/`gamma` is the time one individual requires to recover); and `plock` $\in [0, 1]$ is used to scale down the infection coefficient modeling restrictions to reduce the spread of disease.

Clearly, the outcome of the analysis of the above SIR model is strongly dependent on the parameter values used, as they govern the timing and probability of events of the CTMC describing its semantics. However, in some application domains, parameter values have to be empirically evaluated from a number of

partially-observable executions of the model. A paradigmatic example is the modeling pipeline described in [36], where the parameters of the SIR model in Fig. 1 (left) are estimated based on a definition of the model as ODEs, and later used in an approximation of the original SIR model designed to reduce the state space of the SIR model in Fig. 1 (left). Such modeling pipelines require high technical skills, are error-prone, and are time-consuming, thus limiting the applicability and the user base of model checking tools.

In this work, we address the problem of estimating parameter values of CTMCs expressed as Prism models from a number of partially-observable executions. The expressive power of the Prism language brings two technical challenges: (i) the classic state-space explosion problem due to modular specification, and (ii) the fact that the transition rates of the CTMCs result from the algebraic composition of the rates of different (parallel) modules which are themselves defined as arithmetic expressions over the parameters (*cf.* Fig. 1). We address the second aspect of the problem by considering a class of *parametric* CTMCs (pCTMC) [10,21], which are CTMCs where transition rates are polynomial functions over a fixed set of parameters. In this respect, pCTMCs have the advantage to cover a rich subclass of Prism models and to be closed under the operation of parallel composition implemented by the Prism language.

Following the standard approach, we pursue the maximum likelihood estimate (MLE), i.e., we look for the parameter values that achieve the maximum joint likelihood of the observed execution sequences. However, given the non-convex nature of the likelihood surface, computing the global maximum that defines the MLE is computationally intractable [42].

To deal with this issue we employ a theoretical iterative optimization principle known as MM algorithm [31,32]. The well-known EM algorithm [15] is an instance of MM optimization framework and is a versatile tool for constructing optimization algorithms. MM algorithms are typically easy to design, numerically stable, and in some cases amenable to accelerations [25,44]. The versatility of the MM principle consists in the fact that is built upon a simple theory of inequalities, allowing one to derive optimization procedures. The MM principle is useful to derive iterative procedures for maximum likelihood estimation which increase the likelihood at each iteration and converge to some local optimum.

The main technical contribution of the paper consists in devising a novel iterative maximum likelihood estimation algorithm for pCTMCs. Crucially, our technique is robust to missing data. In contrast with [18,41], where state labels and dwell times are assumed to be observable at each step of the observations while only state variables are hidden, our estimation procedure accepts observations to have information to be missing at some steps.

Notably, when state labels and dwell times are observable and only state variables are hidden, our learning procedure results in a generalization of the Baum-Welch algorithm [38]—an EM algorithm that estimates transition probabilities in hidden Markov models—to pCTMCs.

We demonstrate the effectiveness of our estimation procedure on a case study taken from [36] and show that our technique can be used to simplify modeling

pipelines that involve a number of modifications of the model—possibly introducing approximations—and the re-estimation of its parameters.

2 Preliminaries and Notation

We denote by $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{N}$ respectively the sets of real, rational, and natural numbers, and by Σ^n, Σ^* and, Σ^ω respectively the set of words of length $n \in \mathbb{N}$, finite length, and infinite length, built over the finite alphabet Σ.

We use $\mathcal{D}(\Omega)$ to denote the set of discrete probability distributions on Ω, i.e., functions $\mu\colon \Omega \to [0,1]$, such that $\mu(X) = 1$, where $\mu(E) = \sum_{x \in E} \mu(x)$ for $E \subseteq X$. For a proposition p, we write $[\![p]\!]$ for the Iverson bracket of p, i.e., $[\![p]\!] = 1$ if p is true, otherwise 0.

A labelled continuous-time Markov chain (CTMC) is defined as follows.

Definition 1. *A labelled CTMC is a tuple $\mathcal{M} = (S, R, s_0, \ell)$ where S is a finite set of states, $R\colon S \times S \to \mathbb{R}_{\geq 0}$ is the transition rate function, $s_0 \in S$ the initial states, and $\ell\colon S \to 2^{AP}$ is a labelling function which assigns to each state a subset of atomic propositions that the state s satisfies.*

The transition rate function assigns rates $r = R(s, s')$ to each pair of states $s, s' \in S$ which are to be seen as transitions of the form $s \xrightarrow{r} s'$. A transition $s \xrightarrow{r} s'$ can only occur if $r > 0$. In this case, the probability of this transition to be triggered within $\tau \in \mathbb{R}_{>0}$ time-units is $1 - e^{-r\,\tau}$. When, from a state s, there are more than one outgoing transition with positive rate, we are in presence of a *race condition*. In this case, the first transition to be triggered determines which label is observed as well as the next state of the CTMC. According to these dynamics, the time spent in state s before any transition occurs, called *dwell time*, is exponentially distributed with parameter $E(s) = \sum_{s' \in S} R(s, s')$, called *exit-rate* of s. A state s is called *absorbing* if $E(s) = 0$, that is, s has no outgoing transition. Accordingly, when the CTMC ends in an absorbing state it will remain in the same state indefinitely. The probability that the transition $s \xrightarrow{r} s'$ is triggered from s is $r/E(s)$ and is independent from the time at which it occurs. Accordingly, from the CTMC $\mathcal{M}$, we construct a (labelled) discrete-time Markov chain $emb(\mathcal{M}) = (S, P, s_0, \ell)$ with transition probability function $P\colon S \times S \to [0,1]$ defined as

$$P(s,s') = \begin{cases} R(s,s')/E(s) & \text{if } E(s) \neq 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s = s' \\ 0 & \text{otherwise} \end{cases}$$

Remark 1. A CTMC can be equivalently described as a tuple $(S, \to, s_0, \ell)$ where $\to \subseteq S \times \mathbb{R}_{\geq 0} \times S$ is a transition *relation*. The transition rate function R induced by $\to$ is obtained as, $R(s, s') = \sum\{r \mid s \xrightarrow{r} s'\}$ for arbitrary $s, s' \in S$.

