KAUNAS UNIVERSITY OF TECHNOLOGY

MINDAUGAS BRAŽĖNAS

INVESTIGATION OF PHASE-TYPE DISTRIBUTION OF FOURTH ORDER STRUCTURES FOR FITTING AND MARKOV ARRIVAL PROCESS FITTING BY EXPECTATION MAXIMIZATION ALGORITHM PARALLELIZATION

Doctoral Dissertation
Natural Sciences, Informatics (N 009)

2019, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Mathematics and Natural Sciences, Department of Mathematical Modelling during the period of 2014–2018.

Scientific Supervisor

Prof. Dr. Habil. Eimutis VALAKEVIČIUS (Kaunas University of Technology, Natural Sciences, Informatics, N 009).

Doctoral dissertation has been published in:
http://ktu.edu

Editor:

Armandas Rumšas (Publishing Office "Technologija")

ISBN 978-609-02-1608-8

The bibliographic information about the publication is available in the National Bibliographic Data Bank (NBDB) of Martynas Mažvydas National Library of Lithuania.

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MINDAUGAS BRAŽĖNAS

KETVIRTOS EILĖS FAZINIŲ SKIRSTINIŲ STRUKTŪRŲ ANALIZĖ IR MARKOVO ATVYKIMŲ PROCESO PARAMETRŲ PAIEŠKOS MAKSIMALAUS TIKĖTINUMO METODU ALGORITMO IŠLYGIAGRETINIMAS

Daktaro disertacija
Gamtos mokslai, informatika (N 009)

2019, Kaunas

Disertacija rengta 2014–2018 metais Kauno technologijos universiteto Matematikos ir gamtos mokslų fakultete Matematinio modeliavimo katedroje.

Mokslinis vadovas:

Prof. habil. dr. Eimutis VALAKEVIČIUS (Kauno technologijos universitetas, Gamtos mokslai, Informatika, N 009).

Interneto svetainės, kurioje skelbiama disertacija, adresas:
http://ktu.edu

Editor:

Armandas Rumšas (Publishing Office “Technologija”)

ISBN 978-609-02-1608-8

Leidinio bibliografinė informacija pateikiama Lietuvos nacionalinės Martyno Mažvydo bibliotekos Nacionalinės bibliografijos duomenų banke (NBDB).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

The bold capital letters are used to denote matrices, e.g., $\boldsymbol{Q}$, the $(i,j)^{\text{th}}$ element of matrix is $\boldsymbol{Q}_{i,j}$. For vectors, bold lowercase letters are used, e.g., $\boldsymbol{\alpha}$, $\boldsymbol{d}$, whose elements are $\boldsymbol{\alpha}_i$, $\boldsymbol{d}_i$. The matrices and vectors themselves can be enumerated in different ways, e.g., $\boldsymbol{D_0}$, $\boldsymbol{P}[\begin{smallmatrix} v \\ u \end{smallmatrix}]$, $\boldsymbol{d_1}$, $\boldsymbol{a}[u]$. For random variables, capital letters of calligraphic font are used, e.g., $\mathcal{X}$. Random variables can be enumerated as $\mathcal{B}_i$, $\mathcal{N}_{i,j}$. For sets of random variables, bold capital letters of calligraphic font are used, e.g., $\boldsymbol{\mathcal{H}} = \{\mathcal{B}_i, \mathcal{N}_{i,j}, \mathcal{Z}_i\}$. The expectation of a set of random variables is denoted by bold lowercase letters, e.g., $\boldsymbol{h} = \{b_i, n_{i,j}, z_i\}$. A similar notation follows for lists, for example, the notation $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \dots)$ stands for a list of random variables, and their realizations are $\boldsymbol{x} = (x_1, x_2, \dots)$. Sets of parameters are denoted by brackets, e.g., $(\boldsymbol{\alpha}, \boldsymbol{D_0})$.

The following, more specific, cases of notation are used:

r.v. – random variable,

r.a.d. – relative absolute difference, for any two numbers x, y is defined as $\frac{|y-x|}{|x|}$,

$\mathbb{P}(...)$ – probability,

$\mathbb{E}[...]$ – expectation,

$\boldsymbol{Q}$ – transition rate matrix of CTMC process,

$\boldsymbol{\alpha}$ – initial state probability vector,

$\overline{\boldsymbol{\alpha}}$ – stationary state probability vector of CTMC process,

$\boldsymbol{D_0}$ – transition rates which do not generate arrival events,

$\boldsymbol{D_1}$ – transition rates which generate arrival events,

$\boldsymbol{d_1}$ – termination rates for PH distributions,

$\boldsymbol{d}$ – termination rates for TMAP processes,

$\boldsymbol{r}$ – Erlang branch orders,

$\boldsymbol{\lambda}$ – Erlang branch transition rates,

$\boldsymbol{\pi}$ – Erlang branch initial probabilities,

$\boldsymbol{\Pi}$ – Erlang branch switching probabilities,

$\mathbb{1}$ – a column vector of ones of the appropriate size,

$\boldsymbol{e_i}$ – a row vector of zeros except for i^{th} element, which is 1.

Abbreviations, terms:

CTMC – continuous time Markov chain,

DTMC – discrete time Markov chain,

CHMM – continuous time hidden Markov model,

PH – Phase-type distribution,

ER – Erlang distribution,

HErD – hyper-Erlang distribution,

MAP – Markov Arrival Process,

TMAP – Transient Markov Arrival Process,

ER-CHMM – MAP or TMAP process of hyper-Erlang structure,

ML – Maximum likelihood method,

EM – Expectation Maximization method,

GPU – Graphics Processing Unit,

CPU – Central Processing Unit,

llh – log-likelihood.

INTRODUCTION

Continuous time finite state space Markov process has been used to model stochastic arrival processes for decades. This type of modeling is attractive due to its analytical tractability. The evolution of Markov process depends only on the current system state (i.e., it depends on its past only via this state). This property is known as Markovian property. The holding times of the states are exponentially distributed. The remaining holding time of exponential distribution does not depend on the elapsed time and has the same exponential distribution. This is called as memoryless property of exponential distribution and implies constant rates, i.e., such a distribution is neither aging nor non-aging. However, real life phenomena exhibit diverse behavior including the necessity of modeling non-exponential holding times. The accuracy of modeling non-Markovian arrival processes depends on the accuracy of approximation of the distribution by mixtures and convolutions of exponential distributions. Such distributions are said to be of Phase-type (PH).

Any distribution of a positive random variable can be approximated by a Phase-type distribution. In scientific literature, a lot of effort is being invested in solving PH representational and parameter searching problems. Canonical forms are significant for effective parameter fitting and application. However, these forms are known only for low order PH(n) distributions, i.e., $n = 2, 3$. For higher orders, the analytical analysis of general PH distribution structures is complicated. Instead, properties of various PH sub-classes are investigated and used for parameter fitting. Despite the fact that these sub-classes are not optimal and might require more phases to approximate a given distribution, it has been found that these distribution sub-classes are more applicable for practical use. One of the most limiting Markovian modeling problems is a rapid state space increase to the point where it becomes intractable. This problem can be tackled by using denser, preferably canonical structures.

Phase-type distributions are used to model independent inter-arrival times, however, in practice, certain processes show some kind of dependency. In general, a dependency structure can be very complex, and the model ability to capture it is essential for obtaining realistic statistical properties. For this purpose, the extension of PH distributions, i.e., the Markov Arrival Processes (MAPs) are used. Due to even more complex analytical structural analysis, canonical forms are known only for MAP(n) of order $n = 2$. The parameter fitting problem is more complex compared to PH fitting since autocorrelation also has to be captured.

In the scientific literature, there are few references to the general PH distribution structures because of the complexity of analytical analysis. We investigate PH(4) structures by generating all possible structures and eliminating the redundant ones based on the current results in the field. The obtained generated structure sub-classes are investigated empirically by approximating various benchmark distributions by PH distributions of these structures.

For MAPs, a more pressing issue is the fitting effectiveness. More specifically, in order to capture the autocorrelation structure, long data traces have to be used. There are no documented parallel algorithms for parallel MAP fitting. In order to fulfill this gap, we have implemented various parallel algorithms, including their modifications, and analyzed their execution properties. Some ideas for parallelization have been taken from the analogous CHMM (Continuous time Hidden Markov Model) fitting procedures. Also, we have derived an algorithm for transient Markov arrival process (TMAP) fitting by applying the Expectation-maximization method. Finally, for comparison purposes, we have implemented parallel MAP, TMAP parameter fitting algorithms for GPU computing architecture.

Research Object

Phase-type distributions, Markov arrival processes and transient Markov arrival processes parameter fitting by using the Expectation-maximization method.

Research Aim

To investigate the general Phase-type distribution fitting with a sufficient number of transitions, and to develop parallel algorithms for Markov arrival processes and transient Markov arrival processes fitting.

Research Tasks

1. To develop an algorithm for Phase-type structure generation.
2. To research Phase-type fitting of various structure sub-classes.
3. To develop parallel algorithms for Markov arrival process fitting and to compare their execution properties.
4. To develop an Expectation-maximization algorithm and its parallel version for transient Markov arrival process fitting.
5. To implement parallel algorithms for Markov arrival processes, transient Markov arrival processes fitting for execution on GPU device using CUDA library. To solve numerical issues which arise when huge data sets are fitted.

Methods and software

1. Descriptive statistics has been used to analyze research results.
2. Computations have been performed by programs written in the C++ programming language. Several libraries have been used, i.e., ‘Eigen’ for matrix computations and ‘Boosts’ for statistical distributions.
3. The research programs have been built by the ‘CMake’ tool in the Linux environment. For programs to be executed on a GPU device, the CUDA library has been used.

Scientific Novelty and Practical Relevance

Canonical forms for PH(n) distributions are known only for orders of $n = 2, 3$. By applying a Laplace–Stieltjes transform to the PH distribution density function, it can be shown that PH(n) distribution can be specified by $p = 2n-1$ independent parameters. However, for practical applicability, the PH representation of the matrix form is required. It is convenient to characterize the structure of the matrix form representation by the number of transitions m (the maximum number of independent parameters is $p = m - 1$). The full PH distribution space can be covered by matrix form representations of structures with $m = 2n + \Delta m_n$ transitions, where $0 \leq \Delta m_n \leq \Delta M_n$. In general, the maximum number of additional transitions ΔM_n is not known, except for $\Delta M_2 = 0$ and $\Delta M_3 = 1$. We expect that structures with $m = 2n$ transitions (i.e., $p = 2n-1$ parameters) are sufficient for practical fitting purposes. We formulate a hypothesis that a PH distribution which has a matrix form representation with additional transitions (i.e., $0 < \Delta m_n \leq \Delta M_n$, can be closely approximated by a similar PH distribution without additional transitions (i.e., $\Delta m_n = 0$). An equivalent hypothesis has not been formulated in scientific literature previously. We validate it empirically, for case $n = 4$. In order to achieve that, we have developed a Phase-type distribution matrix form representation structure generation algorithm. The algorithm outputs a set of structures which are used to approximate nine distributions given in related scientific literature. Based on fitting, via the EM method, the results pertaining to the hypothesis are validated.

In scientific literature, the use of certain Phase-type distribution matrix form representation structure classes for more effective fitting is being investigated. However, one specific structure class cannot cover the whole parameter space. In order to search for parameters in the optimal manner, canonical forms have to be used. The canonical forms via the analytical approach are discovered for Phase-type distributions of order $n \leq 3$. We have proposed an alternative combinatorial approach to generate a structure set. It cannot be guaranteed that the obtained structure set is minimal, however, it can be used for further investigation in search for canonical forms.

However, our Phase-type structure generation algorithm is not effective, and it is hardly applicable for higher orders ($n > 4$) due to a huge number of initial structures. Thus we have compared Phase-type fitting using generated structures ($n = 4$) versus the randomly generates sparse ($m = 2n$) structures. Such a comparison in scientific literature could not have been done before because of absence of an analogous structure generation algorithm.

In scientific literature, there is no documented investigation of fitting with general sparse (with $m = 2n$ transitions) structures versus the full structure. We have carried out such an investigation and obtained insights which are beneficial from the practical point point of view. The full strucure is flexible, but, due to a redundant number of parameters, slow convergence can be reached, which prevents from reaching the most potentially likely parameters. On the other hand, by fitting a set of sparse structures, there is a chance that one or more structures will converge faster and will

reach more likely parameter estimates with a smaller number of iterations. Moreover, fitting with a set of sparse structures can be parallelized. For further research, one may raise a question how to determine a suitable structure given the trace data.

Markov arrival processes (MAPs) structure analysis is a complex problem. We attempt to solve more practical problems for faster fitting. The forward-backward algorithm of the expectation maximization (EM) method is serial by nature, since likelihood vectors are evaluated recursively. Parallelism can be achieved by increasing computational complexity. However, there are no documented attempts to parallelize MAP fitting in scientific literature. When the ER-CHMM structure is used, the Baum-Welch algorithm (for CHMM fitting) can be adapted. However, there are various options to formulate the algorithm. In addition, we suggest our own algorithms as well. All algorithms have been implemented, and their execution properties have been compared.

The transient Markov arrival process (TMAP) differs from the Markov arrival process in a sense that it models finite sequences of inter-arrivals (further on referred to as *runs*), while MAPs generate an infinite sequence of inter-arrivals. The EM method application for TMAP fitting is not present in scientific literature. Thus we have developed an algorithm to search for maximum likelihood parameter estimates of such a process. It can be observed that a part of computations for each run can be done independently. Therefore, TMAP fitting can be directly parallelized. We have implemented and compared serial and parallel versions of the algorithm.

The transient Markov arrival processes (TMAPs) fitting algorithm performs a number of independent computations for each run (finite sequences of inter-arrivals), and the results are merged. Therefore, TMAP fitting can be easily parallelized, however, each thread is likely to perform different computational operations at any given moment, which is a limiting factor in effective parallel hardware usage.

Recently, with advances in GPU (Graphics Processing Unit) hardware development and applicability for general computing, temptation has arisen to utilize it for faster parameter fitting. However, there have been no documented attempts to utilize GPU for faster MAP, TMAP process fitting in scientific literature. Therefore, we have implemented the developed algorithms for execution on GPU, which required to find out how certain numerical issues should be solved.

The practical significance of the research results is the following. In situations where general Phase-type distributions with a big number of phases has to be fitted, sparse structures with $m = 2n$ transitions can be used to obtain more likely parameter estimates faster, especially in those cases when a set of sparse structures is fitted in parallel. If a certain real life phenomenon generates a finite sequence of events at random time instances, the collected inter-arrival data can be used to fit a TMAP process. In case a non-terminating sequence is observed, a MAP process can be fitted. While using our developed algorithms, such fitting can be done faster, which is of major importance given that the inter-arrival data is huge.

Information for Defense

1. The algorithm for generating PH(n) structures.
2. An empirical validation of hypothesis that PH matrix form representation with $m = 2n$ transitions is sufficient to cover almost the entire PH distribution space.
3. Comparison of Phase-type fitting using sparse (with $m = 2n$ transitions) and full structures.
4. The developed parallel algorithms for MAP fitting.
5. The derived EM algorithm for TMAP fitting.
6. The parallel MAP, TMAP fitting algorithm implementations for execution on GPU while using CUDA library.

Approbation of the Research Results

The thesis work has been presented in 2 scientific papers (ISI) and 2 international conferences as well as a few national conferences. The PH(n) structure generation algorithm along with some fitting research was presented in ISI paper 'On Structured Initial Solution Generation for Phase-Type Fitting with EM Method'. The presented work was extended by adding more thorough research to find out how different subclasses of the generated structures perform fitting. The MAP fitting algorithms and results on fitting performance were presented in ISI paper 'Parallel algorithms for fitting Markov arrival processes'. The derived EM algorithm for TMAP fitting with a research of fitting performance was presented at ASMTA16 conference in a conference proceedings journal, and a paper under the title 'Efficient implementations of the EM-algorithm for transient Markov arrival processes' was published. The derived EM algorithm for TMAP fitting was presented at ASMTA16 conference, and a paper under the title 'Efficient implementations of the EM-algorithm for transient Markov arrival processes' was published in the conference proceedings. Finally, a simple finite queue model was published in IARA2014 conference proceedings under the title 'Software reliability Markovian model based on phase type distributions'. We have revisited the presented work and used it in the thesis as a practical application context for PH fitting.

The Structure and Volume of the Dissertation

This doctoral dissertation consists of an introduction, 4 major sections, conclusions, a list of references and a list of the author's publications. The total length of the dissertation is 148 pages. The thesis features 54 figures, 44 tables and a list of 94 cited sources (10 sources were published in the time period from 2014 to 2018).

1. LITERATURE REVIEW

1.1. Continuous time Markov chains

Markov chains are named after a prominent Russian mathematician Andrei Andreevich Markov (1856–1922). The term was used for the first time in a paper by S. N. Bernstein in 1926 [1]. Since then, the theory of Markov chains has been extensively researched, and many of its applications discovered.

Markov process is a stochastic process which satisfies the Markov property. If a process has this property, its future state depends only on the current state.

We investigate stochastic process $\{\mathcal{X}_t\}_{t\geq 0}$ which takes values in a finite set $\boldsymbol{\mathcal{S}} = \{0, 1, \ldots, n\}$, and index $t \in \mathbb{R}_+$ is associated with time. We shall briefly denote the process by $\{\mathcal{X}_t\}$.

Definition 1.1 [2] *Process $\{\mathcal{X}_t\}$ is called a Markov process if it satisfies the simple Markov property, i.e., for any $s, t > 0$ and any $i, j \in \boldsymbol{\mathcal{S}}$:*

$$\mathbb{P}(\mathcal{X}_{s+t} = j \mid \mathcal{X}_s = i, \mathcal{X}_u, u \leq s) = \mathbb{P}(\mathcal{X}_{s+t} = j \mid \mathcal{X}_s = i).$$

The probability of being in state j after time t only depends on current state $\mathcal{X}_s$.

Definition 1.2 [2] *Markov process $\{\mathcal{X}_t\}$ is time-homogeneous if transition probability from state i to state j after time t does not depend on elapsed time s, i.e., for any $t, s > 0$ and each $i, j \in \boldsymbol{\mathcal{S}}$:*

$$\mathbb{P}(\mathcal{X}_{s+t} = j \mid \mathcal{X}_s = i) = \mathbb{P}(\mathcal{X}_t = j \mid \mathcal{X}_0 = i).$$

We consider time-homogeneous Markov processes, for brevity, we call such processes homogeneous.

Definition 1.3 [2] *A homogeneous Markov process $\{\mathcal{X}_t\}$ which takes values in state space $\boldsymbol{\mathcal{S}}$ is called a homogeneous continuous time Markov chain (CTMC).*

The time spent in state i before jumping to another state j (which might be the same) is a r.v. H_i. We define the process $\{\mathcal{X}_t\}$ as right-continuous, i.e., the jump times are

$$T_1 = \inf\{t : \mathcal{X}_t \neq \mathcal{X}_0\},$$
$$T_n = \inf\{t : t \geq T_{n-1}, \mathcal{X}_t \neq \mathcal{X}_{T_{n-1}}\}.$$

Proposition 1.1 [3] *In the homogeneous CTMC process, the holding time r.v.'s H_i are exponentially distributed.*

The state transition probability is denoted by

$$\boldsymbol{P}_{i,j}(s, s+t) = \mathbb{P}(\mathcal{X}_{s+t} = j \mid \mathcal{X}_s = i),$$

since the process is homogeneous, the probability depends only on time difference t

$$\boldsymbol{P}_{i,j}(s, s+t) = \boldsymbol{P}_{i,j}(0, t) = \boldsymbol{P}_{i,j}(t).$$

The transition probability matrix is defined as $\boldsymbol{P}(t) = \{\boldsymbol{P}_{i,j}(t)\}_{i,j\in\boldsymbol{\mathcal{S}}\times\boldsymbol{\mathcal{S}}}$.

Proposition 1.2 [3] *For the probability transition matrix $\boldsymbol{P}(t)$ of homogeneous CTMC there exists an infinitesimal generator (or simply a generator) $\boldsymbol{Q}$ defined as:*

$$\boldsymbol{Q} = \lim_{t\to 0} \frac{\boldsymbol{P}(t) - \boldsymbol{I}}{t}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{P}(t) = \boldsymbol{Q}\boldsymbol{P}(t) = \boldsymbol{P}(t)\boldsymbol{Q}.$$

From the definition of $\boldsymbol{Q}$, it follows that

$$\boldsymbol{P}(t) = e^{\boldsymbol{Q}t} = \sum_{i=0}^{\infty} \frac{\boldsymbol{Q}^i t^i}{i!}.$$

The infinitesimal generator $\boldsymbol{Q} = \{\boldsymbol{Q}_{i,j}\}_{i,j\in\boldsymbol{\mathcal{S}}\times\boldsymbol{\mathcal{S}}}$ is called the rate matrix.

Proposition 1.3 *If $\boldsymbol{Q}$ is the rate matrix of homogeneous CTMC, then:*

(i) $\sum_{j\in\boldsymbol{\mathcal{S}}} \boldsymbol{Q}_{i,j} = 0$ *for all $i \in \boldsymbol{\mathcal{S}}$,*

(ii) $-\boldsymbol{Q}_{i,i}$ *is the parameter of exponential distribution associated with state i sojourn time.*

The distribution of staying in state i is exponential, i.e., $\mathbb{P}(H_i \le t) = 1 - e^{-\boldsymbol{Q}_{i,i}t}$.

The embedded process of state jumps $\{\mathcal{Y}_k\}_{k\in\mathbb{N}}$ is specified as

$$\mathcal{Y}_k = \mathcal{X}_{T_k},$$

and is a discrete time Markov chain (DTMC). This DTMC process $\{\mathcal{Y}_k\}$ is specified by the transition probability matrix $\boldsymbol{P} = \{\boldsymbol{P}_{i,j}\}_{i,j\in\boldsymbol{\mathcal{S}}\times\boldsymbol{\mathcal{S}}}$ and the initial probability vector $\boldsymbol{p}(t)$. Transition probability matrix $\boldsymbol{P}$ can be obtained from rate matrix $\boldsymbol{Q}$ by

$$\boldsymbol{P}_{i,j} = \begin{cases} -\dfrac{\boldsymbol{Q}_{i,j}}{\boldsymbol{Q}_{i,i}}, \text{ for } i \neq j, \boldsymbol{Q}_{i,i} \neq 0, \\ 0, \text{ for } i \neq j, \boldsymbol{Q}_{i,i} = 0, \\ 1, \text{ for } i = j. \end{cases}$$

Definition 1.4 [2] *State i, for which $\boldsymbol{Q}_{i,i} = 0$, is called the absorbing state.*

When a process gets into the absorbing state, it stays in it for indefinite time. Also, the probabilities of transition from absorbing state i to another state j is zero, i.e., $\boldsymbol{P}_{i,j} = 0$.

Example 1.1 Let us consider a three state homogeneous CTMC process $\{\mathcal{X}_t\}$ defined by the rate matrix

$$\boldsymbol{Q} = \begin{bmatrix} -3 & 1 & 2 \\ 0 & -1 & 1 \\ 1 & 1 & -2 \end{bmatrix},$$

whose CTMC schema is shown in Figure 1.1.

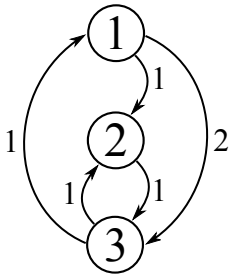

Fig. 1.1. State schema of CTMC given in Example 1.1.

The transition probability matrices for time moments $t = 0$ and $t = 1$ are

$$\begin{aligned} \boldsymbol{P}(0) =& e^{\boldsymbol{Q}\cdot 0} = \boldsymbol{I}, \\ \boldsymbol{P}(1) =& e^{\boldsymbol{Q}\cdot 1} \approx \begin{bmatrix} 0.1703 & 0.4323 & 0.3974 \\ 0.0935 & 0.5677 & 0.3389 \\ 0.1520 & 0.4323 & 0.4157 \end{bmatrix}. \end{aligned}$$

The stationary distribution $\overline{\boldsymbol{\alpha}}$ of the process $\{\mathcal{X}_t\}$ is

$$\overline{\boldsymbol{\alpha}} = \begin{bmatrix} \frac{1}{8} & \frac{1}{2} & \frac{3}{8} \end{bmatrix}.$$

It can be easily checked that

$$\overline{\boldsymbol{\alpha}}\boldsymbol{P}(0) = \overline{\boldsymbol{\alpha}}, \quad \overline{\boldsymbol{\alpha}}\boldsymbol{P}(1) = \overline{\boldsymbol{\alpha}}.$$

The transition probability matrix of embedded jump process $\{\mathcal{Y}_{T_k}\}$ is

$$\boldsymbol{P} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}.$$

There is no absorbing state in this homogeneous CTMC. △

Definition 1.5 [2] *States i, j are said to be communicating if state i is reachable from state j and state j is reachable from state i.*

Definition 1.6 [2] *A subset of states $\mathcal{C} \subset \mathcal{S}$ is closed if $\boldsymbol{P}_{i,j} = 0$ for $i \in \mathcal{C}, j \notin \mathcal{C}$.*

Definition 1.7 [2] *A closed subset $\mathcal{C}$ is said to be communicating closed subset $\mathcal{C}$ if all the states in it are communicating.*

Definition 1.8 [2] *State i is said to be an absorbing state if it is the only state in closed subset $\mathcal{C}$.*

Definition 1.9 [2] *State $i \in \mathcal{S}$ is said to be a transient state if the probability of returning to it after leaving it is less than one.*

Definition 1.10 [2] *A homogeneous CTMC is called an absorbing Markov chain if each of its states is either transient or absorbing.*

Definition 1.11 [2] *A hitting time is a time it takes for the process to reach the absorbing state.*

For absorbing state i we have $\boldsymbol{P}_{i,i} = 1$, i.e., once the process reaches the absorbing state, it remains in it.

1.2. Phase-type distributions

The term *Phase-type distributions* as such was introduced in [4].

Definition 1.12 [4, 5, 2] *The Phase-type distribution is defined as the distribution of hitting time $\mathcal{X}$ it takes for the absorbing continuous time Markov chain process to reach the absorbing state from the set of transient states.*

The Phase-type distribution is represented by the corresponding absorbing Markov chain which is specified by rate matrix $\boldsymbol{Q}$ and initial probability vector $\boldsymbol{\alpha}$.

To make the analysis easier, the states are rearranged in such a way that the absorbing state is clearly separated from the transient ones, i.e.,

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{D_0} & \boldsymbol{d_1} \\ \boldsymbol{0} & 0 \end{bmatrix}, \tag{1}$$

and the state space is composed as $\mathcal{S} = \mathcal{S}_T \cup \mathcal{S}_A$, where $\mathcal{S}_T = \{1, 2, \ldots, n\}$ are the transient states, and set $\mathcal{S}_A$ contains absorbing state $n + 1$.

Matrix $\boldsymbol{D_0}$ describes transitions between transient states $\mathcal{S}_T$, rate vector $\boldsymbol{d_1}$ describes transitions from transient states to absorbing state $n+1$. Other elements of rate matrix $\boldsymbol{Q}$ are zeros, which indicates that there is no transition from the absorbing state.

Definition 1.13 [4] *The transient states in a continuous time Markov chain, which is interpreted as a Phase-type distribution, are called phases.*

Definition 1.14 [6] *The order of Phase-type distribution is a number of transient states (phases).*

Example 1.2 A transient CTMC is given by the parameters

$$\boldsymbol{Q} = \begin{bmatrix} -2 & 0 & 1 & 1 \\ 3 & -3 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{\alpha} = \begin{bmatrix} 0.5 & 0.3 & 0.2 & 0 \end{bmatrix}, \qquad \triangle$$

specifies a Phase-type distribution whose CTMC is shown in Figure 1.2.

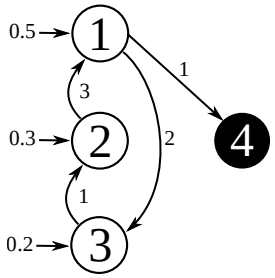

Fig. 1.2. An example schema of trasient CTMC of Phase-type distribution.

The first, second and third states are transient, whereas the fourth one is absorbing.

Matrix $\boldsymbol{M} = (-\boldsymbol{D_0})^{-1}$ is a fundamental matrix of Phase-type distribution as defined in [7].

The value of $\{(-\boldsymbol{D_0})^{-1}\}_{i,j}$ is the expected total time spent in state j before reaching the absorbing state when the initial state is i.

Based on Proposition 1.3 about rate matrix $\boldsymbol{Q}$ we have

$$\boldsymbol{D_0}\mathbb{1} + \boldsymbol{d_1} = \boldsymbol{0}. \tag{2}$$

Rate matrix $\boldsymbol{D_0}$ is *substochastic* since for $i \in \boldsymbol{\mathcal{S}}_T$ holds

$$\sum_{j \in \boldsymbol{\mathcal{S}}_T} \boldsymbol{D_0}_{i,j} \leq 0.$$

Vector $\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 & \dots & \boldsymbol{\alpha}_n \end{bmatrix}$ denotes the initial probability of starting in the transient state $i \in \boldsymbol{\mathcal{S}}_T$. The case when the process starts (and ends at the same time) in the

absorbing state is not considered. That does not reduce the practical applicability of PH distributions but simplifies further investigation.

Definition 1.15 [3] *Parameter set* $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ *fully specifies a Phase-type distribution and is called the matrix form representation.*

Indeed, from (2) it follows that $\boldsymbol{d_1} = -\boldsymbol{D_0}\mathbb{1}$.

Example 1.3 The Phase-type distribution given in Example 1.2 has the following parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0})$

$$\boldsymbol{D_0} = \begin{bmatrix} -2 & 0 & 1 \\ 3 & -3 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \ \boldsymbol{\alpha} = \begin{bmatrix} 0.5 & 0.3 & 0.2 \end{bmatrix}. \qquad \triangle$$

Definition 1.16 [8] *Representation* $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ *is Markovian if for* $i \in \boldsymbol{\mathcal{S}}_T$

$$\begin{aligned} \boldsymbol{\alpha}_i \geq 0, \boldsymbol{\alpha}\mathbb{1} = 1, \\ \boldsymbol{D_0}_{i,j} \geq 0 \text{ for } j \in \boldsymbol{\mathcal{S}}_T, i \neq j, \\ \boldsymbol{D_0}_{i,i} < 0. \end{aligned}$$

In other words, representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ is Markovian if it specifies a valid CTMC with rate matrix $\boldsymbol{Q}$ in (1).

Proposition 1.4 [9, 3] *Two representations* $(\boldsymbol{\alpha}^{(a)}, \boldsymbol{D}_{\mathbf{0}}^{(a)})$ *and* $(\boldsymbol{\alpha}^{(b)}, \boldsymbol{D}_{\mathbf{0}}^{(b)})$ *with distribution functions*

$$F^{(a)}(x) = 1 - \boldsymbol{\alpha}^{(a)} e^{\boldsymbol{D}_{\mathbf{0}}^{(a)} x}\mathbb{1} \ \text{ and } \ F^{(b)}(x) = 1 - \boldsymbol{\alpha}^{(b)} e^{\boldsymbol{D}_{\mathbf{0}}^{(b)} x}\mathbb{1}$$

are equivalent if there exists a non-singular matrix $\boldsymbol{B}$ *(*$\boldsymbol{B}\mathbb{1} = \mathbb{1}$*) such that*

$$\boldsymbol{\alpha}^{(b)} = \boldsymbol{\alpha}^{(a)}\boldsymbol{B} \ \text{ and } \ \boldsymbol{D}_{\mathbf{0}}^{(b)} = \boldsymbol{B}^{-1}\boldsymbol{D}_{\mathbf{0}}^{(a)}\boldsymbol{B}.$$

In general, for a given representation, we can find infinitely many equivalent representations, however, it is not guaranteed that the obtained representation will be Markovian. To this day, there is no known procedure to explicitly determine whether transformation matrix $\boldsymbol{B}$ exists for transforming a given representation to another representation of any given structure.

Proposition 1.5 [10] *The distribution function of Phase-type distribution given by matrix form representation* $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ *is*

$$F(x) = 1 - \boldsymbol{\alpha} e^{\boldsymbol{D_0} x}\mathbb{1}, \tag{3}$$

and its density function is

$$f(x) = \boldsymbol{\alpha} e^{\boldsymbol{D}_0 x} \boldsymbol{d_1}, \tag{4}$$

for $x \geq 0$.

Continuous Phase-type distributions are dense in the $\mathbb{R}_{\geq 0}$ and thus can approximate arbitrarily closely any non-negative distribution [11].

The Laplace–Stieltjes transform of density function is [3]

$$f^*(s) = \int_0^\infty e^{-sx} \mathrm{d}F(x) = \boldsymbol{\alpha}(s\boldsymbol{I} - \boldsymbol{D_0})^{-1} \boldsymbol{d_1}. \tag{5}$$

It is interesting to note that in case representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ is such that $\boldsymbol{d_1} = -\boldsymbol{D_0}\mathbb{1} = \lambda\mathbb{1}$ for some $\lambda > 0$, then the Phase-type distribution is exponential with rate λ regardless of the choice of initial probability vector $\boldsymbol{\alpha}$.

Definition 1.17 *Parameter set* $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}})$ *specifies possible transitions in a continuous time Markov chain of the Phase-type distribution and is called the structure of matrix form representation* $(\boldsymbol{\alpha}, \boldsymbol{D_0})$.

Such Phase-type distribution structure notation (Definition 1.17) has not been introduced earlier. The most likely reason is that general Phase-type fitting (of orders $n > 4$) is not effective [12, 13], and that a more effective alternative Hyper-Erlang distribution fitting has been investigated more. We use this structure notation as a base for presenting further literature review.

