Matching Distributions under Structural Constraints

Aaron Bies, Holger Hermanns, Maximilian A. Köhl, and Andreas Schmidt

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{bies,hermanns,koehl,andreas.schmidt}@cs.uni-saarland.de

Abstract. Phase-type distributions, the probability distributions generated by the time to absorption in a continuous-time Markov chain, are a popular tool for modeling time-dependent system behaviour. They are well understood mathematically, and so is the problem of identifying a matching distribution if given information about its moments, as well as fitting to a given distribution or a set of distribution samples. This paper looks at the problem of finding distributions from a structural perspective, namely where system behaviour is known to have a specific structure comprising parallelism, sequencing, and first-to-finish races. We present a general method that, given the coarse system structure with annotations regarding the moments of some fragments, finds a concrete phase-type distribution that fulfils the specification, if one exists. We develop the foundational underpinning in terms of constraint solving with satifiability modulo theories, spell out the algorithmic details of a divide-and-conquer solution approach, and provide empirical evidence of feasibility, presenting a prototypical solution engine for structural distribution matching.

1 Introduction

Continuous probability distributions are an important topic in statistics, stochastics, and data science because they allow to model and analyze data that are influenced by continuously varying quantities. This is particularly important for real-world phenomena where the continuous variable of interest is time.

Phase-type distributions are a class of continuous probability distributions that are often used in this context. A phase-type (PH) distribution can be thought of as describing the time to absorption in an absorbing continuous-time Markov chain (CTMC) if starting in its initial state at time zero. These distributions have been proven effective in modelling many time-dependent systems, since they can, in principle, represent any distributions on the positive axis with arbitrary precision [3], while the representation size is straightforward to control.

The behaviour of PH distributions is well understood. Given a PH distribution, one can compute the corresponding probability density function, cumulative distribution function as well as any statistical moment using closed

```rust
#[async_std::main]
async fn main() {
    let d1 = download_bytes("https://.../assets/orig.svg");              // X1
    let d2 = download_bytes("https://.../assets/orig.png");              // X2
    let d3 = download_bytes("https://.../assets/flat.png");              // X3

    let d4 = async {                                                     // X4.X5
        let text = download_text("https://.../assets/style.css").await;  // X4
        async_std::fs::write("style.css", &text).await.unwrap();         // X5
        text
    };

    let image = d1.race(d2).race(d3);                                    // X1 + X2 + X3
    let (image_bytes, css_text) = image.join(d4).await;                  // ... || X4.X5

    dbg!(image_bytes.len(), css_text.len());
}
```

Listing 1: An example program written in Rust

formulae. They are also composable, as they are closed under typical operations on distributions such as minimum, maximum, convolution, and mixture.

Finding a phase-type distribution for certain data involves estimating the parameters of the absorbing CTMC including its size, shape, and parameters. There are a number of sophisticated algorithms for doing so, which come in two flavors: They either fit to data points [1,9,10,14,23,21,4] available from sampling, or aim at matching the moments [13,2,11,22,12] of the distribution. They usually aim at finding a PH distribution that resembles the observed behaviour as accurately as possible, but is otherwise as small and simple as possible.

Now, when considering real-world applications of finding PH distributions, we might have more than only some measured characteristics at hand – and this is the twist we are considering worthwhile to investigate: In particular, we may also have insights about the coarse structure of the system. This may refer to the parallel or sequential nature of the program execution studied, together with measurements that relate to fragments of this structure.

Motivating Example. Consider the program in Lst. 1, which is written in Rust using the `async-std` library. Here, the functions `download_bytes` and `download_text` return futures (promises in other languages) that can be composed with each other to create structural concurrency. `race` runs two futures in parallel and outputs the result of the future that finished first. `join` runs two futures in parallel and returns both results, i.e., only finishes once both futures finish. After downloading the stylesheet, we also write its content to a local file.

Now suppose we have benchmarked the program and determined that the program takes 0.15 seconds to terminate on average. Additionally, we measured that the `style.css` file only takes about 0.1 seconds to download and save. Using these sparse measurements and our knowledge of the program structure, we would like to construct a complete model for the execution time of this program.

This program can be abstractly expressed as

$$\overline{(X_1+X_2+X_3) \parallel \overline{X_4.X_5}\big|_{E=0.1}}\big|_{E=0.15}$$

Here, we use some common process calculi operators $\{\,.\,,+,\parallel\}$ for sequence, choice, and parallelism, and $\overline{\Psi}\big|_{E=c}$ to constrain the expected value for the over-lined process Ψ to c. Choice and parallelism echo the Rust constructs `race` and `join`, the variables X_1 to X_4 represent the individual download durations, and X_5 represents the time it takes to store the file in the example.

We are not aware of methods that allow for such structural constraints to be taken into account. This paper pioneers the consideration of structural constraints in finding phase-type distributions, and it does so focusing on moment matching techniques. The paper develops the theory and algorithmic ingredients to fill in small PH distributions for X_1 to X_5 so that altogether the constraints present are satisfied. Using our prototype solver StruMPh, we find that the exponential distribution $Exp(3.25)$ for X_1 to X_3, approximately $Exp(10.965)$ for X_4, and approximately $Exp(113.655)$ for X_5 is a valid solution for this example.

Contribution. In this paper, we discuss the problem of finding a PH distribution subject to a set of structural and moment constraints. We restrict to the first two moments for the sake of simplicity, and need to only work with acyclic PH distributions (APH). To do so, we assume that structural knowledge determining the problem at hand is given in a formal language which we call CccSat—a variant of the Ccc process calculus [20] that exploits the closure properties discussed above. We discuss properties and transformations of CccSat problems that are leveraged by a general semi-decision procedure. This procedure decomposes the problem at hand in a divide-and-conquer manner and is guaranteed to find the smallest APH solution, if one exists.

In short, the contribution of this paper is fourfold: a) We formalize the problem of moment matching under structural constraints as a CccSat problem. b) We propose an approach for determining an APH distribution of smallest possible size solving a CccSat problem. c) We present a prototypical implementation in the tool StruMPh that embedds SMT solving into a problem specific iterative procedure. d) We report on empirical evidence regarding the practicality of the approach.

Organization of the Paper. In Sec. 2, we give related work and the mathematical background, after which we introduce our CccSat calculus in Sec. 3. Techniques to find constraint-satisfying variable assignments for CccSat expressions are discussed in Sec. 4. Our implementation StruMPh is used for evaluation in Sec. 5. Finally, Sec. 6 concludes the paper and provides an outlook.