An *infinite path* of a CTMC $\mathcal{M}$ is a sequence $s_0\tau_0 s_1\tau_1 s_2\tau_2 \cdots \in (S \times \mathbb{R}_{>0})^\omega$ where $R(s_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. A *finite path* is a sequence $s_0\tau_0 \cdots s_{k-1}\tau_{k-1}s_k$

where $R(s_i, s_{i+1}) > 0$ and $\tau_i \in \mathbb{R}_{>0}$ for all $i \in \{1, \ldots, k-1\}$ and s_k is absorbing. The meaning of a path is that the system started in state s_0, where it stayed for time τ_0, then transitioned to state s_1 where it stayed for time τ_1, and so on. For a finite path the system eventually reaches an absorbing state s_k, where it remains. We denote by $\mathbf{Path}_{\mathcal{M}}$ the set of all (infinite and finite) paths of $\mathcal{M}$. The formal definition of the probability space over $\mathbf{Path}_{\mathcal{M}}$ induced by $\mathcal{M}$ can be given by following the classical cylinder set construction (see e.g., [6,28]).

Finally, we define the random variables S_i, L_i, and T_i ($i \in \mathbb{N}$) that respectively indicate the i-th state, its label, and i-th dwell time of a path.

The MM Algorithm. The MM algorithm is an iterative optimization method. The acronym MM has a double interpretation: in minimization problems, the first M stands for majorize and the second for minimize; dually, in maximization problems, the first M stands for minorize and the second for maximize. In this paper we only focus on maximizing an objective function $f(\mathbf{x})$, hence we tailor the presentation of the general principles of the MM framework to maximization problems. The MM algorithm is based on the concept of *surrogate function*. A surrogate function $g(\mathbf{x} \mid \mathbf{x}_m)$ is said to *minorize* a function $f(\mathbf{x})$ at $\mathbf{x}_m$ if

$$f(\mathbf{x}_m) = g(\mathbf{x}_m \mid \mathbf{x}_m)\,, \tag{1}$$

$$f(\mathbf{x}) \geq g(\mathbf{x} \mid \mathbf{x}_m) \quad \text{for all } \mathbf{x} \neq \mathbf{x}_m\,. \tag{2}$$

In the MM optimization framework, we maximize the surrogate minorizing function $g(\mathbf{x} \mid \mathbf{x}_m)$ rather than the actual function $f(\mathbf{x})$. If $\mathbf{x}_{m+1}$ denotes the maximum of the surrogate $g(\mathbf{x} \mid \mathbf{x}_m)$, then the next iterate $\mathbf{x}_{m+1}$ forces $f(\mathbf{x})$ uphill, Indeed, the inequalities

$$f(\mathbf{x}_m) = g(\mathbf{x}_m \mid \mathbf{x}_m) \leq g(\mathbf{x}_{m+1} \mid \mathbf{x}_m) \leq f(\mathbf{x}_{m+1})$$

follow directly from the definition of $\mathbf{x}_{m+1}$ and the axioms (1) and (2).

Because piecemeal composition of minorization works well, the derivations of surrogate functions are typically achieved by applying basic minorizations to strategic parts of the objective function, leaving other parts untouched. Finally, another aspect that can simplify the derivation of MM algorithms comes from the fact that the iterative maximization procedure hinges on finding $\mathbf{x}_{m+1} = \arg\max_{\mathbf{x}} g(\mathbf{x} \mid \mathbf{x}_m)$. Therefore, $g(\mathbf{x} \mid \mathbf{x}_m)$ can be replaced by any other surrogate function $g'(\mathbf{x} \mid \mathbf{x}_m)$ satisfying $\arg\max_{\mathbf{x}} g(\mathbf{x} \mid \mathbf{x}_m) = \arg\max_{\mathbf{x}} g'(\mathbf{x} \mid \mathbf{x}_m)$ for all $\mathbf{x}_m$. This is for instance the case when $g(\mathbf{x} \mid \mathbf{x}_m)$ and $g'(\mathbf{x} \mid \mathbf{x}_m)$ are equal up to some (irrelevant) constant c, that is $g(\mathbf{x} \mid \mathbf{x}_m) = g'(\mathbf{x} \mid \mathbf{x}_m) + c$.

3 Parametric Continuous-Time Markov Chains

As mentioned in the introduction, the PRISM language offers constructs for the modular design of CTMCs within a uniform framework that represents synchronous and asynchronous module interaction. For example, consider the PRISM models depicted in Fig. 1. The behavior of each module is described by a set of

commands which take the form [`action`] `guard` $\rightarrow$ `rate`: `update` representing a set of transitions of the module. The guard is a predicate over the state variables in the model. The update and the rate describe a transition that the module can make if the guard is true. The command optionally includes an action used to force two or more modules to make transitions simultaneously (i.e., to synchronize). For example, in the model in Fig. 1 (right), in state $(50, 20, 5)$ (i.e., $s = 50$, $i = 20$, and $r = 5$), the model can move to state $(49, 21, 5)$ by synchronizing over the action `infection`. The rate of this transition is equal to the product of the individual rates of each module participating in an `infection` transition, which in this case amounts to $0.01 \cdot$ `beta` $\cdot$ `plock`. Commands that do not have an action represent asynchronous transitions that can be taken independently (i.e., asynchronously) from other modules.

By default, all modules are combined following standard parallel composition in the sense of the parallel operator from Communicating Sequential Processes algebra (CPS), that is, modules synchronize over all their common actions. The Prism language offers also other CPS-based operators to specify the way in which modules are composed in parallel.

Therefore, a parametric representation of a CTMC described by a Prism model shall consider *transition rate expressions* which are closed under finite sums and finite products: sums deal with commands with overlapping guards and updates, while products take into account synchronization. In line with [10,21] we employ *parametric* CTMCs (pCTMCs).

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector of parameters. We write $\mathcal{E}$ for the set of polynomial maps $f\colon \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ of the form $f(\mathbf{x}) = \sum_{i=1}^{m} b_i \prod_{j=1}^{n} x_j^{a_{ij}}$, where $b_i \in \mathbb{R}_{\geq 0}$ and $a_{ij} \in \mathbb{N}$ for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. $\mathcal{E}$ is a commutative semiring satisfying the above-mentioned requests for transition rate expressions.

Definition 2. *A pCTMC is a tuple $\mathcal{P} = (S, R, s_0, \ell)$ where S, s_0, and ℓ are defined as for CTMCs, and $R\colon S \times S \to \mathcal{E}$ is a parametric transition rate function.*

Intuitively, a pCTMC $\mathcal{P} = (S, R, s_0, \ell)$ defines a family of CTMCs arising by plugging in concrete values for the parameters $\mathbf{x}$. Given a parameter evaluation $\mathbf{v} \in \mathbb{R}^n_{\geq 0}$, we denote by $\mathcal{P}(\mathbf{v})$ the CTMC associated with $\mathbf{v}$, and $R(\mathbf{v})$ for its rate transition function. Note that by construction $R(\mathbf{v})(s, s') \geq 0$ for all $s, s' \in S$, therefore $\mathcal{P}(\mathbf{v})$ is a proper CTMC.