Elements of $\dot{\boldsymbol{\alpha}}$, $\dot{\boldsymbol{D}}_{\boldsymbol{0}}$, $\dot{\boldsymbol{d}}_{\boldsymbol{1}}$ are from set $\{0, 1\}$. Structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}})$ is characterized by the number of transitions

$$m = \sum_{j=1}^{n} \dot{\boldsymbol{\alpha}}_j + \sum_{i=1}^{n} \sum_{j=1}^{n} \dot{\boldsymbol{D}}_{\boldsymbol{0}i,j} + \sum_{i=1}^{n} \dot{\boldsymbol{d}}_{\boldsymbol{1}i}.$$

Example 1.4 The structure of the Phase-type distribution matrix form representation given in Example 1.3 is

$$(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}}) = \left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)$$

and the number of transitions is $m = 7$. △

Definition 1.18 *A Phase-type distribution is said to have a full structure if all valid transitions are enabled.*

A full structure has the maximum number of transitions $m = n^2 + n$ [14].

Example 1.5 The full structure of order $n = 3$ Phase-type distribution is specified by

$$(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}) = \left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right)$$

and is shown in Figure 1.3. The number of transitions is $m = n^2 + n = 12$.

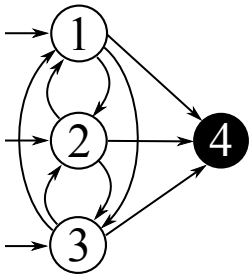

Fig. 1.3. The CTMC of Phase-type distribution matrix form representation of full structure.

△

Definition 1.19 *The matrix form representation parametrization is function vector* $\boldsymbol{f}$ *which specifies how transition rates in a corresponding CTMC depend on* p *parameters.*

In other words, for a given parameter vector $\boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_1 & \dots & \boldsymbol{p}_p \end{bmatrix}$, a given parametrization specifies a single matrix form representation

$$(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}}) = \boldsymbol{f}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})}(\boldsymbol{p}).$$

For each given structure, an undefinite number of parametrizations exists. Only a few are of particular interest.

Definition 1.20 *The full parametrization of structure* $\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}\right)$ *has* $p = m - 1$ *independent parameters.*

The number of independent parameters is one fewer than the number of transitions because of constraint $\boldsymbol{\alpha}\mathbb{1} = 1$.

Example 1.6 Let us consider a structure specified by

$$(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}) = \left(\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right),$$

whose two parametrizations are shown in Figure 1.4.

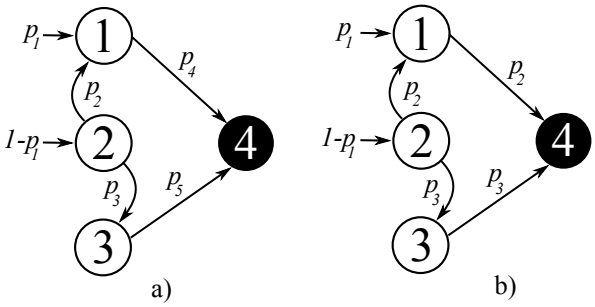

Fig. 1.4. Two parametrizations of the Phase-type distribution structure given in Example 1.6.

Parametrizations (a), (b) have $p = 5$ and $p = 3$ parameters, respectively. However, parametrization (b) is a full one. △

Theorem 1.1 [15] *A Phase-type distribution of order n is determined by $2n-1$ parameters at most.*

Representation $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$ with $n^2 + n - 1$ parameters is highly redundant (based on Theorem 1.1.). Actually, the same Phase-type distribution can be represented by many different matrix representations [11], even of different representation orders [16].

Let $\boldsymbol{\mathcal{P}}_n$ be a set of all possible parametrizations with $p = 2n - 1$ parameters.

Definition 1.21 [17] *A canonical form of Phase-type distribution matrix form representation is a finite set $\boldsymbol{\mathcal{C}}_n$ of parametrizations with $p = 2n - 1$ parameters which can be used to uniquely represent any Phase-type distribution of order n.*

Consequently, we have $\boldsymbol{\mathcal{C}}_n \subseteq \boldsymbol{\mathcal{P}}_n$. It is useful to express $\boldsymbol{\mathcal{C}}_n$ as a union of sets

$$\boldsymbol{\mathcal{C}}_n = \boldsymbol{\mathcal{C}}_n^{(2n)} \cup \cdots \cup \boldsymbol{\mathcal{C}}_n^{M_n},$$

where $\boldsymbol{\mathcal{C}}_n^{(m)}$ is a set of parametrizations with m transitions; M_n is the maximum number of transitions

$$M_n = 2n + \Delta M_n,$$

where ΔM_n is the maximum number of additional transitions.

Definition 1.22 [18, 19] *If transition rate matrix $\boldsymbol{D_0}$ in representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ can be transformed into an upper triangular matrix, the represented Phase-type distribution is acyclic (APH).*

Every state in the acyclic $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ is visited no more than once. We will overview a few sub-classes of APH.

The simplest case is a single phase, i.e., an exponential distribution whose distribution function is

$$F(x) = 1 - e^{-\lambda x}$$

and density function is

$$f(x) = \lambda e^{-\lambda x}$$

for $x \geq 0$.

The sum of r random values which are exponentially distributed with the same rate parameter λ has an Erlang distribution (ER). The Erlang distribution function is

$$F(x) = 1 - \sum_{k=0}^{r-1} \frac{(\lambda x)^k}{k!} e^{-\lambda x}$$

and its density is

$$f(x) = \frac{\lambda^n}{(r-1)!} t^{r-1} e^{-\lambda x}$$

for $x \geq 0$.

The CTMC process of Erlang distribution starts in the first state with probability 1, i.e., the initial probability vector is

$$\boldsymbol{\alpha} = \boldsymbol{\beta}(r) = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}_{1 \times r}$$

and rate matrix $\boldsymbol{D_0}$ has the form

$$\boldsymbol{D_0} = \boldsymbol{B}(r, \lambda) = \begin{bmatrix} -\lambda & \lambda & \dots & 0 & 0 \\ 0 & -\lambda & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -\lambda & \lambda \\ 0 & 0 & \dots & 0 & -\lambda \end{bmatrix}_{r \times r}.$$

The Erlang distribution has the least variance [20], i.e., when $r \to \infty$, it approaches a deterministic distribution.

The Hyper-Erlang (HErD) [21] is a mixture of R Erlang distributions weighted by $\boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \dots & \boldsymbol{\pi}_R \end{bmatrix}$. The i^{th} Erlang distribution is called the Erlang branch with $\boldsymbol{r}_i$ states. Its distribution function is

$$F(x) = 1 - \sum_{i=1}^{R} \boldsymbol{\pi}_i \sum_{j=0}^{\boldsymbol{r}_i - 1} \frac{(\boldsymbol{\lambda}_i x)^j}{j!} e^{-\boldsymbol{\lambda}_i x}, \; x \geq 0,$$

and its density function is

$$f(x) = \sum_{i=1}^{R} \boldsymbol{\pi}_i \frac{(\boldsymbol{\lambda}_i x)^{\boldsymbol{r}_i - 1}}{(\boldsymbol{r}_i - 1)!} \boldsymbol{\lambda}_i e^{-\boldsymbol{\lambda}_i x}, \; x \geq 0,$$

where $\boldsymbol{\lambda}_i$ is the transition rate of i^{th} Erlang branch. The HErD structure is denoted by a list of Erlang branch orders $\boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_R \end{bmatrix}$. Also, it is convenient to define vector $\boldsymbol{s}$ of branch start state indices with elements

$$\boldsymbol{s}_i = 1 + \boldsymbol{r}_1 + \cdots + \boldsymbol{r}_{i-1} \quad \text{for } i = 1, \ldots, R.$$

Hyper-Erlang distribution can be represented in the matrix form notation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ in the following way

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\pi}_1 \boldsymbol{\beta}(\boldsymbol{r}_1) & \boldsymbol{\pi}_2 \boldsymbol{\beta}(\boldsymbol{r}_2) & \ldots & \boldsymbol{\beta}(\boldsymbol{r}_R) \end{bmatrix},$$

$$\boldsymbol{D_0} = \begin{bmatrix} \boldsymbol{B}(\boldsymbol{r}_1, \boldsymbol{\lambda}_1) & 0 & \ldots & 0 & 0 \\ 0 & \boldsymbol{B}(\boldsymbol{r}_2, \boldsymbol{\lambda}_2) & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \boldsymbol{B}(\boldsymbol{r}_{R-1}, \boldsymbol{\lambda}_{R-1}) & 0 \\ 0 & 0 & \ldots & 0 & \boldsymbol{B}(\boldsymbol{r}_R, \boldsymbol{\lambda}_R) \end{bmatrix},$$

where $\boldsymbol{B}(\boldsymbol{r}_i, \boldsymbol{\lambda}_i)$ is the infinitesimal generator of the i^{th} Erlang distribution.

Example 1.7 Let us consider a Hyper-Erlang distribution with branch orders of $\boldsymbol{r} = \begin{bmatrix} 2 & 1 \end{bmatrix}$ which is shown in Figure 1.5 and is specified by the following matrix form representation

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\pi}_1 & 0 & \boldsymbol{\pi}_2 \end{bmatrix}, \boldsymbol{D_0} = \begin{bmatrix} -\lambda_1 & \lambda_1 & 0 \\ 0 & -\lambda_1 & 0 \\ 0 & 0 & -\lambda_2 \end{bmatrix}.$$

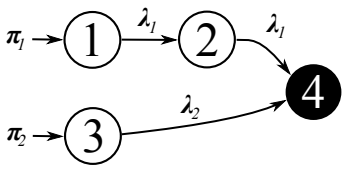

Fig. 1.5. The CTMC of Hyper-Erlang distribution with branch orders of $\boldsymbol{r} = [2\ 1]$.

△

A Coxian distribution has n phases, the process starts at the first phase with probability 1 and transitions to the next phase with probability $\boldsymbol{g}_i$ or to the absorbing state with probability $(1-\boldsymbol{g}_i)$. Transition rates $\boldsymbol{\lambda}_i$ can be different in general.

The matrix form representation of Coxian distribution is

$$\boldsymbol{\alpha} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}_{1\times n},$$

$$\boldsymbol{D_0} = \begin{bmatrix} -\boldsymbol{\lambda}_1 & \boldsymbol{g}_1 & 0 & \dots & & 0 \\ 0 & -\boldsymbol{\lambda}_2 & \boldsymbol{g}_2\boldsymbol{\lambda}_2 & \dots & & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & -\boldsymbol{\lambda}_{n-1} & \boldsymbol{\lambda}_{n-1}\boldsymbol{g}_{n-1} & 0 \\ 0 & \dots & & & 0 & -\boldsymbol{\lambda}_n \end{bmatrix}_{n\times n}.$$

If $\boldsymbol{\lambda}_1 \geq \boldsymbol{\lambda}_2 \geq \cdots \geq \boldsymbol{\lambda}_n$, such Coxian distribution is said to be ordered.

Theorem 1.2 [22] *Any acyclic Phase-type distribution has the representation of ordered Coxian distribution.*

In the general Coxian distribution, the process can start in any phase, i.e., $\boldsymbol{\alpha}_i \geq 0$ for $i = 1, 2, \dots, n$.

1.3. Expectation Maximization method for Phase-type fitting

The expectaction-maximization (EM) algorithm for the Phase-type of the general structure is given in [14].

The observed random variables are interpreted as hitting times

$$\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_T)$$

in the corresponding CTMC of Phase-type distribution. The realization of $\boldsymbol{\mathcal{X}}$ is denoted as

$$\boldsymbol{x} = (x_1, x_2, \dots, x_T).$$

The task is to find the most likely parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ of Phase-type distribution so that the distribution of its hitting times is as close to one of the observed random variables as possible. This can be achieved by maximizing the following likelihood function

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{D_0} \mid \boldsymbol{\mathcal{X}}) = \prod_{k=1}^{T} f(x_k) = \prod_{k=1}^{T} \boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{d_1} \tag{6}$$

by applying the expectation maximization method [23].

Hitting times $\boldsymbol{x}$ constitute the only available information. Yet, in order to determine the parameters of CTMC, it is necessary to have full information regarding how

each x_i in $\boldsymbol{x}$ has been generated. Let us assume that random variable $\mathcal{X}_k$ is generated by the continuous time process $\mathcal{J}_k^{(k)}$ which is fully characterized by the discrete time process of the states being visited

$$\mathcal{I}_0^{(k)}, ..., \mathcal{I}_{\mathcal{M}^{(k)}-1}^{(k)}$$

and the discrete time process of sojourn times

$$\mathcal{S}_0^{(k)}, ..., \mathcal{S}_{\mathcal{M}^{(k)}-1}^{(k)},$$

which specify the time spent in a certain state before leaving it. Here, $\mathcal{M}^{(k)}$ is a random number of transitions it takes to reach the absorbing state. Meanwhile, the observed random variable $\mathcal{X}_k$ is interpreted as the total time spent before reaching the absorbing state, i.e.,

$$\mathcal{X}_k = \mathcal{S}_0^{(k)} + ... + \mathcal{S}_{\mathcal{M}^{(k)}-1}^{(k)}.$$

The full information $\boldsymbol{\mathcal{Y}}$ about T observations consists of

$$\boldsymbol{y} = \left(i_0^{(1)}, ..., i_{m^{(0)}-1}^{(1)}, s_0^{(1)}, ..., s_{m^{(0)}-1}^{(1)}, ..., i_0^{(T)}, ..., i_{m^{(T)}-1}^{(T)}, s_0^{(T)}, ..., s_{m^{(T)}-1}^{(T)}\right).$$

In theory, based on observations, $\boldsymbol{x}$ the most likely parameters could be found in the following way. With initial parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0})$, a sufficient number of hitting times is sampled. The process realizations whose hitting times match the observed values in $\boldsymbol{x}$ are selected. Then the full process realization information of these hitting times is used to estimate the more likely parameters. The procedure would be repeated until parameter likelihood stops increasing.

Actually, it is not necessary to have full information $\boldsymbol{\mathcal{Y}}$ about process realizations, it suffices to have its characteristics

$$\boldsymbol{\mathcal{H}} = \{\mathcal{B}_i, \mathcal{N}_{i,j}, \mathcal{Z}_i\},$$

where $\mathcal{B}_i$ is the number of realizations starting in state i,

$$\mathcal{B}_i = \sum_{k=1}^{T} \mathcal{B}_i[k], \;\; \mathcal{B}_i[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(k)}=i\right\}},$$

$\mathcal{N}_{i,j}$ is the number of transitions from state i to state j

$$\mathcal{N}_{i,j} = \sum_{k=1}^{T} \mathcal{N}_{i,j}[k], \;\; \mathcal{N}_{i,j}[k] = \sum_{l=0}^{\mathcal{M}^{(k)}} \mathbb{1}_{\left\{\mathcal{I}_l^{(k)}=i, \mathcal{I}_{l+1}^{(k)}=j\right\}},$$

and $\mathcal{Z}_i$ is the total time spent in state i

$$\mathcal{Z}_i = \sum_{k=1}^{T} \mathcal{Z}_i[k], \;\; \mathcal{Z}_i[k] = \sum_{l=0}^{\mathcal{M}^{(k)}} \mathcal{S}_l^{(k)} \mathbb{1}_{\left\{\mathcal{I}_l^{(k)}=i\right\}}.$$

The estimate of statistics $\boldsymbol{\mathcal{H}}$ for given parameters $(\boldsymbol{\alpha D_0})$ and observations $\boldsymbol{x}$ with their weights $\boldsymbol{w}$ can be evaluated by the following formulas. The estimate of the number of realizations starting in state i is

$$b_i = \mathbb{E}[\mathcal{B}_i \mid \boldsymbol{\mathcal{X}}] = \sum_{k=1}^{T} w_k \mathbb{E}[\mathcal{B}_i[k] \mid \mathcal{X}_k] = \sum_{k=1}^{T} w_k \frac{\boldsymbol{\alpha}_i \boldsymbol{e}_i^{\intercal} e^{\boldsymbol{D_0} x_k} \boldsymbol{d_1}}{\boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{d_1}}, \tag{7}$$

the estimate of the number of transitions from state i to state j for $i = 1, \ldots, n$, $j = 1, \ldots, n$, $i \neq j$ is

$$\begin{aligned} n_{i,j} &= \mathbb{E}[\mathcal{N}_{i,j} \mid \boldsymbol{\mathcal{X}}] = \sum_{k=1}^{T} w_k \mathbb{E}[\mathcal{N}_{i,j}[k] \mid \mathcal{X}_k] \\ &= \sum_{k=1}^{T} w_k \frac{\int_0^{x_k} \boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{e}_i \boldsymbol{D_0}_{i,j} \boldsymbol{e}_j^{\intercal} e^{\boldsymbol{D_0}(x_k - u)} \boldsymbol{d_1} \mathrm{d}u}{\boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{d}_1}, \end{aligned} \tag{8}$$

the estimate of the number of state i to absorbing state $n+1$ for $i = 1, ..., n$ is

$$\begin{aligned} n_{i,n+1} &= \mathbb{E}[\mathcal{N}_{i,n+1} \mid \boldsymbol{\mathcal{X}}] = \sum_{k=1}^{T} w_k \mathbb{E}[\mathcal{N}_{i,n+1}[k] \mid \mathcal{X}_k] \\ &= \sum_{k=1}^{T} w_k \frac{\boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{e}_i \boldsymbol{d_1}_i}{\boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{d_1}} \end{aligned} \tag{9}$$

and the estimate of the total time spent in state i is

$$\begin{aligned} z_i &= \mathbb{E}[\mathcal{Z}_i \mid \boldsymbol{\mathcal{X}}] = \sum_{k=1}^{T} w_k \mathbb{E}[\mathcal{Z}_i[k] \mid \mathcal{X}_k] \\ &= \sum_{k=1}^{T} w_k \frac{\int_0^{x_k} \boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{e}_i \boldsymbol{e}_j^{\intercal} e^{\boldsymbol{D_0}(x_k - u)} \boldsymbol{d_1} \mathrm{d}u}{\boldsymbol{\alpha} e^{\boldsymbol{D_0} x_k} \boldsymbol{d_1}}. \end{aligned} \tag{10}$$

The next step is to compute the maximum likelihood parameter estimates based on $\boldsymbol{h} = \{b_i, n_{i,j}, z_i\}$, which is done according to [24]

$$\begin{aligned} \boldsymbol{\alpha}_i &= \frac{b_i}{\sum_{k=1}^{T} w_k}, && i = 1, \ldots, n \\ \boldsymbol{D_0}_{i,j} &= \frac{n_{i,j}}{z_i}, && i, j = 1, \ldots, n,\ i \neq j \\ \boldsymbol{d_1}_i &= \frac{n_{i,n+1}}{z_i}, && i = 1, \ldots, n \\ \boldsymbol{D_0}_{i,i} &= -\boldsymbol{d_1}_i - \sum_{j=1, j \neq i}^{n} \boldsymbol{D_0}_{i,j}, && i = 1, \ldots, n. \end{aligned} \tag{11}$$

Weights $\boldsymbol{w}$ can be used to perform discretized distribution fitting, otherwise $w_k = 1$ for $k = 1, \ldots, T$.

The application of the maximum expectation method for Phase-type fitting is summarized in Algorithm 1.

Algorithm 1 Algorithm for Phase-type fitting by EM method.

procedure PH-EM-fitting ($\boldsymbol{x}$, $\dot{\boldsymbol{\alpha}}$, $\dot{\boldsymbol{D}}_{\mathbf{0}}$, $\dot{\boldsymbol{d}}_{\mathbf{1}}$)
    initial parameters $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$ of structure $\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}\right)$ are generated by Algorithm 2
    likelihood of $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$ is computed by (6)
    repeat
        estimation of sufficient characteristics $\boldsymbol{h}$ by (7), (8), (9), (10)
        maximum likelihood parameters $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$ are found by (11)
        likelihood of $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$ is computed by (6)
    until likelihood keeps increasing
    return $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$
end procedure

Computation of estimate $\boldsymbol{h}$ of suffcient characteristics $\boldsymbol{\mathcal{H}}$ of the full information $\boldsymbol{\mathcal{Y}}$ is a computationally demanding task. [14] suggested to compute

$$\boldsymbol{\alpha} e^{\boldsymbol{D}_0 x},\ e^{\boldsymbol{D}_0 x}\boldsymbol{d}_{\mathbf{1}},\ \int_0^x \boldsymbol{\alpha} e^{\boldsymbol{D}_0 u}\boldsymbol{e}_{\boldsymbol{i}} e^{\boldsymbol{D}_0 (x-u)}\boldsymbol{d}_{\mathbf{1}}\mathrm{d}u \quad \text{for } i = 1, \ldots, n \tag{12}$$

by numerically solving the following system of differential equations

$$\begin{cases} \left(\boldsymbol{\alpha} e^{\boldsymbol{D}_0 x}\right)' = \boldsymbol{\alpha} e^{\boldsymbol{D}_0 x}\boldsymbol{D}_{\mathbf{0}} \\ \left(e^{\boldsymbol{D}_0 x}\boldsymbol{d}_{\mathbf{1}}\right)' = \boldsymbol{D}_{\mathbf{0}} e^{\boldsymbol{D}_0 x}\boldsymbol{d}_{\mathbf{1}} \\ \left(\int_0^x \boldsymbol{\alpha} e^{\boldsymbol{D}_0 u}\boldsymbol{e}_{\boldsymbol{i}} e^{\boldsymbol{D}_0 (x-u)}\boldsymbol{d}_{\mathbf{1}}\mathrm{d}u\right)' = \boldsymbol{D}_{\mathbf{0}} \int_0^x \boldsymbol{\alpha} e^{\boldsymbol{D}_0 u}\boldsymbol{e}_{\boldsymbol{i}} e^{\boldsymbol{D}_0 (x-u)}\boldsymbol{d}_{\mathbf{1}}\mathrm{d}u + \left\{\boldsymbol{\alpha} e^{\boldsymbol{D}_0 x}\right\}_i, \\ i = 1, \ldots, n. \end{cases} \tag{13}$$

[14] applies the fourth order Runge-Kuta method to solve (13). While [25] applies a randomization technique to compute (12), however, we have experienced numerical issues when dealing with sparse structures.

The initial Phase-type distribution parameters can be generated by Algorithm 2. A sample drawn from uniform distribution with the range from 0 to 1 is denoted by $U[0; 1]$. The randomly generated initial parameters are rescaled to match the given trace data mean $\mathbb{E}[\mathcal{X}]$.

Algorithm 2 Algorithm for generating initial Phase-type distribution parameters.

procedure PH-parameter-generation ($\mathbb{E}[\mathcal{X}]$, $\dot{\boldsymbol{\alpha}}$, $\dot{\boldsymbol{D}}_{\mathbf{0}}$, $\dot{\boldsymbol{d}}_{\mathbf{1}}$)
▷ initial probability vector generation
  $\forall i : \boldsymbol{\alpha}_i = \dot{\boldsymbol{\alpha}}_i U[0;1]$
  rescale $\boldsymbol{\alpha}$ so that $\boldsymbol{\alpha}\mathbb{1} = 1$
▷ initial rate matrix generation
  $\forall i : \tilde{\boldsymbol{d}}_{\mathbf{1}\,i} = \dot{\boldsymbol{d}}_{\mathbf{1}\,i} U[0;1]$
  $\forall i, j, i \neq j : \tilde{\boldsymbol{D}}_{\mathbf{0}\,i,j} = \dot{\boldsymbol{D}}_{\mathbf{0}\,i,j} U[0;1]$
  $\forall i : \tilde{\boldsymbol{D}}_{\mathbf{0}\,i,i} = -\tilde{\boldsymbol{d}}_{\mathbf{1}\,i} - \sum_{j,i\neq j} \tilde{\boldsymbol{D}}_{\mathbf{0}\,i,j}$
▷ computation of generated representation mean
  $\mathbb{E}[\tilde{\mathcal{X}}] = \boldsymbol{\alpha}\tilde{\boldsymbol{D}}_{\mathbf{0}}{}^{-1}\mathbb{1}$
▷ rate matrix rescaling to match the given mean
  $\forall i : \boldsymbol{d}_{\mathbf{1}\,i} = \tilde{\boldsymbol{d}}_{\mathbf{1}\,i} \left(\mathbb{E}[\mathcal{X}]/\mathbb{E}[\tilde{\mathcal{X}}]\right)$
  $\forall i, j : \boldsymbol{D}_{\mathbf{0}\,i,j} = \tilde{\boldsymbol{D}}_{\mathbf{0}\,i,j} \left(\mathbb{E}[\mathcal{X}]/\mathbb{E}[\tilde{\mathcal{X}}]\right)$
  return $(\boldsymbol{\alpha}, \boldsymbol{D}_{\mathbf{0}})$
end procedure

1.4. Expectation Maximization method for Hyper-Erlang fitting

An algorithm of the EM method can be simplified by imposing certain restrictions on the structure. One of good examples is the Hyper-Erlang distributions [21] which can be used to approximate any positive random variable distribution [26, 13]. Hyper-Erlang distribution is specified by parameters $(\boldsymbol{\pi}, \boldsymbol{\lambda})$, and its structure is specified by Erlang branch orders $\boldsymbol{r}$. The EM algorithm for this class of Phase-type distributions is given in [2].

In this case a continuous time Markov process $\mathcal{J}_t^{(k)}$ which generates a random variable $\mathcal{X}_k$ can be fully described by the initial state $\mathcal{I}_0^{(k)}$. That is because $\mathcal{I}_0^{(k)}$ is sufficient to determine the chosen Erlang branch. More specifically, let us say that $i_0^{(k)}$ state was randomly chosen for generating x_k. Index i of the Erlang branch can be looked up (i.e., by $i^{(k)} = \boldsymbol{s}_i$), and the states ($\boldsymbol{r}_i$ of them) which are to be visited are determined. However, the sojourn times spent in each branch state are not known, but that is not a problem given that the exit from each branch state has the same transition rate $\boldsymbol{\lambda}_i$. Therefore, the sufficient characterization of full process information $\boldsymbol{\mathcal{Y}}$ is

$$\boldsymbol{\mathcal{H}} = \{\mathcal{Q}_i[k]\},$$

where $\mathcal{Q}_i[k]$ is an indicator of $\mathcal{X}_k$ being generated by Erlang branch i

$$\mathcal{Q}_i[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(k)} = \boldsymbol{s}_i\right\}}.$$

Estimate $\boldsymbol{h}$ can be found by

$$\begin{aligned} q_i[k] &= \mathbb{E}[\mathcal{Q}_i[k] \mid \mathcal{X}_k] \\ &= \mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i \mid \mathcal{X}_k) = \frac{\mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{X}_k)}{\mathbb{P}(\mathcal{X}_k)} = \frac{\boldsymbol{\pi}_i f_i(x_k)}{\sum_{j=1}^{R} \boldsymbol{\pi}_j f_j(x_k)}, \end{aligned} \tag{14}$$

where $f_i(x)$ is the density of Erlang branch i

$$f_i(x) = \frac{(\boldsymbol{\lambda}_i x)^{\boldsymbol{r}_i - 1}}{(\boldsymbol{r}_i - 1)!} \boldsymbol{\lambda}_i e^{-\boldsymbol{\lambda}_i x}.$$

Next, maximum likelihood parameter estimates for the given observations $\boldsymbol{x}$ with weights $\boldsymbol{w}$ are found as follows:

$$\boldsymbol{\pi}_i = \frac{\sum_{k=1}^{T} q_i[k]}{\sum_{k=1}^{T} w_k} \tag{15}$$

$$\boldsymbol{\lambda}_i = \frac{\boldsymbol{r}_i \sum_{k=1}^{T} w_k q_i[k]}{\sum_{k=1}^{T} w_k q_i[k] x_k}. \tag{16}$$

The expression of the likelihood function gets to be

$$\mathcal{L}\left(\boldsymbol{\pi}, \boldsymbol{\lambda} \mid \boldsymbol{\mathcal{X}}\right) = \prod_{k=1}^{T} \sum_{i=1}^{R} \boldsymbol{\pi}_i f_i(x_k). \tag{17}$$

All the steps for Hyper-Erlang fitting by EM method are summarized in Algorithm 3.

Algorithm 3 Algorithm of EM method for the Hyper-Erlang distribution fitting.

procedure HErD-EM-fitting ($\boldsymbol{x}$, $\boldsymbol{r}$)
    initial parameter $(\boldsymbol{\pi}, \boldsymbol{\lambda})$ of structure $\boldsymbol{r}$ generation by Algorithm 4
    likelihood of $(\boldsymbol{\pi}, \boldsymbol{\lambda})$ computation by (17)
    repeat
        estimation of $\boldsymbol{h}$ by (14)
        maximum likelihood estimation of $(\boldsymbol{\pi}, \boldsymbol{\lambda})$ by (15), (16)
        likelihood of $(\boldsymbol{\pi}, \boldsymbol{\lambda})$ computation by (17)
    until likelihooh keeps increasing
    return $(\boldsymbol{\pi}, \boldsymbol{\lambda})$
end procedure

In case of Hyper-Erlang fitting to theoretical distributions, observations x_k are obtained by discretization, and their weights w_k can be assigned the probabilities.

The initial parameters can be generated by Algorithm 4. The generated Erlang branch rates are rescaled to match the given inter-arrival mean $\mathbb{E}[\mathcal{X}]$.

Algorithm 4 Algorithm for generating initial parameters of Hyper-Erlang distribution.

procedure HErD-parameter-generation ($\mathbb{E}[\mathcal{X}]$, $\boldsymbol{r}$)
▷ initial Erlang branch rate generation
$\forall i : \tilde{\boldsymbol{\pi}}_i = U[0;1]$
$\forall i : \boldsymbol{\pi}_i = \tilde{\boldsymbol{\pi}}_i / \sum_{k=1}^{R} \tilde{\boldsymbol{\pi}}_k$
▷ initial Erlang branch rate generation
$\forall i : \tilde{\boldsymbol{\lambda}}_i = U[0;1]$
▷ computation of generated representation mean
$\mathbb{E}[\tilde{\mathcal{X}}] = \sum_{k=1}^{R} \boldsymbol{\pi}_k \boldsymbol{r}_k / \tilde{\boldsymbol{\lambda}}_k$
▷ Erlang branch rate rescaling to match the given mean
$\forall i : \boldsymbol{\lambda}_i = \tilde{\boldsymbol{\lambda}}_i \left(\mathbb{E}[\mathcal{X}] / \mathbb{E}[\tilde{\mathcal{X}}] \right)$
return $(\boldsymbol{\pi}, \boldsymbol{\lambda})$
end procedure

1.5. Markov arrival processes

Let us consider a certain phenomenon which generates events (i.e., arrivals) at time instances t_k. If the inter-arrivals ($x_k = t_k - t_{k-1}$) are independent, Phase-type distributions could be used for modeling. However, if these inter-arrivals exhibit any kind of dependency, a more general process should be used to capture it.

More specifically, the distribution of the initial state should depend on how the current inter-arrival has been generated. This can be achieved by a continuous time Markov chain which has a number of arrival generating transitions. The ordinary transitions are given by rate matrix $\boldsymbol{D_0}$, and the ones that generate arrivals are given by rate matrix $\boldsymbol{D_1}$.

Example 1.8 Let us consider a three state continuous time Markov chain specified by

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.3 & 0 & 0.7 \end{bmatrix}, \boldsymbol{Q} = \begin{bmatrix} -2 & 2 & 0 \\ 2 & -6 & 4 \\ 0 & 1 & -1 \end{bmatrix},$$

which is depicted in Figure 1.6.

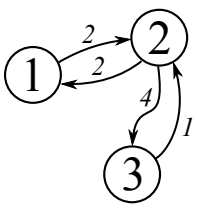

Fig. 1.6. An example of three state continuous time Markov chain.

Next, the arrival generating transitions are picked and indicated in Figure 1.7.

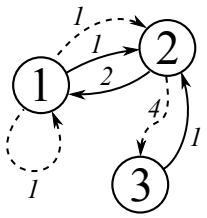

Fig. 1.7. Continuous time Markov chain schema with arrival generating transitions indicated by dashes.

The transition from the first state to itself in Markov chains, we consider, is not valid. This discrepancy can be resolved by agreeing to transition (by $\boldsymbol{D_1}$) into a duplicated set of states at the moment of arrival. The set of duplicated states is called a *level*. The process within states of a level is said to be the background process governed by matrix $\boldsymbol{D_0}$. The concept of the levels and the background process for this example is illustrated in Figure 1.8.

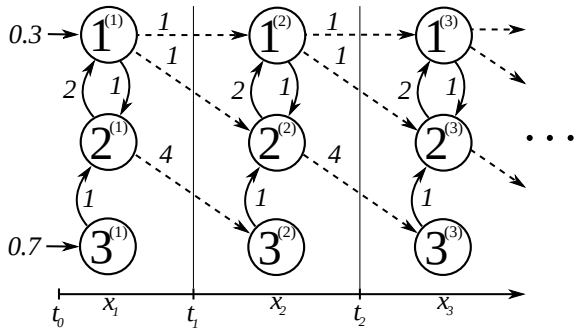

Fig. 1.8. An infinite continuous time Markov chain of MAP process.

The parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0}, \boldsymbol{D_1})$ which define CTMC (Figure 1.8) are

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.3 & 0 & 0.7 \end{bmatrix}, \boldsymbol{D_0} = \begin{bmatrix} -3 & 1 & 0 \\ 2 & -6 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \boldsymbol{D_1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix}.$$

It is clear that the choice distribution of the initial state for the next level depends on the last visited state in the previous level. △

Definition 1.23 [27, 28, 2] *Markov arrival process specified by parameters* $(\boldsymbol{\alpha}, \boldsymbol{D_0}, \boldsymbol{D_1})$ *is an irreducible Markov chain in a finite set of states* $\boldsymbol{\mathcal{S}}$ *with generator* $\boldsymbol{Q} = \boldsymbol{D_0} + \boldsymbol{D_1}$, *where* $\boldsymbol{D_1} > \boldsymbol{0}$, $\boldsymbol{D_1} \neq \boldsymbol{0}$, $\boldsymbol{D_0}_{i,j} \geq 0$ *for* $i \neq j$.