2 Preliminaries

Moments. For a (continuous) random variable X, characterized by probability density function f_X, the *expected value of* X(also called mean) is $E(X) =$

$\int_{\mathbb{R}} x\, f_X(x)\, dx$, and its *k-th moment* is $E(X^k)$. The *variance* of X is $V(X) = E((X - E(X))^2) = E(X^2) - E(X)^2$, and the *coefficient of variation* c_X is defined as $\sqrt{V(X)}\,/\,E(X)$.

Furthermore, if D is a probability distribution and X is a random variable distributed according to D, then $E[D]$ refers to $E(X)$, and $V[D]$ refers to $V(X)$. We say a distribution is *Dirac* if it assigns full probability to a single outcome.

Phase-Type Distributions. A *Phase-Type Distribution* [17], is a probability distribution modeled by the time from start to absorption in an absorbing *continuous-time Markov chain* (CTMC). In an absorbing CTMC, every state is either *absorbing*, meaning it has no outgoing transitions, or *transient*, in which case it can reach an absorbing state. Note that an *acyclic* CTMC is always absorbing.

Definition 1. *The* (continuous) n-phase phase-type distribution $PH(\vec{\alpha}, \boldsymbol{A})$ *is the distribution of the* time-until-absorption *in an* $(n+1)$*-state CTMC* $(S, \boldsymbol{R}, \vec{\pi})$ *with parameters*

$$\vec{\pi} = (\vec{\alpha}, 0)\,, \qquad \boldsymbol{R} = \begin{pmatrix} \boldsymbol{A} & \vec{A} \\ \vec{0} & 0 \end{pmatrix}$$

where all but the last state are transient.

The tuple $(\vec{\alpha}, \boldsymbol{A})$ is commonly referred to as the *phase-type representation* of the associated PH distribution. The parameter n is called the *order* of the representation. A PH distribution is *acyclic* if it is represented by an acyclic CTMC.

These distributions are mathematically well-understood and easy to analyze. Given an n-phase PH distribution $PH(\vec{\alpha}, \boldsymbol{A})$, we can compute its

- cumulative distribution function (CDF) with $F(t) = 1 - \vec{\alpha} \exp(t\,\boldsymbol{A})\,\vec{1}$,
- probability density function (PDF) with $f(t) = \vec{\alpha} \exp(t\,\boldsymbol{A})\,\vec{A}$,
- Laplace-Stieltjes transform (LST) with $\tilde{f}(s) = \vec{\alpha}_{n+1} + \vec{\alpha}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\vec{A}$, and
- k-th moment with $E(X^k) = (-1)^k k!\, \vec{\alpha}\, \boldsymbol{A}^{-k} \vec{1}$ (see [18]).

Here $\exp(\cdot)$ refers to the matrix exponential. We remark that the column vector $\vec{A}$ can be computed from the $n \times n$-matrix $\boldsymbol{A}$ since every row in $\boldsymbol{R}$ must sum to zero. The set of (acyclic) phase-type distributions is closed under convolution, minimum, maximum, and mixture (aka convex combination) [18].

Cox & Convenience Calculus. To conveniently describe acyclic phase-type distributions (APH) in a structured manner, we recall the *Cox & Convenience Calculus* (Ccc) [20] – a simple stochastic process calculus, spanned by all expressions that adhere to the following grammar[1]

$$P,\ Q \ \in\ \mathcal{L} \quad ::= \quad (\lambda) \quad | \quad (\mu) \lhd (\lambda) P \quad | \quad P.Q \quad | \quad P + Q \quad | \quad P \parallel Q$$

[1] Listed here in order of operator precedence, from highest to lowest. The three binary operators are left-associative.

with rates $\lambda \in \mathbb{R}_+$ and $\mu \in \mathbb{R}_{\geq 0}$. Any Ccc process P represents an acyclic PH-distribution, formally denoted by $PH(P)$. We refer to [20] for details of the semantic mapping $PH(\cdot)$. The three binary operators sequence (.), choice (+), and parallel ($\|$) correspond to the convolution, minimum, and maximum on PH distribution respectively, exploiting the closure properties discussed above. Intuitively, given processes P and Q,

- $P.Q$ terminates once Q terminates which is started only after P terminates.
- $P \parallel Q$ terminates once the *last* of P and Q terminates, with both P and Q started at the same time.
- $P + Q$ terminates once the *first* of P and Q terminates, with both P and Q started at the same time.

The remaining two operators are used to model more primitive distributions, where (λ) stands for a simple exponential distribution, while the operator $(\mu) \lhd (\lambda)P$ (called the "Cox"-operator) assures completeness. It allows one to express any possible Cox[2] distribution [6], and hence any possible APH distribution [20]. Practically speaking, the following distribution families can be encoded into Ccc inductively, for $n > 0$. We will make use of their encodings in the sequel.

Exponential distribution: $\mathrm{Exp}(\lambda) = (\lambda)$;
Erlang distribution: $\mathrm{Erl}(n+1, \lambda) = (\lambda).\mathrm{Erl}(n, \lambda)$, and $\mathrm{Erl}(1, \lambda) = (\lambda)$;
Hypoexponential distribution: $\mathrm{Hypo}(\lambda_1, R) = (\lambda_1).\mathrm{Hypo}(R)$ where R is of shape $\lambda_2, \ldots, \lambda_n$, and $\mathrm{Hypo}(\lambda) = (\lambda)$;
Cox distribution: $\mathrm{Cx}([\lambda_1, p_1], R) = ((1-p_1)\lambda_1) \lhd (p_1\lambda_1).\mathrm{Cx}(R)$ where R is of shape $[\lambda_2, p_2], \ldots, [\lambda_{n-1}, p_{n-1}], \lambda_n$, and $\mathrm{Cx}(\lambda) = (\lambda)$.

Notably, choice (+) and parallel ($\|$) do not occur in the above encodings. They are added for modeling convenience as they echo typical usage scenarios.

Related Work. Moment matching, the problem of finding a PH distribution with a given set of moments, is well-explored. There have been too many contributions to list here, but some we would like to highlight.