As for CTMCs, parametric transitions rate functions can be equivalently described by means of a transition relation $\rightarrow \subseteq S \times \mathcal{E} \times S$, where the parametric transition rate from s to s' is $R(s, s')(\mathbf{x}) = \sum\{f(\mathbf{x}) \mid s \xrightarrow{f} s'\}$.

Example 1. Consider the model in Fig. 1 parametric in `beta`, `gamma`, and `plock`. The semantics of this model is a pCTMC with states $S = \{(s, i, r) \mid s, i, r \in \{0, \ldots, 10^5\}\}$ and initial state $(99936, 48, 16)$. For example, the initial state has two outgoing transitions: one that goes to $(99935, 49, 16)$ with rate $47.96928 \cdot$ `beta`$\cdot$`plock`, and the other that goes to $(99935, 48, 17)$ with rate $49 \cdot$`gamma`$\cdot$`plock`.

One relevant aspect of the class of pCTMCs is the fact that it is closed under parallel composition in the sense described above. This justifies the study

of parameter estimation of PRISM models from observed data via maximum likelihood estimation for pCTMCs.

4 Estimating Parameters from Partial Observations

In this section, we present an algorithm to estimate the parameters of a pCTMC $\mathcal{P}$ from a collection of i.i.d. observation sequences $\mathcal{O} = \mathbf{o}_1, \ldots, \mathbf{o}_J$. Notably, the algorithm is devised to be robust to missing dwell time values. In this line, we consider *partial observations* of the form $p_{0:k}, \tau_{0:k-1}$ representing a finite sequence $p_0\tau_0 \cdots \tau_{k-1}p_k$ of consecutive dwell time values and atomic propositions observed during a random execution of $\mathcal{M}$. Here, for uniformity of treatment, the dwell times τ_t that are missing are denoted as $\tau_t = \emptyset$.

We follow a maximum likelihood approach: the parameters $\mathbf{x}$ are estimated to maximize the joint likelihood $\mathcal{L}(\mathcal{P}(\mathbf{x})|\mathcal{O}) = \prod_{j=1}^{J} l(\mathbf{o}_j|\mathcal{P}(\mathbf{x}))$ of the observed data. When $\mathcal{P}$ and $\mathcal{O}$ are clear from the context, we write $\mathcal{L}(\mathbf{x})$ for the joint likelihood and $l(\mathbf{o}|\mathbf{x})$ for the likelihood of the observation $\mathbf{o}$.

According to the assumption that some dwell time values may be missing, the likelihood of a partial observation $\mathbf{o} = p_{0:k}, \tau_{0:k-1}$ for a generic CTMC $\mathcal{M}$ is

$$l(\mathbf{o}|\mathcal{M}) = \sum_{s_{0:k}} P[S_{0:k} = s_{0:k}, L_{0:k} \ni p_{0:k}|\mathcal{M}] \cdot l(S_{0:k} = s_{0:k}, T_{0:k-1} = \tau_{0:k-1}|\mathcal{M})$$

$$= \sum_{s_{0:k}} \left([\![\ell(s_{0:k}) \ni p_{0:k}]\!] \prod_{t=0}^{k-1} R(s_t, s_{t+1})/E(s_t) \right) \left(\prod_{t \in \mathcal{T}(\mathbf{o})} E(s_t)\, e^{-E(s_t)\tau_t} \right), \quad (3)$$

where $\mathcal{T}(\mathbf{o}) = \{t \mid 1 \leq t < k, \tau_t \neq \emptyset\}$ denotes the subset of indices of the observation $\mathbf{o}$ that correspond to actual dwell time measurement.

Our solution to the maximum likelihood estimation problem builds on the MM optimization framework [31,32]. In this line, our algorithm starts with an initial hypothesis $\mathbf{x}_0$ and iteratively improves the current hypothesis $\mathbf{x}_m$, in the sense that the likelihood associated with the next hypothesis $\mathbf{x}_{m+1}$ enjoys the inequality $\mathcal{L}(\mathbf{x}_m) \leq \mathcal{L}(\mathbf{x}_{m+1})$. The procedure terminates when the improvement does not exceed a fixed threshold ϵ, namely when $\mathcal{L}(\mathbf{x}_m) - \mathcal{L}(\mathbf{x}_{m-1}) \leq \epsilon$.

Before proceeding with the formulation of the surrogate function, we find it convenient to introduce some notation. Let $\mathcal{P} = (S, \rightarrow, s_0, \ell)$, we write f_ρ for the rate function of a transition $\rho \in \rightarrow$, and write $s \rightarrow \cdot$ for the set of transitions departing from $s \in S$.

Without loss of generality, we assume that the rate function f_ρ of a transition is either a constant map, i.e., $f_\rho(\mathbf{x}) = c_\rho$ for some $c_\rho \geq 0$ or a map of the form $f_\rho(\mathbf{x}) = c_\rho \prod_{i=1}^{n} x_i^{a_{\rho i}}$ for some $c_\rho > 0$ and $a_{\rho i} > 0$ for some $i \in \{1, \ldots, n\}$; we write a_ρ for $\sum_{i=1}^{n} a_{\rho i}$. We denote by $\xrightarrow{c}$ the subset of transitions with constant rate function and $\xrightarrow{\mathbf{x}}$ for the remaining transitions.

To maximize $\mathcal{L}(\mathbf{x})$ we propose to employ an MM algorithm based on the following surrogate function $g(\mathbf{x}|\mathbf{x}_m) = \sum_{i=1}^{n} g(x_i|\mathbf{x}_m)$ where

$$g(x_i|\mathbf{x}_m) = \sum_{\rho \in \xrightarrow{\mathbf{x}}} \xi_\rho a_{\rho i} \ln x_i - \sum_s \sum_{\rho \in s \xrightarrow{\mathbf{x}}} \frac{f_\rho(\mathbf{x}_m) a_{\rho i} \gamma_s}{a_\rho (x_{mi})^{a_\rho}} x_i^{a_\rho}. \tag{4}$$

Here the coefficients γ_s and ξ_ρ are respectively defined as

$$\gamma_s = \sum_{j=1}^{J} \sum_{t=0}^{k_j - 1} \gamma_s^j(t) \left([\![\tau_t^j \neq \emptyset]\!] \tau_t^j + [\![\tau_t^j = \emptyset]\!] E_m(s)^{-1} \right) \tag{5}$$

$$\xi_\rho = \sum_{j=1}^{J} \sum_{t=0}^{k_j - 1} \xi_\rho^j(t) \tag{6}$$

where $\gamma_s^j(t)$ denotes the likelihood that having observed $\mathbf{o}_j$ on a random execution of $\mathcal{P}(\mathbf{x}_m)$ the state $S_t = s$; and $\xi_\rho^j(t)$ is the likelihood that for such random execution the transition performed from state S_t is ρ.