State i is chosen with probability $\boldsymbol{\alpha}_i$, in which, the process stays for an exponen-

tially distributed time at a rate

$$\boldsymbol{\lambda}_i = \sum_{i \neq j} \boldsymbol{D_0}_{i,j} + \sum_j \boldsymbol{D_1}_{i,j} = -\boldsymbol{D_0}_{i,i}.$$

The arrival is generated with the probability $\sum_j \boldsymbol{D_1}_{i,j}/\boldsymbol{\lambda}_i$, and the next state j is chosen with probability $(\boldsymbol{D_0}_{i,j} + \boldsymbol{D_1}_{i,j})/\boldsymbol{\lambda}_i$ for $i \neq j$ and with probability $\boldsymbol{D_1}_{i,j}$ for $i = j$.

The stationary Markov arrival process is specified by parameters $(\boldsymbol{D_0}, \boldsymbol{D_1})$, whereas initial stationary probability distribution $\boldsymbol{\alpha}$ is found by solving the system of equations (18)

$$\begin{cases} \boldsymbol{\alpha P} = \boldsymbol{\alpha}, \\ \boldsymbol{\alpha}\mathbb{1} = 1, \end{cases} \tag{18}$$

where $\boldsymbol{P} = (-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}$.

For a valid MAP representation $(\boldsymbol{\alpha}, \boldsymbol{D_0}, \boldsymbol{D_1})$, $(\boldsymbol{\alpha}, \boldsymbol{D}_0)$ represents a valid PH distribution. MAP with representation $(\boldsymbol{\alpha}, \boldsymbol{D_0}, \boldsymbol{d_1}\boldsymbol{\alpha})$, where $\boldsymbol{d_1} = -\boldsymbol{D_0}\mathbb{1}$, generates events whose inter-arrival times have a Phase-type distribution with parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0})$.

Let us consider a possibly correlated sequence of inter-arrivals

$$\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_T)$$

and denote the random inter-arrival time with $\mathcal{X}$. The joint density of inter-arrival sequence

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_T)$$

is

$$f(\boldsymbol{x}) = \boldsymbol{\alpha} e^{\boldsymbol{D_0} x_0} \boldsymbol{D_1} e^{\boldsymbol{D_0} x_1} \boldsymbol{D_1} \ldots e^{\boldsymbol{D_0} x_T} \boldsymbol{D_1}\mathbb{1}, \text{ for } x_1, x_2, \ldots, x_T \geq 0.$$

Then, the likelihood of MAP parameters $(\boldsymbol{D_0}, \boldsymbol{D_1})$ is

$$\mathcal{L}(\boldsymbol{D_0}, \boldsymbol{D_1} \mid \boldsymbol{\mathcal{X}}) = \boldsymbol{\alpha} \left(\prod_{k=1}^{T} \boldsymbol{A}[u] \right) \mathbb{1}, \tag{19}$$

where $\boldsymbol{A}[k] = e^{\boldsymbol{D_0} x_k} \boldsymbol{D_1}$. The joint moments of k consecutive inter-arrivals with orders i_l $(l = 1, \ldots, k)$ are given by

$$\mathbb{E}[\mathcal{X}_1^{i_1} \mathcal{X}_2^{i_2} \ldots \mathcal{X}_k^{i_k}] = \int_0^\infty \cdots \int_0^\infty (x_1)^{i_1} \ldots (x_k)^{i_k} f(\boldsymbol{x}) \mathrm{d}x_1 \ldots \mathrm{d}x_k$$

and can be calculated for a MAP by

$$\mathbb{E}[\mathcal{X}_1^{i_1} \mathcal{X}_2^{i_2} \ldots \mathcal{X}_k^{i_k}] = i_1! i_2! \ldots i_k! \boldsymbol{\alpha} (-\boldsymbol{D_0})^{-i_1} \boldsymbol{P} (-\boldsymbol{D_0})^{-i_2} \ldots \boldsymbol{P} (-\boldsymbol{D_0})^{-i_k} \mathbb{1}.$$

The first order autocorrelation coefficient $-1 \geq \rho_k \geq 1$ between the first event and the k^{th} event (i.e., lag-k) is

$$\rho_k = \frac{\mathbb{E}[\mathcal{X}_1, \mathcal{X}_{1+k}] - \mathbb{E}[\mathcal{X}])^2}{\mathbb{E}[\mathcal{X}^2] - \mathbb{E}^2[\mathcal{X}]} = \frac{\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\boldsymbol{P}^k(-\boldsymbol{D_0})^{-1}\mathbb{1} - (\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\mathbb{1})^2}{2\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-2}\mathbb{1} - (\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\mathbb{1})^2}.$$

For the uncorrelated events, the value of the autocorrelation coefficient is zero. The sign of the autocorrelation coefficient indicates the positive/negative autocorrelation between inter-arrivals. Also, we have $\lim_{k\to\infty} \rho_k = 0$.

[29] discusses how MAP properties are derived from the general theory of Markov processes.

The canonical forms are only known for order two MAPs as presented in [30]. Matrix $\boldsymbol{P}$ of MAP(2) has two eigenvalues 1, γ where $-1 \geq \gamma \geq 1$, and the autocorrelation can be written as

$$\rho_k = \frac{\gamma^k}{2}\frac{\mathbb{E}[\mathcal{X}_1, \mathcal{X}_{1+k}] - 2\mathbb{E}[\mathcal{X}])^2}{\mathbb{E}[\mathcal{X}^2] - \mathbb{E}^2[\mathcal{X}]} = \frac{2(\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-2}\mathbb{1} - (\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\mathbb{1})^2)}{2\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-2}\mathbb{1} - (\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\mathbb{1})^2}.$$

The first form of the canonical MAP(2) representation for the case $\gamma > 0$ is

$$\boldsymbol{D_0} = \begin{bmatrix} -\lambda_1 & (1-a)\lambda_1 \\ 0 & -\lambda_2 \end{bmatrix},\ \boldsymbol{D_1} = \begin{bmatrix} a\lambda_1 & 0 \\ (1-b)\lambda_2 & b\lambda_2 \end{bmatrix},\ \gamma = ab \tag{20}$$

and the second form, for case $\gamma < 0$, is

$$\boldsymbol{D_0} = \begin{bmatrix} -\lambda_1 & (1-a)\lambda_1 \\ 0 & -\lambda_2 \end{bmatrix},\ \boldsymbol{D_1} = \begin{bmatrix} 0 & a\lambda_1 \\ b\lambda_2 & (1-b)\lambda_2 \end{bmatrix},\ \gamma = -ab, \tag{21}$$

where $0 < \lambda_1 \leq \lambda_2$, $0 \geq a \geq 1$ and $0 \geq a \geq 1$. The additional constraints for the first form are $a, b \neq 1$. For the second form, the constraints are $b \neq 0$ and $\lambda_1 \neq \lambda_2$ if $a = 1$. Since the canonical form does not impose any restrictions on the initial probability vector, we shall refer to these canonical forms as cases of the NMAP(2) process.

Next, we overview the MAP process of the ER-CHMM structure. In this arrival process, the inter-arrival times have Erlang's distribution (ER) and are modulated by a discrete time hidden Markov chain (HMM, hence the name of the structure). An important qualitative property of this structural restriction is that given the state of the modulating Markov chain, the inter-arrival time and the next state of the modulating Markov chain are independent random variables, which is not the case with general MAPs. Despite this independence, the inter-arrival times generated by the ER-CHMM structure are still correlated (in general).

The ER-CHMM process is specified by Erlang branch order vector $\boldsymbol{r}$ and the parameters $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$. Matrix $\boldsymbol{\Pi}$ denotes Erlang branch switching probabilities at the time instance of arrival.

Example 1.9 CTMC schema of ER-CHMM process with structure $\boldsymbol{r} = \begin{bmatrix} 2 & 1 \end{bmatrix}$ is shown in Figure 1.9.

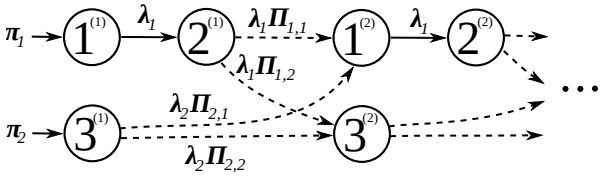

Fig. 1.9. An infinite continuous time Markov chain of ER-CHMM process.

△

Again, we choose a stationary initial Erlang branch probability distribution $\boldsymbol{\pi}$ by solving (22)

$$\begin{cases} \boldsymbol{\pi}\boldsymbol{\Pi} = \boldsymbol{\pi}, \\ \boldsymbol{\pi}\mathbb{1} = 1. \end{cases} \tag{22}$$

The stationary ER-CHMM process can be specified by parameters $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$. Further in the thesis, by 'ER-CHMM process', we shall refer to the stationary version of it.

For the given parameters $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ and structure $\boldsymbol{r}$, it is easy to obtain the matrix form representation $(\boldsymbol{D_0}, \boldsymbol{D_1})$ which is expressed by

$$\begin{aligned} \boldsymbol{D_0} &= \begin{bmatrix} \boldsymbol{B}(\boldsymbol{r}_1, \boldsymbol{\lambda}_1) & & \\ & \ddots & \\ & & \boldsymbol{B}(\boldsymbol{r}_R, \boldsymbol{\lambda}_R) \end{bmatrix}, \\ \boldsymbol{D_1} &= \begin{bmatrix} -\boldsymbol{B}(\boldsymbol{r}_1, \boldsymbol{\lambda}_1)\mathbb{1}\boldsymbol{\Pi}_{1,1}\boldsymbol{\beta}(\boldsymbol{r}_1) & \dots & -\boldsymbol{B}(\boldsymbol{r}_1, \boldsymbol{\lambda}_1)\mathbb{1}\boldsymbol{\Pi}_{1,R}\boldsymbol{\beta}(\boldsymbol{r}_R) \\ \vdots & \ddots & \vdots \\ -\boldsymbol{B}(\boldsymbol{r}_R, \boldsymbol{\lambda}_R)\mathbb{1}\boldsymbol{\Pi}_{R,1}\boldsymbol{\beta}(\boldsymbol{r}_1) & \dots & -\boldsymbol{B}(\boldsymbol{r}_R, \boldsymbol{\lambda}_R)\mathbb{1}\boldsymbol{\Pi}_{R,R}\boldsymbol{\beta}(\boldsymbol{r}_R) \end{bmatrix}. \end{aligned} \tag{23}$$

From the computational point of view, the most beneficial feature of the ER-CHMM structure is that in the computation of matrices $\boldsymbol{A}[k]$, the key element of the joint density function is significantly simpler with the ER-CHMM structure than in the case of general MAPs due to the conditional independence of the inter-arrival time and the next state of the modulating Markov chain, which is provided by the ER-CHMM structure. As a consequence, the computation of matrices $\boldsymbol{A}[k]$ does not rely on the computationally heavy matrix-exponential function, but it is simplified to a scalar product of $\boldsymbol{\Pi}_{i,j}$ and $f_i(x_k)$ as it is demonstrated by the relevant elements of $\boldsymbol{A}[k]$ in the following Example 1.10.

Example 1.10 Let us continue Example 1.9. The ER-CHMM process with Erlang branch orders $\boldsymbol{r} = \begin{bmatrix} 2 & 1 \end{bmatrix}$ can be specified by the following matrix form representation

$(\boldsymbol{D_0}, \boldsymbol{D_1})$

$$\boldsymbol{D_0} = \begin{bmatrix} -\boldsymbol{\lambda}_1 & \boldsymbol{\lambda}_1 & 0 \\ 0 & -\boldsymbol{\lambda}_1 & 0 \\ 0 & 0 & -\boldsymbol{\lambda}_2 \end{bmatrix}, \quad \boldsymbol{D_1} = \begin{bmatrix} 0 & 0 & 0 \\ \boldsymbol{\Pi}_{1,1}\boldsymbol{\lambda}_1 & 0 & \boldsymbol{\Pi}_{1,2}\boldsymbol{\lambda}_1 \\ \boldsymbol{\Pi}_{2,1}\boldsymbol{\lambda}_2 & 0 & \boldsymbol{\Pi}_{2,2}\boldsymbol{\lambda}_2 \end{bmatrix},$$

also, matrices $\boldsymbol{A}[k]$ are

$$\boldsymbol{A}[k] = e^{\boldsymbol{D_0} x_k} \boldsymbol{D_1} = \begin{bmatrix} \boldsymbol{\Pi}_{1,1} f_1(x_k) & 0 & \boldsymbol{\Pi}_{1,2} f_1(x_k) \\ \bullet & 0 & \bullet \\ \boldsymbol{\Pi}_{2,1} f_2(x_k) & 0 & \boldsymbol{\Pi}_{2,2} f_2(x_k) \end{bmatrix},$$

where $\bullet$ indicates matrix entries which are irrelevant because, after an arrival event, the phase is either 1 or 3 due to the zero column. This fact allows for dimension reduction. $\triangle$

The dimension reduction, as demonstrated in Example 1.10, can be utilized as follows: instead of relying on size n matrices $\boldsymbol{A}[k]$, matrices $\boldsymbol{P}[k]$ of size R defined as

$$\boldsymbol{P}[k] = \begin{bmatrix} \boldsymbol{\Pi}_{1,1} f_1(x_k) & \dots & \boldsymbol{\Pi}_{1,R} f_1(x_k) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Pi}_{R,1} f_R(x_k) & \dots & \boldsymbol{\Pi}_{R,R} f_R(x_k) \end{bmatrix}, \tag{24}$$

where $f_i(x)$ is the density function of i^{th} Erlang branch (distribution)

$$f_i(x) = \frac{(\boldsymbol{\lambda}_i x)^{\boldsymbol{r}_i - 1}}{(\boldsymbol{r}_i - 1)!} \boldsymbol{\lambda}_i e^{-\boldsymbol{\lambda}_i x}. \tag{25}$$

Then, the likelihood of parameters $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ can be computed by

$$\mathcal{L}\left(\boldsymbol{\Pi}, \boldsymbol{\lambda} \mid \boldsymbol{\mathcal{X}}\right) = \boldsymbol{\pi} \left(\prod_{k=1}^{T} \boldsymbol{P}[k] \right) \mathbb{1}. \tag{26}$$

We observe that, since $R \leq N$ holds, (26) involves smaller matrices and lacks matrix-exponential functions compared to (19).

1.6. Expectation Maximization method for Markov arrival process fitting (ER-CHMM)

The expectation maximization method for Markov arrival process fitting is applied in a similar way to Phase-type distributions.

Let us refer to $\mathcal{X}_k$ as a random time (i.e., inter-arrival) spent in k^{th} level. Since sequence

$$\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_T)$$

can be correlated, individual processes $\mathcal{J}_t^{(k)}$ generating each $\mathcal{X}_k$ have to be investigated as a whole

$$\mathcal{J}_t^{(1)}, \mathcal{J}_t^{(2)}, ..., \mathcal{J}_t^{(T)}.$$

At the time instance of $t = \mathcal{X}_1 + \cdots + \mathcal{X}_k$, process $\mathcal{J}_t^{(k)}$ is switched to $\mathcal{J}_t^{(k+1)}$. This sequence of processes is fully characterized by discrete state process

$$\mathcal{I}_0^{(1)}, ..., \mathcal{I}_{\mathcal{M}^{(1)}-1}^{(1)}, ..., \mathcal{I}_{\mathcal{M}^{(k)}-1}^{(k)}, ..., \mathcal{I}_{\mathcal{M}^{(T)}-1}^{(T)}$$

and sojourn time discrete process

$$\mathcal{S}_0^{(1)}, ..., \mathcal{S}_{\mathcal{M}^{(1)}-1}^{(1)}, ..., \mathcal{S}_{\mathcal{M}^{(k)}-1}^{(k)}, ..., \mathcal{S}_{\mathcal{M}^{(T)}-1}^{(T)}$$

which model the time spent in the visited states. Here, $\mathcal{M}^{(k)}$ is a random number of transitions within the k^{th} level, before the arrival generating transition occurs. Thus full information $\boldsymbol{\mathcal{Y}}$ about the process which generates the observed sequence $x_1, ..., x_T$ is

$$\boldsymbol{y} = \left(i_0^{(1)}, ..., i_{m^{(0)}-1}^{(1)}, ..., i_{m^{(T)}-1}^{(T)}, s_0^{(1)}, ..., s_{m^{(0)}-1}^{(1)}, ..., s_{m^{(T)}-1}^{(T)} \right).$$

The fitting algorithm can be substantially simplified by using a restricted class of Markov arrival processes, the ER-CHMM process. Considering the fact that in order to accurately capture autocorrelation, a long sequence of inter-arrivals has to be used, it is reasonable to use a stationary distribution of choosing the Erlang branch. Thus the parameters to be searched for are $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$.

In this case, full information $\boldsymbol{\mathcal{Y}}$ can be characterized by

$$\boldsymbol{\mathcal{H}} = \{\mathcal{Q}_i[k], \mathcal{Q}_{i,j}[k]\},$$

where $\mathcal{Q}_i[k]$ is the indicator of $\mathcal{X}_k$ being generated by Erlang branch i

$$\mathcal{Q}_i[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(k)} = \boldsymbol{s}_i\right\}}$$

and $\mathcal{Q}_{i,j}[k]$ is the indicator of transition to Erlang branch j after an arrival has been generated by branch i

$$\mathcal{Q}_{i,j}[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{I}_0^{(k+1)} = \boldsymbol{s}_j\right\}}.$$

Estimate $\boldsymbol{h}$ of sufficient characteristics $\boldsymbol{\mathcal{H}}$ is found by

$$\begin{aligned}
q_i[k] &= \mathbb{E}[\mathcal{Q}_i[k] \mid \boldsymbol{\mathcal{X}}] = \mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i \mid \boldsymbol{\mathcal{X}}) \\
&= \frac{\mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \boldsymbol{\mathcal{X}})}{\mathbb{P}(\boldsymbol{\mathcal{X}})} = \frac{\boldsymbol{a}_i[k-1]\boldsymbol{b}_i[k]}{\boldsymbol{\pi}\boldsymbol{b}[1]}, \\
q_{i,j}[k] &= \mathbb{E}[\mathcal{Q}_{i,j}[k] \mid \boldsymbol{\mathcal{X}}] = \mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{I}_0^{(k+1)} = \boldsymbol{s}_j \mid \boldsymbol{\mathcal{X}}) \\
&= \frac{\mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{I}_0^{(k+1)} = \boldsymbol{s}_j, \boldsymbol{\mathcal{X}})}{\mathbb{P}(\boldsymbol{\mathcal{X}})} \\
&= \frac{\boldsymbol{a}_i[k-1] f_i(x_k) \boldsymbol{\Pi}_{i,j} \boldsymbol{b}_j[k+1]}{\boldsymbol{\pi}\boldsymbol{b}[1]},
\end{aligned} \tag{27}$$

where $\boldsymbol{a}[k]$, $\boldsymbol{b}[k]$ are likelihood vectors of choosing the particular Erlang branch

$$\mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{X}_1, ..., \mathcal{X}_k) = \boldsymbol{a}_i[k],$$
$$\mathbb{P}(\mathcal{I}_0^{(k)} = \boldsymbol{s}_i, \mathcal{X}_k, ..., \mathcal{X}_T) = \boldsymbol{b}_i[k],$$

which can be evaluated by the recurrent formulas

$$\begin{aligned} \boldsymbol{a}_i[k] &= \begin{cases} \boldsymbol{\pi}_i, & k = 0, \\ \sum_{j=1}^{R} \boldsymbol{a}_j[k-1] f_j(x_k) \boldsymbol{\Pi}_{j,i}, & 1 \leq k \leq T, \end{cases} \\ \boldsymbol{b}_j[k] &= \begin{cases} \sum_{i=1}^{R} f_j(x_k) \boldsymbol{\Pi}_{j,i} \boldsymbol{b}_i[k+1], & 1 \leq k \leq T, \\ 1, & k = T+1. \end{cases} \end{aligned} \tag{28}$$

A similar recursion can be defined in case of full MAPs, too, see equations (13) and (14) in [31].

Example 1.11 Probability of inter-arrival $\mathcal{X}_2$ being generated by Erlang branch i, given $\mathcal{X}_1$ is

$$\mathbb{P}(\mathcal{I}_0^{(2)} = \boldsymbol{s}_i \mid \mathcal{X}_1, \mathcal{X}_2) = \frac{\mathbb{P}(\mathcal{I}_0^{(2)} = \boldsymbol{s}_i, \mathcal{X}_1, \mathcal{X}_2)}{\mathbb{P}(\mathcal{X}_1, \mathcal{X}_2)} = \frac{\boldsymbol{a}_i[2]}{\boldsymbol{a}[2]\mathbb{1}}. \qquad \triangle$$

Next, the maximum likelihood parameter $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ estimates are found by

$$\boldsymbol{\lambda}_i = \frac{\boldsymbol{r}_i \sum_{k=1}^{T} q_i[k]}{\sum_{k=1}^{T} x_k q_i[k]}, \tag{29}$$

$$\boldsymbol{\Pi}_{i,j} = \frac{\sum_{k=1}^{T} q_{i,j}[k]}{\sum_{k=1}^{T} q_i[k]}. \tag{30}$$

Stationary initial probability distribution $\boldsymbol{\pi}$ can be found by solving (22), or, alternatively, by

$$\boldsymbol{\pi}_i = \frac{\sum_{u=1}^{T} \boldsymbol{a}_i[u-1] \boldsymbol{b}_i[u]}{T \boldsymbol{a}[T] \mathbb{1}}. \tag{31}$$

The likelihood of $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ using likelihood vectors can be computed simply as

$$\mathcal{L}(\boldsymbol{\Pi}, \boldsymbol{\lambda} \mid \boldsymbol{\mathcal{X}}) = \boldsymbol{a}[T]\mathbb{1} = \boldsymbol{\pi}\boldsymbol{b}[1]. \tag{32}$$

The steps of ER-CHMM fitting by the EM method are summarized in Algorithm 5.

Algorithm 5 ER-CHMM process fitting by expectation maximization method.

procedure ER-CHMM-EM-fitting ($\boldsymbol{x}$, $\boldsymbol{r}$)
    initial parameter $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ of structure $\boldsymbol{r}$ generation by Algorithm 6
    likelihood of $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ is computated by (32)
    repeat
        estimate $\boldsymbol{h}$ is computed by (27)
        maximum likelihood parameters $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ are computed by (29), (30)
        likelihood of $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$ is computated by (32)
    until likelihood keeps increasing
    return $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$
end procedure

Some heuristics for choosing Erlang branch orders $\boldsymbol{r}$ is presented in [32].

The initial parameters can be generated by Algorithm 6. After generating initial Erlang branch switching probabilities, the stationary initial probability vector is computed. Then the inter-arrival mean of the process with generated parameters is computed and used to rescale Erlang branch rates so that to match the given inter-arrival mean $\mathbb{E}[\mathcal{X}]$.

Algorithm 6 Algorithm for generating initial parameters of stationary MAP process of structure ER-CHMM.

procedure ER-CHMM-parameter-generation ($\mathbb{E}[\mathcal{X}]$, $\boldsymbol{r}$)
▷ initial Erlang branch rate generation
    $\forall i : \tilde{\boldsymbol{\lambda}}_i = U[0; 1]$
▷ initial Erlang branch switching probability matrix generation
    $\forall i, j : \tilde{\boldsymbol{\Pi}}_{i,j} = U[0; 1]$
    $\forall i, j : \boldsymbol{\Pi}_{i,j} = \tilde{\boldsymbol{\Pi}}_{i,j} / \sum_{k=1}^{R} \tilde{\boldsymbol{\Pi}}_{i,k}$
▷ stationary initial probability vector computation
    $\begin{cases} \boldsymbol{\pi}\boldsymbol{\Pi} = \boldsymbol{\pi} \\ \boldsymbol{\pi}\mathbb{1} = 1 \end{cases} \rightarrow \boldsymbol{\pi}$
▷ computation of generated representation mean
    $\mathbb{E}[\tilde{\mathcal{X}}] = \sum_{k=1}^{R} \boldsymbol{\pi}_k \boldsymbol{r}_k / \tilde{\boldsymbol{\lambda}}_k$
▷ Erlang branch rate rescaling to match the given mean
    $\forall i : \boldsymbol{\lambda}_i = \tilde{\boldsymbol{\lambda}}_i \left(\mathbb{E}[\mathcal{X}] / \mathbb{E}[\tilde{\mathcal{X}}] \right)$
    return $(\boldsymbol{\Pi}, \boldsymbol{\lambda})$
end procedure

1.7. Transient Markov arrival processes

Transient Markovian arrival processes (TMAPs) are continuous time terminating point processes where the inter-arrival times depend on a background Markov chain, hence they can be dependent.

TMAPs can be characterized by an initial probability vector, $\boldsymbol{\alpha}$, i.e., $\boldsymbol{\alpha}\mathbb{1} = 1$, holding the initial state distribution of the background Markov chain at time 0 and

two matrices, $\boldsymbol{D_0}$ and $\boldsymbol{D_1}$. Matrix $\boldsymbol{D_0}$ contains the rates of the internal transitions that are not accompanied by an arrival, whereas matrix $\boldsymbol{D_1}$ consists of the rates of those transitions that generate an arrival. However, contrary to non-terminating MAPs, the generator matrix of the background Markov chain of TMAPs, $\boldsymbol{D} = \boldsymbol{D_0} + \boldsymbol{D_1}$, is transient, that is, $\boldsymbol{D}\mathbb{1} \neq 0$, and the non-negative vector $\boldsymbol{d} = -\boldsymbol{D}\mathbb{1}$ describes the termination rates of the background Markov chain. For practical considerations, it is assumed that the termination is an observed event (an arrival), which means that a TMAP generates at least one arrival. If only the 'arrival events' are known, which is commonly the case in practice, the TMAPs which do not generate any arrival are not observed.

Matrix $\boldsymbol{P} = (-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}$ holds the state transition probabilities of a transient discrete time Markov chain (DTMC) with termination vector $\boldsymbol{p} = \mathbb{1} - \boldsymbol{P}\mathbb{1}$. We should note that $\boldsymbol{P}$ is a sub-stochastic matrix (it has non-negative elements and $\boldsymbol{P}\mathbb{1} \leq \mathbb{1}$), and $(\boldsymbol{I} - \boldsymbol{P})^{-1}\boldsymbol{p} = \mathbb{1}$ holds.

Example 1.12 Let us consider a transient Markov arrival process specified by parameters

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.3 & 0 & 0.7 \end{bmatrix}, \boldsymbol{D_0} = \begin{bmatrix} -3 & 1 & 0 \\ 2 & -8 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \boldsymbol{D_1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix},$$

whose continuous time Markov chain is shown in Figure 1.10.

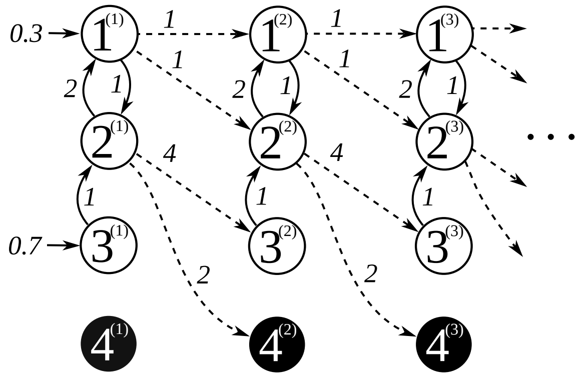

Fig. 1.10. Continuous time Markov chain of transient Markov arrival process given in Example 1.12.

The vector of absorption rates is

$$\boldsymbol{d} = -(\boldsymbol{D_0} + \boldsymbol{D_1})\mathbb{1} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}. \qquad \triangle$$

In case of TMAPs, not only the statistical quantities related to the inter-arrival times are of interest, but also the ones related to the number of generated arrivals attract interest as well.

The number of arrivals $\mathcal{K}$ is characterized by a discrete Phase-type (DPH) distribution with initial vector $\boldsymbol{\alpha}$ and transition probability matrix $\boldsymbol{P}$. Hence, the mean number of arrivals is given by

$$\mathbb{E}[\mathcal{K}] = \sum_{k=1}^{\infty} \boldsymbol{\alpha} k \boldsymbol{P}^{k-1} \boldsymbol{p} = \boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-2}\boldsymbol{p} = \boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-1}\mathbb{1}. \tag{33}$$

If the inter-arrival times are denoted by $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_K)$, then the joint density function of the inter-arrival times is

$$\begin{aligned} f(\boldsymbol{x}) &= \lim_{\Delta \to 0} \frac{1}{\Delta} \mathbb{P}(\mathcal{X}_1 \in (x_1, x_1+\Delta), \ldots, \mathcal{X}_k \in (x_k, x_k+\Delta)) \\ &= \boldsymbol{\alpha} e^{\boldsymbol{D}_0 x_1} \boldsymbol{D}_1 e^{\boldsymbol{D}_0 x_2} \boldsymbol{D}_1 \cdots e^{\boldsymbol{D}_0 x_k} (\boldsymbol{D}_1 \mathbb{1} + \boldsymbol{d}). \end{aligned} \tag{34}$$

If it exists, the n^{th} moment of $\mathcal{X}_{k+1}$ is

$$\mathbb{E}[\mathcal{X}_{k+1}^n | \mathcal{X}_{k+1} < \infty] = \frac{\mathbb{E}[\mathcal{X}_{k+1}^n \mathbb{1}_{\{\mathcal{X}_{k+1} < \infty\}}]}{\mathbb{P}(\mathcal{X}_{k+1} < \infty)} = \frac{n! \boldsymbol{\alpha} \boldsymbol{P}^k (-\boldsymbol{D_0})^{-n} \mathbb{1}}{\boldsymbol{\alpha} \boldsymbol{P}^k \mathbb{1}}. \tag{35}$$

The mean of the inter-arrival times $\mathbb{E}[\mathcal{X}]$ is not as easy to express as for ordinary MAPs, it is obtained from $\mathbb{E}[\mathcal{X}] = \mathbb{E}\left[\sum_{k=1}^{\mathcal{K}} \mathcal{X}_k\right] / \mathbb{E}[\mathcal{K}]$ where the numerator is derived as

$$\begin{aligned} \mathbb{E}\left[\sum_{k=1}^{\mathcal{K}} \mathcal{X}_k\right] &= \sum_{\kappa=1}^{\infty} \mathbb{E}\left[\mathbb{1}_{\{\mathcal{K}=\kappa\}} \sum_{k=1}^{\mathcal{K}} \mathcal{X}_k\right] = \sum_{\kappa=1}^{\infty} \sum_{i=0}^{\kappa-1} \boldsymbol{\alpha} \boldsymbol{P}^i \boldsymbol{M} \boldsymbol{P}^{\kappa-1-i} \boldsymbol{p} \\ &= \sum_{i=0}^{\infty} \sum_{\kappa=0}^{\infty} \boldsymbol{\alpha} \boldsymbol{P}^i \boldsymbol{M} \boldsymbol{P}^{\kappa} \boldsymbol{p} = \boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-1} \boldsymbol{M} (\boldsymbol{I}-\boldsymbol{P})^{-1} \boldsymbol{p}, \end{aligned} \tag{36}$$

where $\boldsymbol{M} = (-\boldsymbol{D}_0)^{-1}$, and the denominator is given by (33). As a result, the mean inter-arrival time is

$$\mathbb{E}[\mathcal{X}] = \frac{\mathbb{E}\left[\sum_{k=1}^{\mathcal{K}} \mathcal{X}_k\right]}{\mathbb{E}[\mathcal{K}]} = \frac{\boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-1}\boldsymbol{M}(\boldsymbol{I}-\boldsymbol{P})^{-1}\boldsymbol{p}}{\boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-2}\boldsymbol{p}} = \frac{\boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-1}\boldsymbol{M}\mathbb{1}}{\boldsymbol{\alpha}(\boldsymbol{I}-\boldsymbol{P})^{-1}\mathbb{1}}. \tag{37}$$

In order to discuss the correlation of the inter-arrival times, we introduce the notation $\hat{\mathcal{X}}_k = \mathcal{X}_k \mid \mathcal{X}_k < \infty$. We should note that $\hat{\mathcal{X}}_1 = \mathcal{X}_1$ due to the modeling assumption of at least one arrival. By this notation from (35), we have

$$\mathbb{E}\left[\hat{\mathcal{X}}_{k+1}^n\right] = \frac{n! \boldsymbol{\alpha} \boldsymbol{P}^k \boldsymbol{M}^n \mathbb{1}}{\boldsymbol{\alpha} \boldsymbol{P}^k \mathbb{1}}.$$

The expectation of the product of two subsequent inter-arrival times is

$$\begin{aligned}\mathbb{E}\left[\mathcal{X}_1\hat{\mathcal{X}}_{k+1}\right] &= \frac{\mathbb{E}\left[\mathcal{X}_1\mathcal{X}_{k+1}\mathbb{1}_{\{\mathcal{X}_{k+1}<\infty\}}\right]}{\mathbb{P}(\mathcal{X}_{k+1}<\infty)} \\ &= \frac{\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-2}\boldsymbol{D_1}\boldsymbol{P}^{k-1}(-\boldsymbol{D_0})^{-2}(\boldsymbol{D_1}\mathbb{1}+\boldsymbol{d})}{\boldsymbol{\alpha}(-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}\boldsymbol{P}^{k-1}(-\boldsymbol{D_0})^{-1}(\boldsymbol{D_1}\mathbb{1}+\boldsymbol{d})} = \frac{\boldsymbol{\alpha}\boldsymbol{M}\boldsymbol{P}^k\boldsymbol{M}\mathbb{1}}{\boldsymbol{\alpha}\boldsymbol{P}^k\mathbb{1}},\end{aligned} \tag{38}$$

where we state that $(-\boldsymbol{D_0})^{-1}(\boldsymbol{D_1}\mathbb{1}+\boldsymbol{d}) = \mathbb{1}$ due to $\boldsymbol{D_0}\mathbb{1}+\boldsymbol{D_1}\mathbb{1}+\boldsymbol{d} = 0$. Based on the joint expectation, the correlation is

$$\rho_k = \frac{\mathbb{E}\left[\mathcal{X}_1\hat{\mathcal{X}}_{k+1}\right] - \mathbb{E}\left[\mathcal{X}_1\right]\mathbb{E}\left[\hat{\mathcal{X}}_{k+1}\right]}{\sqrt{\mathbb{E}\left[\mathcal{X}_1^2\right]-\mathbb{E}^2\left[\mathcal{X}_1\right]}\sqrt{\mathbb{E}\left[\hat{\mathcal{X}}_{k+1}^2\right]-\mathbb{E}^2\left[\hat{\mathcal{X}}_{k+1}\right]}}. \tag{39}$$

Given the inter-arrivals $\boldsymbol{x} = \left(x_1^{(1)},\ldots,x_{K_1}^{(1)},\ldots,x_1^{(U)},\ldots,x_{K_u}^{(U)}\right)$ of U observed runs, the likelihood of parameters $(\boldsymbol{\alpha},\boldsymbol{D_0},\boldsymbol{D_1})$ is

$$\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{D_0},\boldsymbol{D_1}\mid\boldsymbol{\mathcal{X}}) = \prod_{u=1}^{U}\boldsymbol{\alpha}e^{\boldsymbol{D_0}x_1^{(u)}}\boldsymbol{D_1}\cdots e^{\boldsymbol{D_0}x_{K_u}^{(u)}}\boldsymbol{d}. \tag{40}$$

As with MAPs, TMAP fitting can be greatly simplified by imposing the ER-CHMM structure. In this case, the process is specified by parameters $(\boldsymbol{\pi},\boldsymbol{\Pi},\boldsymbol{\lambda})$, where $\boldsymbol{\pi}_i$ is a probability of choosing the i^{th} Erlang branch, and $\boldsymbol{\Pi}_{i,j}$ is a probability that Erlang branch j will be chosen after the arrival by Erlang branch i has been generated. In addition, the probability $\boldsymbol{p}$ run termination can be derived as

$$\boldsymbol{p} = \mathbb{1}-\boldsymbol{\Pi}\mathbb{1} = (\boldsymbol{I}-\boldsymbol{\Pi})\,\mathbb{1}.$$

Example 1.13 Let us consider a transient Markov arrival process of ER-CHMM structure specified by the parameters

$$\boldsymbol{\pi} = \begin{bmatrix}0.8\\0.2\end{bmatrix},\ \boldsymbol{\Pi} = \begin{bmatrix}0.75 & 0.25\\0 & 0.5\end{bmatrix},\ \boldsymbol{\lambda} = \begin{bmatrix}4 & 2\end{bmatrix},$$

whose continous time Markov chain is shown in Figure 1.11.