Johnson and Taaffe investigated the use of mixtures of Erlang distributions of common order in the context of moment matching [13], showing among others that, except for corner cases, this class of phase-type distributions can be used to match the first k finite moments, for any choice of k. Asmussen et. al. developed an extended version of the expectation-maximization algorithm to approximate sampled data by phase-type distributions [1]. The paper [22] discusses minimal representations and moment matching methods for Markovian Arrival Processes (MAP), and [5] looks into similarity transformations and investigates representations maximizing the first joint moment of MAP. Building on those results, [4]

[2] A Cox distribution $PH(\vec{\alpha}, \boldsymbol{A})$ corresponds to an acyclic absorbing CTMC with bidiagonal $\boldsymbol{A}$, Dirac α, and ascending (but negative) diagonal. Pictorially it can be seen as a sequence of exponential distributions (like hypo-exponential distributions) of decreasing rate with escape options to the absorbing state at each state. Any APH distribution can be represented as a Cox distribution of equal order.

presents a variation of the EM algorithm, for online estimation of the parameters of PH distributions. Finally, for a summary of existing phase-type approximation techniques, we would like to highlight the Bachelor's thesis of Komárková [15].

The above methods let us construct PH distributions that fulfill given moment constraints, but they do not allow for structural constraints to be taken into account. We believe to be the first to explore PH distributions with structural and moment constraints in this manner.

3 The Ccc Satisfiability Problem (CccSat)

In this section, we will formally define the problem statement of this paper. To do so, we will first introduce CccSat, a derivative of the Ccc process calculus, and define its semantics as well as a notion of satisfiability.

Syntax. Based on the Ccc language, we define the following syntax.

Definition 2. *Let $\mathcal{P}$ be a language defined by the following grammar*

$$\Psi,\ \Phi\ \in\ \mathcal{P} \quad ::= \quad X \quad | \quad \overline{\Psi}|_{E=c} \quad | \quad \overline{\Psi}|_{V=c} \quad | \quad \Psi.\Phi \quad | \quad \Psi + \Phi \quad | \quad \Psi \parallel \Phi$$

where $X \in \mathrm{Var}$ is a variable for some Ccc process, and $c \in \mathbb{R}_{\geq 0}$ is a constant.

We will refer to an element of $\mathcal{P}$ as a CccSat *problem instance* or a *problem* for short. While $\overline{\Psi}|_{E=c}$ will constrain the expected value of Ψ (thus $E[\Psi]$) to be c, $\overline{\Psi}|_{V=c}$ will constrain the variance $V[\Psi]$ to be c.[3]

Semantics. We define a constraint-oblivious semantics for CccSat problems, assuming an assignment Γ assigning to each variable a specific Ccc process.

Definition 3. *Let $\Gamma \in \mathrm{Var} \to \mathcal{L}$ be an* assignment*. For problem instance $\Psi \in \mathcal{P}$, we define $[\![\Psi]\!]_\Gamma$ as the Ccc process acquired by the following substitutions.*

- *Variables are substituted, i.e., $[\![X]\!]_\Gamma = \Gamma(X)$ for any $X \in \mathrm{Var}$.*
- *Constraint operators are dropped, i.e., $[\![\,\overline{\Psi}|_{\dots}\,]\!]_\Gamma = [\![\Psi]\!]_\Gamma$ for any $\Psi \in \mathcal{P}$.*
- *$[\![\Psi \circ \Phi]\!]_\Gamma = [\![\Psi]\!]_\Gamma \circ [\![\Phi]\!]_\Gamma$ for any $\Psi, \Phi \in \mathcal{P}$ and $\circ \in \{\,.\,, +, \parallel\,\}$.*

By defining the semantics of $\mathcal{P}$ using the function $[\![\cdot]\!]_\Gamma \in \mathcal{P} \to \mathcal{L}$, many statements that have been shown for Ccc processes and PH distributions are immediately applicable to $\mathcal{P}$. In particular, the semantics of the three binary operators sequence (.), choice (+), and parallel ($\parallel$) are inherited from Ccc.

Satisfiability. The significance of the constraint operators that have been discarded so far manifests in the following notion of satisfiability.

[3] In all that follows, there will be no need to distinguish $\overline{\overline{\Psi}|_{E=c}}|_{V=d}$ from $\overline{\overline{\Psi}|_{V=d}}|_{E=c}$.

Definition 4. *For a given problem $\Psi \in \mathcal{P}$ and assignment $\Gamma \in \mathrm{Var} \to \mathcal{L}$, we say Γ* satisfies *Ψ, written as $\Gamma \vDash \Psi$, if the following hold*

$$\frac{}{\Gamma \vDash X} \qquad \frac{\Gamma \vDash \Psi \qquad E[\, PH(\llbracket \Psi \rrbracket_\Gamma)\,] = c}{\Gamma \vDash \overline{\Psi}|_{E=c}}$$

$$\frac{\Gamma \vDash \Psi \qquad \Gamma \vDash \Phi}{\Gamma \vDash \Psi \circ \Phi} \qquad \frac{\Gamma \vDash \Psi \qquad V[\, PH(\llbracket \Psi \rrbracket_\Gamma)\,] = c}{\Gamma \vDash \overline{\Psi}|_{V=c}}$$

with $X \in \mathrm{Var}$, problem instances $\Psi, \Phi \in \mathcal{P}$, and operators $\circ \in \{\,.\,, +, \parallel\}$.

Example 1. Consider the problem instance $\Psi \in \mathcal{P}$ defined as follows.

$$\Psi = \overline{\overline{X_1 + \overline{X_2}|_{E=2}}|_{V=3}}|_{E=5} \parallel \overline{X_3}|_{E=4}.X_4$$

By the definition above, the statement $\Gamma \vDash \Psi$ expands to the following.

$$E[\,\Gamma(X_1) + \Gamma(X_2)\,] = 5 \quad \wedge \quad V[\,\Gamma(X_1) + \Gamma(X_2)\,] = 3$$
$$\wedge \quad E[\,\Gamma(X_2)\,] = 2 \quad \wedge \quad E[\,\Gamma(X_3)\,] = 4$$

Note that some variables, such as X_1 and X_2, are constrained multiple times, which will make solving for those variables harder. On the other hand, X_3 can be solved for independently, as it appears in just one constraint by itself. It is easy to see that X_4 may be chosen arbitrarily, as it is completely unconstrained.

The goal of this paper is to develop techniques for finding an assignment Γ for a given problem instance Ψ, such that $\Gamma \vDash \Psi$, if one exists. We will see later that our methods tend to find the simplest solution for a given problem instance. We call a problem $\Psi \in \mathcal{P}$ *satisfiable*, denoted by $\mathrm{SAT}(\Psi)$, if there is an assignment Γ such that $\Gamma \vDash \Psi$. Otherwise we call Ψ *unsatisfiable*, denoted by $\mathrm{UNSAT}(\Psi)$.