The following theorem states that the surrogate function $g(\mathbf{x}|\mathbf{x}_m)$ is a minorizer of the log-likelihood relative to the observed dataset $\mathcal{O}$.

Theorem 1. *The surrogate function $g(\mathbf{x}|\mathbf{x}_m)$ minorizes $\ln \mathcal{L}(\mathbf{x})$ at $\mathbf{x}_m$ up to an irrelevant constant.*

By Theorem 1 and the fact that the logarithm is an increasing function, we obtain that the parameter valuation that achieves the maximum of $g(\mathbf{x}|\mathbf{x}_m)$ improves the current hypothesis $\mathbf{x}_m$ relative to likelihood function $\mathcal{L}(\mathbf{x})$.

Corollary 1. *Let $\mathbf{x}_{m+1} = \arg\max_{\mathbf{x}} g(\mathbf{x}|\mathbf{x}_m)$, then $\mathcal{L}(\mathbf{x}_m) \leq \mathcal{L}(\mathbf{x}_{m+1})$.*

The surrogate function $g(\mathbf{x}|\mathbf{x}_m)$ is easier to maximize than $\mathcal{L}(\mathbf{x})$ because its parameters are separated. Indeed, maximization of $g(\mathbf{x}|\mathbf{x}_m)$ is done by point-wise maximization of each univariate function $g(x_i|\mathbf{x}_m)$. This has two main advantages: first, it is easier to handle high-dimensional problems [31,32]; second, one can choose to fix the value of some parameters and perform the maximization of $g(\mathbf{x}|\mathbf{x}_m)$ only on the corresponding subexpressions $g(x_i|\mathbf{x}_m)$.

The maxima of $g(x_i|\mathbf{x}_m)$ are found among the *non-negative* roots[1] of the polynomial function $P_i \colon \mathbb{R} \to \mathbb{R}$

$$P_i(y) = \sum_s \sum_{\rho \in s \xrightarrow{\mathbf{x}}} \frac{f_\rho(\mathbf{x}_m) a_{\rho i} \gamma_s}{(x_{mi})^{a_\rho}} y^{a_\rho} - \sum_{\rho \in \xrightarrow{\mathbf{x}}} \xi_\rho a_{\rho i} \tag{7}$$

Remark 2. There are some cases when (7) admits a closed-form solution. For instance, when the parameter index i satisfies the property $\forall \rho \in \xrightarrow{\mathbf{x}}. a_{\rho i} > 0 \implies a_\rho = C$ for some constant $C \in \mathbb{N}$, then maximization of $g(x_i|\mathbf{x}_m)$ leads to the following update

$$x_{(m+1)i} = \left[\frac{(x_{mi})^C \sum_{\rho \in \xrightarrow{\mathbf{x}}} \xi_\rho a_{\rho i}}{\sum_s \sum_{\rho \in s \xrightarrow{\mathbf{x}}} f_\rho(\mathbf{x}_m) a_{\rho i} \gamma_s} \right]^{1/C}$$

[1] Note that P_i always admits non-negative roots. Indeed, $P_i(0) \leq 0$ and $P_i(M) > 0$ for $M > 0$ sufficiently large. Therefore, by the intermediate value theorem, there exists $y_0 \in [0, M)$ such that $P_i(y_0) = 0$.

A classic situation when the above condition is fulfilled occurs when all transitions ρ where x_i appear (i.e., $a_{\rho i} > 0$), the transition rate is $f_\rho(\mathbf{x}) = c_\rho x_i$ (i.e., $a_{\rho i} = a_\rho = 1$). In that case, the above equation simplifies to

$$x_{(m+1)i} = \frac{\sum_{\rho \in \xrightarrow{\mathbf{x}}} \xi_\rho}{\sum_s \sum_{\rho \in s \xrightarrow{\mathbf{x}}} c_\rho \gamma_s}$$

For example, the pCTMC associated with the SIR models in Fig. 1 satisfies the former property for all parameters, because all transition rates are expressions either of the form $c \cdot \texttt{plock} \cdot \texttt{beta}$ or the form $c \cdot \texttt{plock} \cdot \texttt{gamma}$ for some constant $c > 0$. Furthermore, if we fix the value of the parameter `plock` the remaining parameters satisfy the latter property. In Sect. 6, we will take advantage of this fact for our calculations. □

Finally, we show how to compute $\gamma_s^j(t)$ and $\xi_\rho^j(t)$ w.r.t. the observation $\mathbf{o}_j = p_0^j \tau_0^j \cdots \tau_{k_j-1}^j p_{k_j}^j$ by using standard forward and backward procedures. We define the forward function $\alpha_s^j(t)$ and the backward function $\beta_s^j(t)$ respectively as

$$\alpha_s^j(t) = l(L_{0:t} \ni p_{0:t}^j, T_{0:t} = \tau_{0:t}^j, S_t = s | \mathcal{P}(\mathbf{x}_m)) \,, \text{ and}$$
$$\beta_s^j(t) = l(L_{t+1:k_j} \ni p_{t+1:k_j}^j, T_{t+1:k_j-1} = \tau_{t+1:k_j-1}^j | S_t = s, \mathcal{P}(\mathbf{x}_m)) \,.$$

These can be computed using dynamic programming according to the following recurrences. Let $\mathcal{P}(\mathbf{x}_m) = (S, R, s_0, \ell)$, then

$$\alpha_s^j(t) = \begin{cases} [\![s = s_0]\!] \, \omega_s^j(t) & \text{if } t = 0 \\ \omega_s^j(t) \sum_{s' \in S} \frac{R(s',s)}{E(s')} \, \alpha_{s'}^j(t-1) & \text{if } 0 < t \leq k_j \end{cases} \tag{8}$$

$$\beta_s^j(t) = \begin{cases} 1 & \text{if } t = k_j \\ \sum_{s' \in S} \frac{R(s,s')}{E(s)} \, \beta_{s'}^j(t+1) \, \omega_{s'}^j(t+1) & \text{if } 0 \leq t < k_j \end{cases} \tag{9}$$

where

$$\omega_s^j(t) = \begin{cases} [\![\ell(s) \in p_t^j]\!] E(s) e^{-E(s)\tau_t^j} & \text{if } 0 \leq t < k_j \text{ and } \tau_t^j \neq \emptyset \\ [\![\ell(s) \in p_t^j]\!] & \text{if } t = k_j \text{ or } \tau_t^j = \emptyset. \end{cases} \tag{10}$$