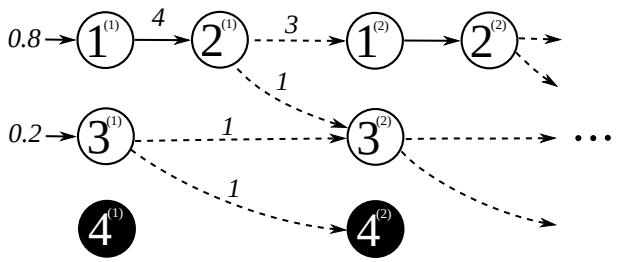

Fig. 1.11. Continuous Markov chain of the transient Markov arrival process given in Example 1.13.

Rate vector $\tilde{\boldsymbol{d}}$ of transitions to the absorbing state from Erlang branches is

$$\tilde{\boldsymbol{d}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

which can be recomputed for absorption probability vector

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}. \qquad \triangle$$

1.8. State-of-the-art overview

Phase-type (PH) distributions and Markov arrival processes (MAP) have been extensively used to create stochastic models for decades due to their analytical tractability. PH distributions are used to incorporate a certain approximated distribution into the analytical model. To name a few among many, they are used in queueing [33], modeling failure times [34], or in risk theory [35], in modeling wireless networks [21]. The information about autocorrelation can be incorporated in a model by using MAPs. These processes are used for modeling correlated workload for traffic, performance and reliability analysis in several fields [36, 37]. Contrary to MAPS, the transient Markov arrival processes (TMAPs) generate finite sequences of inter-arrivals. TMAPs can be used across a wide range of application fields from traffic modeling of computer systems to risk analysis, including population dynamics in biological systems. For example, TMAPs were applied to model women's lifespan in several countries in [38].

There are a lot of open problems for PHs, MAPs and related processes. Canonical forms are known only for low order processes. Various structure sub-classes are being investigated. Until representation problems have been solved, it is not realistic to expect to develop optimal fitting procedures. Currently, fitting is performed by several methods, including moment matching, ML based, general optimization and hybrid ones. However, one has to consider a trade-off of fast fitting with sub-class structures versus slow fitting using a general structure. Depending on the algorithm, certain numerical issues also have to be solved in case big data traces are being fitted as well.

These processes are related to other mathematical fields, and some extensions are developed, too. PH distribution relation to positive systems is investigated in [22, 39]. PH distribution generators are being investigated by using polytopes [40]. [41] discusses the usage of bivariate Phase-type distributions for modeling relapse time in clinical patients. An extension of PH distributions supporting negative values is introduced in [42]. [43] presents the properties of acyclic bilateral Phase-type (ABPH) distributions. The version of PH distribution with a finite support (FSPH) is introduced in [44].

PH distribution representations have not been fully investigated yet. PH can be fully specified by $2n-1$ moments [15]. However, for practical usage, the matrix form

representation is necessary. The PH(n) canonical matrix form is known only for orders $n = 2, 3$ [17, 8]. The authors of [8] tried to investigate the canonical forms of PH(4), but arrived at a conclusion that the problem is too complex to be solved analytically. Certain PH sub-classes are much better investigated in this regard. For example, any acyclic Phase-type (APH) distribution can be represented by a Coxian distribution [45]; in addition, the ability to represent general PH distribution by Coxian distribution is also investigated in this paper. Two algorithms for constructing bi-diagonal representations are given in [46]. Another subclass of APH distributions is the Hyper-Erlang (HErD) distribution, for which, effective fitting algorithms are developed. The HErD distribution is a mixture of Erlang distributions, which itself turns out to be the least variable one among PH distributions [20]. A hybrid subclass, constructed as a convolution of Erlang and Coxian distributions (EC), is presented in [47, 48]. As for cyclic PH subclasses, the unicyclic, Feedback Erlang and Feedback Hypo-Exponential distributions are investigated in [49]. In [11], a conjecture is given that any PH distribution has a unicyclic representation, however, in [49], a counter example is given to disprove that.

There are two main approaches for PH fitting. The first one is to perform moment matching. PH distributions can be specified by $2n - 1$ moments. However, closed form PH(n) fitting procedures are known only for order $n = 2, 3$ distributions. These results are limited due to unknown canonical forms for higher orders. For PH(n) of orders $n > 3$, moment matching is a complex optimization problem. To simplify the problem, researchers investigate PH sub-classes and/or aim to fit a small number of moments. For example, [50, 18] introduce procedures for APH distribution moment matching. [47] introduces the Erlang-Coxian (EC) distribution subclass for fitting the first three moments. The usage of EC for moment matching is further investigated in [48]. For certain applications (i.e., simple queueing models), it is enough to match a few moments, but, for more general usage, this approch is limited.

Another, a more general, approach is to use the ML parameter estimation procedure. The first EM procedure for general PH fitting is presented in [14]. The main disadvantage of the presented EM procedure is that it requires high computational effort. The most direct attempt to address the problem is to use the uniformization (randomization) technique for computing matrix exponentials and the convolution integral; such a procedure is presented in [25]. However, from our experience, in order to use the procedure for sparse structure fitting, a numerical issue related to number representation and scaling has to be addressed. Other similar results are presented in [51], where the EM method is applied to the discretized data, and it also uses the uniformization technique. [52] gives an EM fitting procedure based on uniformization and MCMC. There are numerous attempts to optimize the EM procedure for specific subclasses in scientific literature. These results proved to be useful from the practical point of view, i.e., they enabled PH fitting with many more phases. The effectiveness of using a smaller subclass for PH fitting is suggested by empirical results in [13]. [53] presents APH canonical forms and their fitting by using the EM procedure. The EM algorithms for the Hyper-Exponential distribution fitting are presented in [54, 55, 56].

The EM algorithm for Hyper-Erlang distribution fitting is presented in [2]. [12] introduced a hybrid algorithm SS&IAGA-EM Hyper-Erlang distribution fitting. The algorithm SS&IAGA-EM uses the expectation maximization (EM) method for the local search and scatter search algorithm (SS) and the improved adaptive genetic algorithm (IAGA) for better global parameter space exploration, including a number of branches and phases. The authors claim that the proposed algorithm is more effective (it explores the Hyper-Erlang structure space faster and better) compared to G-FIT [13] while producing similar fitting results.

There are more specific procedures developed for certain cases. A review of PH, MAP fitting algorithms is presented in [57]. The collection of nine distributions for testing and comparing different PH fitting approaches is presented in [58]. Certain models are sensitive to the fitting quality of heavy tailed distributions. The PH fitting to heavy tailed distributions is investigated in [59, 60, 55]. Other less traditional approaches include Hyper-Exponential distribution fitting in the Laplace domain [61]. Also, general optimization algorithms are used, for example, [62] uses Quasi-Newton and Nelder Mead algorithms to fit a Coxian distribution.

It is important to have canonical matrix forms (i.e., minimal and unique representations) of MAP(n) for an efficient MAP application. The matrix form representation with all possible transitions is redundant, and its parameter search with the EM algorithm is computationally demanding [63]. On the other hand, a simpler EM algorithm can be derived for a certain MAP subclass (for example, ER-CHMM), but it may not cover the whole MAP(n) space with the same number of phases. A unique representation makes the whole parameter search process simpler, which results in a faster convergence to a set of parameters. Also, it is preferable to represent a certain MAP process with the minimal number of phases ([64] introduces MAP order determination) because redundant phases escalate the so-called state explosion problem. These representation problems are investigated in [9].

However, canonical matrix forms are known only for MAP(2); they are given in [30, 65].

Alternatively, MAP can be represented by moments and the autocorrelation structure. For practical purposes, it is necessary to be able to determine whether given moments define a Markovian process [9] and to construct a corresponding matrix form representation. [66] presents moment representation of MEPs (matrix exponential processes), and [67] investigates non-Markovian representation, i.e., a MEP transformation to a Markovian one, i.e., MAP. MAP characterization by their moments is presented in [68]. The usage of KPC (Kronecker product composition) for constructing MAPs with predefined moments and correlation is presented in [69]. This approach relies on the explicit construction of MAP(2) given three marginal moments and correlation; this is discussed in [70, 71, 30, 72]. The construction of the MAP matrix form representation based on moments is one of the possible ways to perform MAP fitting.

Explicit results for MAP matching methods exist only for the second-order case

[30]. For larger models, a combined two-step procedure for the MAP fitting problem has been developed, e.g., in [73]. In the first step, a PH distribution is created to fit the marginal distribution, and, in the second step, a lag-k autocorrelation fitting is performed. Another two-step matching approach is proposed in [74], where an acyclic PH distribution is extended into a MAP by adding correlations based on the lag-1 joint moments of the inter-arrival times. The first (PH fitting) step can be improved by representation transformation (see [75, 76]) to make the PH representation more appropriate for correlation fitting. [77] presents a procedure for building MAP from PH. However, the common flaw of these two-step algorithms is that they are not able to take long-range correlations into account and that the underlying optimization problems are not easy to solve even with the most recent optimization software.

A comparison of many MAP fitting procedures in survey [78] concluded that EM-based algorithms are more beneficial than the matching ones, including the combined methods, for many reasons. The main reason for this is that the EM method considers all the information carried by the samples, while the matching methods consider only the statistical parameters which are matched.

However, the full potential of the EM method is not used. One of the reasons for that is the unsolved representation problem. Due to redundant representation, the solution can be easily attracted to a local maxima, which increases the overall fitting result dependence on the initial guess. To overcome the problem for a subclass MMPP (Markov modulated Poison process), a DAEM algorithm is introduced in [79], which effectively avoids being trapped by the local maxima. Another issue is related to non-uniqueness of representation [9] which causes the solution to go back and forth between MAP representations that are almost identical with respect to the underlying MAP but have very different matrix representations.

In order to partially overcome inefficient fitting due to representation problems and EM computational complexity, various MAP subclasses are used. Since a canonical matrix form is not known (for $n > 2$), a certain sub-class may not cover the MAP space or might be redundant. However, it has been observed that a structurally restricted subclass is more able to overcome local maxima problems, and a simpler, thus faster, EM algorithm can be derived for it. An EM procedure to perform MAP fitting of a general structure is presented in [63]. That procedure requires a massive computational effort, and, consequently, it is applicable only to fit small data traces consisting of a few thousand observations. To address the issue, an EM procedure based on the aggregation of the inter-arrival times is presented in [80]. Similarly, [81] extends the EM algorithm for fitting MAPs to group data. To reduce the computational demand of EM-based MAP fitting algorithms, a special MAP structure is introduced in [31] called Erlang distributed – continuous time hidden Markov chain (ER-CHMM). A generalization of the ER-CHMM structure is presented in [32], which has relaxed some structural restriction of the ER-CHMM structure but increased the number of the model parameters. While using the generalized ER-CHMM structure, slightly higher likelihood values can occasionally be achieved at the cost of increased numerical com-

plexity.

A transient Markov arrival process (TMAP) is a terminating stochastic process [82]. The TMAP stochastic process generates a finite number of possibly correlated inter-arrivals. It can be compared to a PH distrubution, which is also a terminating process but generates only one event. TMAP is related to the MAP process in a sense that it models the inter-arrival correlation as well. Basic TMAP properties are given in [83], and moment characterization is presented in [84].

In this context, our work contributes to the field in several ways.

The general tendency is towards the investigation of PH subclasses, and it is reasonable given their performance advantages. However, in order to fully solve the fitting problem, we believe the general PH structures have to be investigated, as well. We rely more on the combinatorics approach rather than on the analytical one. However, our insights and developed tools might facilitate further advancement in the field.

As for MAPs, we have not found documented results on the parallelization of MAP maximum likelihood parameter estimation algorithms. The reason behind this research gap is unclear, especially considering the fact that huge data sets have to be fitted to accurately capture autocorrelation. We have considered several parallelization options and gave recommendations when a particular algorithm could be used.

There is no presented EM algorithm for TMAP fitting. To address that, we have developed one.

To the best of our knowledge, we are the first to develop MAP, TMAP fitting parallel procedures for parallel execution on GPU devices.

2. PHASE-TYPE DISTRIBUTION FITTING

Is is a known that EM converges to a local solution which depends on the initial PH representation parameter set (i.e., solution). It is very likely that, for different representation structures, the set of local solutions which can be reached by the EM method starting from them is different. This can be proven very easily, by taking two different sparse structures. The structures represent different subspaces of PH distributions. Therefore, the EM method might converge to different local solutions. Thus by starting from the initial solutions of various structures, the chances to find a better solution might increase.

The general structure of PH distribution representation is over-parameterized. It can be shown that $p = 2n - 1$ parameters are sufficient to describe any PH distribution (Theorem 1.1). However, canonical forms of Phase-type distribution representations are known only for orders $n = 2, 3$ [17]. Analytical analysis of the representation structures, even for $n = 4$, is quite complex [85], partly due to the fact that explicit expression for eigenvalues is unknown. It was observed that the EM method can be more robust when fitting with a restricted class of PH distributions. For example, an EM method procedure for Hyper-Exponential distribution fitting is given in [55], which is extended for Hyper-Erlang distribution fitting in [13]. However, one specific structure may not cover the whole class of PH distributions [85], even though its usage is more practical.

In this section, we present Phase-type structure generation algorithms and empirically validate a hypothesis (for case $n = 4$) about the sufficient number of transitions in the matrix form representation. Also, we will research Phase-type fitting with randomly generated structures with $m = 2n$ transitions.

2.1. Phase-type structure generation algorithm

We introduce an algorithm to generate the structures of matrix form PH(n) representations.

Our generation of Phase-type distribution matrix form representation structures is based on the canonical forms of PH distributions with two [86] and three [17] states, non-stationary Markov arrival processes (NMAP) with two states [65] and other observations.

We use the PH distribution matrix form representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ structure notation $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}})$ which is explained in Section 1.2.

We present Algorithm 7 to generate a set of PH(n) representation structures with the specified number of transitions m.

Algorithm 7 Algorithm for generating a set of Phase-type distribution matrix form representation structures with m transitions.

1: **procedure** Structure-generation (n, m)
2: (step 1) Generation of all valid structures with m transitions.
3: (step 2) Removal of redundant trivially equivalent structures.
4: (step 3) * Removal of constrained/not-constrained structures.
5: (step 4) * Removal of cyclic/acyclic structures.
6: (step 5) ** Inclusion of Coxian structure in case of $m = 2n$.
7: (step 6) Removal of structures which do not have obviously constrained and non-canonical lower order PH sub-structures.
8: (step 7) Removal of structures which do not have obviously constrained and non-canonical NMAP(2) sub-structures.
9: **return** the resulting set of structures.
10: **end procedure**

Definition 2.1 *The structure of PH distribution (or NMAP process) representation is said to be obviously constrained if the number of transitions m is equal or less than the necessary number of independent parameters to fully specify the distribution/process.*

A structure containing a lower order PH or NMAP substructure that is not obviously constrained and is of non-canonical form is not preferred and therefore is not considered, as such a structure duplicates the other similar structure which has a canonical form sub-structure.

Generation of all valid structures (step 1)

Basically, from a set of all possible structures, the ones with m transitions are taken for further investigation. Let $\tilde{\boldsymbol{v}}$ be a set of all possible binary vectors of size n

$$\tilde{\boldsymbol{v}} = \left\{ [\dot{\boldsymbol{v}}_1 \ \dot{\boldsymbol{v}}_2 \ \dots \ \dot{\boldsymbol{v}}_n] \middle| \dot{\boldsymbol{v}}_i \in \{0,1\}, i = \overline{1,n} \right\}.$$

Similarly, let $\tilde{\boldsymbol{A}}$ be a set of all possible binary matrices of size n

$$\tilde{\boldsymbol{A}} = \left\{ \begin{bmatrix} \dot{\boldsymbol{A}}_{1,1} & \dots & \dot{\boldsymbol{A}}_{1,n} \\ \vdots & \ddots & \vdots \\ \dot{\boldsymbol{A}}_{n,1} & \dots & \dot{\boldsymbol{A}}_{n,n} \end{bmatrix} \middle| \begin{array}{c} \dot{\boldsymbol{A}}_{i,j} \in \{0,1\}, \dot{\boldsymbol{A}}_{i,i} = 0 \\ \text{for } i \neq j, \ i,j = \overline{1,n} \end{array} \right\}.$$

Then the initial set of all structures with m transitions is

$$\left\{ (\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}}) \middle| \begin{array}{r} \boldsymbol{\alpha} \in \tilde{\boldsymbol{v}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}} \in \tilde{\boldsymbol{A}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}} \in \tilde{\boldsymbol{v}}, \\ \sum_{i=1}^{n} \dot{\boldsymbol{\alpha}}_i - 1 + \sum_{i=1}^{n} \sum_{j=1}^{n} \dot{\boldsymbol{D}}_{\boldsymbol{0}i,j} + \sum_{i=1}^{n} \dot{\boldsymbol{d}}_{\boldsymbol{1}i} = m \end{array} \right\}$$

Definition 2.2 [10] *A structure is considered to be valid if the following conditions are met: a) there exists a path to visit every transient state, b) there exists a path to reach an absorbing state from every transient state.*

This set contains many structures which do not define a valid CTMC. All such invalid structures are removed from it. The resulting set still contains many equivalent structures which are investigated further.

Trivial structure equivalence (step 2)

We are interested in a set of PH distributions that contain all the distributions which have at least one representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ with the structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})$. Such a set is denoted as $\mathrm{PH}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})}$.

Two structures $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$ and $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ are equivalent if

$\mathrm{PH}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}} \subseteq \mathrm{PH}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}}$ and $\mathrm{PH}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}} \subseteq \mathrm{PH}_{(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}}$.

The structure with reordered states stands for the same set of PH distributions. This comes from the fact that renumbering the states of a certain representation results in another representation which represents the same PH distribution. Formally, given a list of unique new states indices $(l_1, l_2, \ldots, l_n)$, the equivalent structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ is obtained from $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$ by state reordering

$$\begin{aligned} \dot{\boldsymbol{\alpha}}_i^{(b)} &= \dot{\boldsymbol{\alpha}}_{l_i}^{(a)}, \ \dot{\boldsymbol{D}}_{\mathbf{0}}{}_{i,j}^{(b)} = \dot{\boldsymbol{D}}_{\mathbf{0}}{}_{l_i,l_j}^{(a)}, \\ \dot{\boldsymbol{d}}_{\mathbf{1}}{}_i^{(b)} &= \dot{\boldsymbol{d}}_{\mathbf{1}}{}_{l_i}^{(a)}, \ \ i,j = \overline{1,n}. \end{aligned}$$

Similarly, the time-reversed structure has the same set of PH distributions as the time-reversed representation represents the same PH distribution. Formally, given structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$, the equivalent time-reversed structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ is obtained by

$$\begin{aligned} \dot{\boldsymbol{\alpha}}_i^{(b)} &= \dot{\boldsymbol{d}}_{\mathbf{1}}{}_i^{(a)}, \ \dot{\boldsymbol{D}}_{\mathbf{0}}{}_{i,j}^{(b)} = \dot{\boldsymbol{D}}_{\mathbf{0}}{}_{j,i}^{(a)}, \\ \dot{\boldsymbol{d}}_{\mathbf{1}}{}_i^{(b)} &= \dot{\boldsymbol{\alpha}}_i^{(a)}, \ \ i,j = \overline{1,n}. \end{aligned}$$

Definition 2.3 [9] *A trivial transformation of a structure or a representation is such a transformation which involves the state reordering and/or time reversal.*

In addition, if a structure or a representation was obtained by the trivial transformation, those are said to be trivially equivalent.

Constrained structures (step 3)

Phase-type distribution can be represented by $2n-1$ moments if moment r_{-1} in [66] is fixed to zero and Phase-type distribution is represented with $2n-2$ parameters. Thus the following proposition for a general structure can be formulated.

Proposition 2.1 *If the representation $(\boldsymbol{\alpha}, \boldsymbol{D_0})$ of the structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})$ is such that there is no immediate exit from the states for which the initial probability is non-*

zero, i.e., for all $i = \overline{1,n} : \boldsymbol{\alpha}_i \boldsymbol{d}_{\mathbf{1}i} = 0$*, the represented PH distribution is defined by* $2n-2$ *parameters at most.*

Proof. From (5) we have that

$$det\left(s\boldsymbol{I}-\boldsymbol{D}_{\mathbf{0}}\right) f^{*}(s) = \boldsymbol{\alpha}\left(\boldsymbol{\Gamma}(s)\right)^{\intercal} \boldsymbol{d}_{\mathbf{1}} =$$

$$\begin{bmatrix}\boldsymbol{\alpha}_1 & \dots & \boldsymbol{\alpha}_n\end{bmatrix} \begin{bmatrix}\boldsymbol{\Gamma}_{1,1}(s) & \dots & \boldsymbol{\Gamma}_{n,1}(s) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Gamma}_{1,n}(s) & \dots & \boldsymbol{\Gamma}_{n,n}(s)\end{bmatrix} \begin{bmatrix}\boldsymbol{d}_{\mathbf{1}1} \\ \vdots \\ \boldsymbol{d}_{\mathbf{1}n}\end{bmatrix} =$$

$$\alpha_1\left(\boldsymbol{\Gamma}_{1,1}(s)\,\boldsymbol{d}_{\mathbf{1}1} + \dots + \boldsymbol{\Gamma}_{n,1}(s)\,\boldsymbol{d}_{\mathbf{1}n}\right) + \dots +$$

$$\boldsymbol{\alpha}_n\left(\boldsymbol{\Gamma}_{1,n}(s)\,\boldsymbol{d}_{\mathbf{1}1} + \dots + \boldsymbol{\Gamma}_{n,n}(s)\,\boldsymbol{d}_{\mathbf{1}n}\right).$$

From the condition of Proposition 1.1, it follows that adjuncts $\boldsymbol{\Gamma}_{i,i}(s)$ are multiplied by zero. Since these are the only adjuncts that are polynomials in s of $n-1$ degree, the numerator in (5) does not contain term s^{n-1}, i.e., coefficient c_{n-1} is equal to zero. Consequently, the Laplace transform (and the corresponding PH) is defined by $2n-2$ parameters at most. □

The constrained structures have density function property $f(0) = 0$ and in general are not preferred since they are less flexible.

Structures with acyclic generator and Coxian structure (steps 4 and 5)

Definition 2.4 [2] *Structure* $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$ *represents the acyclic PH distribution if it can be trivially transformed into the structure* $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ *with an upper triangular matrix* $\dot{\boldsymbol{D}}_{\mathbf{0}}{}^{(b)}$.

Any PH distribution representation with a triangular generator matrix can be transformed into an ordered Coxian representation structure. The ordered Coxian representation structure has the following form

$$\dot{\boldsymbol{\alpha}} = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}, \dot{\boldsymbol{d}}_{\mathbf{1}} = \begin{bmatrix}1 \\ 1 \\ \vdots \\ 1\end{bmatrix},$$

$$\dot{\boldsymbol{D}}_{\mathbf{0}} = \begin{bmatrix}0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0\end{bmatrix}.$$

The sub-structures (sub-chains)

Let us recall that PH distribution is defined via a CTMC. We choose a few transient states for analysis. Then we partition the whole original CTMC into two chains. The first one contains the chosen transient states and is called a sub-chain. The second one contains all the remaining states and is called a complimentary chain.

A particular sub-chain with r states is given by a list of unique state indices $(l_1, \ldots, l_r)$. There are the remaining $q = n + 2 - r$ states in the complimentary chain, where $+2$ stands for the starting and for the absorbing states. For the given sub-chain state index list $(l_1, \ldots, l_r)$, the generator matrix of CTMC can be reordered to the form

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{C}^* & \boldsymbol{E} \\ \boldsymbol{X} & \boldsymbol{C} \end{bmatrix}, \tag{41}$$

where the lower right block $\boldsymbol{C}$ is a transient generator matrix of the sub-chain. The input $\boldsymbol{E}$ and output $\boldsymbol{X}$ matrices are the upper right and lower left blocks, respectively. The complimentary chain transient generator $\tilde{\boldsymbol{C}}$ matrix is not investigated further. The structure of the $\boldsymbol{C}$, $\boldsymbol{E}$ and $\boldsymbol{X}$ matrices is encoded by the binary element matrices $\dot{\boldsymbol{C}}$, $\dot{\boldsymbol{E}}$ and $\dot{\boldsymbol{X}}$

$$\begin{gathered} \dot{\boldsymbol{C}} = \begin{bmatrix} \dot{\boldsymbol{C}}_{1,1} & \ldots & \dot{\boldsymbol{C}}_{1,r} \\ \vdots & \ddots & \vdots \\ \dot{\boldsymbol{C}}_{r,1} & \ldots & \dot{\boldsymbol{C}}_{r,r} \end{bmatrix}, \dot{\boldsymbol{E}} = \begin{bmatrix} \dot{\boldsymbol{E}}_{1,1} & \ldots & \dot{\boldsymbol{E}}_{1,r} \\ \vdots & \ddots & \vdots \\ \dot{\boldsymbol{E}}_{q,1} & \ldots & \dot{\boldsymbol{E}}_{q,r} \end{bmatrix}, \\ \dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{X}}_{1,1} & \ldots & \dot{\boldsymbol{X}}_{1,q} \\ \vdots & \ddots & \vdots \\ \dot{\boldsymbol{X}}_{q,1} & \ldots & \dot{\boldsymbol{X}}_{q,q} \end{bmatrix}, \end{gathered} \tag{42}$$

where $\dot{\boldsymbol{C}}_{i,j} \in \{0, 1\}$ for $i \neq j$, $i, j = \overline{1, r}$ indicates the transitions between sub-chain states; $\dot{\boldsymbol{E}}_{i,j} \in \{0, 1\}$ for $i = \overline{1, q}$, $j = \overline{1, r}$ indicates the transitions from a complimentary chain to a sub-chain; $\dot{\boldsymbol{X}}_{i,j} \in \{0, 1\}$ for $i \neq j$, $i, j = \overline{1, q}$ indicates the transitions from a sub-chain to a complimentary chain.

Definition 2.5 *A Phase-type sub-structure is a sub-chain structure of CTMC and is specified by* $\left(\dot{\boldsymbol{E}}, \dot{\boldsymbol{C}}, \dot{\boldsymbol{X}}\right)$.

Alternatively, a sub-structure of Phase-type structure $\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\boldsymbol{0}}, \dot{\boldsymbol{d}}_{\boldsymbol{1}}\right)$ can be specified by the list of states.

We define the following characteristics based on matrices $\dot{\boldsymbol{E}}$ and $\dot{\boldsymbol{X}}$:

- E_c / E_r is the number of states to/from which a sub-chain can be entered,
- X_c / X_r is the number of states to/from which a sub-chain can be left,

- E_{rank} / X_{rank} is the maximal possible rank of input/output matrices found by (43)

$$\begin{aligned} E_{rank} &= \min\left(E_c, E_r\right), \\ X_{rank} &= \min\left(X_c, X_r\right). \end{aligned} \tag{43}$$

A sub-structure is characterized by the number of transitions

$$m^{(c)} = E_c + \sum_{i=1}^{r}\sum_{j=1}^{r} \dot{\boldsymbol{C}}_{i,j} + X_c.$$

Example 2.1 Let us consider a Phase-type distribution representation structure given in Figure 2.1.

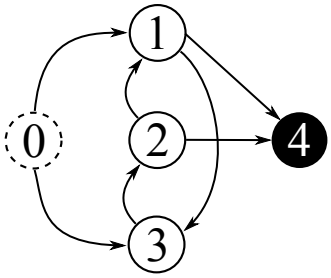

Fig. 2.1. An example of a Phase-type distribution structure.

Let us say we are interested in the sub-structure which consists of $(2, 3)$ transient states. First of all, we reorder the states so that the CTMC generator matrix structure becomes

$$\dot{\boldsymbol{Q}} = \begin{bmatrix} \dot{\boldsymbol{C}}^{*} & \dot{\boldsymbol{E}} \\ \dot{\boldsymbol{X}} & \dot{\boldsymbol{C}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

where

$$\dot{\boldsymbol{C}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \dot{\boldsymbol{E}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \dot{\boldsymbol{X}} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

From matrices $\dot{\boldsymbol{E}}$, $\dot{\boldsymbol{X}}$, the following characteristics are found

$$\begin{aligned} E_r = 2,\ E_c = 1,\ E_{rank} = 1, \\ X_r = 1,\ X_c = 2,\ X_{rank} = 1. \end{aligned}$$

The sub-structure schema is shown in Figure 2.2.

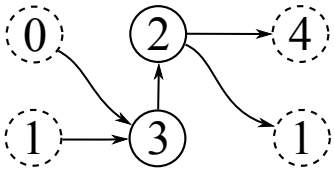

Fig. 2.2. A schema of the sub-structure of the Phase-type distribution given in Example 2.1.

△

The given structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})$ is investigated by analyzing the structure of all of its sub-substructures obtained from the corresponding CTMC (or time-reversed CTMC) structure. We shall note that the order of states in the sub-chain is not important because the state renumbering in CTMC does not affect the PH distribution that it represents.

Identification of non-canonical PH(2) and PH(3) sub-structures (step 6)

The sub-chain of a two-state sub-structure, that is characterized by $E_r = 1$ and $X_r = 1$ can be considered as an order $n = 2$ Phase-type distribution.

If $m^{(c)} = n+1 = 3$, then the sub-chain is a constrained PH(2), and no reasonable conclusion can be drawn about the original PH structure.

It is known that any PH(2) distribution has a canonical form of the ordered Coxian distribution [45]. Therefore, if $m^{(c)} = 2n = 3$ and the sub-structure is not trivially equivalent to the Coxian structure, the original PH structure is not preferred. If $m^{(c)} > 2n = 4$, the sub-structure represents a redundant PH(2) distribution, and the original PH structure is not preferred, either.

If a three-state sub-chain is such that $E_r = 1$ and $X_r = 1$, it can be considered as an order $n = 3$ Phase-type distribution. If the sub-structure is not obviously constrained, it is necessary to check if it has the canonical form given in [17].

If $m^{(c)} < 2n = 6$, then the sub-structure is obviously constrained PH(3), and no reasonable conclusion can be drawn about the original PH structure.

In the case of $m^{(c)} = 2n = 6$, the original PH structure is not preferred if it

cannot be trivially transformed into one of the canonical form structures (44):

$$\begin{aligned}\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}\right)^{(1)} &= \left(\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}, \begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\right),\\ \left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}\right)^{(2)} &= \left(\begin{bmatrix}1 & 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}, \begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\right).\end{aligned} \tag{44}$$

Similarly, if $m^{(c)} = 2n + \Delta M_3 = 7$ and the sub-structure is not trivially equivalent to the canonical form structure (45), the original PH structure is not preferred:

$$\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}}\right)^{(3)} = \left(\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}, \begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\right). \tag{45}$$

Finally, in the case of $m^{(c)} > 2n + \Delta M_3 = 7$, the whole original structure is not preferred since such a PH(3) distribution is redundant.

Identification of non-canonical NMAP(2) sub-structures (step 7)

A two-state sub-chain can be considered as a NMAP(2) process if it can be entered from one complimentary chain state ($E_r = 1$) and exited to two complimentary chain states ($X_r = 2$).

It is necessary to check if the NMAP(2) structure is not constrained; in order to do that, the statistic $z^{(c)}$ is defined as

$$z^{(c)} = z_E + z_X + z_C, \tag{46}$$

where z_E / x_X is the number of zeros in the matrix $\dot{\boldsymbol{E}}$ / $\dot{\boldsymbol{X}}$ excluding rows/columns with only zero elements, and z_C is the number of zeros in matrix $\dot{\boldsymbol{C}}$ excluding the diagonal elements.