4 Semi-Algorithm for Solving CccSat

Now that we have arrived at a formal problem statement, let us walk through how we go about its solution, i.e., given a problem instance $\Psi \in \mathcal{P}$, we want to find $\Gamma \in \mathrm{Var} \to \mathcal{L}$, such that $\Gamma \vDash \Psi$. Our approach is split up into three stages: a) Immediately handle cases that can be solved analytically (Sec. 4.1). b) Divide the problem (recursively) into smaller problems that can be solved independently, if possible (Sec. 4.2). c) Solve the residual cases using an SMT solver (Sec. 4.3).

4.1 Satisfying Simple Cases

First, we will cover some cases that can be solved analytically. In what follows, we abuse notation and abbreviate the assignment that maps each variable on the same constant process P just as P. We call a problem instance $\Psi \in \mathcal{P}$ *unconstrained*, if Ψ does not contain any constraint operator ($\overline{\cdot}|_{\ldots}$).

Theorem 1. *Let $\Psi \in \mathcal{P}$ be unconstrained. This implies*

1. *$\Gamma \vDash \Psi$ where Γ is arbitrary,*
2. *$\Gamma \vDash \overline{\Psi}|_{E=c}$ where $\Gamma = (\hat{c}/c)$ and $\hat{c} = E\big[[\![\Psi]\!]_{(1)}\big]$, and*
3. *$\Gamma \vDash \overline{\Psi}|_{V=d}$ where $\Gamma = (\sqrt{\hat{d}/d})$ and $\hat{d} = V\big[[\![\Psi]\!]_{(1)}\big]$.*

The first statement is obvious in light of how $\vDash$ is defined. For the remaining statements, it suffices to show that scaling all rates in a PH-representation scales the moments accordingly. For the first case, our implementation defaults to the constant assignment (1).

Remark 1. Special cases of Theorem 1 are the problems $\overline{X}|_{E=c}$ and $\overline{X}|_{V=d}$ respectively, where $X \in \mathrm{Var}$. These are well known to be solvable using exponential distributions with rates $\lambda = 1/c$ and $\lambda = 1/\sqrt{d}$, which aligns with the above theorems (since $Exp(1)$ has a mean and variance of one).

The case $\overline{\overline{X}|_{E=c}}|_{V=d}$ also has a well-known solution, which however we need to spend some thought on. Recall the squared coefficient of variation $c_X^2 = d\,/\,c^2$. For $c_X^2 \geq 0.5$, we can solve it with a 2-phase Coxian distribution $\mathrm{Cx}([\lambda_1, p_1], \lambda_2)$ where $\lambda_1 = 2/c$, $p_1 = 1\,/\,2c_X^2$ and $\lambda_2 = \lambda_1 p_1$, due to [16]. This gives us the Ccc process mapping $\Gamma(X) = ((1-p_1)\lambda_1) \lhd (p_1\lambda_1)(\lambda_2)$. For $c_X^2 < 0.5$, we would obtain $p_1 > 1$ however, which would result in negative rates, so an alternative method is required.

A popular approach to create PH distributions with $0 < c_X \leq 1$ is using a mixture of two Erlang distributions with identical rates and sizes differing by one [24]. Yet, due to the absence of a mixture operator and of non-Dirac initial distributions in Ccc, this solution cannot be expressed directly.

One way to handle this case is to instead prefix a 2-phase Coxian distribution with an Erlang distribution, i.e. using $\mathrm{Erl}(k, \lambda).\mathrm{Cx}([\lambda_1, p_1], \lambda_2)$. The idea is to use an Erlang distribution to offset the coefficient of variation required for the Coxian distribution, such that we can find it with the method outlined above. For the parameters k and λ of the Erlang distribution, we require that

$$\frac{d - \frac{k}{\lambda^2}}{\left(c - \frac{k}{\lambda}\right)^2} > \frac{1}{2}, \qquad d > \frac{k}{\lambda^2}, \qquad c > \frac{k}{\lambda}, \qquad 0 < \frac{d}{c^2} < \frac{1}{2}.$$

Due to the use of inequalities, there are infinitely many solutions. One possible solution is to fix $k = \left\lfloor \frac{c^2}{d} \right\rfloor$ and $\lambda = \frac{kc}{c^2 - 2d}$. The 2-phase Coxian distribution is constructed as described above with mean $c - k/\lambda$ and variance $d - k/\lambda^2$. These observations can be harvested by the following Ccc constructions, which are notable for being fully described by their size k and up to 4 rate parameters.

Definition 5. *We define the class of* Tailweight *distributions as follows.*

$$\begin{aligned} Tw(\lambda_1) &= (\lambda_1) \\ Tw([\lambda_3, \lambda_2], \lambda_1) &= (\lambda_3) \lhd (\lambda_2)(\lambda_1) \\ Tw([k, \lambda_4], [\lambda_3, \lambda_2], \lambda_1) &= Erl(k, \lambda_4).(\lambda_3) \lhd (\lambda_2)(\lambda_1) \end{aligned}$$

The above recipe for problems of the form $\overline{\overline{X}|_{E=c}}|_{V=d}$ (where $X \in \text{Var}$ and $c, d \in \mathbb{R}_+$) indeed always constructs a Tailweight distribution.

Corollary 1. $\forall c, d \in \mathbb{R}_+.\ \exists P \in \textit{Tailweight}.\ E[P] = c \wedge V[P] = d.$

4.2 Problem Decomposition

Next, we discuss how we can break large problem instances down into multiple smaller problems that can be solved independently. Let $\circ$ denote any binary operator of $\mathcal{P}$, i.e., $\circ \in \{\,.\,, +, \|\,\}$. For $\Psi \in \mathcal{P}$, we define $\text{Vars}(\Psi)$ as the set of variables in Ψ. Further, we say $\Psi, \Phi \in \mathcal{P}$ are *disjoint*, if $\text{Vars}(\Psi) \cap \text{Vars}(\Phi) = \emptyset$.

Disjointness seems like a strong assumption to make, yet many problem instances we care about in practice use any variable only once. When modeling for instance a production chain, we want to represent each step in the process by a unique variable. The reuse of a variable asserts that two steps have exactly the same distribution, which is hard to guarantee in practice. Therefore, it is not uncommon for sub-expressions of a problem instance to be pairwise disjoint.