Finally, for $s \in S$ and $\rho = (s \xrightarrow{f_\rho} s')$, $\gamma_s^j(t)$ and $\xi_\rho^j(t)$ are related to the forward and backward functions as follows

$$\gamma_s^j(t) = \frac{\alpha_s^j(t) \, \beta_s^j(t)}{\sum_{s' \in S} \alpha_{s'}^j(t) \, \beta_{s'}^j(t)} \,, \quad \xi_\rho^j(t) = \frac{\alpha_s^j(t) f_\rho(\mathbf{x}_m) \, \omega_{s'}^j(t+1) \, \beta_{s'}^j(t+1)}{E(s) \sum_{s'' \in S} \alpha_{s''}^j(t) \, \beta_{s''}^j(t)} \,. \tag{11}$$

The Case of Non-timed Observations. Consider the limit situation when dwell time variables are not observable (i.e., $\tau_t^j = \emptyset$ for all $j = 1 \ldots J$ and $t = 1 \ldots k_j - 1$). Under this assumption, two CTMCs $\mathcal{M}_1$ and $\mathcal{M}_2$ having the same embedded Markov chain satisfy $\mathcal{L}(\mathcal{M}_1 | \mathcal{O}) = \mathcal{L}(\mathcal{M}_2 | \mathcal{O})$. In other words, when

dwell time variables are not observable the MLE objective does not fully capture the continuous-time aspects of the model under estimation.

The next section provides experimental evidence that, when the number of parametric transitions is sufficiently small relative to that of constant transitions, our algorithm can hinge on the value of the transition rates that are fixed, leading the procedure to converge to the real parameter values.

5 Experimental Evaluation

We implemented the algorithm from Sect. 4 as an extension of the `Jajapy` Python library [39], which has the advantage of being compatible with PRISM models. In this section, we present an empirical evaluation of the efficiency of our algorithm as well as the quality of their outcome. To this end, we employ a selection of CTMCs from the QComp benchmark set [22]. Experiments on each model have been designed according to the following setup.

For each model, we selected a set of parameters to be estimated as well as the set of observable atomic propositions[2]. We then estimated the parameter values from a training set consisting of 100 observation sequences of length 30, generated by simulating the original benchmark model. After the estimation, we verify all the formulas associated with the given benchmark model and compare the result with the expected one.

We perform experiments both using timed and non-timed observations. Each experiment is repeated 10 times by randomly re-sampling the initial parameter values $\mathbf{x}_0$ in the range $[0.00025, 0.0025]$. We annotate the running time, the relative error δ_i for each parameter x_i, and the relative error Φ_i for each formula[3].

Table 1. Performance comparison on selected QComp benchmarks [22].

Model	$\lvert S\rvert$	$\lvert\rightarrow\rvert$	Timed Observations				Non-timed Observations			
			Time(s)	Iter	avg δ	avg Φ	Time(s)	Iter	avg δ	avg Φ
polling	240	800	136.430	4	0.053	0.421	33.743	12	1.000	7.146
cluster	276	1120	132.278	3	0.089	1.293	279.853	12	0.313	3.827
tandem	780	2583	1047.746	3	0.043	0.544	4302.197	74	0.161	1.354
philosophers (i)	1065	4141	2404.803	3	0.043	0.119	2232.706	6	0.263	0.235
philosophers (ii)	1065	4141	9865.645	12	0.032	0.026	33265.151	200	0.870	2.573

Table 1 reports the aggregated results of the experiments. The columns $|S|$ and $|\rightarrow|$ provide respectively the number of states and transitions of the model;

[2] The models are available at https://github.com/Rapfff/MM-PCTMC-benchmark-models. The source files contain a description of the parameters and what is observable.

[3] The relative error is $|e - r|/|r|$, where e (resp. r) is the estimated (resp. real) value.

the columns "Time" and "Iter" respectively report the average running time[4] and number of iterations; and the columns "avg δ" and "avg Φ" respectively report the average relative error of the estimated parameters and model checking outcomes. Unsurprisingly, the quality of the estimation is higher for timed observations. Despite in most cases the initial parameter valuation $\mathbf{x}_0$ being picked far from the real parameter values, our method is capable to get close to the expected parameter values by using relatively few observation sequences. Most of the formulas employed in the experiments compute expected accumulated rewards for a time horizon exceeding that of the used training set, as a consequence, also the error tends to build up. The issue can be tamed by having longer observations in the training set. Notably, for timed observations, each iteration is more expensive than non-timed ones, but the additional overhead is largely compensated by a consistently smaller number of iterations.

To understand how, for non-timed observation, the quality of the estimation varies based on the number of constant transitions we ran our algorithm on two variants of the `philosophers` model: (i) with the variable `gammax` as a constant; and (ii) with `gammax` as a parameter. The algorithm clearly benefits from the presence of constant transitions and it converges way faster to better estimates.

Figure 2 reports the results of the experiments performed on the tandem queueing network model from [23] for different sizes of the queue. Each experiment was repeated 10 times by randomly re-sampling the initial valuation $\mathbf{x}_0$ in the interval [0.1, 5.0]. Accordingly, measurements are presented together with their respective error bars. The graph of the running time (*cf.* Fig. 2 bottom) follows a quadratic curve in the number of states both for timed and non-timed observations. However, for non-timed observations, the variance of the measured running times tends to grow with the size of the model. Figure 2 (top) shows how the L_1-norm (resp. L_∞-norm) of the vector $\delta = (\delta_i)$ may vary for different size of the model. The variance of the measured relative errors is larger in the experiments performed with non-timed observations. Notably, for timed observations, the quality of the estimation remained stable despite the size of the model increased relative to the size of the training set. This may be explained by the fact that, in the tandem model, the parameters occur in many transitions.

6 Case Study: SIR Modeling of Pandemic

In this section, we take as a case study the modeling pipeline proposed by Milazzo [36] for the analysis and simulation in Prism of the spread of COVID-19 in presence of lockdown countermeasures. The modeling pipeline includes: (i) parameter estimation from real data based on a modified SIR model described by means of a system of ODEs; (ii) encoding of the modified SIR model into as a Prism model; and (iii) stochastic simulation and model checking with Prism.

The model devised in step (ii) is depicted in Fig. 1 (left). However, to perform the analysis, Milazzo had to apply "a couple of modeling tricks (variable prun-

[4] Experiments were performed on a Linux machine with an AMD-Ryzen 9 3900X 12-Core processor and 32 GB of RAM.