If $z^{(c)} > 2$, such a sub-chain is obviously a constrained NMAP(2) process, and the original PH structure has to be left as possibly contributing.

Otherwise, if $z^{(c)} \leq 2$ and the sub-structure cannot be trivially transformed into one of the two NMAP(2) canonical form (20), (21) structures

$$\begin{aligned}\left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{D}}_{\mathbf{1}}, \right)^{(1)} &= \left(\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}, \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}\right),\\ \left(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{D}}_{\mathbf{1}}, \right)^{(2)} &= \left(\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}, \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\right),\end{aligned}$$

then the original PH structure is not preferred.

2.2. Phase-type random structure generation algorithm

We introduce a much simpler random structure generation algorithm. The subclass of randomly generated structures is specified by the following properties:

- cyclic/acyclic – specifies whether the Phase-type infinitesimal generator structure is cyclic or not;
- ZID (zero initial density) – specifies that the structure is constrained (i.e., it is such that $f(0) = 0$);
- number of transitions, m.

Let us denote the cyclic/acyclic and ZID properties by $\mathbf{\Theta}$. A random structure is generated by constructing the initial structure with the maximum number of transitions that has the properties $\mathbf{\Theta}$, and then randomly chosen transitions are removed until (see Algorithm 8) the number of transitions is m.

Algorithm 8 Removal of a random transition from structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$

procedure Transition-Random-Removal ($(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$, $\mathbf{\Theta}$)
  $\mathcal{T} := \emptyset$ ▷ a set of transitions which can be removed
  for each transition t in $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$ **do**
    obtain structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ by removing t from $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$
    if $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ is valid and has properties $\mathbf{\Theta}$ **then**
      insert t into $\mathcal{T}$
    end if
  end for each
  randomly choose transition t from $\mathcal{T}$
  obtain structure $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$ by removing t from $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(a)}$
  return $(\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{D}}_{\mathbf{0}}, \dot{\boldsymbol{d}}_{\mathbf{1}})^{(b)}$
end procedure

2.3. Distribution discretization algorithm

For a given distribution density function $f(x)$, we need to obtain the weighted observations (x_i, w_i). This is achieved in two steps. First, we discretize the $f(x)$ function for a given interval from x_{start} to x_{end}. After discretization, we have intervals $[z_i, z_{i+1})$ which do not overlap and cover the interval $[x_{start}, x_{end})$. Then the observations (i.e., the trace data) are compiled by Formula (47)

$$\begin{aligned} x_i &= \frac{z_i + z_{i+1}}{2}, \\ w_i &= \int_{z_i}^{z_{i+1}} f(x)\mathrm{d}x \approx \sum_{k=0}^{K-1} \Delta z_i f(z_i + k\Delta z_i), \end{aligned} \tag{47}$$

where $\Delta z_i = \dfrac{z_{i+1} - z_i}{K}$, and K is the number of sampling points.

In order to reduce the number of intervals $[z_i, z_{i+1})$, we discretize the interval $[x_{start}, x_{end})$ with an adaptive algorithm.

The list of discretization points is denoted by $\boldsymbol{\mathcal{D}} = (z_1, \ldots, z_k, \ldots)$; the list of interval errors is $\boldsymbol{\mathcal{E}} = (e_1, \ldots, e_k, \ldots)$.

The error of k^{th} interval is

$$e_i = \left| f\left(\frac{z_i + z_{i+1}}{2}\right) - \frac{f(z_i) + f(z_{i+1})}{2} \right| (z_{i+1} - z_i), \tag{48}$$

which is an absolute difference between the density function value in the middle of interval $[z_i, z_{i+1})$ and the value of the discretized function value multiplied by the length of the interval.

Initially, we set $\boldsymbol{\mathcal{D}} := (x_{start}, x_{end})$ and keep inserting new discretization points (see Algorithm 9) until the total discretization error $\sum_i e_i$ is not bigger than a certain criterion. Also, we specify the minimum number of discretization points to insert at the beginning. The specific values of these parameters are not important, we choose them by hand until the obtained discretization is precise enough based on its statistical properties (mean, standard deviation, skewness).

Algorithm 9 Pseudo-code for inserting a point into discretization $\boldsymbol{\mathcal{D}}$

1: **procedure** Insert-Point ($\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{E}}$)
2: find the interval with the maximum error $k = arg\max_i(e_i)$
3: compute new discretization point $z^* = \frac{z_i + z_{i+1}}{2}$
4: insert z^* into $\boldsymbol{\mathcal{D}}$, so that $\boldsymbol{\mathcal{D}} = (\ldots, z_k, z^*, z_{k+1}, \ldots)$
5: compute new errors e_1^*, e_2^* for intervals $[z_k, z^*)$, $[z^*, z_{k+1})$ by (48)
6: update error list so that $\boldsymbol{\mathcal{E}} = (\ldots, e_{k-1}, e_1^*, e_2^*, e_{k+1}, \ldots)$
7: **return** $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{E}}$
8: **end procedure**

2.4. The benchmark distributions for Phase-type fitting

For comparing Phase-type fitting, we use the benchmark distributions presented in [58].

Table 2.1. The benchmark distributions and their density functions.

Distribution	Density function
Weibull	$f(x;\lambda,k)=\begin{cases}\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-\left(\frac{x}{\lambda}\right)^{k}}, \text{ if } 0\le x,\\ 0, \text{ otherwise.}\end{cases}$
Log-normal	$f(x;\mu,\sigma)=\begin{cases}\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}, \text{ if } 0\le x,\\ 0, \text{ otherwise.}\end{cases}$
Uniform	$f(x;a,b)=\begin{cases}\frac{1}{b-a}, \text{ if } a\le x\le b,\\ 0, \text{ otherwise.}\end{cases}$
Shifted exponential	$f(x)=\begin{cases}\frac{1}{2}e^{-x}, \text{ if } 0<x\le 1,\\ \frac{1}{2}\left(1+e\right)e^{-x}, \text{ if } 1<x,\\ 0, \text{ otherwise.}\end{cases}$
Matrix exponential	$f(x)=\begin{cases}\left(1+\frac{1}{4\pi^2}\right)\left(1-\cos\left(2\pi x\right)\right)e^{-x},\\ \qquad\qquad\qquad \text{if } 0\le x,\\ 0, \text{ otherwise.}\end{cases}$

Table 2.2. The parameters of benchmark distributions.

Distribution	Code	Parameters
Weibull	**W1**	$\lambda=1, k=1.5$
	W2	$\lambda=1, k=0.5$
Log-normal	**L1**	$\mu=-1.8, \sigma=1.8$
	L2	$\mu=-0.32, \sigma=0.8$
	L3	$\mu=-0.02, \sigma=0.2$
Uniform	**U1**	$a=0, b=1$
	U2	$a=1, b=2$
Shifted exponential	**SE**	
Matrix exponential	**ME**	

The statistical properties of benchmark distributions are given in Table 2.3.

Table 2.3. Statistical properties of benchmark distributions.

Dist.	mean	std.	skewness
W1	$9.0275 \cdot 10^{-1}$	$6.1294 \cdot 10^{-1}$	1.0720
W2	2.0000	4.4721	6.6188
L1	$8.3527 \cdot 10^{-1}$	4.1372	$1.3638 \cdot 10^{2}$
L2	1.0000	$9.4683 \cdot 10^{-1}$	3.6893
L3	1.0000	$2.0202 \cdot 10^{-1}$	$6.1429 \cdot 10^{-1}$
U1	$5.0000 \cdot 10^{-1}$	$2.8868 \cdot 10^{-1}$	0.0000
U2	1.5000	$2.8868 \cdot 10^{-1}$	0.0000
SE	1.5000	1.1180	1.4311
ME	1.0494	$9.7623 \cdot 10^{-1}$	2.1420

We have discretized the distributions (given in Tables 2.1, 2.2) by an adaptive algorithm (see Section 2.3). The statistical properties of discretized distributions are given in Table 2.5. The disretization parameters (Table 2.4) have been chosen by hand so that at least two digits of the mean and standard deviation of distributions and their discretizations would match (except for **L1**). A more detailed comparison of discretization precision is given in Table 2.6.

Table 2.4. Discretization parameters of benchmark distributions.

Dist.	range start	range end	num. of observations
W1	$1.2873 \cdot 10^{-3}$	4.2381	69
W2	$8.5974 \cdot 10^{-10}$	$1.1150 \cdot 10^{2}$	144
L1	$2.2782 \cdot 10^{-4}$	$4.1668 \cdot 10^{1}$	288
L2	$3.0571 \cdot 10^{-2}$	$1.8497 \cdot 10^{1}$	97
L3	$4.8017 \cdot 10^{-1}$	2.0047	76
U1	$2.9691 \cdot 10^{-3}$	$9.9703 \cdot 10^{-1}$	12
U2	1.0030	1.9970	12
SE	$3.8554 \cdot 10^{-2}$	9.6552	78
ME	$2.6314 \cdot 10^{-2}$	8.5999	78

Table 2.5. Statistical properties of discretized benchmark distributions.

Dist.	mean	std.	skewness
W1	$9.0237 \cdot 10^{-1}$	$6.1150 \cdot 10^{-1}$	1.0537
W2	1.9985	4.4415	6.2417
L1	$7.5376 \cdot 10^{-1}$	2.2086	8.2388
L2	$9.9975 \cdot 10^{-1}$	$9.4293 \cdot 10^{-1}$	3.4937
L3	$9.9994 \cdot 10^{-1}$	$2.0155 \cdot 10^{-1}$	$5.9995 \cdot 10^{-1}$
U1	$5.0000 \cdot 10^{-1}$	$2.8315 \cdot 10^{-1}$	$9.2254 \cdot 10^{-8}$
U2	1.5000	$2.8315 \cdot 10^{-1}$	$-3.8207 \cdot 10^{-17}$
SE	1.4992	1.1143	1.3897
ME	1.0483	$9.7086 \cdot 10^{-1}$	2.0730

Table 2.6. Comparison of discretized benchmark statistical properties. The absolute relative difference is given in accordance to the benchmark distribution statistics.

Dist.	mean	std.	skewness
W1	$4.2089 \cdot 10^{-4}$	$2.3466 \cdot 10^{-3}$	$1.7067 \cdot 10^{-2}$
W2	$7.6788 \cdot 10^{-4}$	$6.8608 \cdot 10^{-3}$	$5.6973 \cdot 10^{-2}$
L1	$9.7583 \cdot 10^{-2}$	$4.6616 \cdot 10^{-1}$	$9.3959 \cdot 10^{-1}$
L2	$2.4544 \cdot 10^{-4}$	$4.1182 \cdot 10^{-3}$	$5.3006 \cdot 10^{-2}$
L3	$6.1045 \cdot 10^{-5}$	$2.3148 \cdot 10^{-3}$	$2.3354 \cdot 10^{-2}$
U1	$1.3242 \cdot 10^{-8}$	$1.9134 \cdot 10^{-2}$	0.0000
U2	$1.2418 \cdot 10^{-9}$	$1.9134 \cdot 10^{-2}$	0.0000
SE	$5.3461 \cdot 10^{-4}$	$3.3016 \cdot 10^{-3}$	$2.8899 \cdot 10^{-2}$
ME	$1.0585 \cdot 10^{-3}$	$5.5062 \cdot 10^{-3}$	$3.2211 \cdot 10^{-2}$

2.5. Phase-type fitting research

It is known that the matrix form representation of Phase-type distribution is redundant [15].

Based on the fact that the Phase-type distribution of order $n = 3$ has a canonical form [17] which has the maximal number of additional transitions $\Delta M_3 = 1 > 0$, it can be expected that $\Delta M_n \geq 0$ when $n > 3$. If the analytical dependence of ΔM_n on order n were known, it could be used as an upper bound for the number of transitions in the generated structures. However, to be on the safe side, we do not assume about ΔM_n and, thus, investigate structures with up to $m = n^2 + n$ transitions.

We expect that a set of Phase-type distributions which can be represented by the matrix form representations of structures with additional transitions (i.e., $\Delta m_n > 0$) is not dense (i.e., of zero measure). Therefore, we formulate the following hypothesis.

Hypothesis 2.1 *For every Phase-type distribution which has a matrix form representation with $m > 2n$ transitions, it is possible to find another of its representations with $m = 2n$ transitions or to find a sequence of matrix form representations with $m = 2n$ transitions whose Phase-type distributions converge to the given one.*

We validate Hypothesis 2.1 empirically for the case $n = 4$. The validation schema is given in Figure 2.3.

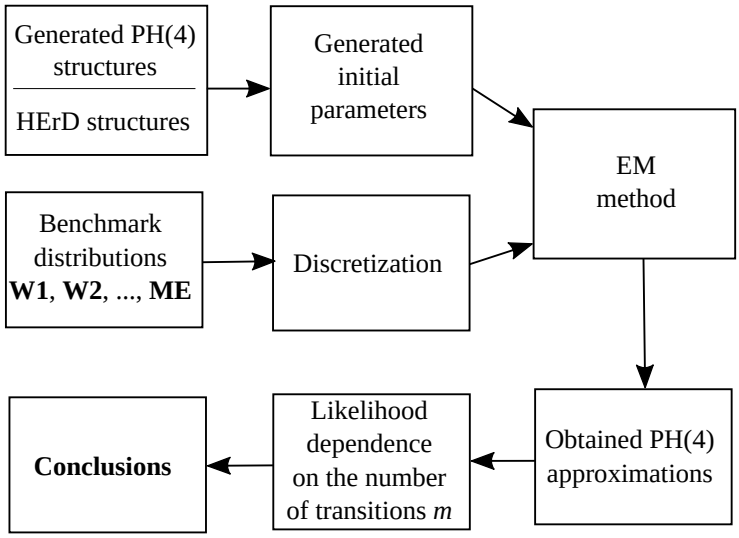

Fig. 2.3. Schema of Hypothesis 2.1 empirical validation.

In case Hypothesis 2.1 has validity, we would expect to find out that the fitting results using structures with $m = 2n$ transitions are not worse than the ones obtained with $m > 2n$ transitions. Of course, given the nature of empirical computations (i.e., limited number of bits for the floating point number representation, etc.), we do not expect exact accuracy. Instead, we look for the general tendency. We will look for the cases which violate Hypothesis 2.1, as well. It is of interest how strong these cases might be, if any.

PH(4) fitting with all generated structures

First of all, we have generated all the PH(4) structures by using our algorithm, as well as the HErD structures with 4 states. The number of the generated structures is summarized in Table 2.7.

Table 2.7. The number of generated PH(4) structures for cyclic and acyclic subclasses.

Tran. count, m	number of structures	Tran. count, m	number of structures
	cyclic		acyclic
6	1	6	2
7	22	7	21
8	118	8	50
9	526	9	93
10	1241	10	109
11	1778	11	63
12	1874	12	26
13	1429	13	7
14	829	14	1
15	362		acyclic-zid
16	132	5	1
17	37	6	8
18	10	7	6
19	2	8	7
20	1	9	10
	cyclic-zid	10	1
6	6		Hyper-Erlang
7	39	5	1
8	178	6	2
9	386	7	1
10	494	8	1
11	443		
12	275		
13	116		
14	35		
15	7		
16	2		

We have performed PH fitting while using the generated strucures (and Coxian structures) for all the benchmark distributions. One initial random parameter guess was generated for each structure, and the EM algorithm was performed for a number of iterations until the relative log-likelihood value change was smaller than 10^{-8}, or the whole computation time exceeded the limit of one minute.

The resulting best log-likehood values from each subclass are presented in the following Tables 2.8, 2.9, 2.10, 2.11.

Table 2.8. Log-likelihood values of **W1**, **W2**, **L1**, **L2**, **L3** distribution fitting results using PH(4) cyclic structures.

Tran. count	**W1**	**W2**	**L1**	**L2**	**L3**
cyclic					
6	−0.8971830	−1.2577138	−0.2475088	−0.9997446	−0.9997466
7	−0.7902727	−1.1631224	−0.2026385	−0.8774560	−0.3062347
8	−0.7859343	−1.1358007	−0.1981078	−0.8770357	−0.3062347
9	−0.7859344	−1.1358237	−0.1981076	−0.8770351	−0.3062347
10	−0.7859344	−1.1358015	−0.1981076	−0.8770349	−0.3062347
11	−0.7859344	−1.1358058	−0.1981076	−0.8770287	−0.3062347
12	−0.7859344	−1.1357553	−0.1981076	−0.8770355	−0.3062347
13	−0.7859355	−1.1358057	−0.1981076	−0.8770365	−0.3062347
14	−0.7859362	−1.1357656	−0.1981076	−0.8770370	−0.3062347
15	−0.7859436	−1.1358563	−0.1981076	−0.8770370	−0.3062347
16	−0.7859473	−1.1358629	−0.1981078	−0.8770378	−0.3062347
17	−0.7859485	−1.1359817	−0.1981078	−0.8770383	−0.3062347
18	−0.7859549	−1.1361987	−0.1981082	−0.8770407	−0.3062347
19	−0.7859539	−1.1363224	−0.1981083	−0.8770382	−0.3062347
20	−0.7859680	−1.1610044	−0.1981084	−0.8770394	−0.3062347
cyclic-zid					
6	−0.7903425	−1.3872123	−0.2484174	−0.8774870	−0.3062347
7	−0.7865590	−1.2837918	−0.2010889	−0.8770366	−0.3062347
8	−0.7865145	−1.2662096	−0.2009460	−0.8770272	−0.3062347
9	−0.7865143	−1.2076030	−0.2008525	−0.8770279	−0.3062347
10	−0.7865143	−1.2136351	−0.2008440	−0.8770348	−0.3062347
11	−0.7865150	−1.2412106	−0.2008045	−0.8770288	−0.3062347
12	−0.7865152	−1.2334210	−0.2008152	−0.8770378	−0.3062347
13	−0.7865157	−1.2550141	−0.2008091	−0.8770378	−0.3062347
14	−0.7865147	−1.2132378	−0.2008160	−0.8770379	−0.3062347
15	−0.7865162	−1.2650951	−0.2009652	−0.8770380	−0.3062347
16	−0.7865178	−1.3226232	−0.2443496	−0.8770437	−0.3062347

Table 2.9. Log-likelihood values of **W1**, **W2**, **L1**, **L2**, **L3** distribution fitting results using PH(4) acyclic structures.

Tran. count	**W1**	**W2**	**L1**	**L2**	**L3**
coxian					
8	-0.7859480	-1.1363519	-0.1981083	-0.8774869	-0.3062347
herd					
5	-0.9892055	-6.4768990	-4.5291807	-1.2050052	-0.3062347
6	-0.7917803	-1.3585821	-0.3052479	-0.8902568	-0.4370118
7	-0.7902732	-1.1768731	-0.2064813	-0.8959456	-0.6334429
8	-0.8971745	-1.1347523	-0.1981082	-0.9997346	-0.9997390
acyclic					
6	-0.7903581	-1.2560894	-0.2466669	-0.8802333	-0.3062347
7	-0.7859344	-1.1631206	-0.2026402	-0.8774862	-0.3062347
8	-0.7859357	-1.1359613	-0.1981078	-0.8774858	-0.3062347
9	-0.7859345	-1.1359405	-0.1981077	-0.8774858	-0.3062347
10	-0.7859377	-1.1359596	-0.1981077	-0.8774859	-0.3062347
11	-0.7859446	-1.1359523	-0.1981076	-0.8774865	-0.3062347
12	-0.7859464	-1.1359454	-0.1981076	-0.8774859	-0.3062347
13	-0.7859506	-1.1360484	-0.1981081	-0.8774866	-0.3062347
14	-0.7859519	-1.1375892	-0.1981082	-0.8774867	-0.3062347
acyclic-zid					
5	-0.7928659	-2.2690005	-0.7650856	-0.8856195	-0.3062347
6	-0.7865645	-1.3875362	-0.2481796	-0.8774858	-0.3062347
7	-0.7865196	-1.2610474	-0.2010922	-0.8774869	-0.3062347
8	-0.7865151	-1.2707656	-0.2011749	-0.8774916	-0.3062347
9	-0.7865211	-1.2778164	-0.2011195	-0.8774868	-0.3062347
10	-0.7865150	-1.2852793	-0.2046147	-0.8774868	-0.3062347

Table 2.10. Log-likelihood values of **U1**, **U2**, **SE**, **ME** distribution fitting results using PH(4) cyclic structures.

Tran. count	**U1**	**U2**	**SE**	**ME**
cyclic				
6	−0.3068655	−1.4054855	−1.4048301	−1.0470473
7	−0.1257613	−0.7073650	−1.3234295	−0.9110613
8	−0.1257613	−0.7073650	−1.3234294	−0.9110613
9	−0.1257613	−0.7073650	−1.3234296	−0.9110613
10	−0.1257614	−0.7073650	−1.3234295	−0.9110613
11	−0.1257614	−0.7073650	−1.3234295	−0.9110613
12	−0.1257614	−0.7073650	−1.3234295	−0.9110613
13	−0.1257614	−0.7073650	−1.3234296	−0.9110613
14	−0.1257615	−0.7073650	−1.3234296	−0.9110613
15	−0.1257618	−0.7073650	−1.3234295	−0.9110613
16	−0.1257618	−0.7073650	−1.3234296	−0.9110613
17	−0.1257618	−0.7073650	−1.3234296	−0.9110613
18	−0.1257618	−0.7073650	−1.3234298	−0.9110621
19	−0.1257632	−0.7073650	−1.3234300	−0.9110621
20	−0.1257624	−0.7073650	−1.3234303	−0.9110621
cyclic-zid				
6	−0.1914623	−0.7073650	−1.3458414	−0.9110613
7	−0.1488286	−0.7073650	−1.3245217	−0.9110613
8	−0.1488286	−0.7073650	−1.3245217	−0.9110613
9	−0.1488286	−0.7073650	−1.3245217	−0.9110613
10	−0.1488235	−0.7073650	−1.3245217	−0.9110613
11	−0.1488283	−0.7073650	−1.3245217	−0.9110613
12	−0.1488262	−0.7073650	−1.3245217	−0.9110615
13	−0.1488286	−0.7073650	−1.3245218	−0.9110617
14	−0.1488287	−0.7073650	−1.3245221	−0.9110619
15	−0.1488366	−0.7073650	−1.3245218	−0.9110621
16	−0.1488801	−0.7073650	−1.3245328	−0.9110621

Table 2.11. Log-likelihood values of **U1**, **U2**, **SE**, **ME** distribution fitting results using PH(4) acyclic structures.

Tran. count	**U1**	**U1**	**SE**	**ME**
coxian				
8	−0.1257633	−0.7073650	−1.3250964	−0.9509505
herd				
5	−0.4285273	−0.7073650	−1.7991042	−1.3507712
6	−0.1448826	−0.8396534	−1.3303970	−0.9519297
7	−0.1928429	−1.0376088	−1.3516318	−0.9787622
8	−0.3068523	−1.4054629	−1.4048118	−1.0470389
acyclic				
6	−0.1257613	−0.7073650	−1.3290435	−0.9360609
7	−0.1257618	−0.7073650	−1.3250949	−0.9184889
8	−0.1257618	−0.7073650	−1.3245220	−0.9184888
9	−0.1257613	−0.7073650	−1.3245218	−0.9184889
10	−0.1257618	−0.7073650	−1.3245220	−0.9184889
11	−0.1257618	−0.7073650	−1.3250784	−0.9184888
12	−0.1257618	−0.7073650	−1.3250797	−0.9184889
13	−0.1257618	−0.7073650	−1.3250804	−0.9184892
14	−0.1257652	−0.7073650	−1.3250805	−0.9184892
acyclic-zid				
5	−0.2124731	−0.7073650	−1.3994257	−0.9360611
6	−0.1488287	−0.7073650	−1.3245217	−0.9184889
7	−0.1488287	−0.7073650	−1.3245213	−0.9360618
8	−0.1488286	−0.7073650	−1.3245217	−0.9436604
9	−0.1488287	−0.7073650	−1.3245217	−0.9184889
10	−0.1488287	−0.7073650	−1.3245330	−0.9184892

Table 2.12. The PH(4) structure classes which obtained the highest log-likelihood. Compiled from Tables 2.8, 2.9, 2.10, 2.11.

Dist.	structure class	log-likelihood
W1	cyclic(8)	-0.7859343
W2	herd(8)	-1.1347523
L1	cyclic(9)	-0.1981076
L2	cyclic-zid(8)	-0.8770272
L3	cyclic(8)	-0.3062347
U1	cyclic(7)	-0.1257613
U2	cyclic(7)	-0.7073650
SE	cyclic(8)	-1.3234294
ME	cyclic(7)	-0.9110613

The first thing we look for when analyzing the fitting results given in Tables 2.8, 2.9, 2.10, 2.11, 2.12 is the biggest contradictions to Hypothesis 2.1. These are the cases when it was not enough to have $m = 8$ (i.e., $m = 2n$) transitions to obtain the best fitting result (among the rest of the cases). We have found only one such case. For **L1** distribution, the higher log-likelihood value of -0.1981076 was obtained by using a structure with $m = 9$ transitions, while fitting with a structure of $m = 8$ transitions gave -0.1981078. The absolute log-likelihood difference is 0.0000002, which we think is negligible. For other distributions, the best log-likelihood values have been obtained while using structures with $m \leq 8$ transitions. Therefore, based on these empirical results, we state that Hypothesis 2.1 is valid. However, we would be interested to see if anybody were able to find a counter example.

Next, it is interesting to look into the classes of structures of the highest log-likelihood values. In all the cases but two, the best structure was cyclic (not constrained, i.e., non-ZID). In the case of **W2**, the best parameters (log-likelihood of -1.1347523) were found by fitting the Hyper-Erlang distribution with $m = 8$ transitions. We can raise a question why fitting the Coxian distribution gave a smaller log-likelihood value of -1.1363519 (absolute difference of 0.0016). One would expect the Coxian distribution to be as good as (or better than) any other acyclic Phase-type distribution [19]. However, when fitting with the EM method, an acyclic (without Coxian) PH can yield better results than the Coxian distribution, as pointed out by [14]. In the case of comparing our results of Coxian, Hyper-Erlang and acyclic-PH fitting, we also observe that tendency. In another case of **L2** distribution fitting, the maximum log-likelihood value of -0.8770272 was obtained with a constrained cyclic (i.e., cyclic-zid) structure. Meanwhile, the log-likelihood value of a cyclic structure with $m = 8$ transitions is -0.8770357, which is smaller by 0.0000085. The reason behind this can be related to the numerical computational nature or/and the fact that the density function of **L2** is close to zero at zero (i.e., $f(0) \approx 0$).

In Table 2.13 the fitting results are compared from the perspective of cyclic structures with $m = 8$ transitions. The results here are more accurate in the sense that the digits which are not shown are used in the computation of the relative absolute error. Based on these results, there are three cases when Hyper-Erlang distribution yielded better results with $m \leq 8$ transitions. In theory, one would not expect such an outcome since cyclic structures are more dense. In our case, such a twist could be an influence of the fact that Hyper-Erlang fitting is much faster (and simpler, in the sense of the number of parameters), which might be significant given that for each fitting instance, only one minute of time is allocated.

Other minor details can be inspected by investigating the obtained results shown in Figures 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 2.10, 2.11, 2.12.

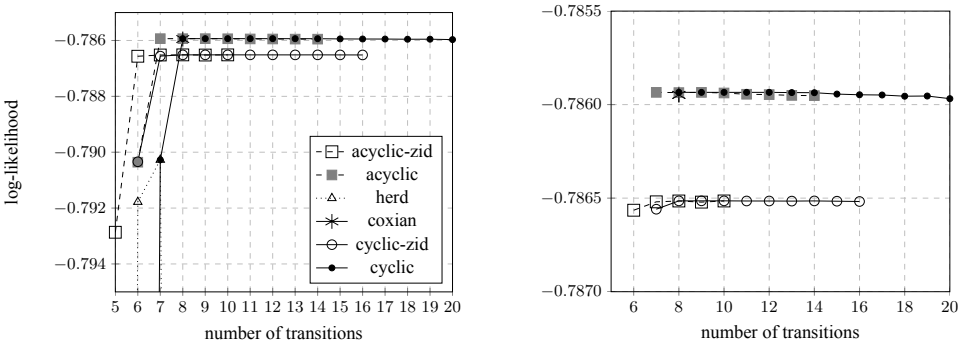

Fig. 2.4. Fitting **W1** distribution by PH(4).

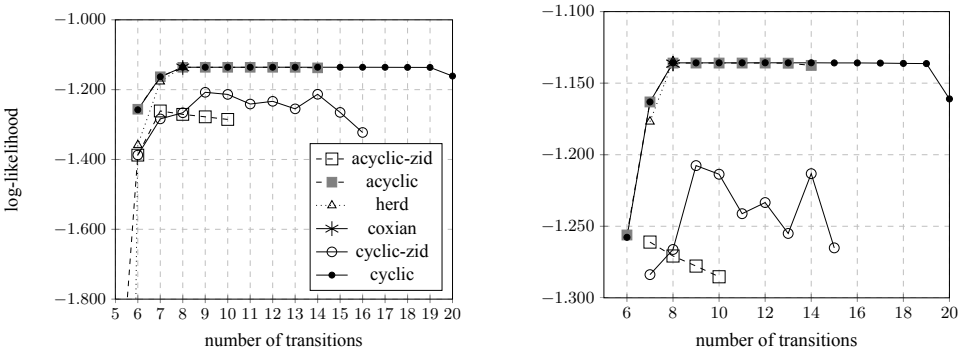

Fig. 2.5. Fitting **W2** distribution by PH(4).

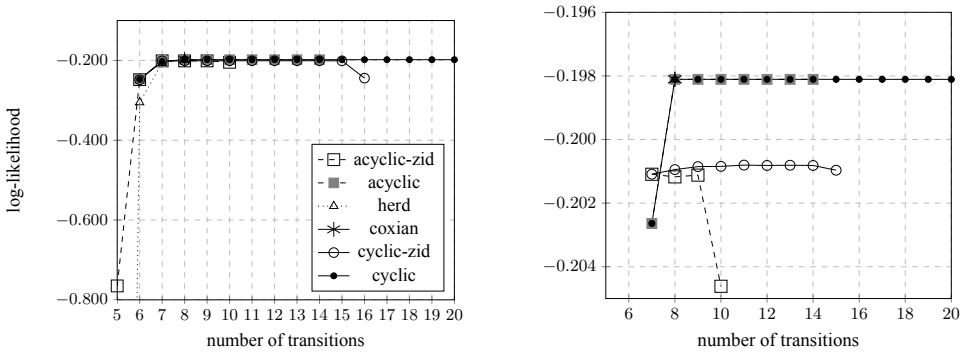

Fig. 2.6. Fitting **L1** distribution by PH(4).

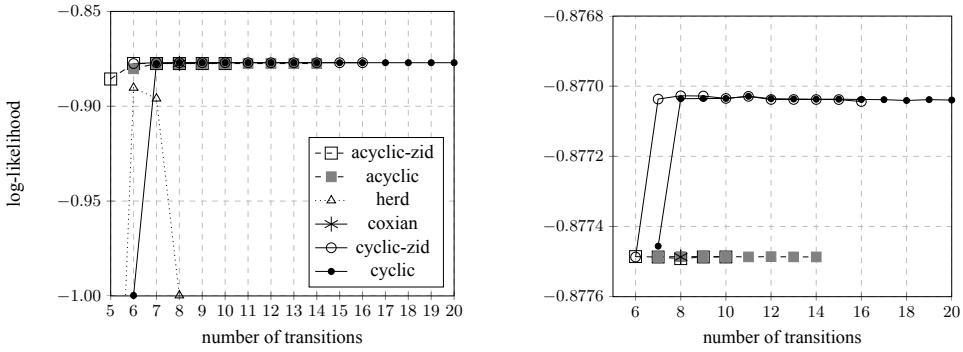

Fig. 2.7. Fitting **L2** distribution by PH(4).

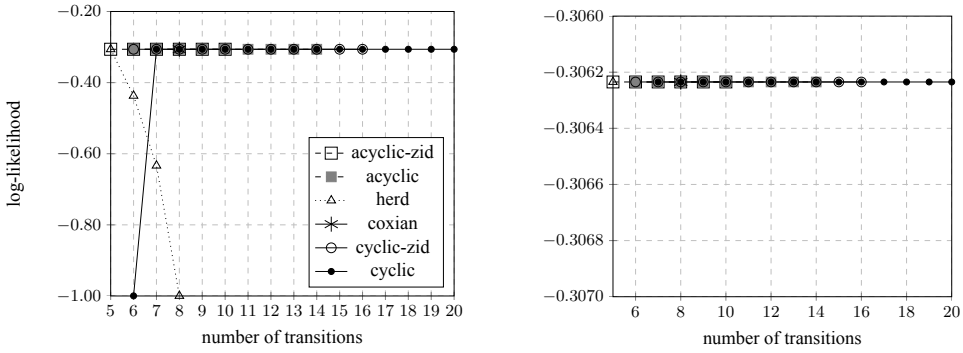

Fig. 2.8. Fitting **L3** distribution by PH(4).

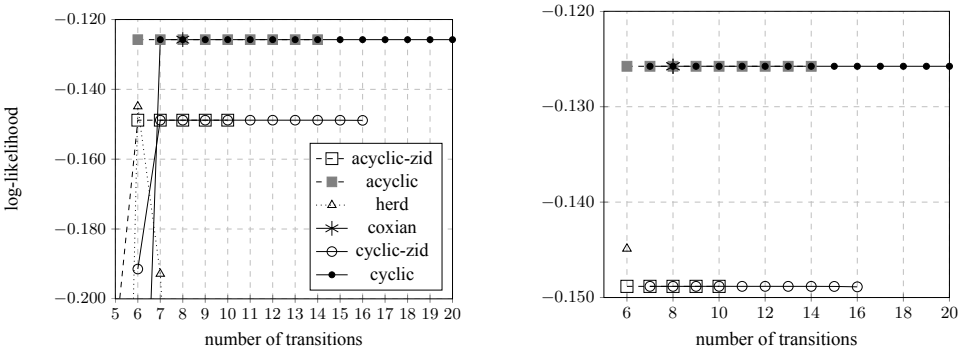

Fig. 2.9. Fitting **U1** distribution by PH(4).