Theorem 2. $\forall \Psi, \Phi \in \mathcal{P}.\ \textit{disjoint}(\Psi, \Phi) \wedge \textit{SAT}(\Psi) \wedge \textit{SAT}(\Phi) \rightarrow \textit{SAT}(\Psi \circ \Phi)$

This holds since the function $\Gamma_\Psi \cup \Gamma_\Phi$ is well-formed due to disjointness, and solves $\Psi \circ \Phi$. Using the above theorem, we can decompose problem instances where the topmost operation is a binary operator and work on both operands independently, as long as they are disjoint. Note that this theorem can be applied recursively, and thus truly is the basis for a divide-and-conquer approach.

Theorem 3. *Using only Theorems 1 and 2, we can solve all* CccSat *problems containing neither nested constraints nor repeated variables.*

The above can be shown by structural induction on $\mathcal{P}$. Yet, Theorem 2 only allows us to decompose until we reach the first constraint operator. To deal with constraints, additional techniques are required.

Remark 2. To show that $\Psi \in \mathcal{P}$ is UNSAT, it suffices to show that any of its sub-expressions is UNSAT. This means while refuting Ψ, we are permitted to check all sub-expressions independently.

- $\forall \Psi, \Phi \in \mathcal{P}.\ \text{UNSAT}(\Psi) \vee \text{UNSAT}(\Phi) \rightarrow \text{UNSAT}(\Psi \circ \Phi)$
- $\forall \Psi \in \mathcal{P}.\ \text{UNSAT}(\Psi) \rightarrow \text{UNSAT}(\overline{\Psi}|_{\ldots})$

Constraint propagation through sequences. The following theorem allows us to rewrite specific problems involving the sequence operator between four forms.

Theorem 4. *For problems $\Psi, \Phi \in \mathcal{P}$, assignment Γ and property $p \in \{E, V\}$ the following statements are pairwise equivalent:*

$$\Gamma \vDash \overline{\overline{\Psi}|_{p=c}.\overline{\Phi}|_{p=d}}|_{p=c+d} \qquad \Gamma \vDash \overline{\overline{\Psi}|_{p=c}.\Phi}|_{p=c+d}$$

$$\Gamma \vDash \overline{\Psi.\overline{\Phi}|_{p=d}}|_{p=c+d} \qquad \Gamma \vDash \overline{\Psi}|_{p=c}.\overline{\Phi}|_{p=d}$$

This follows from the fact that both the mean and variance are cumulative. The last form gives us an unconstrained sequence operator at the top of the expression, which potentially allows us to decompose the problem further using Theorem 2. Once again, this theorem can be applied recursively to push constraints further into the problem.

Remark 3. Theorem 4 also yields some rejection criteria for CccSat problems. In cases where all three constraints around a sequence operator are given but do not add up as shown, the problem admits no solutions.

If only two constraints are given, we may still reject the problem if we infer that the third constraint must be negative, since the mean and variance of PH distributions are always positive.

Example 2. Consider $\Psi = \overline{X_1 . \overline{X_2}\big|_{E=5}}\big|_{E=3}$. Due to Theorem 4, $\Gamma \vDash \Psi$ is equivalent to $\Gamma \vDash \overline{X_1}\big|_{E=-2} . \overline{X_2}\big|_{E=5}$, which has no solution.

Choice and parallel are less well-behaved. If the remaining two operators behaved like the sequence operator (.), i.e., if a theorem similar to Theorem 4 could be shown for operators choice (+) and parallel ($\|$), we would be able to push all constraints to the innermost sub-expressions of a given CccSat problem. This means problems could be broken down recursively until only variable expressions with constraints remain, which can be easily solved using the methods discussed in Sec. 4.1. Alas, no such rule exists. We will outline why this is the case here.

Theorem 5. *For unknown $P, Q \in \mathcal{L}$, $E[P + Q]$, $V[P + Q]$, $E[P \parallel Q]$ and $V[P \parallel Q]$ are not fully determined by $E[P]$, $V[P]$, $E[Q]$ and $V[Q]$.*

Counterexamples are easy to find, for instance by setting Q to (2) and comparing $P = (1).(1/3)$ and $P' = (1/10) \lhd (2/5)(2/5)$. Here $E[P] = E[P']$, and $V[P] = V[P']$ hold, but the compound expressions appearing in the theorem give different values if P is replaced by P'. As a result, we will need to treat the moments of choice (+) and parallel ($\|$) as completely opaque in this paper.

Remark 4. The above is equivalent to showing that equality of the first two moments does not yield a congruence relation on Ccc.

PH-equivalence ($\approx_{PH}$) on the other hand is a congruence relation for all operators of Ccc [19] and defined as $\forall P, Q \in \mathcal{L}.\ PH(P) = PH(Q) \rightarrow P \approx_{PH} Q$. Using [25], two Ccc processes are PH-equivalent if and only if their first $2n$ moments agree, where n is the order of the larger PH representation.

4.3 Reduction to SMT Instances

We now present a general procedure for solving a CccSat problem $\Psi \in \mathcal{P}$ based on iteration over *template assignments* $\Gamma_\Lambda \in \mathrm{Var} \rightarrow \mathcal{L}[\Lambda]$, i.e., assignments with rate parameters $\Lambda \in (\mathbb{R}_{\geq 0})^d$ for a template with d parameters to fill in. For each such template assignment, we present an encoding of the statement $\Gamma_\Lambda \vDash \Psi$ as a system of equations to be solved for Λ by an SMT solver. We show that this procedure is exhaustive, i.e., guaranteed to find a solution Γ for Ψ, if one exists, and that it finds the smallest solution for each variable.

Enumerating assignments. To iterate over template assignments, we need a function $\gamma \in \mathbb{N} \to \mathrm{Var} \to \mathcal{L}[\Lambda]$ producing a template assignment $\Gamma_\Lambda \in \mathrm{Var} \to \mathcal{L}[\Lambda]$ in each step $i \in \mathbb{N}$. We first construct *template generators* $\tau \in \mathbb{N}_+ \to \mathcal{L}[\Lambda]$ and then discuss how to use them to iterate over template assignments.

For templates, we make use of Coxian distributions because they are canonical forms for general APH distributions, i.e., every APH distribution can be transformed into a Coxian distribution while preserving its CDF [7]. This allows us to restrict our search to the set of Coxian distributions without missing out on possible solutions which guarantees exhaustiveness.