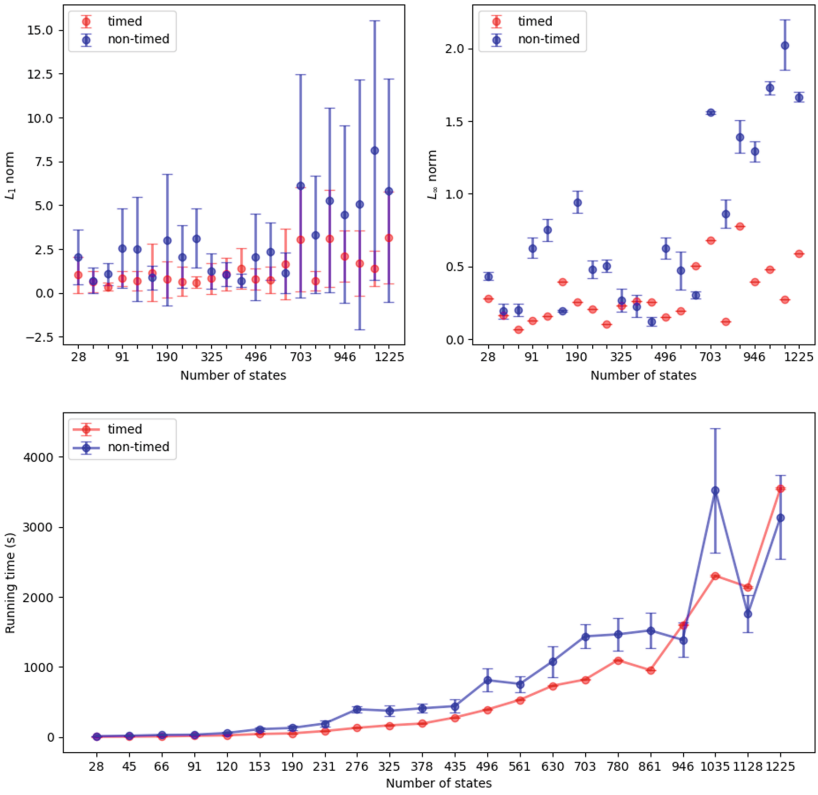

Fig. 2. Comparison of the performance of the estimation for timed and non-timed observations on the tandem queueing network with different size of the queue.

ing and upper bounds) that allowed state space of the model [..] to be reduced by several orders of magnitude. " [36]. These kinds of modeling tricks are not uncommon in formal verification, but they require the modeler to ensure that the parameter values estimated for the original model are still valid in the approximated one. In this section, we showcase the use of our algorithm to simplify this task. Specifically, we generate two training sets by simulating the SIR model in Fig. 1 using PRISM and, based on that, we re-estimate `beta`, `gamma`, and `plock` on an approximated version of the model (*cf.* Fig. 3).

The first training set represents the spread of the disease without lockdown (i.e., `plock` $= 1$), while the second one is obtained by fixing the value of `plock` estimated in [36] (i.e., `plock` $= 0.472081$). In line with the data set used in [36], both training sets consist of one (timed) observation reporting the number of infected individuals for a period of 30 days.

The estimation of the parameters `beta`, `gamma` and `plock` is performed on the model depicted in Fig. 3. As in [36], we use an approximated version of the original SIR model (*cf.* Fig. 1) obtained by employing a few modeling tricks:

```
ctmc
 // bounds
const int ubound_i; const int lbound_i; const int nb_r = 10;
const double size_r = 500/nb_r; const int SIZE = 100000;
const double beta; const double gamma; const double plock;  // SIR model parameters

module SIR
    i : [lbound_i..ubound_i] init 48;
    r : [0..nb_r − 1] init 0;

    [infection] i>0 & i <ubound_i  → i ∗ (SIZE − (i + (r + 0.5) ∗ size_r)) : (i′=i + 1);
    [recovery]  i>0 & r<nb_r − 1  → i ∗ ((size_r) − 1)/(size_r) : (i′=i − 1);
    [recovery]  i>0 & r<nb_r − 1  → i ∗ 1/(size_r) : (r′=r + 1) & (i′=i − 1);
    [recovery]  i>0 & r=nb_r − 1  → i : (i′=i − 1);
endmodule

module Rates
    [infection] true → beta ∗ plock/SIZE : true;
    [recovery]  true → gamma ∗ plock : true;
endmodule
```

Fig. 3. Approximated SIR model.

Table 2. Parameter estimation on the approximated SIR model.

Parameter	Expected Value	Estimated Value	Absolute Error
`beta`	0.122128	0.135541	0.013413
`gamma`	0.127283	0.128495	0.001212
`plock`	0.472081	0.437500	0.034581

variable pruning, set upper bounds on the state variable `i`, and re-scaling of the variable `r` in the interval $[0, \texttt{nb\_r} - 1]$. These modeling tricks have the effect to reduce the state space of the underlying CTMC, speeding-up in this way parameter estimation and the following model analysis.

We perform the estimation in two steps. First, we estimate the values of `beta` and `gamma` on the first training set with `plock` set to 1 (i.e., with no restrictions). Then, we estimate the value of `plock` on the second training set with `beta` and `gamma` set to the values estimated in the first step. Each step was repeated 10 times by randomly re-sampling the initial values of each unknown parameter in the interval $[0, 1]$. Table 2 reports the average estimated values and absolute errors relative to each parameter. The running time of each estimation was on average 89.94 seconds[5].

Notably, we were able to achieve accurate estimations of all the parameters from training sets consisting of a single partially-observable execution of the original SIR model. As observed in Sect. 5, this may be due to the fact that each parameter occurs in many transitions.

This case study demonstrates that our estimation procedure can be effectively used to simplify modeling pipelines that involve successive modifications of the model and the re-estimation of its parameter values.

[5] Experiments were performed on a Linux machine with an AMD-Ryzen 9 3900X 12-Core processor and 32 GB of RAM.

7 Related Work

Literature on parameter estimation for CTMCs follows two approaches. The first approach is based on Bayesian inference and assumes a probability distribution over parameters which in turn produces an *uncertain* CTMC [5,20]. In this line of work, Georgoulas et al. [19,20] proposed ProPPA, a stochastic process algebra with inference capabilities. Using probabilistic inference, the ProPPA model is combined with the observations to derive updated probability distributions over rates. Uncertain pCTMCs require dedicated model checking techniques [5].