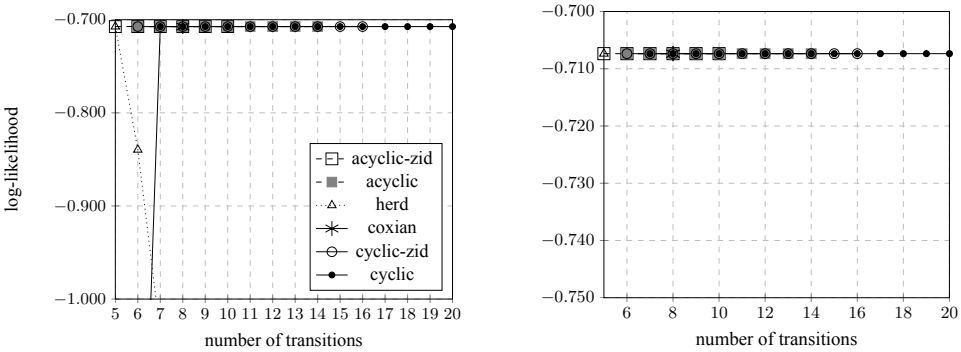

Fig. 2.10. Fitting **U2** distribution by PH(4).

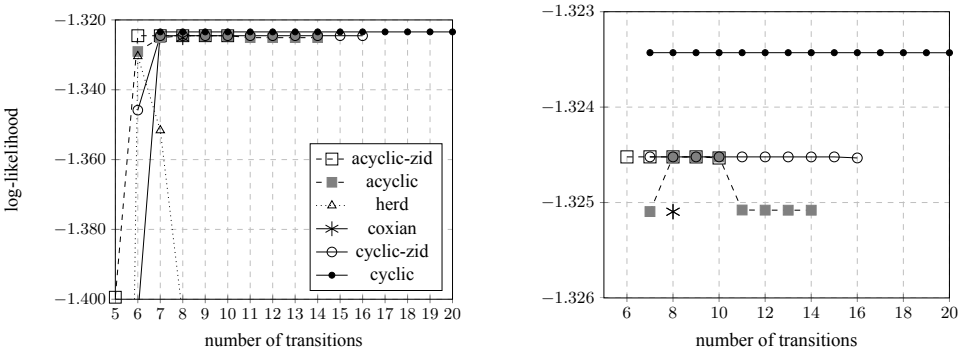

Fig. 2.11. Fitting **SE** distribution by PH(4).

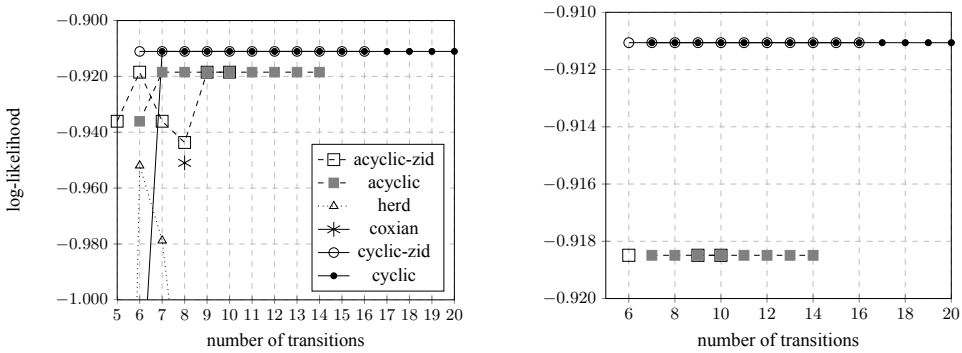

Fig. 2.12. Fitting **ME** distribution by PH(4).

Table 2.13. Comparison of fitted PH(4) cyclic(8) structures to the best ones.

Dist.	structure	mean	std.	skewness	llh
W1	cyclic(8)	$9.0237 \cdot 10^{-1}$	$6.1272 \cdot 10^{-1}$	1.0964	$-7.859343 \cdot 10^{-1}$
best:	cyclic(8)	$9.0237 \cdot 10^{-1}$	$6.1272 \cdot 10^{-1}$	1.0964	$-7.859343 \cdot 10^{-1}$
r.a.d.		0.0000	0.0000	0.0000	0.0000
W2	cyclic(8)	1.9953	4.1002	4.3393	-1.1358007
best:	herd(8)	1.9985	4.0133	4.0646	-1.1347523
r.a.d.		$1.6073 \cdot 10^{-3}$	$2.1649 \cdot 10^{-2}$	$6.7568 \cdot 10^{-2}$	$9.2397 \cdot 10^{-4}$
L1	cyclic(8)	$7.5376 \cdot 10^{-1}$	2.1846	8.3085	$-1.981078 \cdot 10^{-1}$
best:	cyclic(11)	$7.5376 \cdot 10^{-1}$	2.1844	8.3042	$-1.981076 \cdot 10^{-1}$
r.a.d.		$1.6561 \cdot 10^{-10}$	$1.2297 \cdot 10^{-4}$	$5.1381 \cdot 10^{-4}$	$6.7195 \cdot 10^{-7}$
L2	cyclic(8)	$9.9975 \cdot 10^{-1}$	$9.3464 \cdot 10^{-1}$	3.1925	$-8.770357 \cdot 10^{-1}$
best:	cyclic-zid(8)	$9.9975 \cdot 10^{-1}$	$9.3451 \cdot 10^{-1}$	3.1881	$-8.770272 \cdot 10^{-1}$
r.a.d.		$3.6460 \cdot 10^{-1}$	$1.3849 \cdot 10^{-4}$	$1.3681 \cdot 10^{-3}$	$9.6676 \cdot 10^{-6}$
L3	cyclic(8)	$9.9994 \cdot 10^{-1}$	$4.9997 \cdot 10^{-1}$	1.0000	$-3.062347 \cdot 10^{-1}$
best:	herd(5)	$9.9994 \cdot 10^{-1}$	$4.9997 \cdot 10^{-1}$	1.0000	$-3.062347 \cdot 10^{-1}$
r.a.d.		$1.3895 \cdot 10^{-7}$	$1.6275 \cdot 10^{-7}$	$7.1418 \cdot 10^{-8}$	$5.2074 \cdot 10^{-8}$
U1	cyclic(8)	$5.0000 \cdot 10^{-1}$	$3.1887 \cdot 10^{-1}$	$7.9719 \cdot 10^{-1}$	$-1.257613 \cdot 10^{-1}$
best:	cyclic(7)	$5.0000 \cdot 10^{-1}$	$3.1887 \cdot 10^{-1}$	$7.9719 \cdot 10^{-1}$	$-1.257613 \cdot 10^{-1}$
r.a.d.		$1.2847 \cdot 10^{-11}$	$8.3788 \cdot 10^{-7}$	$4.1990 \cdot 10^{-6}$	$3.1468 \cdot 10^{-9}$
U2	cyclic(8)	1.5000	$7.5000 \cdot 10^{-1}$	1.0000	$-7.073650 \cdot 10^{-1}$
best:	herd(5)	1.5000	$7.5000 \cdot 10^{-1}$	1.0000	$-7.073650 \cdot 10^{-1}$
r.a.d.		$2.3680 \cdot 10^{-7}$	$2.5942 \cdot 10^{-7}$	$6.7844 \cdot 10^{-8}$	$2.1940 \cdot 10^{-8}$
SE	cyclic(8)	1.4992	1.1108	1.2897	-1.3234294
best:	cyclic(8)	1.4992	1.1108	1.2897	-1.3234294
r.a.d.		0.0000	0.0000	0.0000	0.0000
ME	cyclic(8)	1.0483	1.0112	2.2941	$-9.110613 \cdot 10^{-1}$
best:	cyclic(10)	1.0483	1.0112	2.2941	$-9.110613 \cdot 10^{-1}$
r.a.d.		$3.7925 \cdot 10^{-8}$	$1.9893 \cdot 10^{-8}$	$1.2801 \cdot 10^{-9}$	$4.2002 \cdot 10^{-9}$

2.6. Phase-type fitting with randomly generated structures

In the previous section, we investigated how the fitting quality depends on the number of transitions. After empirically validating Hypothesis 2.1, it is of interest to investigate the fitting with $m = 2n$ transitions further. Considering the fact that our structure generation algorithm is hardly applicable for cases $n > 4$, we have introduced a simple random structure generation algorithm in Section 2.2.

Before starting numerical experiments, let us conduct a brief discussion. We may ask ourselves what advantages we expect from using $m = 2n$ transitions instead of $m = n^2 + n$. We expect a faster convergence due to a smaller number of parameters. Yet to find the most suitable structure, one would need to check a number of them. Also, it can be expected that the initial parameters of various structures would better explore the parameter space without being trapped in local maximums. To continue reasoning, one could consider fitting several structures in parallel, while combining that with a faster convergence would look like an attractive approach.

Contrary to that, although full structure fitting might be slower (from the computational and convergence points of view), it is universal and flexible.

Let us consider the number of iterations it takes for a likelihood to converge. A more practical scenario involving runtime is presented in Section 5.

The fitting procedure is stopped when the relative likelihood increase (compared with the likelihood value before iteration) is less than 10^{-8}. We generate 1180 initial parameter sets for fitting the full structure (FULL), the generated cyclic and acyclic structures (GEN), as well as randomly generated cyclic and acyclic structures (RND). We make sure that the structures are not constrained, either. Then we perform fitting until convergence is reached.

Table 2.14. Log-likelihood of approximated **W1** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.43274	$-7.85934 \cdot 10^{-1}$	$-7.91078 \cdot 10^{-1}$	$2.34399 \cdot 10^{-2}$
num. of iters.	127	22331	4574.24	4527.32
GEN:acyc-2n	$-7.92710 \cdot 10^{-1}$	$-7.85934 \cdot 10^{-1}$	$-7.87564 \cdot 10^{-1}$	$1.53709 \cdot 10^{-3}$
num. of iters.	76	12714	1889.29	2151.44
RND:cyc-2n	$-9.03057 \cdot 10^{-1}$	$-7.85934 \cdot 10^{-1}$	$-7.99615 \cdot 10^{-1}$	$3.27831 \cdot 10^{-2}$
num. of iters.	121	24470	4773.32	4573.64
RND:acyc-2n	$-8.97176 \cdot 10^{-1}$	$-7.85934 \cdot 10^{-1}$	$-7.88485 \cdot 10^{-1}$	$6.52271 \cdot 10^{-3}$
num. of iters.	10	9780	1132.02	1476.19
FULL	$-7.87448 \cdot 10^{-1}$	$-7.85950 \cdot 10^{-1}$	$-7.86003 \cdot 10^{-1}$	$2.45251 \cdot 10^{-4}$
num. of iters.	727	18536	5827.76	3051.21

Table 2.15. Log-likelihood of approximated **W2** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	$-5.97261 \cdot 10^{1}$	-1.13581	-1.39815	2.85502
num. of iters.	99	10852	2476.16	2010.84
GEN:acyc-2n	$-1.29736 \cdot 10^{1}$	-1.13589	-1.41205	1.36130
num. of iters.	24	9347	1338.43	1649.04
RND:cyc-2n	$-5.70556 \cdot 10^{1}$	-1.13582	-1.37338	2.21655
num. of iters.	13	11941	2789.83	2502.87
RND:acyc-2n	$-5.94120 \cdot 10^{1}$	-1.13588	-1.54531	2.86070
num. of iters.	23	8134	963.13	1253.20
FULL	-3.69040	-1.13597	-1.46150	$3.43201 \cdot 10^{-1}$
num. of iters.	88	936	324.94	114.82

Table 2.16. Log-likelihood of approximated **L1** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.55279	$-1.98108 \cdot 10^{-1}$	$-2.14636 \cdot 10^{-1}$	$8.02157 \cdot 10^{-2}$
num. of iters.	69	13193	1513.17	1697.82
GEN:acyc-2n	-4.00517	$-1.98108 \cdot 10^{-1}$	$-2.21754 \cdot 10^{-1}$	$2.32945 \cdot 10^{-1}$
num. of iters.	58	8785	1146.12	1437.99
RND:cyc-2n	-4.01233	$-1.98108 \cdot 10^{-1}$	$-2.21805 \cdot 10^{-1}$	$1.44771 \cdot 10^{-1}$
num. of iters.	46	12134	1728.60	2013.43
RND:acyc-2n	-1.55603	$-1.98108 \cdot 10^{-1}$	$-2.08444 \cdot 10^{-1}$	$7.68505 \cdot 10^{-2}$
num. of iters.	63	8278	887.03	1124.68
FULL	$-2.03026 \cdot 10^{-1}$	$-1.98108 \cdot 10^{-1}$	$-1.98358 \cdot 10^{-1}$	$1.07832 \cdot 10^{-3}$
num. of iters.	84	3558	756.21	432.08

Table 2.17. Log-likelihood of approximated **L2** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.53534	$-8.77034 \cdot 10^{-1}$	$-8.88775 \cdot 10^{-1}$	$2.92762 \cdot 10^{-2}$
num. of iters.	130	20813	2528.37	3084.88
GEN:acyc-2n	$-9.99735 \cdot 10^{-1}$	$-8.77486 \cdot 10^{-1}$	$-8.85208 \cdot 10^{-1}$	$9.46271 \cdot 10^{-3}$
num. of iters.	122	4646	1087.40	727.63
RND:cyc-2n	-1.92702	$-8.77034 \cdot 10^{-1}$	$-8.98453 \cdot 10^{-1}$	$4.90460 \cdot 10^{-2}$
num. of iters.	142	25019	2982.72	3580.39
RND:acyc-2n	$-9.99736 \cdot 10^{-1}$	$-8.77486 \cdot 10^{-1}$	$-8.87166 \cdot 10^{-1}$	$1.03743 \cdot 10^{-2}$
num. of iters.	42	3882	839.45	590.90
FULL	$-8.89936 \cdot 10^{-1}$	$-8.77037 \cdot 10^{-1}$	$-8.77191 \cdot 10^{-1}$	$9.80945 \cdot 10^{-4}$
num. of iters.	1732	14815	6898.62	2542.72

Table 2.18. Log-likelihood of approximated **L3** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.53525	$-3.06235 \cdot 10^{-1}$	$-4.42611 \cdot 10^{-1}$	$1.41878 \cdot 10^{-1}$
num. of iters.	23	24003	847.70	2358.76
GEN:acyc-2n	$-6.33443 \cdot 10^{-1}$	$-3.06235 \cdot 10^{-1}$	$-4.16492 \cdot 10^{-1}$	$1.08100 \cdot 10^{-1}$
num. of iters.	20	711	73.80	47.33
RND:cyc-2n	-1.00662	$-3.06235 \cdot 10^{-1}$	$-5.06302 \cdot 10^{-1}$	$2.00977 \cdot 10^{-1}$
num. of iters.	26	20366	1672.40	3260.66
RND:acyc-2n	$-9.99739 \cdot 10^{-1}$	$-3.06235 \cdot 10^{-1}$	$-4.77754 \cdot 10^{-1}$	$1.13840 \cdot 10^{-1}$
num. of iters.	4	773	63.50	38.93
FULL	$-4.37012 \cdot 10^{-1}$	$-3.06235 \cdot 10^{-1}$	$-3.06346 \cdot 10^{-1}$	$3.80545 \cdot 10^{-3}$
num. of iters.	76	347	122.93	29.73

Table 2.19. Log-likelihood of approximated **U1** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	$-8.42470 \cdot 10^{-1}$	$-1.25761 \cdot 10^{-1}$	$-1.54397 \cdot 10^{-1}$	$3.69528 \cdot 10^{-2}$
num. of iters.	58	37047	2167.33	5333.83
GEN:acyc-2n	$-1.92843 \cdot 10^{-1}$	$-1.25762 \cdot 10^{-1}$	$-1.47388 \cdot 10^{-1}$	$2.11722 \cdot 10^{-2}$
num. of iters.	35	5200	524.91	375.17
RND:cyc-2n	$-3.10585 \cdot 10^{-1}$	$-1.25761 \cdot 10^{-1}$	$-1.70302 \cdot 10^{-1}$	$5.12941 \cdot 10^{-2}$
num. of iters.	60	38289	3157.04	6188.91
RND:acyc-2n	$-3.06852 \cdot 10^{-1}$	$-1.25762 \cdot 10^{-1}$	$-1.59073 \cdot 10^{-1}$	$2.44619 \cdot 10^{-2}$
num. of iters.	7	4368	405.36	381.18
FULL	$-1.25766 \cdot 10^{-1}$	$-1.25762 \cdot 10^{-1}$	$-1.25764 \cdot 10^{-1}$	$1.09720 \cdot 10^{-6}$
num. of iters.	588	1633	689.68	109.15

Table 2.20. Log-likelihood of approximated **U2** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.94108	$-7.07365 \cdot 10^{-1}$	$-8.44990 \cdot 10^{-1}$	$1.43075 \cdot 10^{-1}$
num. of iters.	23	31374	893.87	2757.28
GEN:acyc-2n	-1.03761	$-7.07365 \cdot 10^{-1}$	$-8.18787 \cdot 10^{-1}$	$1.09130 \cdot 10^{-1}$
num. of iters.	19	689	73.13	46.63
RND:cyc-2n	-1.41234	$-7.07365 \cdot 10^{-1}$	$-9.09390 \cdot 10^{-1}$	$2.02371 \cdot 10^{-1}$
num. of iters.	25	27636	1760.61	3737.77
RND:acyc-2n	-1.40546	$-7.07365 \cdot 10^{-1}$	$-8.80638 \cdot 10^{-1}$	$1.14815 \cdot 10^{-1}$
num. of iters.	4	765	63.07	38.60
FULL	$-7.07365 \cdot 10^{-1}$	$-7.07365 \cdot 10^{-1}$	$-7.07365 \cdot 10^{-1}$	$3.70587 \cdot 10^{-9}$
num. of iters.	88	342	121.71	29.75

Table 2.21. Log-likelihood of approximated **SE** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.94038	-1.32343	-1.33500	$2.25813 \cdot 10^{-2}$
num. of iters.	100	28843	2014.33	2675.55
GEN:acyc-2n	-1.39747	-1.32452	-1.33192	$9.08830 \cdot 10^{-3}$
num. of iters.	81	9568	1302.69	834.39
RND:cyc-2n	-1.40868	-1.32343	-1.34286	$2.46830 \cdot 10^{-2}$
num. of iters.	96	28635	2918.37	3802.25
RND:acyc-2n	-1.40481	-1.32452	-1.33644	$1.13667 \cdot 10^{-2}$
num. of iters.	12	5437	903.70	685.24
FULL	-1.33113	-1.32343	-1.32353	$8.76791 \cdot 10^{-4}$
num. of iters.	999	4370	1948.51	397.25

Table 2.22. Log-likelihood of approximated **ME** distribution by PH(4) distributions.

	min.	max.	mean	std.
GEN:cyc-2n	-1.58259	$-9.11061 \cdot 10^{-1}$	$-9.36605 \cdot 10^{-1}$	$2.95785 \cdot 10^{-2}$
num. of iters.	179	17239	1561.01	1964.33
GEN:acyc-2n	$-9.53523 \cdot 10^{-1}$	$-9.18489 \cdot 10^{-1}$	$-9.42556 \cdot 10^{-1}$	$1.40954 \cdot 10^{-2}$
num. of iters.	120	7227	981.47	990.98
RND:cyc-2n	-1.05212	$-9.11061 \cdot 10^{-1}$	$-9.48006 \cdot 10^{-1}$	$3.70084 \cdot 10^{-1}$
num. of iters.	127	19610	1990.67	2776.28
RND:acyc-2n	-1.04704	$-9.18489 \cdot 10^{-1}$	$-9.47486 \cdot 10^{-1}$	$1.34718 \cdot 10^{-2}$
num. of iters.	36	6443	838.13	973.18
FULL	$-9.50924 \cdot 10^{-1}$	$-9.11061 \cdot 10^{-1}$	$-9.15131 \cdot 10^{-1}$	$1.19459 \cdot 10^{-2}$
num. of iters.	1107	8796	1938.76	821.81

The very first observation (see Tables 2.14, 2.15, 2.16, 2.17, 2.18, 2.19, 2.20, 2.21, 2.22) is that the standard deviation of log-likelihood obtained by the full structure fitting is smaller in all cases. To add, the mean of log-likelihood obtained by the full structure fitting is the biggest (except for case **W2**). All of that is due to the full structure flexibility.

Regarding sparse structures, the minimum number of iterations to reach a convergence is smaller compared to the full structure. Yet, the maximum number of iterations to reach convergence can be substantially higher for some structures. This kind of behavior is expected. In case a suitable structure is used, one would expect a fast convergence; on the other hand, if a structure is quite different from the optimal one, the convergence could be slow; that should be checked.

We may wonder whether fitting with sparse structures helps to find more likely parameter estimates. We selected the cases when the biggest log-likelihood value was

obtained and computed, and the absolute difference was compared to the one obtained by the full structure. A summary is presented in Table 2.23.

Table 2.23. The difference of the highest log-likelihood values of the sparse structures and the ones of the full structure.

Distribution	best subclass	log-likelihood difference
W1	GEN:acyc-2n	$1.57728 \cdot 10^{-5}$
W2	GEN:cyc-2n	$1.59915 \cdot 10^{-4}$
L1	RND:acyc-2n	$3.82625 \cdot 10^{-8}$
L2	RND:cyc-2n	$3.12390 \cdot 10^{-6}$
L3	RND:cyc-2n	$1.21557 \cdot 10^{-8}$
U1	GEN:cyc-2n	$4.75536 \cdot 10^{-7}$
U2	GEN:acyc-2n	$1.24849 \cdot 10^{-8}$
SE	GEN:cyc-2n	$5.96125 \cdot 10^{-8}$
ME	RND:cyc-2n	$2.58314 \cdot 10^{-7}$

The results summary in Table 2.23 reveals that the highest log-likelihood values have been obtained by sparse structures. However, in most cases, the difference compared to the highest log-likelihood of the full structure is small. The likely reason for this is that the fitting of the full structure is stopped when a zone of slow convergence has been reached.

It is interesting to see how log-likelihood values relate to the number of iterations. We may wonder whether our assumption that a near optimal structure converges faster is correct. To check that, we have plotted log-likelihooh dependency on a number of iterations graphs (see Figures 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19, 2.20, 2.21). The black dots represent cyclic structures (GEN:cyc-2n), the red dots denote acyclic structures (GEN:acyc-2n), and the blue dots represent the full structure.

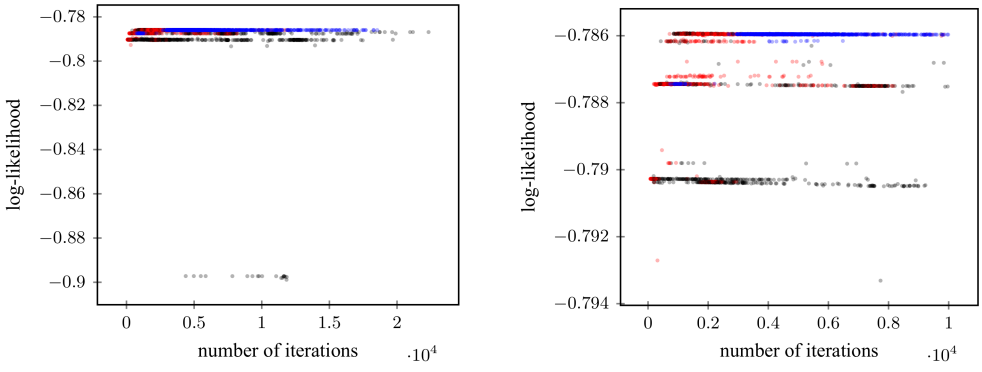

Fig. 2.13. Fitting **W1** distribution by PH(4) using generated structures and the full structure.

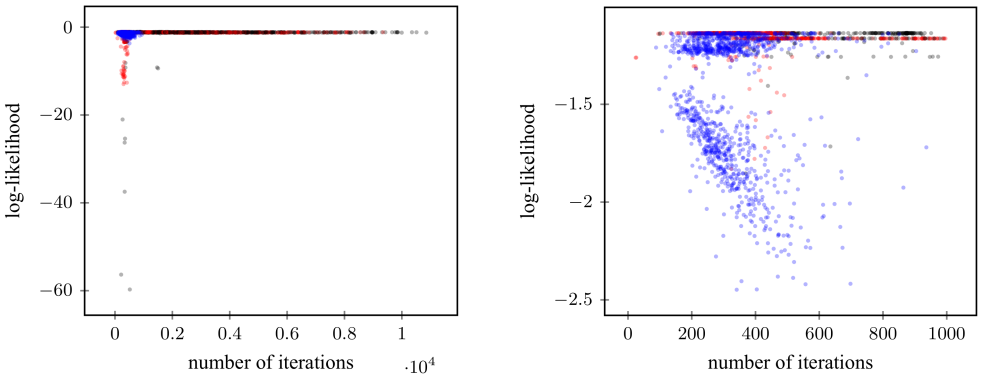

Fig. 2.14. Fitting **W2** distribution by PH(4) using generated structures and the full structure.

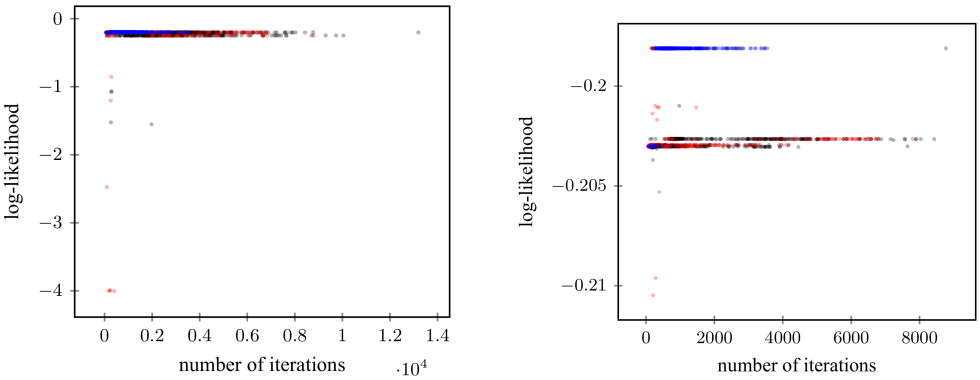

Fig. 2.15. Fitting **L1** distribution by PH(4) using generated structures and the full structure.

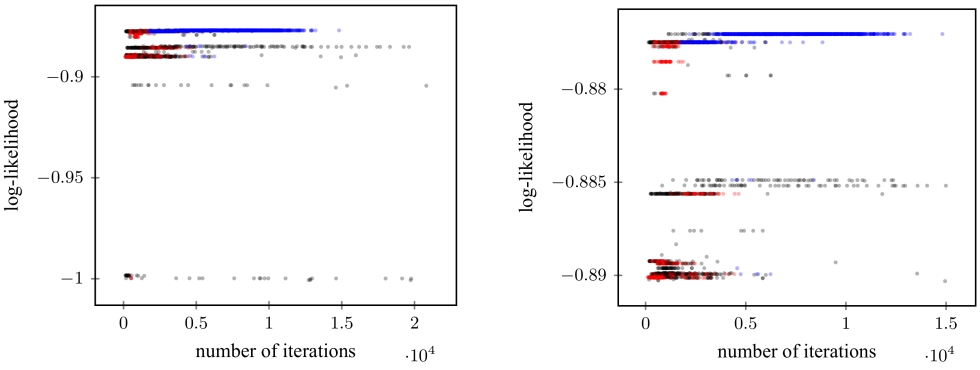

Fig. 2.16. Fitting **L2** distribution by PH(4) using generated structures and the full structure.

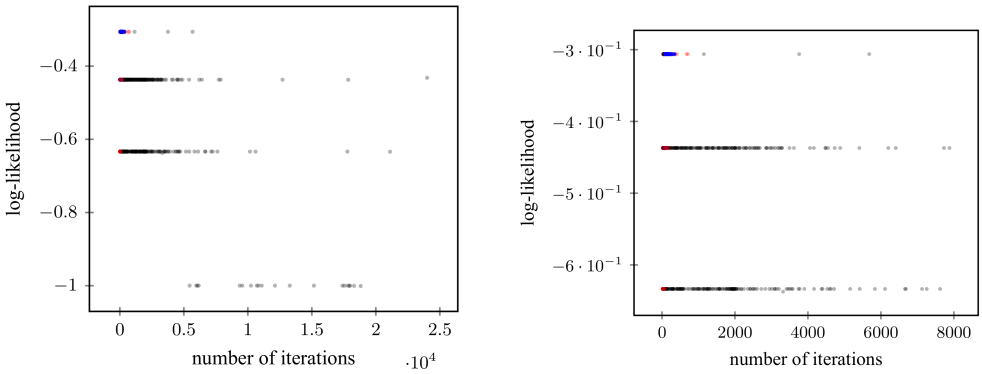

Fig. 2.17. Fitting **L3** distribution by PH(4) using generated structures and the full structure.

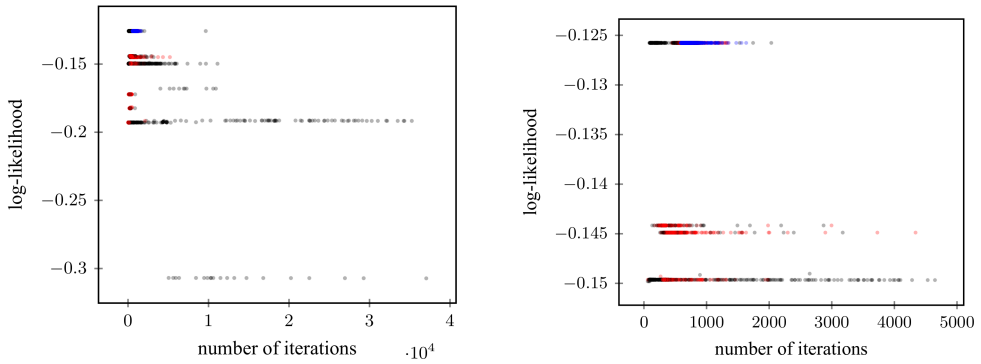

Fig. 2.18. Fitting **U1** distribution by PH(4) using generated structures and the full structure.

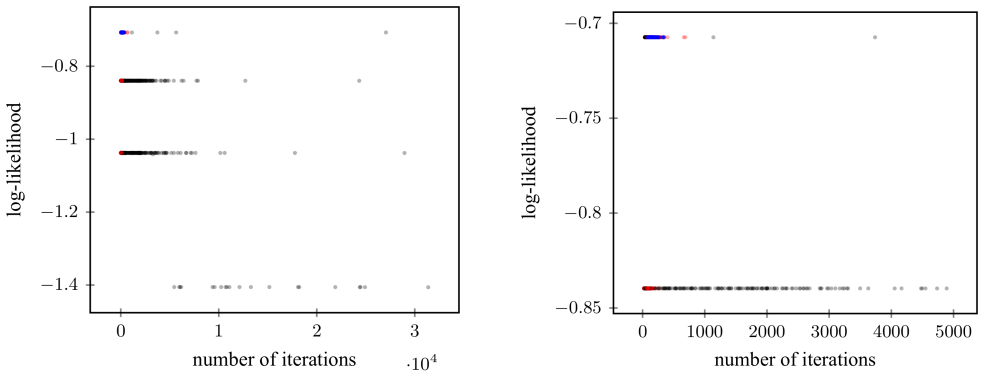

Fig. 2.19. Fitting **U2** distribution by PH(4) using generated structures and the full structure.

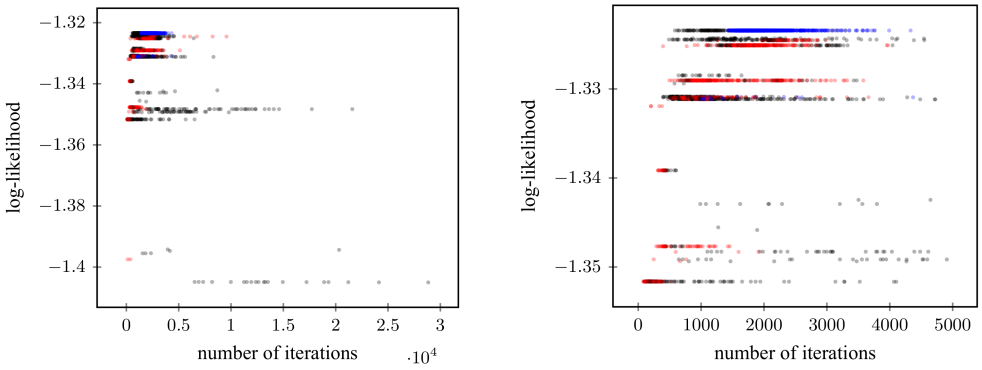

Fig. 2.20. Fitting **SE** distribution by PH(4) using generated structures and the full structure.

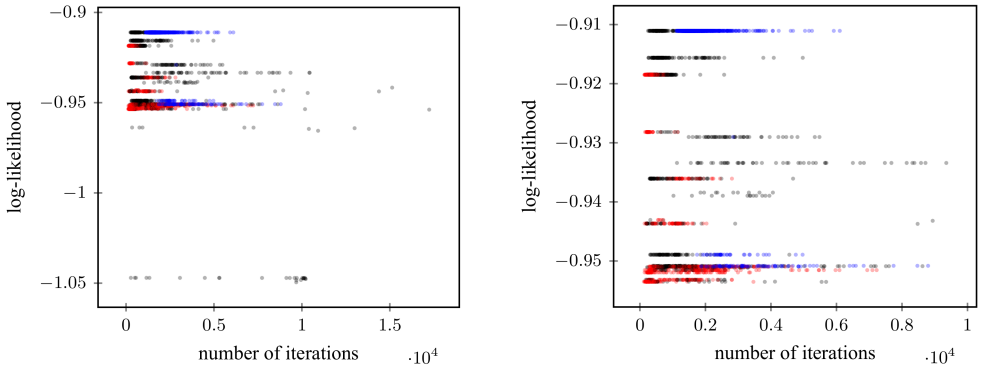

Fig. 2.21. Fitting **ME** distribution by PH(4) using generated structures and the full structure.