Definition 6. *We define the Coxian* template generator $\tau_{\mathrm{Cx}} \in \mathbb{N}_+ \to \mathcal{L}[\Lambda]$ *by*

$$\tau_{\mathrm{Cx}}(1) = (\lambda_1), \qquad \tau_{\mathrm{Cx}}(j+1) = (\lambda_{2j+1}) \lhd (\lambda_{2j})\, \tau_{\mathrm{Cx}}(j).$$

While τ_{Cx} covers all APH distributions, the resulting templates $\tau_{\mathrm{Cx}}(j)$ have $2j-1$ rate parameters λ_1 to λ_{2j-1}, so here $\Lambda \in (\mathbb{R}_{\geq 0})^{2j-1}$. Our experiments (Sec. 5) show that the number of rate parameters has a significant impact on SMT solving time. Hence, we also define the following alternative template generators with fewer parameters. In contrast to the Coxian generator, using them does, however, not guarantee exhaustiveness.

Definition 7. *We define the Erlang (τ_{Erl}), the Hypoexponential (τ_{Hypo}), and the Tailweight (τ_{Tw}) template generators as follows.*

$$\tau_{\mathrm{Erl}}(1) = (\lambda_1), \quad \tau_{\mathrm{Erl}}(j+1) = (\lambda_1).\tau_{\mathrm{Erl}}(j)$$

$$\tau_{\mathrm{Hypo}}(1) = (\lambda_1), \quad \tau_{\mathrm{Hypo}}(j+1) = (\lambda_{j+1}).\tau_{\mathrm{Hypo}}(j)$$

$$\tau_{\mathrm{Tw}}(1) = (\lambda_1), \quad \tau_{\mathrm{Tw}}(2) = (\lambda_3) \lhd (\lambda_2)(\lambda_1), \quad \tau_{\mathrm{Tw}}(j+2) = (\lambda_4).\tau_{\mathrm{Tw}}(j+1)$$

To lift a template generator τ to a function $\gamma \in \mathbb{N} \to \mathrm{Var} \to \mathcal{L}[\Lambda]$ for generating template assignments, we use linear search thereby starting with the smallest assignment and working our way upward. Note that for some of the template generators defined above, there are distributions that can only be represented using a specific order, i.e. parameter j. Thus, we must consider all possible combinations of template orders for each variable. Further, we want to choose the sequence of template assignments such that we find the smallest solution for each variable first. To this end, we define the following auxiliary function f.

$$f(1,s) = \big\{(s)\big\}, \qquad f(N,s) = \bigcup_{x=0}^{s} \big\{(x,t) \mid t \in f(N-1, s-x)\big\}$$

Here, $f(N,s)$ is the set of all tuples $\mathbb{N}^N$ whose elements sum up to $s \in \mathbb{N}$. Now, we take the tuples from $f(|\mathrm{Vars}(\Psi)|, s)$ for increasing values of s, obtaining a sequence of tuples with monotonically increasing sums.[4] Finally, by mapping

[4] As the tuples within each set have equal sums, the order in which we linearise each set when taking tuples is arbitrary.

every number x in each of the tuple to $\tau(x+1)$, we get our enumeration of template assignments $\gamma \in \mathbb{N} \to \mathrm{Var} \to \mathcal{L}[\Lambda]$.

This algorithm is guaranteed to find the smallest APH distribution for each variable, but as the set of templates we need to search is countably infinite, the algorithm diverges if Ψ is UNSAT.

Remark 5. Rather than minimizing the order of templates, i.e., the order of the CCC processes substituted for the variables, one may be interested in the assignment that produces the smallest process once all variables are substituted. Using [18], the order of CCC processes can be computed recursively.

$$\mathrm{ord}(P.Q) = \mathrm{ord}(P) + \mathrm{ord}(Q), \qquad \mathrm{ord}(P+Q) = \mathrm{ord}(P) \cdot \mathrm{ord}(Q),$$
$$\mathrm{ord}(P \parallel Q) = \mathrm{ord}(P) \cdot \mathrm{ord}(Q) + \mathrm{ord}(P) + \mathrm{ord}(Q)$$

Finding the next tuple in the sequence then amounts to minimizing a multivariate polynomial over the natural numbers while not repeating solutions.

SMT encoding. So far, we defined a linear search function $\gamma \in \mathbb{N} \to \mathrm{Var} \to \mathcal{L}[\Lambda]$ which gives us a template assignment $\Gamma_\Lambda = \gamma(i)$ for every step i in our search. Now, assuming $\Gamma_\Lambda \in \mathrm{Var} \to \mathcal{L}[\Lambda]$ is given, we need to encode $\Gamma_\Lambda \vDash \Psi$ into a system of equations, which can be solved using an SMT solver. As described in Sec. 3, this can be done by generating an equation for every occurrence of the constraint operator $(\overline{\cdot}\rceil_{\ldots})$ where the left-hand side is the computed moment of $[\![\Psi]\!]_{\Gamma_\Lambda}$ and the right-hand side is a constant given by the constraint. The final SMT encoding is the conjunction of all generated equations.

To compute the k-th moment of a random variable $X \sim PH(\vec{\alpha}, \boldsymbol{A})$, we can use the equation $E(X^k) = (-1)^k k!\, \vec{\alpha}\, \boldsymbol{A}^{-k} \vec{1}$ mentioned in Sec. 2. As remarked in [25], we can avoid the matrix inverse by computing the first k moments iteratively. First we solve for $\vec{\beta}_1$ and continue by inductively solving for $\vec{\beta}_k$ in

$$\vec{\beta}_1 \boldsymbol{A} = -\vec{\alpha}, \qquad \vec{\beta}_{k+1} \boldsymbol{A} = -(k+1)\vec{\beta}_k.$$

In every step, we can compute the k-th moment $E(X^k) = \vec{\beta}_k \vec{1}$. Since the PH-representations constructed from CCC processes are acyclic, $\boldsymbol{A}$ is an upper-triangular matrix. This means if $\boldsymbol{A}$ is of order n, each $\vec{\beta}_k$ can be computed using backward substitution in $\mathcal{O}(n^2)$.

5 Empirical Evaluation

We have implemented the methods discussed in Sec. 4 in a prototypical solver for CCCSAT problems, called STRUMPH (Structural Matching of PH-distributions). The problems we consider are more general than those that can be attacked by moment matching or phase-type fitting methods (unless constraint are present on the outermost level only). Therefore no meaningful comparison with existing tools is possible. This section provides empirical observations of the solution of STRUMPH on a selection of example problems.

Overview. StruMPh is a CLI application written in Rust, which takes a problem instance either in the CccSat syntax or a simplified JSON format and returns a valid assignment to stdout, and stores it as a JSON dictionary. We use the rsmt2 library to interface with any solver that conforms to the SMT-LIB v2 standard.