The second approach aims at estimating parameter values producing concrete CTMCs via maximum likelihood estimation. In this line, Geisweiller proposed EMPEPA [18], an expectation-maximization algorithm that estimates the rate values inside a PEPA model. Wei et al. [43] learn the infinitesimal generator of a continuous-time hidden Markov model by first employing the Baum-Welch algorithm [38] to estimate the transition probability matrix of its (embedded) hidden Markov model from a set of periodic observations.

A large body of literature studies parameter estimation for stochastic reaction networks (SRN) (*cf.* [35,40] and references therein). According to Gillespie's theory of stochastic chemical kinetics, SRNs can be represented using CTMCs with states in $\mathbb{N}^d$. An SRN describes the dynamics of a population of d chemical species by means of a number of *chemical reaction rules*. Notably, the SIR model in Fig. 1 was encoded from an SRN. The parameter estimation problem for SRNs focuses on estimating the rate values associated with each reaction rule. Due to the nature of the models, estimation algorithms for SRNs aim at being able to scale on the number of species as well as the size of the population. In this respect, (i) Andreychenko et al. [2] employs numerical approximations of the likelihood function (and its derivatives) w.r.t. reaction rate constants by dynamically truncating the state space in an on-the-fly fashion, considering only those states that significantly contribute to the likelihood in a given time interval, while (ii) Bayer et al. [8] combines the Monte Carlo version of the expectation-maximization algorithm [34] with the forward-reverse technique developed in [9] to efficiently simulate SRN bridges conditional on the observed data. Compared with our method, the estimation algorithms of [2,8] scale better in the number of species and population size. The main limitation of the efficiency of our method is the computation of the forward and backward functions, whose complexity grows quadratically in the number of states. In contrast with our technique both [2] and [8] assume to observe *all* the coordinates of the state (in [2] these are additionally assumed to be subject to Gaussian noise). In our opinion, this limits the applicability of their methods to scenarios where states are partially observed. Such an example is the case study of Sect. 6 where the available data set was only reporting the number of infected individuals (i.e., two components out of three were not observable). Daigle et al. [27] developed an efficient version of the Monte Carlo expectation-maximization technique which employs modified cross-entropy methods to account for rare events. Notably, their technique is executable also on data sets with missing species but, as for our algorithm, such flexibility comes at the expense of efficiency. As pointed out in [8], the techniques

used in [27] have some analogies with those of [8], but its run-time performance is comparably slower than that of [8]: the algorithm of [27] took 8.7 days for the parameter estimation on an SRN describing an auto-regulatory gene network with five species, while the method of [8] took 2 days[6].

Sen et al. [41] presented a variant of ALERGIA [11] to learn a (transition-labeled) CTMC from timed observations. In contrast with our work, [41] does not perform parameter estimation over structured models, but learns an unstructured CTMC. Hence, [41] suits better for learning a single component CTMC when no assumption can be made on the structure or the size of the model.

Another related line of research is parameter synthesis of Markov models [26]. In particular, [12,21] consider parametric CTMCs, but are generally restricted to a few parameters. In contrast with our work, parameter synthesis revolves around the problem of finding (some or all) parameter instantiations of the model that satisfy a given logical specification.

8 Conclusion and Future Work

We presented a novel technique to estimate parameter values of CTMCs expressed as PRISM models from partially-observable executions. We demonstrated, with a case study, that our solution is a concrete aid in applications involving modeling and analysis, especially when the model under study requires successive approximations that require re-estimation of the parameters. The major strengths of our algorithm are (i) its interoperability with the model checking tools PRISMand STORM, and (ii) the fact that it accepts partially-observable data sets where both state and dwell times can be missing; However, the generality of our approach comes at the expense of efficiency. The computations of the forward and backward functions which are required to update the coefficients of the surrogate function (4) have a time and space complexity that grows quadratically in the number of states of the pCTMC, thus limiting the number of components that our implementation can currently handle. In future work, we consider investigating how to speed up the computation of the forward and backward functions either by integrating GPU-accelerated techniques from [37] or by replacing their exact computation in favor of numerical approximations obtained through Monte Carlo simulations in line with the idea employed in Monte Carlo EM algorithm [34].

Notably, the algorithm presented in this paper was devised following simple optimization principles borrowed from the MM optimization framework. We suggest that similar techniques can be employed to other modeling languages (e.g., Markov automata [16,17]) and metric-based approximate minimization [3,7]. An interesting future direction of research consists in extending our techniques to Markov decision processes by integrating the active learning strategies [4].

[6] Details on the experiments can be found in the respective papers.

References

1. Alur, R., Henzinger, T.A.: Reactive modules. Formal Methods Syst. Des. **15**(1), 7–48 (1999). https://doi.org/10.1023/A:1008739929481
2. Andreychenko, A., Mikeev, L., Spieler, D., Wolf, V.: Parameter identification for Markov models of biochemical reactions. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 83–98. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_8
3. Bacci, G., Bacci, G., Larsen, K.G., Mardare, R.: On the metric-based approximate minimization of Markov chains. In: Chatzigiannakis, I., Indyk, P., Kuhn, F., Muscholl, A. (eds.) 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, 10–14 July 2017, Warsaw, Poland. LIPIcs, vol. 80, pp. 104:1–104:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ICALP.2017.104
4. Bacci, G., Ingólfsdóttir, A., Larsen, K.G., Reynouard, R.: Active learning of markov decision processes using Baum-Welch algorithm. In: Wani, M.A., Sethi, I.K., Shi, W., Qu, G., Raicu, D.S., Jin, R. (eds.) 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021, pp. 1203–1208. IEEE (2021). https://doi.org/10.1109/ICMLA52953.2021.00195
5. Badings, T.S., Jansen, N., Junges, S., Stoelinga, M., Volk, M.: Sampling-based verification of CTMCs with uncertain rates. In: Shoham, S., Vizel, Y. (eds.) CAV 2022, Part II. LNCS, vol. 13372, pp. 26–47. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13188-2_2
6. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Software Eng. **29**(6), 524–541 (2003). https://doi.org/10.1109/TSE.2003.1205180
7. Balle, B., Lacroce, C., Panangaden, P., Precup, D., Rabusseau, G.: Optimal spectral-norm approximate minimization of weighted finite automata. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, 12–16 July 2021, Glasgow, Scotland (Virtual Conference). LIPIcs, vol. 198, pp. 118:1–118:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.118
8. Bayer, C., Moraes, A., Tempone, R., Vilanova, P.: An efficient forward-reverse expectation-maximization algorithm for statistical inference in stochastic reaction networks. Stoch. Anal. Appl. **34**(2), 193–231 (2016). https://doi.org/10.1080/07362994.2015.1116396
9. Bayer, C., Schoenmakers, J.: Simulation of forward-reverse stochastic representations for conditional diffusions. Ann. Appl. Probab. **24**(5), 1994–2032 (2014). https://doi.org/10.1214/13-AAP969
10. Calinescu, R., Ceska, M., Gerasimou, S., Kwiatkowska, M., Paoletti, N.: Efficient synthesis of robust models for stochastic systems. J. Syst. Softw. **143**, 140–158 (2018). https://doi.org/10.1016/j.jss.2018.05.013
11. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS, vol. 862, pp. 139–152. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58473-0_144