The very first observation is that log-likelihood values form distinctive clusters. Also, the results obtained by fitting with the full structure tend to be in the highest log-likelihood cluster. Meanwhile, fitting sparse structures results in a number of other clusters, as well. Thus it seems that even though sparse structures do better explore the parameter space, it does not result in distinctively more likely parameter estimates. But, in most cases, a properly selected sparse structure finds the best parameter estimates quicker. On the other hand, the selection of a not-so-suitable structure can result in slow convergence and/or convergence to less likely parameter estimates.

We can raise an interesting research question. Given the trace data, how do we find the optimal (or a set of near optimal) sparse structure? If such a structure could be efficiently determined (with negligible computational effort), the fitting time could be reduced due to the faster convergence and smaller computational effort. We will explore in this direction in Section 5.

2.7. Conclusions

An algorithm for generating a Phase-type distribution representing matrix form structure has been developed. However, it is not guaranteed that the obtained structure set (of a specified number of transitions m) is not minimal. The algorithm initially generates all the possible structures and then eliminates the redundant ones based on various insights. Due to this fact, for higher order ($n > 4$) structures, the initial number of structures increases exponentially, which is not effective. Despite these drawbacks, the generated structure set was successfully used in the empirical validation of the following hypothesis.

The hypothesis of a sufficient number of transitions ($m = 2n$) in a Phase-type distribution structure has been formulated and empirically validated for case $n = 4$. The expectation maximization method has been used to perform Phase-type fitting of nine benchmark distributions given in scientific literature while using the set of the generated structures. Based on the likelihood dependence on the number of transitions m, it was concluded that the stated hypothesis is valid. The only contradiction against the hypothesis statement was obtained for **L1** distribution, for which, slightly more likely parameter estimates have been obtained while using a structure with one additional transition (i.e., $m = 2n+1 = 9$). However, the log-likelihood is only 0.0000002 higher compared to the one obtained using $m = 2n = 8$ transitions, which can be considered to be negligible, especially considering the nature of numerical computations.

In addition, our results compiled for hypothesis validation also revealed other tendencies. In scientific literature, it has been observed that acyclic structure fitting via the EM method can yield more likely parameter estimates than with the Coxian structure even despite the fact that the latter covers the whole Phase-type distribution representations with acyclic structure. This is justified simply by stating that for the EM method, the Coxian structure is not necessarily optimal. Our results go along these lines, too. Another observation is regarding the constrained structure specifying Phase-type distributions with $p = 2n-2$ parameters at most. These structures are less flexible, and this fact shows up consistently in most of our results.

Next, the hypothesis regarding the sufficient number of transitions has been applied for the Phase-type fitting. Phase-type fitting with sparse (i.e., $m = 2n$ transitions) structures versus the full structure has been compared for case $n = 4$. In addition to our structure generation algorithm, a more robust random structure generation algorithm has been used. The following key observations have been outlined:

1. Due to the full structure flexibility, the likelihood of the obtained parameter estimates depends less on the initial parameters. With a sparse structure, the opposite tendency has been observed.
2. A full structure, due to a redundant number of parameters, converges slower, and can be terminated too early.
3. The fitting characteristics using a sparse structure depend much more on the

specific structure and trace data. If a proper structure is chosen, it can converge much faster compared to the full structure. In case of a not-so-optimal structure, the convergence can be even slower.

Sparse structure fitting may potentially have an advantage over the full structure in case it is necessary to find more likely parameter estimates with a smaller number of iterations. However, the result reliability depends on the number of sparse structures checked, which can be done in parallel. To better utilize the faster sparse structure convergence, it is necessary to develop a reliable method for the sparse structure identification.

3. MARKOV ARRIVAL PROCESS OF ER-CHMM STRUCTURE FITTING

The work in this section is mainly based on the results presented in the paper 'Parallel algorithms for fitting Markov Arrival Processes'.

The fitting of Markov arrival processes (MAPs) with the expectation-maximization (EM) algorithm is a computationally demanding task. There are attempts in the literature to reduce the computational complexity by introducing special MAP structures instead of the general representation. Another possibility to improve the efficiency of MAP fitting is to reformulate the inherently serial classical EM algorithm to exploit massively parallel hardware architectures. In this section, we present three different EM-based fitting procedures that can take advantage of the parallel hardware (such as Graphics Processing Units, GPUs) and apply a special MAP structure, the Erlang distributed continuous-time hidden Markov chain (ER-CHMM) structure for reducing the computational complexity. Both [31] and [32] arrived at the interesting conclusion that the structurally restricted ER-CHMM-based EM algorithm, while being significantly faster than the EM algorithm operating on the full (dense) MAP class, often achieves significantly higher likelihood values.

There are a few EM algorithms published in the literature for hidden Markov model (HMM) fitting that are also based on dependent samples [87, 88, 89]. In all these papers, similar to our work, formalizing the problem with matrices was the key idea enabling the parallel implementation. [87, eq. (30)] provides an important relation for the iterative parallel computation of the performance indices for the HMM problem with continuous observations, but this idea has not been put forward to an efficient parallel implementation in [87]. [88] and [89] proposed parallel solutions without using [87, eq. (30)] for a simpler HMM setting with discrete observations. These HMM models differ from the MAP fitting problem in the meaning of model parameters, but their computational approaches are similar to ours. Our contribution is that we adapt the principal relation [87, eq. (30)] for the MAP fitting problem, based on which, we develop parallel algorithms organized similarly to the ones in [88, 89]. By mixing the elements of these algorithms in different ways, we arrive to three different algorithm variants, which covers the practically relevant spectrum of the computation time – memory requirement trade-off. Having these algorithms defined and implemented in a unified manner enabled us to compare their performance fairly.

3.1. The serial algorithm

The standard, naive implementation of the EM algorithm is serial. Computing and saving likelihood vectors $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ for $u = 1, \ldots, T$ based on (28) implies $2T$ vector-matrix multiplication operations of size R and saving $2T$ vectors of size R in the memory.

Having likelihood vectors $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ for $u = 1, \ldots, T$, the $\boldsymbol{\lambda}_i$, $\boldsymbol{\Pi}_{i,j}$, $\boldsymbol{\pi}_i$ estimates ($R^2 + 2R$ parameters) can be computed based on (29), (30) and (31) that needs

$\mathcal{O}(T)$ scalar multiplications for all parameters.

There are some straightforward, albeit limited, possibilities to make this essentially serial computation parallel. Vectors $\boldsymbol{a}[u]$, for $u = 1, \ldots, T$, and vectors $\boldsymbol{b}[u]$, for $u = T, \ldots, 1$, can be obtained simultaneously, by two execution threads. The recursive definition of these vectors does not allow a higher level of parallelism. Updating the estimates in the M-step, i.e., computing (29), (30) and (31), can also benefit from parallel hardware, $\boldsymbol{\lambda}_i$ and $\boldsymbol{\pi}_i$ can be computed simultaneously for $i = 1, \ldots, R$ by $2R$ execution threads, and $\boldsymbol{\Pi}_{i,j}$ for $i, j = 1, \ldots, R$ by R^2 execution threads.

In the subsequent sections, several options are considered to transform this algorithm to a massively parallel one with a different number of passes through the data set.

3.2. The parallel algorithm with one pass

We start by reformulating the basic forward-backward algorithm with the matrix notation, which is an important step towards making the algorithm massively parallel.

By defining matrix $\boldsymbol{P}[\begin{smallmatrix} v \\ u \end{smallmatrix}]$ for $1 \le u \le v \le T$ as

$$\boldsymbol{P}[\begin{smallmatrix} v \\ u \end{smallmatrix}] = \boldsymbol{P}[u]\boldsymbol{P}[u+1]\ldots\boldsymbol{P}[v] = \prod_{z=u}^{v} \begin{bmatrix} f_1(x_z)\boldsymbol{\Pi}_{1,1} & \ldots & f_1(x_z)\boldsymbol{\Pi}_{1,R} \\ \vdots & \ddots & \vdots \\ f_R(x_z)\boldsymbol{\Pi}_{R,1} & \ldots & f_R(x_z)\boldsymbol{\Pi}_{R,R} \end{bmatrix}, \quad (49)$$

and for $u > v$ as $\boldsymbol{P}[\begin{smallmatrix} v \\ u \end{smallmatrix}] = \boldsymbol{I}$, the forward and backward likelihood vectors can be expressed by

$$\boldsymbol{a}[u] = \boldsymbol{\pi}\boldsymbol{P}[\begin{smallmatrix} u \\ 1 \end{smallmatrix}], \text{ and } \boldsymbol{b}[u] = \boldsymbol{P}[\begin{smallmatrix} T \\ u \end{smallmatrix}]\mathbb{1}. \quad (50)$$

We divide the T inter-arrival times into L partitions: inter-arrival times $x_1, \ldots, x_K$ are assigned to partition 1, $x_{K+1}, \ldots, x_{2K}$ to partition 2, and so on, where the number of inter-arrival times in the first $L-1$ partitions is $K = \lceil T/L \rceil$, and the size of the last partition is $T - (L-1)K$ (which can be smaller than K). If u belongs to the ℓth partition, that is, $(\ell-1)K + 1 \le u \le \ell K$, then we have

$$\boldsymbol{a}[u] = \boldsymbol{\pi}\left(\prod_{z=1}^{\ell-1} \boldsymbol{P}[\begin{smallmatrix} zK \\ (z-1)K+1 \end{smallmatrix}]\right) \boldsymbol{P}[\begin{smallmatrix} u \\ (\ell-1)K+1 \end{smallmatrix}], \quad (51)$$

$$\boldsymbol{b}[u] = \boldsymbol{P}[\begin{smallmatrix} \ell K \\ u \end{smallmatrix}]\left(\prod_{z=\ell+1}^{L-1} \boldsymbol{P}[\begin{smallmatrix} zK \\ (z-1)K+1 \end{smallmatrix}]\right) \boldsymbol{P}[\begin{smallmatrix} T \\ (L-1)K+1 \end{smallmatrix}]\mathbb{1}. \quad (52)$$

The main observation that enables the parallel implementation is that matrices $\boldsymbol{P}[\begin{smallmatrix} zK \\ (z-1)K+1 \end{smallmatrix}]$ can be calculated simultaneously for $\ell = 1, \ldots, L$. To simplify the notation, we introduce the *likelihood matrix* corresponding to partition ℓ as

$$\boldsymbol{U}[\ell] = \boldsymbol{P}[\begin{smallmatrix} \ell K \\ (\ell-1)K+1 \end{smallmatrix}], \text{ for } \ell = 1, \ldots, L-1, \text{ and } \boldsymbol{U}[L] = \boldsymbol{P}[\begin{smallmatrix} T \\ (L-1)K+1 \end{smallmatrix}].$$

The likelihood vectors corresponding to the inter-arrival times up to partition ℓ and from partition ℓ on can be expressed from the likelihood matrices by

$$\boldsymbol{\pi}^{\ell} = \boldsymbol{a}[\ell K] = \boldsymbol{\pi}\boldsymbol{P}[\,{}^{\ell K}_{1}\,] = \boldsymbol{\pi}\left(\prod_{u=1}^{\ell}\boldsymbol{U}[u]\right),$$

$$\mathbb{1}^{\ell} = \boldsymbol{b}[\ell K+1] = \boldsymbol{P}[\,{}^{T}_{\ell K+1}\,]\mathbb{1} = \left(\prod_{u=\ell+1}^{L}\boldsymbol{U}[u]\right)\mathbb{1},$$

respectively. By this notation, for $(\ell-1)K+1 \le u \le \ell K$, likelihood vectors corresponding to each inter-arrival time simplify to

$$\boldsymbol{a}[u] = \boldsymbol{\pi}^{\ell-1}\boldsymbol{P}[\,{}^{u}_{(\ell-1)K+1}\,], \text{ and } \boldsymbol{b}[u] = \boldsymbol{P}[\,{}^{\ell K}_{u}\,]\mathbb{1}^{\ell+1}, \tag{53}$$

where $\boldsymbol{P}[\,{}^{u}_{(\ell-1)K+1}\,]$ and $\boldsymbol{P}[\,{}^{\ell K}_{u}\,]$ are likelihood matrices within partition ℓ.

To compute $\boldsymbol{\lambda}_i$, $\boldsymbol{\Pi}_{i,j}$ and $\boldsymbol{\pi}_i$ in parallel, we rewrite (29), (30) and (31) as

$$\boldsymbol{\lambda}_i = \frac{\sum_{u=1}^{T}\boldsymbol{r}_i\boldsymbol{a}_i[u-1]\boldsymbol{b}_i[u]}{\sum_{u=1}^{T}x_u\boldsymbol{a}_i[u-1]\boldsymbol{b}_i[u]} = \frac{\boldsymbol{r}_i S_i^{(1)}}{S_i^{(2)}}, \tag{54}$$

$$\boldsymbol{\Pi}_{i,j} = \frac{\sum_{u=1}^{T-1}\boldsymbol{a}_i[u-1]f_i(x_u)\boldsymbol{\Pi}_{i,j}\boldsymbol{b}_j[u+1]}{\sum_{u=1}^{T-1}\boldsymbol{a}_i[u-1]\boldsymbol{b}_i[u]} = \frac{S_{i,j}^{(3)}}{S_i^{(3)}}, \tag{55}$$

$$\boldsymbol{\pi}_i = \frac{\sum_{u=1}^{T}\boldsymbol{a}_i[u-1]\boldsymbol{b}_i[u]}{T\boldsymbol{a}[T]\mathbb{1}} = \frac{1}{T\boldsymbol{a}[T]\mathbb{1}}S_i^{(1)}, \tag{56}$$

where $S_i^{(1)}$ and $S_i^{(3)}$ differs only in a single element

$$S_i^{(1)} = S_i^{(3)} + \boldsymbol{a}_i[T-1]\boldsymbol{b}_i[T], \tag{57}$$

and $S_i^{(3)}$ can be obtained from $S_{i,j}^{(3)}$ as

$$S_i^{(3)} = \sum_{j=1}^{R} S_{i,j}^{(3)}, \tag{58}$$

by which we focus only on $S_i^{(2)}$ and $S_{i,j}^{(3)}$.

To simplify further notation, we define the number of inter-arrivals before the ℓ^{th} partition as

$$\iota_\ell = (\ell-1)K.$$

To make the parallel computation of $S_i^{(2)}$ possible, we separate these sums into parts corresponding to the partitions as

$$
\begin{aligned}
S_i^{(2)} &= \sum_{\ell=1}^{L-1}\sum_{u=1}^{K} \boldsymbol{a}_i[\iota_\ell + u - 1]x_{\iota_\ell+u}\boldsymbol{b}_i[\iota_\ell + u] \\
&+ \sum_{u=1}^{T-(L-1)K} \boldsymbol{a}_i[\iota_L + u - 1]x_{\iota_L+u}\boldsymbol{b}_i[\iota_L + u] \\
&= \sum_{\ell=1}^{L-1} \boldsymbol{\pi}^{\ell-1} \underbrace{\sum_{u=1}^{K} \boldsymbol{P}\big[{}^{\iota_\ell+u-1}_{\iota_\ell+1}\big]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}x_{\iota_\ell+u}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}\big[{}^{\ell K}_{\iota_\ell+u}\big]}_{\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,K]=\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]} \mathbb{1}^{\ell+1} \\
&+ \boldsymbol{\pi}^{L-1} \underbrace{\sum_{u=1}^{T-(L-1)K} \boldsymbol{P}\big[{}^{\iota_L+u-1}_{\iota_L+1}\big]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}x_{\iota_L+u}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}\big[{}^{T}_{\iota_L+u}\big]}_{\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[L,T-(L-1)K]=\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[L]} \mathbb{1},
\end{aligned}
\tag{59}
$$

where

$$
\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell, z] = \sum_{u=1}^{z} \boldsymbol{P}\big[{}^{\iota_\ell+u-1}_{\iota_\ell+1}\big]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}x_{\iota_\ell+u}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}\big[{}^{\iota_\ell+z}_{\iota_\ell+u}\big]
$$

and we make use of (53). The first (double) summation corresponds to partitions $1, \ldots, L-1$, and the second one to the last (L^{th}) partition having potentially fewer than K elements. Matrices $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell, z]$ represent the sum in partition ℓ up to term z, while we introduce the shorthand notation $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ to denote the sum over all terms of partition ℓ. The main observation is that $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ can be computed simultaneously for each partition.

Similarly, for $S_{i,j}^{(3)}$, we have

$$
\begin{aligned}
S_{i,j}^{(3)} &= \sum_{\ell=1}^{L-1}\sum_{u=1}^{K} \boldsymbol{a}_i[\iota_\ell+u-1]\boldsymbol{P}_{i,j}[\iota_\ell+u]\boldsymbol{b}_j[\iota_\ell+u+1] \qquad (60)\\
&+ \sum_{u=1}^{T-(L-1)K-1} \boldsymbol{a}_i[\iota_L+u-1]\boldsymbol{P}_{i,j}[\iota_L+u]\boldsymbol{b}_j[\iota_L+u+1]\\
&= \sum_{\ell=1}^{L-1} \boldsymbol{\pi}^{\ell-1} \underbrace{\sum_{u=1}^{K} \boldsymbol{P}[{}^{\iota_\ell+u-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell+u]\boldsymbol{e}_{\boldsymbol{j}}^{\intercal}\boldsymbol{e}_{\boldsymbol{j}}\boldsymbol{P}[{}^{\ell K}_{\iota_\ell+u+1}]}_{\bar{\boldsymbol{\Omega}}_{i,j}^{(3)}[\ell,K]=\boldsymbol{\Omega}_{i,j}^{(3)}[\ell]} \mathbb{1}^{\ell+1}\\
&+ \boldsymbol{\pi}^{L-1} \underbrace{\sum_{u=1}^{T-(L-1)K-1} \boldsymbol{P}[{}^{\iota_L+u-1}_{\iota_L+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_L+u]\boldsymbol{e}_{\boldsymbol{j}}^{\intercal}\boldsymbol{e}_{\boldsymbol{j}}\boldsymbol{P}[{}^{T}_{\iota_L+u+1}]}_{\bar{\boldsymbol{\Omega}}_{i,j}^{(3)}[L,T-(L-1)K-1]=\boldsymbol{\Omega}_{i,j}^{(3)}[L]} \mathbb{1},
\end{aligned}
$$

where

$$
\bar{\boldsymbol{\Omega}}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell,z] = \sum_{u=1}^{z} \boldsymbol{P}[{}^{\iota_\ell+u-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell+u]\boldsymbol{e}_{\boldsymbol{j}}^{\intercal}\boldsymbol{e}_{\boldsymbol{j}}\boldsymbol{P}[{}^{\iota_\ell+z}_{\iota_\ell+u+1}].
$$

The next theorem provides an efficient way to compute matrices $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\Omega}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell]$ by forward-only recursions. Similar recursive computation of cumulated measures is used in [87] for hidden Markov chain fitting.

Theorem 3.1 *For $0 < z \leq K$ and $\ell < L$, as well as for $0 < z \leq T-(L-1)K$ and $\ell = L$, matrices $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,z]$ and $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell,z]$ satisfy the recursive relations*

$$
\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,z] = \bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,z-1]\boldsymbol{P}[\iota_\ell+z] + \boldsymbol{P}[{}^{(\ell-1)K+z-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}x_z\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell+z]
$$

and

$$
\bar{\boldsymbol{\Omega}}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell,z] = \bar{\boldsymbol{\Omega}}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell,z-1]\boldsymbol{P}[\iota_\ell+z] + \boldsymbol{P}[{}^{\iota_\ell+z-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell+z]\boldsymbol{e}_{\boldsymbol{j}}^{\intercal}\boldsymbol{e}_{\boldsymbol{j}}
$$

with the initial value $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,0] = \mathbf{0}$ and $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell,0] = \mathbf{0}$.

Proof. Starting with the definition of matrix $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell,z]$ and separating the last term of

the sum as

$$\begin{aligned}
\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell, z] &= \sum_{u=1}^{z} \boldsymbol{P}[\,{}^{\iota_\ell+u-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal} x_{\iota_\ell+u}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\,{}^{\iota_\ell+z}_{\iota_\ell+u}] \\
&= \sum_{u=1}^{z-1} \boldsymbol{P}[\,{}^{\iota_\ell+u-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal} x_{\iota_\ell+u}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\,{}^{\iota_\ell+z-1}_{\iota_\ell+u}] \\
&\quad + \boldsymbol{P}[\,{}^{\iota_\ell+z-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal} x_{\iota_\ell+z}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell + z] \\
&= \bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell, z-1]\boldsymbol{P}[\iota_\ell + z] + \boldsymbol{P}[\,{}^{\iota_\ell+z-1}_{\iota_\ell+1}]\boldsymbol{e}_{\boldsymbol{i}}^{\intercal} x_z\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{P}[\iota_\ell + z]
\end{aligned}$$

proves the recursive relation for matrices $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i}}^{(2)}[\ell, z]$. The relation for matrices $\bar{\boldsymbol{\Omega}}_{\boldsymbol{i,j}}^{(3)}[\ell, z]$ can be proven similarly. □

With these notations, the algorithm that takes a single pass through the data set consists of the following two phases:

1. The *parallel* computation of matrices $\boldsymbol{U}[\ell]$, $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\Omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ for $\ell = 1, \ldots, L$ is such that each partition is computed by a different parallel thread.

Algorithm 10 Pseudo-code for partition ℓ ($\ell < L$) in the one-pass algorithm.

```
1:  procedure Compute-Density-And-Omega-Matrices-for ℓ (x_u, u = (ℓ−1)K+1, ..., ℓK, r, λ, Π)
2:      U = I
3:      ∀i: Ω_i^(2) = 0
4:      ∀i, j: Ω_{i,j}^(3) = 0
5:      for z = 1 to K do
6:          M = P[(ℓ−1)K + z]
7:          ∀i: Ω_i^(2) = Ω_i^(2) M + U x_{(ℓ−1)K+z} e_i^T e_i M
8:          ∀i, j: Ω_{i,j}^(3) = Ω_{i,j}^(3) M + U e_i^T e_i M e_j^T e_j
9:          U = Normalize(UM)
10:     end for
11:     return U, Ω_i^(2), Ω_{i,j}^(3)
12: end procedure
```

Algorithm 10 provides the formal description of the steps for $\ell < L$. For partition L, the procedure differs only by the range of the **for** loop (z goes from 1 to $T - (L-1)K$) and by the fact that $\boldsymbol{\Omega}_{\boldsymbol{i,j}}^{(3)}$ is summed only up to $T - 1$. Operation *Normalize*() in line 9 applies a special scaling on the matrix to improve the numerical behavior, as detailed in Section 3.5.

2. The *serial* computation of the forward and backward likelihood vectors $\boldsymbol{\pi}^{\ell}$ and

$\mathbb{1}^\ell$, for $\ell = 1, \ldots, L$, recursively according to

$$\boldsymbol{\pi}^\ell = \begin{cases} \boldsymbol{\pi}, & \text{if } \ell = 0, \\ \boldsymbol{\pi}^{\ell-1}\boldsymbol{U}[\ell], & \text{if } \ell \geq 1, \end{cases} \text{ and } \quad \mathbb{1}^\ell = \begin{cases} \mathbb{1}, & \text{if } \ell = L+1, \\ \boldsymbol{\pi}^{\ell-1}\boldsymbol{U}[\ell], & \text{if } \ell \geq 1, \end{cases} \tag{61}$$

and the sums $S_i^{(2)}$ and $S_{i,j}^{(3)}$ for $i, j = 1, \ldots, R$ according to

$$S_i^{(2)} = \sum_{\ell=1}^{L} \boldsymbol{\pi}^{\ell-1}\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]\mathbb{1}^{\ell+1}, \; S_{i,j}^{(3)} = \sum_{\ell=1}^{L} \boldsymbol{\pi}^{\ell-1}\boldsymbol{\Omega}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell]\mathbb{1}^{\ell+1}, \tag{62}$$

from which $\boldsymbol{\lambda}_i$, $\boldsymbol{\Pi}_{i,j}$ and $\boldsymbol{\pi}_i$, are obtained by (54), (55) and (56).

For emphasizing the special step due to the fact that $\boldsymbol{\Omega}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}$ is summed only up to $T-1$, we note that in order to compute $S_i^{(1)}$ from $S_i^{(3)}$, it is necessary to compute vectors $\boldsymbol{a}[T-1]$ and $\boldsymbol{b}[T]$ that can be obtained from $\boldsymbol{\pi}^{L-1}$ and $\mathbb{1}^L$ with negligible extra cost.

The computational bottleneck of the procedure is that matrices $\boldsymbol{\Omega}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}$ must be computed for all partitions, which means that the memory complexity is $\mathcal{O}(LR^4)$, and the computational complexity is $\mathcal{O}(KR^5)$ in each thread due to the matrix-matrix multiplications in line 8 of Algorithm 10. A more detailed complexity analysis is provided below in Section 3.6. This algorithm, described formally in Algorithm 11, will be referred to as `P-1` in the sequel.

Algorithm 11 Pseudo-code of algorithm `P-1`.

1: **procedure** EM-fitting-by `P-1` $(x_u, u = 1, \ldots, T, \boldsymbol{r})$
2: $\boldsymbol{\lambda}, \boldsymbol{\Pi}$ = random initial guess
3: **while** relative change of log-likelihood $> \epsilon$ **do**
4: **parallel for** $\ell = 1$ **to** L **do**
5: Compute matrices $\boldsymbol{U}[\ell]$, $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\Omega}_{\boldsymbol{i},\boldsymbol{j}}^{(3)}[\ell]$ by Algorithm 10
6: **end parallel for**
7: **for** $\ell = 1$ **to** L **do**
8: Compute vectors $\boldsymbol{\pi}^\ell$ and $\mathbb{1}^\ell$ for $\ell = 1, \ldots, L$ based on (61)
9: **end for**
10: Compute sums $S_i^{(1)}, S_i^{(2)}, S_i^{(3)}$ and $S_{i,j}^{(3)}$ for $i, j = 1, \ldots, R$ based on (62), (57) and (58)
11: $\boldsymbol{\lambda}, \boldsymbol{\Pi}$ = new estimates based on (29), (30) and (31)
12: Compute the log-likelihood
13: **end while**
14: **return** $\boldsymbol{\lambda}, \boldsymbol{\Pi}$
15: **end procedure**

3.3. The parallel algorithm with two passes

The single pass algorithm in Section 3.2 goes through the inter-arrival times only once in each iteration. If traversing the input twice does not result in an overwhelming

extra cost, it is possible to develop a variant of the algorithm with different performance characteristics.

The main drawback of the single-pass algorithm is that the computation of matrices $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}$ and $\boldsymbol{\Omega}_{\boldsymbol{i,j}}^{(3)}$ requires matrix-matrix multiplications, and a significant amount of memory is required to store them for every partition. The main improvement of the two-pass algorithm is that these matrices are replaced by vectors during the second pass of the computation.

A single EM-iteration of the two-pass algorithm consists of the following phases:

1. The *parallel* computation of matrices $\boldsymbol{U}[\ell]$, for $\ell = 1, \ldots, L$, is based on

$$\boldsymbol{U}[\ell] = \prod_{u=1}^{K} \boldsymbol{P}[(\ell-1)K + u],$$

such that each partition is computed by a different thread simultaneously. More precisely, $\boldsymbol{U}[\ell]$ is computed according to Algorithm 12.

Algorithm 12 Pseudo-code for computing $\boldsymbol{U}[\ell]$ ($\ell < L$) in the two-pass algorithm.

```
1: procedure Compute-Density-Matrix-for ℓ (x_u, u = (ℓ − 1)K + 1, ..., ℓK, r, λ, Π)
2:     U = I
3:     for z = 1 to K do
4:         U = Normalize(U P[(ℓ − 1)K + z])
5:     end for
6:     return U
7: end procedure
```

2. The *serial* computation of the forward and backward likelihood vectors $\boldsymbol{\pi}^{\ell}$ and $\mathbb{1}^{\ell}$ for partitions $\ell = 1, \ldots, L$ recursively according to

$$\boldsymbol{\pi}^{\ell} = \begin{cases} \boldsymbol{\pi}, & \text{if } \ell = 0, \\ \boldsymbol{\pi}^{\ell-1}\boldsymbol{U}[\ell], & \text{if } \ell \geq 1, \end{cases}, \text{ and } \quad \mathbb{1}^{\ell} = \begin{cases} \mathbb{1}, & \text{if } \ell = L+1, \\ \boldsymbol{U}[\ell]\mathbb{1}^{\ell+1}, & \text{if } \ell \leq L. \end{cases} \tag{63}$$

3. The *parallel* computation of vectors $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}[\ell] = \boldsymbol{\pi}^{\ell}\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}[\ell] = \boldsymbol{\pi}^{\ell}\boldsymbol{\Omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ for $\ell = 1, \ldots, L$ is such that each partition is computed by a different thread. The details are provided in Algorithm 13.

Algorithm 13 Pseudo-code for computing $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ ($\ell < L$) in the two-pass algorithm.

1: **procedure** Compute-Omega-Vectors-for ℓ ($x_u, u = (\ell-1)K+1, \ldots, \ell K, \boldsymbol{\pi}^{\ell-1}, \boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\Pi}$)
2: $\quad \boldsymbol{u} = \boldsymbol{\pi}^{\ell-1}$
3: $\quad \forall i$: $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)} = \mathbf{0}$
4: $\quad \forall i, j$: $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)} = \mathbf{0}$
5: $\quad$ **for** $z = 1$ **to** K **do**
6: $\quad\quad \boldsymbol{M} = \boldsymbol{P}[(\ell-1)K + z]$ (based on the stored or recomputed $f_i(x_{(\ell-1)K+z})$ values)
7: $\quad\quad \forall i$: $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)} = \boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}\boldsymbol{M} + \boldsymbol{u}x_z\boldsymbol{e}_{\boldsymbol{i}}^{\mathsf{T}}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{M}$
8: $\quad\quad \forall i, j$: $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)} = \boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}\boldsymbol{M} + \boldsymbol{u}\boldsymbol{e}_{\boldsymbol{i}}^{\mathsf{T}}\boldsymbol{e}_{\boldsymbol{i}}\boldsymbol{M}\boldsymbol{e}_{\boldsymbol{j}}^{\mathsf{T}}\boldsymbol{e}_{\boldsymbol{j}}$
9: $\quad\quad \boldsymbol{u} = \mathit{Normalize}(\boldsymbol{u}\boldsymbol{M})$
10: $\quad$ **end for**
11: $\quad$ **return** $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}, \boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}$.
12: **end procedure**

Again, for partition L, the procedure differs by the range of the **for** loop and by the fact that $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}$ is summed only up to $T-1$.

4. The *serial* computation of sums $S_i^{(2)}$ and $S_{i,j}^{(3)}$ for $i, j = 1, \ldots, R$ according to

$$S_i^{(2)} = \sum_{\ell=1}^{L} \boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}[\ell]\mathbb{1}^{\ell+1},\ S_{i,j}^{(3)} = \sum_{\ell=1}^{L} \boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}[\ell]\mathbb{1}^{\ell+1}, \tag{64}$$

from which $\boldsymbol{\lambda}_i$, $\boldsymbol{\Pi}_{i,j}$ and $\boldsymbol{\pi}_i$ is obtained by (54), (55) and (56).

With this method, a computational bottleneck of the single-pass algorithm has been eliminated. Instead of matrices, only vectors $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}$ are stored and used during the second parallel pass of the computations (i.e., in phase 3), which means that the memory complexity is now $\mathcal{O}(LR^3)$, and the computational complexity in each thread is $\mathcal{O}(KR^4)$ due to the vector-matrix multiplications in line 8 of Algorithm 13. A more detailed complexity analysis is provided below in Section 3.6. This algorithm is referred to as `P-2` in the sequel; the corresponding pseudo-code is provided by Algorithm 14.

Algorithm 14 Pseudo-code of algorithm P-2.

```
 1: procedure EM-fitting-by P-2 (x_u, u = 1,...,T, r)
 2:     λ, Π = random initial guess
 3:     while relative change of log-likelihood > ε do
 4:         parallel for ℓ = 1 to L do
 5:             Compute matrices U[ℓ] by Algorithm 12
 6:         end parallel for
 7:         for ℓ = 1 to L do
 8:             Compute vectors π^ℓ and 𝟙^ℓ for ℓ = 1,...,L based on (63)
 9:         end for
10:         parallel for ℓ = 1 to L do
11:             Compute vectors ω_i^(2)[ℓ], ω_{i,j}^(3)[ℓ] for i, j = 1,...,R by Algorithm 13
12:         end parallel for
13:         Compute sums S_i^(1), S_i^(2), S_i^(3) and S_{i,j}^(3) for i, j = 1,...,R based on (64), (57) and (58)
14:         λ, Π = new estimates based on (29), (30) and (31)
15:         Compute the log-likelihood
16:     end while
17:     return λ, Π
18: end procedure
```

The single-pass algorithm evaluates the branch densities $f_i(x_u)$ (see (25)) only once for each inter-arrival time (when creating matrix $\boldsymbol{P}$ in line 6 of Algorithm 10), while the two-pass algorithm needs these densities twice. In the first pass (phase 1), these densities are needed to obtain likelihood matrices $\boldsymbol{U}[\ell]$, and in the second pass (phase 3), they are needed for the calculation of the $\boldsymbol{\omega}$ vectors. Hence, to save computational effort, it makes sense to compute the branch densities only once and store them in an auxiliary vector, which comes at the expense of the increased memory requirement of storing TR floating point numbers. The variant of the algorithm which computes the branch densities only once and stores them is referred to as P-2-D hereafter.