The benchmarks in this section have been performed on a desktop PC with an Intel Core i7-6700K CPU at 4 GHz and 24 GB of RAM running Windows 10. We use the SMT solver `z3` [8] as a backend. To measure execution times of our solver, we use the `Measure-Command` utility that comes with Powershell. Executions taking longer than 1 hour were considered as timeout.

Decomposable Cases. The easiest CccSat problems for our solver are those where constraints do not overlap, i.e., every subexpression is part of at most one constraint. In this case, our implementation can always decompose the problem until each subproblem only has one constraint, which makes it possible to use moment matching for each of them.

Example 3. The following problem can be solved by decomposing it into three smaller problems, which are solved independently using moment matching (see Theorem 1). Our implementation solves this in about 15 ms.

$$\overline{X_1.X_2}\big|_{E=5} \parallel \overline{X_3}\big|_{V=3}.X_4$$

If a problem does not have any constraints, our implementation outputs the assignment $\Gamma = (1)$ immediately.

Large sequential problems. In the subset of CccSat problems constructed without choice ($+$) and parallel ($\|$), the solver performs well even on larger problem instances. For problems with many moment constraints, StruMPh quickly manages to propagate constraints to the innermost subexpressions which are then solved by moment matching.

Example 4. Our implementation can solve problem Ψ below only using constraint propagation and moment matching in about 103 ms.

$$\begin{aligned}
\Psi_1 &= \overline{\overline{\overline{X_1}\big|_{E=2}}\big|_{V=3/2}.X_2}\Big|_{E=4}\Big|_{V=3} \\
\Psi_2 &= \overline{\overline{\overline{\overline{X_3}\big|_{E=1/2}}\big|_{V=1}.X_4}\Big|_{E=5}\Big|_{V=2}.\,\overline{\overline{X_5}\big|_{E=2}}\big|_{V=2}.X_6}\Big|_{E=10}\Big|_{V=5} \\
\Psi_3 &= \overline{\overline{X_7}\big|_{E=1/2}}\big|_{V=1}.X_8 \\
\Psi &= \overline{\overline{\Psi_1.\Psi_2.\Psi_3}\big|_{E=16}}\big|_{V=10}
\end{aligned}$$

If constraints are too sparse, StruMPh must resort to SMT solving, but the SMT instances generated in these cases are easily solved.

Example 5. Consider the following CccSat problem.

$$\Psi = \overline{X_1.\overline{X_2.X_3.X_4.X_5.X_6.X_7}\big|_{V=3}.X_8}\Big|_{E=5}$$

	Ψ_n									Ψ'_n					Ψ''_n		
n	1	2	3	4	5	6	7	8	9	1	2	3	4	5	1	2	3
Erl.	39	75	117	185	372	534	639	812	962	39	87	130	4545	-	42	148	-
Hypo.	41	74	138	268	291	690	2400	-	-	45	72	121	4436	-	40	271	-
Tail.	42	66	147	323	462	-	-	-	-	37	89	126	-	-	41	158	-
Cox.	54	73	344	-	-	-	-	-	-	38	74	464	-	-	47	169	-

Table 1. Solution times for varying sizes n measured in ms or timeout (-).

When we lower $\Gamma \vDash \Psi$ into an SMT problem with $\Gamma(X_k) = (\lambda_k)$, we get the following system of equations.

$$\frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} + \frac{1}{\lambda_4} + \frac{1}{\lambda_5} + \frac{1}{\lambda_6} + \frac{1}{\lambda_7} + \frac{1}{\lambda_8} = 5,$$
$$\frac{1}{\lambda_2^2} + \frac{1}{\lambda_3^2} + \frac{1}{\lambda_4^2} + \frac{1}{\lambda_5^2} + \frac{1}{\lambda_6^2} + \frac{1}{\lambda_7^2} = 3,$$

Using `z3`, we find the following solution (consistently in roughly 58 ms).

$$\lambda_1 = \lambda_4 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = 2, \qquad \lambda_2 = \lambda_3 = 1$$

Challenging constraints in sequences. We now turn to synthetic cases that are meant to pose challenges to the solution engine. We consider the following example parametric in n.

$$\Psi_n = \overline{X_2.\ \overline{X_1\big|_{E=n}}\big|_{V=n}.X_2}\Big|_{E=n+1}$$

By construction we know that the smallest solution for Ψ_n is given by $\Gamma(X_1) = Erl(n, 1)$ and $\Gamma(X_2) = (2)$. The outer constraint and variable X_2 however prevent StruMPh from decomposing the problem[5] and solving it within a few milliseconds. Table 1 shows solution times for Ψ_n using different templates. As we can see, the number of parameters as well as the size of the templates has a significant impact on the execution times of our solver.

We also notice that our solver gets stuck for over an hour on certain problems, even though it can solve just slightly smaller problems within a few seconds. After further investigation, we find the z3 SMT solver is quick to reject incorrect template assignments, but stalls while trying to compute the exact rates for the correct solution. We believe this is due to the SMT solver internally choosing the wrong strategy for those cases, yet we had little success figuring out what causes this behavior or how to prevent it.

Choice & Parallel. To investigate the performance of the other operators, we repeat the above experiment with two slightly modified versions of Ψ_n:

$$\Psi'_n = \overline{\overline{X_1\big|_{E=n}}\big|_{V=n} + X_2}\Big|_{E=n/2} \qquad \Psi''_n = \overline{\overline{X_1\big|_{E=n}}\big|_{V=n} \parallel X_2}\Big|_{E=n+1}$$

[5] Specifically, decomposition fails here because the occurrences of variable X_2 makes the subexpressions of the sequence non-disjoint.

As above, the problem is deliberately constructed as a challenge for the solver. It must be solved using templates of size at least n, since choice ($+$) and parallel ($\|$) do not allow for constraints to be propagated. The smallest solution for X_1 is $Erl(n, 1)$. Table 1 displays the solution times for both Ψ'_n and Ψ''_n. In both cases, our implementation reaches the timeout a lot sooner than in the previous example for most templates. Since choice and parallel quickly result in CTMCs of higher order, the equations for both mean and variance quickly become more complex and harder for the SMT solver to handle. We expect that a variety of tailored strategies can be devised to overcome this. Notably the issue is entirely absent if parallel processes are decomposable, enabling divide-and-conquer.

6 Conclusion

In this paper, we discussed the problem of finding a PH distribution that fulfils structural and moment constraints. To formalize the problem, we introduced CccSat, a derivative of the Ccc process calculus. We discussed how to decompose CccSat problems into smaller problems, how to transform problems into one another, and how to apply existing moment matching methods to solve base cases. This discussion has culminated in the creation of a general semi-decision procedure, which is guaranteed to find the smallest acyclic solution to any given problem, if one exists. We presented StruMPh, a prototypical implementation of this procedure, and studied its performance on challenging cases.