12. Češka, M., Dannenberg, F., Paoletti, N., Kwiatkowska, M., Brim, L.: Precise parameter synthesis for stochastic biochemical systems. Acta Informatica **54**(6), 589–623 (2016). https://doi.org/10.1007/s00236-016-0265-2
13. Ciocchetta, F., Hillston, J.: Bio-PEPA: a framework for the modelling and analysis of biological systems. Theor. Comput. Sci. **410**(33–34), 3065–3084 (2009). https://doi.org/10.1016/j.tcs.2009.02.037
14. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
15. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. **39**(1), 1–38 (1977)
16. Eisentraut, C., Hermanns, H., Zhang, L.: Concurrency and composition in a stochastic world. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 21–39. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_3
17. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11–14 July 2010, Edinburgh, United Kingdom, pp. 342–351. IEEE Computer Society (2010). https://doi.org/10.1109/LICS.2010.41
18. Geisweiller, N.: Finding the most likely values inside a PEPA model according to partially observable executions. Ph.D. thesis, LAAS (2006)
19. Georgoulas, A., Hillston, J., Milios, D., Sanguinetti, G.: Probabilistic programming process algebra. In: Norman, G., Sanders, W. (eds.) QEST 2014. LNCS, vol. 8657, pp. 249–264. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10696-0_21
20. Georgoulas, A., Hillston, J., Sanguinetti, G.: Proppa: probabilistic programming for stochastic dynamical systems. ACM Trans. Model. Comput. Simul. **28**(1), 3:1–3:23 (2018). https://doi.org/10.1145/3154392
21. Han, T., Katoen, J., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS 2008, Barcelona, Spain, 30 November–3 December 2008, pp. 173–182. IEEE Computer Society (2008). https://doi.org/10.1109/RTSS.2008.19
22. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 344–350. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_20
23. Hermanns, H., Meyer-Kayser, J., Siegle, M.: Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In: Plateau, B., Stewart, W., Silva, M. (eds.) Proceedings of 3rd International Workshop on Numerical Solution of Markov Chains (NSMC 1999), pp. 188–207. Prensas Universitarias de Zaragoza (1999)
24. Hillston, J.: A compositional approach to performance modelling. Ph.D. thesis, University of Edinburgh, UK (1994). http://hdl.handle.net/1842/15027
25. Jamshidian, M., Jennrich, R.I.: Acceleration of the EM algorithm by using quasi-newton methods. J. Roy. Stat. Soc. Ser. B (Methodol.) **59**(3), 569–587 (1997). http://www.jstor.org/stable/2346010

26. Jansen, N., Junges, S., Katoen, J.: Parameter synthesis in Markov models: a gentle survey. In: Raskin, J., Chatterjee, K., Doyen, L., Majumdar, R. (eds.) Principles of Systems Design. LNCS, vol. 13660, pp. 407–437. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22337-2_20
27. Daigle, B.J., Roh, M.K., Petzold, L.R., Niemi, J.: Accelerated maximum likelihood parameter estimation for stochastic biochemical systems. BMC Bioinform. **13**, 68 (2012). https://doi.org/10.1186/1471-2105-13-68
28. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72522-0_6
29. Kwiatkowska, M.Z., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. SIGMETRICS Perform. Evaluation Rev. **35**(4), 14–21 (2008). https://doi.org/10.1145/1364644.1364651
30. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
31. Lange, K.: Optimization, 2nd edn. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5838-8
32. Lange, K.: MM Optimization Algorithms. SIAM (2016). http://bookstore.siam.org/ot147/
33. Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C.: Quantitative System Performance - Computer System Analysis Using Queueing Network Models. Prentice Hall, Hoboken (1984)
34. Levine, R.A., Casella, G.: Implementations of the Monte Carlo EM algorithm. J. Comput. Graph. Stat. **10**(3), 422–439 (2001). http://www.jstor.org/stable/1391097
35. Loskot, P., Atitey, K., Mihaylova, L.: Comprehensive review of models and methods for inferences in bio-chemical reaction networks. Front. Genet. **10** (2019). https://doi.org/10.3389/fgene.2019.00549
36. Milazzo, P.: Analysis of COVID-19 data with PRISM: parameter estimation and SIR modelling. In: Bowles, J., Broccia, G., Nanni, M. (eds.) DataMod 2020. LNCS, vol. 12611, pp. 123–133. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70650-0_8
37. Ondel, L., Lam-Yee-Mui, L.M., Kocour, M., Corro, C.F., Burget, L.: GPU-accelerated forward-backward algorithm with application to lattice-free mmi. In: ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8417–8421 (2022). https://doi.org/10.1109/ICASSP43922.2022.9746824
38. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989). https://doi.org/10.1109/5.18626
39. Reynouard, R.: Jajapy (v 0.10) (2022). https://github.com/Rapfff/jajapy
40. Schnoerr, D., Sanguinetti, G., Grima, R.: Approximation and inference methods for stochastic biochemical kinetics-a tutorial review. J. Phys. A Math. Theor. **50**(9), 093001 (2017). https://doi.org/10.1088/1751-8121/aa54d9
41. Sen, K., Viswanathan, M., Agha, G.: Learning continuous time Markov chains from sample executions. In: 1st International Conference on Quantitative Evaluation of Systems (QEST 2004), pp. 146–155. IEEE Computer Society (2004). https://doi.org/10.1109/QEST.2004.1348029

42. Terwijn, S.A.: On the learnability of hidden Markov models. In: Adriaans, P., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 261–268. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45790-9_21
43. Wei, W., Wang, B., Towsley, D.F.: Continuous-time hidden Markov models for network performance evaluation. Perform. Evaluation **49**(1/4), 129–146 (2002)
44. Zhou, H., Alexander, D.H., Lange, K.: A quasi-Newton acceleration for high-dimensional optimization algorithms. Stat. Comput. **21**(2), 261–273 (2011). https://doi.org/10.1007/s11222-009-9166-3