Future Work. Programs with structural concurrency, such as the one shown in Lst. 1, can be translated into CccSat in a very literal and direct way. We believe it is possible for this translation to be performed completely automatic, and this could also interface with benchmarking utilities to infer moment constraints. This is especially interesting in the context of Rust, due to its rich type system and macro support.

The methods for solving CccSat problems discussed in this paper assume moment constraints must be matched exactly. Yet in reality, these constraints are likely acquired via real-world measurements, which are notoriously inexact. This leaves room for follow-up work, for instance on a CccSat solver which emphasizes solution size and speed over solution accuracy.

Acknowledgements. This work has received support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 389792660 – TRR 248 – CPEC, see `https://perspicuous-computing.science`.

References

1. Asmussen, S., Nerman, O., Olsson, M.: Fitting phase-type distributions via the em algorithm. Scandinavian Journal of Statistics **23**(4), 419–441 (1996), `http://www.jstor.org/stable/4616418`
2. Bobbio, A., András Horváth and Miklós Telek: Matching three moments with minimal acyclic phase type distributions. Stochastic Models **21**(2-3), 303–326 (2005). https://doi.org/10.1081/STM-200056210
3. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Steady-State Solutions of Markov Chains, chap. 3, pp. 103–151. John Wiley and Sons, Ltd (1998). https://doi.org/10.1002/0471200581.ch3
4. Buchholz, P., Dohndorf, I., Kriege, J.: An online approach to estimate parameters of phase-type distributions. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 100–111 (June 2019). https://doi.org/10.1109/DSN.2019.00024
5. Buchholz, P., Felko, I., Kriege, J.: Transformation of acyclic phase type distributions for correlation fitting. In: Dudin, A., De Turck, K. (eds.) Analytical and Stochastic Modeling Techniques and Applications. pp. 96–111. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39408-9_8
6. Cox, D.R.: A use of complex probabilities in the theory of stochastic processes. Mathematical Proceedings of the Cambridge Philosophical Society **51**(2), 313–319 (1955). https://doi.org/10.1017/S0305004100030231
7. Cumani, A.: On the canonical representation of homogeneous markov processes modelling failure-time distributions. Microelectronics Reliability **22**(3), 583–602 (1982). https://doi.org/10.1016/0026-2714(82)90033-6
8. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
9. Feldmann, A., Whitt, W.: Fitting mixtures of exponentials to long-tail distributions to analyze network. Perform. Evaluation **31**(3-4), 245–279 (1998). https://doi.org/10.1016/S0166-5316(97)00003-5
10. Horváth, A., Telek, M.: Phfit: A general phase-type fitting tool. In: Field, T., Harrison, P.G., Bradley, J.T., Harder, U. (eds.) Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2324, pp. 82–91. Springer (2002). https://doi.org/10.1007/3-540-46029-2_5
11. Horváth, A., Telek, M.: Matching more than three moments with acyclic phase type distributions. Stochastic Models **23**(2), 167–194 (2007). https://doi.org/10.1080/15326340701300712
12. Horváth, G.: Moment matching-based distribution fitting with generalized hyper-erlang distributions. In: Dudin, A.N., Turck, K.D. (eds.) Analytical and Stochastic Modelling Techniques and Applications - 20th International Conference, ASMTA 2013, Ghent, Belgium, July 8-10, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7984, pp. 232–246. Springer (2013). https://doi.org/10.1007/978-3-642-39408-9_17
13. Johnson, M.A., Taaffe, M.R.: Matching moments to phase distributions: Mixtures of erlang distributions of common order. Communications in Statistics. Stochastic Models **5**(4), 711–743 (1989). https://doi.org/10.1080/15326348908807131

14. Khayari, R.E.A., Sadre, R., Haverkort, B.R.: Fitting world-wide web request traces with the em-algorithm. Perform. Evaluation **52**(2-3), 175–191 (2003). https://doi.org/10.1016/S0166-5316(02)00179-7
15. Komárková, Z.: Phase-type approximation techniques (2012), `https://is.muni.cz/th/ysfsq/thesis.pdf`, Bachelor Thesis, Masaryk University, Faculty of Informatics
16. Marie, R.A.: Calculating equilibrium probabilities for λ(n)/ck/1/n queues. In: Proceedings of the 1980 International Symposium on Computer Performance Modelling, Measurement and Evaluation. pp. 117–125. PERFORMANCE '80, Association for Computing Machinery, New York, NY, USA (1980). https://doi.org/10.1145/800199.806155
17. Neuts, M.F.: Probability distributions of phase type. Liber Amicorum Prof. Emeritus H. Florin (1975)
18. Neuts, M.F.: Matrix-geometric solutions in stochastic models: An algorithmic approach. The Johns Hopkins University Press (1981). https://doi.org/10.1002/net.3230130219
19. Pulungan, M.R.: Reduction of Acyclic Phase-Type Representations. Ph.D. thesis, Saarland University (2009). https://doi.org/10.22028/D291-25951
20. Pulungan, R., Hermanns, H.: A construction and minimization service for continuous probability distributions. Int. J. Softw. Tools Technol. Transf. **17**(1), 77–90 (2015). https://doi.org/10.1007/s10009-013-0296-8
21. Reinecke, P., Krauß, T., Wolter, K.: Phase-type fitting using hyperstar. In: Balsamo, M.S., Knottenbelt, W.J., Marin, A. (eds.) Computer Performance Engineering - 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8168, pp. 164–175. Springer (2013). https://doi.org/10.1007/978-3-642-40725-3_13
22. Telek, M., Horváth, G.: A minimal representation of markov arrival processes and a moments matching method. Perform. Evaluation **64**(9-12), 1153–1168 (2007). https://doi.org/10.1016/j.peva.2007.06.001
23. Thümmler, A., Buchholz, P., Telek, M.: A novel approach for phase-type fitting with the EM algorithm. IEEE Trans. Dependable Secur. Comput. **3**(3), 245–258 (2006). https://doi.org/10.1109/TDSC.2006.27
24. Tijms, H.C.: Stochastic models : an algorithmic approach. John Wiley & Sons, New York (1994)
25. Wolf, V.: Equivalences on Phase Type Processes. Ph.D. thesis, University of Mannheim (2008), `https://madoc.bib.uni-mannheim.de/1911/`