3.4. The parallel algorithm with three passes

The single pass and the two-pass algorithms are similar in spirit to each other in the sense that they recursively compute partition-based cumulated measures (matrix $\boldsymbol{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\Omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ in the one-pass algorithm, and vector $\boldsymbol{\omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ in the two-pass algorithm). The algorithm introduced in this section differs significantly. In fact, this algorithm is rather similar to the naive sequential method (Section 3.1) which does not apply recursively computed partition-based cumulated measures but computes $S_i^{(2)}[\ell]$ and $S_{i,j}^{(3)}[\ell]$ directly from the forward and backward likelihood vectors.

The phases of the algorithm are as follows:

1. The *parallel* computation of matrices $\boldsymbol{U}[\ell]$, for $\ell = 1, \ldots, L$ is based on

$$\boldsymbol{U}[\ell] = \prod_{u=1}^{K} \boldsymbol{P}[(\ell-1)K + u],$$

such that each partition is computed by a different thread simultaneously. This phase is the same as the first phase in the two-pass method.

2. The *serial* computation of the forward and backward likelihood vectors $\boldsymbol{\pi}^{\ell}$ and $\mathbb{1}^{\ell}$ for the partitions $\ell = 1, \ldots, L$ proceeds according to recursions

$$\boldsymbol{\pi}^{\ell} = \begin{cases} \boldsymbol{\pi}, & \text{if } \ell = 0, \\ \boldsymbol{\pi}^{\ell-1}\boldsymbol{U}[\ell], & \text{if } \ell \geq 1, \end{cases} \text{ and } \quad \mathbb{1}^{\ell} = \begin{cases} \mathbb{1}, & \text{if } \ell = L+1, \\ \boldsymbol{U}[\ell]\mathbb{1}^{\ell+1}, & \text{if } \ell \leq L. \end{cases} \tag{65}$$

3. The *parallel* computation of the individual likelihood vectors $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ belonging to partition ℓ for $\ell = 1, \ldots, L$ is based on

$$\boldsymbol{a}[u] = \boldsymbol{\pi}^{\ell}\boldsymbol{P}[\,{}_{(\ell-1)K+1}^{u}\,], \text{ and } \boldsymbol{b}[u] = \boldsymbol{P}[\,{}_{u}^{\ell K}\,]\mathbb{1}^{\ell+1}, \tag{66}$$

such that each partition is computed by a different thread. Again, for the last partition L, the procedure differs slightly by the number of samples. More precisely, $\boldsymbol{a}[u]$ for $u = (\ell-1)K+1, \ldots, \ell K$ is computed as in Algorithm 15, and $\boldsymbol{b}[u]$ is computed in a similar manner with backward iteration.

Algorithm 15 Pseudo-code for computing $\boldsymbol{a}[u]$ ($\ell < L$, $u = (\ell-1)K+1, \ldots, \ell K$) in the three-pass algorithm.

```
1: procedure Compute-Forward-Likelihood-Vectors-for ℓ (x_u, u = (ℓ − 1)K + 1, ..., ℓK, π^{ℓ−1}, r, λ, Π)
2:     a[0] = π^{ℓ−1}
3:     for z = 1 to K do
4:         a[z] = Normalize(a[z − 1]P[(ℓ − 1)K + z]) (based on the stored or recomputed f_i(x_{(ℓ−1)K+z}) values)
5:     end for
6:     return a.
7: end procedure
```

4. The *parallel* computation of the partition related partial sums $S_i^{(2)}[\ell]$ and $S_{i,j}^{(3)}[\ell]$ proceeds according to

$$\begin{aligned} S_i^{(2)}[\ell] &= \sum_{u=1}^{K} \boldsymbol{a}_i[(\ell-1)K+u-1]x_{(\ell-1)K+u}\boldsymbol{b}_i[(\ell-1)K+u], \\ S_{i,j}^{(3)}[\ell] &= \sum_{u=1}^{K} \boldsymbol{a}_i[(\ell-1)K+u-1]\boldsymbol{P}_{i,j}[(\ell-1)K+u]\boldsymbol{b}_j[(\ell-1)K+u+1], \end{aligned} \tag{67}$$

using the stored $\boldsymbol{a}[u]$, $\boldsymbol{b}[u]$ values.

5. The *serial* computation of the sums $S_i^{(2)}$ and $S_{i,j}^{(3)}$ proceeds according to

$$S_i^{(2)} = \sum_{\ell=1}^{L} S_i^{(2)}[\ell], \text{ and } S_{i,j}^{(3)} = \sum_{\ell=1}^{L} S_{i,j}^{(3)}[\ell], \tag{68}$$

from which $\boldsymbol{\lambda}_i$, $\boldsymbol{\Pi}_{i,j}$ and $\boldsymbol{\pi}_i$ are obtained by (54), (55), and (56).

The computational bottleneck of this procedure is phase 1 which consists of matrix-matrix multiplications of size R leading to $\mathcal{O}(KR^3)$ for each parallel thread. However, the memory consumption increases significantly compared to the previously described `P-1` and `P-2` algorithms. Vectors $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ need to be stored for each inter-arrival time that gives $2TR$ floating point numbers. We should note that the memory consumption of neither `P-1` nor `P-2` is proportional to T in a direct way; they are proportional only to the number of partitions L.

Algorithm 16 Pseudo-code of algorithm `P-3`.

procedure EM-fitting-by `P-3` ($x_u, u = 1, \ldots, T, \boldsymbol{r}$)
  $\boldsymbol{\lambda}, \boldsymbol{\Pi}$ = random initial guess
  while relative change of log-likelihood $> \epsilon$ **do**
    parallel for $\ell = 1$ **to** L **do**
      Compute matrices $\boldsymbol{U}[\ell]$ by Algorithm 12
    end parallel for
    for $\ell = 1$ **to** L **do**
      Compute vectors $\boldsymbol{\pi}^\ell$ and $\mathbb{1}^\ell$ for $\ell = 1, \ldots, L$ based on (65)
    end for
    parallel for $\ell = 1$ **to** L **do**
      Compute vectors $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ for $u = (\ell - 1)K + 1, \ldots, \ell K$, by Algorithm 15
    end parallel for
    parallel for $\ell = 1$ **to** L **do**
      Compute partition sums $S_i^{(2)}[\ell]$ and $S_{i,j}^{(3)}[\ell]$ according to (67)
    end parallel for
    Compute sums $S_i^{(1)}, S_i^{(2)}, S_i^{(3)}$ and $S_{i,j}^{(3)}$ for $i, j = 1, \ldots, R$ based on (64), (57) and (58)
    $\boldsymbol{\lambda}, \boldsymbol{\Pi}$ = new estimates based on (29), (30) and (31)
    Compute the log-likelihood
  end while
  return $\boldsymbol{\lambda}, \boldsymbol{\Pi}$
end procedure

This procedure consists of three parallel phases through the data set (phase 1, phase 3 and phase 4 are parallel), hence we refer to it as the three-pass algorithm, denoted as `P-3`, and describe it in Algorithm 16. In this procedure, the branch densities need to be evaluated three times (in phases 1, 3 and 4), thus the execution time can benefit from computing the branch densities once and storing them similarly to the two-pass algorithm. This variant of the algorithm called `P-3-D` has an even higher memory requirement.

3.5. Numerical techniques

We overview two techniques to address numerical issues which occur especially when less precise number representations are used (i.e., single precision fixed point number, or a 'float' in C++ programming language), and/or when long sequences of multiplication and/or addition operations are performed.

The first technique is used to overcome numerical issues when likelihood vectors/matrices are computed (for MAP, TMAP fitting). We note that all entries of these matrices are non-negative, and their elements are only multiplied and added during the computation. In order to avoid underflow and overflow problems of floating point numbers, we represent each matrix of size $n \times m$ by $nm + 1$ values in the form of

$$\boldsymbol{A} = \mathring{\boldsymbol{A}} \cdot 2^{\dot{a}} = \left\{ \mathring{\boldsymbol{A}}_{i,j} \right\} \cdot 2^{\dot{a}},$$

and we say that the representation is normalized when $0.5 < \max_{i,j} \mathring{\boldsymbol{A}}_{i,j} \leq 1$ holds.

The multiplication with such a matrix representation is straightforward since the element (i, j) of matrix $\boldsymbol{AB}$ is computed as

$$\{\boldsymbol{AB}\}_{i,j} = \sum_k \mathring{\boldsymbol{A}}_{i,k} \mathring{\boldsymbol{B}}_{k,j} \cdot 2^{\dot{a}+\dot{b}} = \mathring{\boldsymbol{C}}_{i,j} \cdot 2^{\dot{c}}.$$

If the representations of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are normalized, then the resulting $\mathring{\boldsymbol{C}}_{i,j} = \sum_k \mathring{\boldsymbol{A}}_{i,k} \mathring{\boldsymbol{B}}_{k,j}$, $\dot{c} = \dot{a} + \dot{b}$ representation does not necessarily satisfy $0.5 < \max_{i,j} \left(\mathring{\boldsymbol{C}}_{i,j} \right) \leq 1$, i.e., the results are not necessarily normalized.

The same applies for matrix summation. For computing matrix $\boldsymbol{A} + \boldsymbol{B}$, let $\hat{c} = \max\left(\dot{a}, \dot{b}\right)$ be the biggest of the two exponents.

Then, element (i, j) of matrix $\boldsymbol{A} + \boldsymbol{B}$ is computed as

$$\{\boldsymbol{A} + \boldsymbol{B}\}_{i,j} = \left(\mathring{\boldsymbol{A}}_{i,j} \cdot 2^{\dot{a}-\hat{c}} + \mathring{\boldsymbol{B}}_{i,j} \cdot 2^{\dot{b}-\hat{c}} \right) \cdot 2^{\hat{c}} = \mathring{\boldsymbol{C}}_{i,j} \cdot 2^{\dot{c}}.$$

Similar to the result of matrix multiplication, the obtained representation is not necessarily normalized.

To obtain a normalized representation from any $\left\{ \mathring{\boldsymbol{C}}_{i,j} \right\} \cdot 2^{\dot{c}}$ representation, let $c_n = \left\lfloor \log_2 \left(\max_{i,j} \left(\mathring{\boldsymbol{C}}_{i,j} \right) \right) \right\rfloor$, then the normalized representation is

$$\mathring{\boldsymbol{C}}_{i,j} \leftarrow \mathring{\boldsymbol{C}}_{i,j} 2^{-1-c_n} \text{ and } \dot{c} \leftarrow \dot{c} + 1 + c_n. \tag{69}$$

We refer to this computational step as *Normalize*() in the description of the algorithms. Since n and m in the matrix dimension can be equal to one, the same scaling technique applies for vectors as well.

There is another numerical pitfall in the algorithms related to floating point summations involving terms with different orders of magnitudes. The issue occurs when the sum becomes too large and does not change when further small values are added one-by-one, even if the sum of these small values is non-negligible. The root of the problem is that when two numbers $x = \mathring{x} \cdot 2^{\dot{x}}$ and $y = \mathring{y} \cdot 2^{\dot{y}}$ having different orders of magnitudes are added together, some precision is lost. If $x > y$, the exponent of the sum $z = x + y = \mathring{z} \cdot 2^{\dot{z}}$ will be $\dot{z} = \dot{x}$, and the mantissa will be $\mathring{z} = \mathring{x} + \mathring{y} \cdot 2^{-(\dot{x}-\dot{y})}$. Since the number of bits available to represent mantissa $\mathring{z}$ is fixed, $\dot{x} - \dot{y}$ precious bits will be lost from $\mathring{y}$.

To improve the accuracy of such numerically sensitive summations, we apply the following simple solution. When a new number y is to be added to the sum x, the loss of precision is calculated as $\phi = |\log_2 x/y|$. When ϕ is greater than a predefined threshold, y is not added to x. Instead, a new accumulator variable is created (initialized to zero), and the further terms (including y) are added to the newly created accumulator variable as long as the precision loss ϕ is below the threshold. In the end, all the accumulator variables are summed up.

3.6. Detailed comparison of algorithm complexities

Table 3.1. The memory consumption of algorithms measured in floating point values to store.

Data	`SERIAL`	`P-1`	`P-2`	`P-3`
parameters $\mathbf{\Pi}$, $\boldsymbol{r}$, $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$	$R^2 + 3R$	$R^2 + 3R$	$R^2 + 3R$	$R^2 + 3R$
inter-arrival times x_u	T	T	T	T
sums $S^{(3)}_{i,j}$, $S^{(n)}_i, n=1,2,3$	$R^2 + 3R$	$R^2 + 3R$	$R^2 + 3R$	$R^2 + 3R$
vectors $\boldsymbol{\pi}^\ell$, $\mathbb{1}^\ell$		$2LR(+2L)$	$2LR(+2L)$	$2LR(+2L)$
matrices $\boldsymbol{U}[\ell]$		$LR^2(+L)$	$LR^2(+L)$	$LR^2(+L)$
partial sums		$\mathbf{\Omega}^{(2)}_{\boldsymbol{i}}[\ell]$: $LR^3(+L)$	$\boldsymbol{\omega}^{(2)}_{\boldsymbol{i}}[\ell]$: $LR^2(+L)$	$S^{(2)}_i[\ell]$: $LR(+L)$
partial sums		$\mathbf{\Omega}^{(3)}_{\boldsymbol{i,j}}[\ell]$: $LR^4(+L)$	$\boldsymbol{\omega}^{(3)}_{\boldsymbol{i,j}}[\ell]$: $LR^3(+L)$	$S^{(3)}_{i,j}[\ell]$: $LR^2(+L)$
vectors $\boldsymbol{a}[u]$, $\boldsymbol{b}[u]$	$2TR(+2T)$			$2TR(+2T)$
branch densities $f_i(x)$	TR		`P-2-D` only: TR	`P-3-D` only: TR
overall complexity	$\mathcal{O}(TR)$	$\mathcal{O}\left(T + LR^4\right)$	$\mathcal{O}\left(T + LR^3\right)$	$\mathcal{O}\left(TR + LR^2\right)$

Table 3.1 depicts the memory consumption of the algorithms presented in Sections 3.1 - 3.4. In case of the serial forward-backward (`SERIAL`) method (Section 3.1), the branch density vectors and the likelihood vectors occupy the majority of the memory space. The $+2T$ term in the parentheses corresponds to the scaling exponents ($\dot{a}$) introduced in Section 3.5. Similarly, the memory requirement of the scaling exponents is given in parentheses in the consecutive columns.

In case of the single-pass method (Section 3.2), on the one hand, storing matrices $\mathbf{\Omega}_{\boldsymbol{i}}^{(2)}[\ell]$ and $\mathbf{\Omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$ for each partition can be very memory consuming when R and L are large; on the other hand, the only component that directly depends on T is the space occupied by the inter-arrival times. Interestingly, as $R << L << T$ typically holds, `P-1` algorithm needs much less memory than the `SERIAL` algorithm (and, at the same time, as shown later, it provides much better execution speed).

Compared to the `P-1` algorithm, the memory requirement of the `P-2` algorithm (Section 3.3) is identical in all the components except the partial sums whose memory consumption reduces by a factor of R. The memory consumption of the `P-2-D` variant, which is expected to improve on the running times by caching the branch densities, is still better than the one of the `SERIAL` algorithm (in case of the typical setting: $R << L << T$).

Methods `P-3` and `P-3-D` (introduced in Section 3.4) are the most memory intensive ones among the parallel algorithms due to the storage of the likelihood vectors, $\boldsymbol{a}[u]$ and $\boldsymbol{b}[u]$ for $u = 1, \ldots, T$. In the `P-3-D` variant, the branch densities increase the memory consumption even further.

The comparison of the computational complexities is more difficult because the algorithms are composed of multiple parallel and serial computation steps. For a reasonable comparison, we divide the parallel executed computation steps by the number of parallel threads. This approach might be inaccurate in some specific computational environments, e.g., with a high overhead of parallel execution, but, without focusing on a particular computational infrastructure, we resort to the assumption that parallel execution with L threads is exactly L times faster than the serial one.

The `orders` of the complexity of the algorithms are summarized in Table 3.2, where the rightmost column refers to serial (S) and parallel execution in pass one (P1), two (P2) and three (P3). The table displays only the order of the highest order terms.

Table 3.2. The `order` of computational complexity of algorithms measured in floating point multiplications.

Data to compute	`SERIAL`	`P-1`	`P-2-D`	`P-3-D`	
branch densities $f_i(x)$	RT	RT/L	RT/L	RT/L	P1
matrices $\boldsymbol{U}[\ell]$		R^3T/L	R^3T/L	R^3T/L	
partial sums		$\mathbf{\Omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$: R^5T/L			
vectors $\boldsymbol{\pi}^{\ell}$, $\mathbb{1}^{\ell}$		R^2L	R^2L	R^2L	S
branch densities $f_i(x)$			`P-2` only: RT/L	`P-3` only: RT/L	P2
partial sums			$\boldsymbol{\omega}_{\boldsymbol{i,j}}^{(3)}[\ell]$: R^4T/L		
$\boldsymbol{a}[u]$, $\boldsymbol{b}[u]$	R^2T			R^2T/L	
branch densities $f_i(x)$				`P-3` only: RT/L	P3
partial sums				$S_{i,j}^{(3)}[\ell]$: R^2T/L	
$\mathbf{\Pi}$, $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$	R^2T	R^4L	R^3L		S
overall complexity	$\mathcal{O}\left(R^2T\right)$	$\mathcal{O}\left(R^5T/L\right)$	$\mathcal{O}\left(R^4T/L\right)$	$\mathcal{O}\left(R^3T/L\right)$	

Similar to the memory consumption, the relation of the three characterizing parameters R, L, and T determines the relation of the computational complexity of the algorithms. One can draw general conclusions based on the tables of memory consumption and execution time. E.g., when a sufficiently large memory is available, then the `P-3-D` method gives the lowest computation time order, but already this basic statement needs to be handled with care. For example, when R is small (e.g., $R = 2$), the difference between R^2 and R^4 might be negligible with respect to other constant terms of the execution time which remain hidden in the table of computational complexity `orders`.

However, there is an even more dominant factor of the computation time, i.e., the hardware and software implementation of the algorithms. One of the most decisive elements with this respect is the memory access times of different computing units, which is known to affect the running time significantly. The next section summarizes our numerical experiments, which essentially follows the main trends presented in this section but shows particular differences in some cases.

3.7. Numerical experiments

It is widely known that computing the likelihood for a long sequence of observations is often affected by numerical issues. The common 'trick' to overcome these issues is to compute the logarithm of the likelihood instead. As long as the observations are independent, the log-likelihood can be easily obtained by simple summation, as it was done in many EM algorithms for PH distributions including [14] and [13].

Unfortunately, in case of MAPs, the likelihood function is obtained through matrix multiplications (see (19) and (26)) whose logarithm cannot be obtained by a simple summation, and we need to cope with the arising numerical issues. Namely, due to the finite resolution of the machine representation of the floating point numbers, the large number of matrix multiplications can cause underflow or overflow.

The EM algorithm for MAP fitting is a special case of this numerical phenomenon; due to the definition of the forward and backward likelihood vectors (28), exponential functions are multiplied a large number of times (T), which makes the occurrence of underflow or overflow almost certain. A common way to overcome this numerical issue is to adopt an appropriate scaling technique (see Section 3.5).

We have implemented the algorithms in the C++ language. All the developed procedures use single precision floating point numbers for storing real values (i.e., the type of 'float' in C++). The `SERIAL` procedure is executed on the CPU (i5-4690, 3.5GHz), and the parallel procedures, in addition, utilize a GeForce GTX 1070 graphics processing unit (GPU) having $L^* = 1920$ parallel processing elements clocked at 1.5GHz and 8GB of RAM.

The GPU architecture facilitates several levels of memory having different capacities and latencies. The global memory has the largest capacity (in our case, it can

store up to 8GB of data). We use it for storing all the inter-arrival times copied from the host environment. Reading/writing the global memory is executed for 128 bytes per transaction, and these operations are cached; there are both L2 and L1 caches available. To improve the memory latency, we have implemented coalesced access where it was possible. We did not find a use for shared memory, and, since technically it is the same memory as used for L1 cache, we configured the GPU device to dedicate more memory to L1 cache instead. The fastest type of memory is the register. To utilize it, we have statically allocated arrays in the register file by using C++ parametrized templates. Our GPU device supports up to 255 registers per thread. Luckily, we have not exceeded this per-thread register usage limitation up to $R = 10$. For $R > 10$, however, register spilling might occur, which would result in the usage of the local memory; this has negative consequences for the preformance. Furthermore, to store the parameters of the ER-CHMM structure, we have used the constant memory which is cached, too. Also, to exploit instruction level parallelism (ILP), we have statically unrolled all the loops for R.

We do not provide more specific GPU related implementation details here; we just note that we did our best to optimize the algorithms for as fast execution as possible and made the source code available at `https://github.com/minbraz/parmap`.

For testing and demonstration purposes, we used a data set of $T = 50\,000\,000$ inter-arrival time samples, which we obtained by simulating a MAP. The samples were generated by an ER-CHMM with parameters

$$\boldsymbol{r} = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \end{bmatrix}, \boldsymbol{\lambda} = \begin{bmatrix} 1 & 3 & 2 & 1 & 3 \end{bmatrix}, \boldsymbol{\Pi} = \frac{1}{10} \begin{bmatrix} 1 & 2 & 1 & 4 & 2 \\ 3 & 1 & 2 & 1 & 3 \\ 2 & 3 & 1 & 1 & 3 \\ 4 & 1 & 2 & 2 & 1 \\ 2 & 2 & 1 & 3 & 2 \end{bmatrix}.$$

To point out how high this T parameter is, we note that one of the first papers on the topic ([63]) reported that the EM-based MAP fitting procedure is limited to a few thousand of samples due to the high computation effort.

A critical parameter of the procedures is L, the number of parallel threads used by the implementation. The parallel computing ability is not fully utilized when $L < L^*$, and, according to Section 3.6, the memory requirement increases with L, suggesting that $L = L^*$ is the optimal choice. However, when the amount of the available memory is not a bottleneck, there might be cases when a higher level of parallelism decreases the computation time. It is due to the pipeline executions of the parallel threads which might benefit from utilizing the computational resources while other threads are blocked. This feature is closely related to the particular HW/SW implementation, and we do not consider it in detail; we only conclude that $L = L^*$ is an almost optimal choice, and in the case of large memory resources, one can choose $L = nL^*$, where n is a small integer number.

In the rest of the section, we provide the memory consumption and the computation time for 100 iterations of all the presented algorithms for $L = L^* = 1920$, $L = 2L^* = 3840$, and $L = 4L^* = 7680$. In all of the numerical experiments, starting from the same initial guess and using the same samples, the resulted MAP produced by the different versions of the algorithm were the same up to the first 6 digits. In Tables 3.3, 3.4, 3.5, we present the highest log-likelihood values of the structures with a certain number of branches R. The log-likelihood differences across different algorithm implementations are negligible. The execution times are presented in Tables 3.6, 3.7, 3.8 and graphically depicted in Figures 3.1, 3.3, 3.5. The total memory usage is depicted in Figures 3.2, 3.4, 3.6.

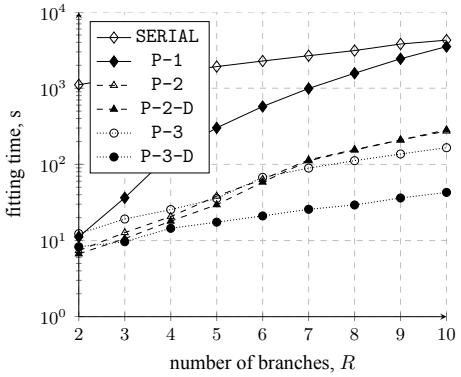

Fig. 3.1. The runtimes of MAP fitting procedures for $L = L^*$.

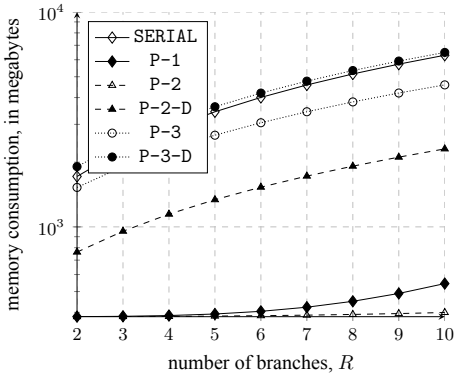

Fig. 3.2. The memory consumption of MAP fitting procedures for $L = L^*$.

Table 3.3. The best log-likelihood values of MAP fitting procedures for $L = L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	−64505468	−64505468	−64505468	−64505468	−64505468	−64505468
3	−64432232	−64432240	−64432240	−64432240	−64432212	−64432212
4	−64409492	−64409472	−64409472	−64409472	−64409500	−64409500
5	−64397848	−64397824	−64397828	−64397828	−64397852	−64397852
6	−64412172	−64412168	−64412168	−64412168	−64412168	−64412168
7	−64431896	−64431876	−64431872	−64431872	−64431892	−64431892
8	−64446364	−64446368	−64446368	−64446368	−64446368	−64446368
9	−64462516	−64462516	−64462516	−64462516	−64462516	−64462516
10	−64513032	−64530836	−64537680	−64537680	−64521580	−64521580

Table 3.4. The best log-likelihoods values of MAP fitting procedures for $L = 2L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	−64505468	−64505472	−64505468	−64505468	−64505472	−64505472
3	−64432232	−64432236	−64432236	−64432236	−64432224	−64432224
4	−64409492	−64409484	−64409488	−64409488	−64409496	−64409496
5	−64397848	−64397840	−64397840	−64397840	−64397848	−64397848
6	−64412172	−64412168	−64412168	−64412168	−64412168	−64412168
7	−64431896	−64431888	−64431888	−64431888	−64431896	−64431896
8	−64446364	−64446368	−64446368	−64446368	−64446368	−64446368
9	−64462516	−64462516	−64462516	−64462516	−64462516	−64462516
10	−64513032	−64528404	−64530128	−64530128	−64522040	−64522040

Table 3.5. The best log-likelihood values of MAP fitting procedures for $L = 4L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	−64505468	−64505468	−64505468	−64505468	−64505464	−64505464
3	−64432232	−64432232	−64432236	−64432236	−64432228	−64432228
4	−64409492	−64409492	−64409492	−64409492	−64409492	−64409492
5	−64397848	−64397848	−64397848	−64397848	−64397848	−64397848
6	−64412172	−64412168	−64412172	−64412172	−64412168	−64412168
7	−64431896	−64431892	−64431892	−64431892	−64431896	−64431896
8	−64446364	−64446368	−64446368	−64446368	−64446368	−64446368
9	−64462516	−64462516	−64462516	−64462516	−64462520	−64462520
10	−64513032	−64520404	−64527300	−64527300	−64525644	−64525644

Table 3.6. The runtimes (in seconds) of MAP fitting procedures for $L = L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	1111.521	11.178	7.379	6.682	12.292	8.281
3	1395.008	36.419	12.694	10.589	19.174	9.646
4	1605.503	137.304	20.621	17.977	25.496	14.476
5	1923.365	301.676	38.271	29.512	35.601	17.421
6	2276.253	575.146	63.607	58.477	67.553	21.042
7	2671.527	991.859	114.546	111.423	89.302	25.715
8	3122.541	1572.324	156.444	154.105	111.893	29.312
9	3803.840	2437.085	209.106	209.086	136.840	36.206
10	4278.771	3518.714	273.834	284.140	165.523	42.861

Table 3.7. The runtimes (in seconds) of MAP fitting procedures for $L = 2L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	1111.521	6.297	3.870	3.644	6.557	4.775
3	1395.008	26.018	6.539	5.611	10.033	5.607
4	1605.503	91.137	10.267	9.399	13.206	8.195
5	1923.365	201.369	48.999	38.758	18.121	10.181
6	2276.253	380.215	63.872	64.392	33.734	12.336
7	2671.527	657.038	97.559	98.059	44.649	14.734
8	3122.541	1063.828	138.394	140.038	55.413	17.078
9	3803.840	1650.562	189.625	192.288	68.378	20.859
10	4278.771	2469.702	255.107	260.547	82.898	24.186

Table 3.8. The runtimes (in seconds) of MAP fitting procedures for $L = 4L^*$.

R	SERIAL	P-1	P-2	P-2-D	P-3	P-3-D
2	1111.521	3.715	2.061	2.017	3.797	3.243
3	1395.008	29.077	4.062	3.785	5.676	4.157
4	1605.503	82.138	12.769	7.789	7.675	5.616
5	1923.365	243.007	37.551	38.056	10.319	6.979
6	2276.253	439.618	62.213	62.809	18.224	8.397
7	2671.527	778.323	100.350	95.212	40.851	9.973
8	3122.541	1230.419	144.994	139.035	54.492	12.376
9	3803.840	1665.161	191.378	196.341	66.897	14.785
10	4278.771	2489.385	255.603	268.444	81.033	20.436

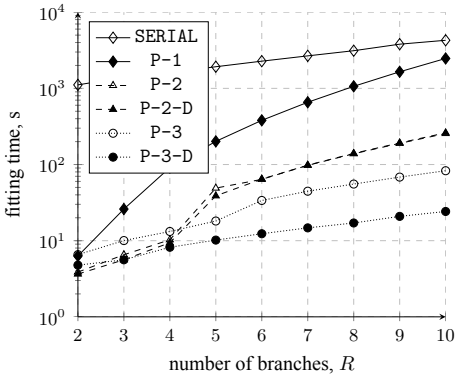

Fig. 3.3. The runtimes of procedures for $L = 2L^*$.

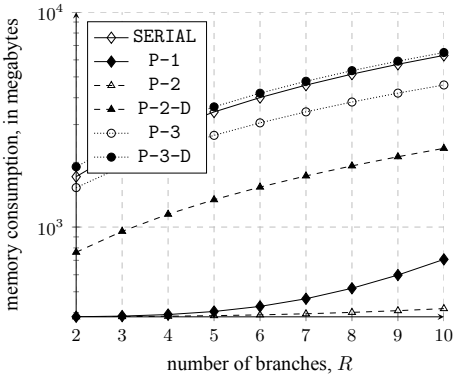

Fig. 3.4. The memory consumption of MAP fitting procedures for $L = 2L^*$.

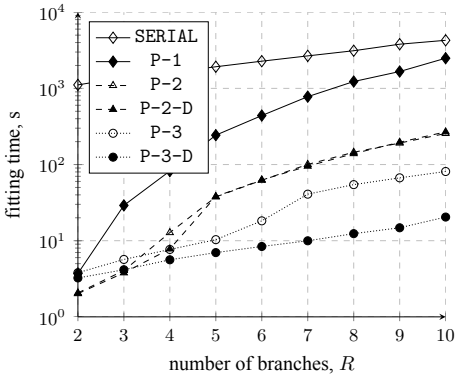

Fig. 3.5. The runtimes of procedures for $L = 4L^*$.

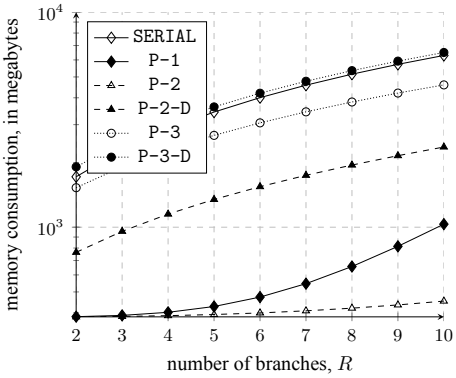

Fig. 3.6. The memory consumption of MAP fitting procedures for $L = 4L^*$.

The figures indicate the following conclusions. The speed improvement over the serial algorithm is essential (up to two orders of magnitude), the presented algorithms do benefit from the parallel hardware indeed. In line with our analysis in Section 3.6, method `P-1` slows down the fastest as R increases. Switching to vector-based operations (`P-2` and `P-2-D`) leads to better running times, especially when R is large. In our implementation and hardware environment, it turned out that storing the branch densities (method `P-2-D`) does not lead to further improvement. When R is large, the fastest procedures are the 3-pass methods (`P-3` and `P-3-D`). In this case, storing the branch densities leads to a significant improvement.

According to Section 3.6, methods `P-2-D` and `P-3-D` need extra memory to store the branch densities, while `P-3` and `P-3-D` need a large amount of memory to store the forward and backward likelihood vectors for every sample of the trace. Our numerical experiments confirm these considerations and demonstrate that `P-1` and `P-2` have

the lowest memory demand, in fact, one order of magnitude lower than the SERIAL algorithm.

All the parallel procedures did profit from the higher number of partitions (L), especially when R is small. At the same time, the memory consumption for the memory efficient procedures P-1 and P-2 increased linearly with L in accordance with theoretical evaluations (see Table 3.1).

Let us inspect algorthm P-2 and P-3-D runtimes more closely for case $L = 4L^*$. These algorithms have computational complexities (Table 3.2) $\mathcal{O}\left(R^4T/L\right)$ and $\mathcal{O}\left(R^3T/L\right)$, respectively. The runtimes given in Table 3.8 can be compared with theoretical complexities by fitting functions

$$\begin{aligned} f^{(\mathrm{P}-2)} &= a^{(\mathrm{P}-2)}R^4 + b^{(\mathrm{P}-2)}, \\ f^{(\mathrm{P}-3-\mathrm{D})} &= a^{(\mathrm{P}-3-\mathrm{D})}R^3 + b^{(\mathrm{P}-3-\mathrm{D})}. \end{aligned} \tag{70}$$

The coefficients have been found using the least square method and approximately are

$$\begin{aligned} a^{(\mathrm{P}-2)} &\approx 0.0256, \qquad b^{(\mathrm{P}-2)} \approx 17.9668, \\ a^{(\mathrm{P}-3-\mathrm{D})} &\approx 0.0159, \quad b^{(\mathrm{P}-3-\mathrm{D})} \approx 4.2191. \end{aligned}$$

The algorthm P-2, P-3-D runtimes and the fitted functions $f^{(\mathrm{P}-2)}$, $f^{(\mathrm{P}-3-\mathrm{D})}$ are shown in Figures 3.7, 3.8.

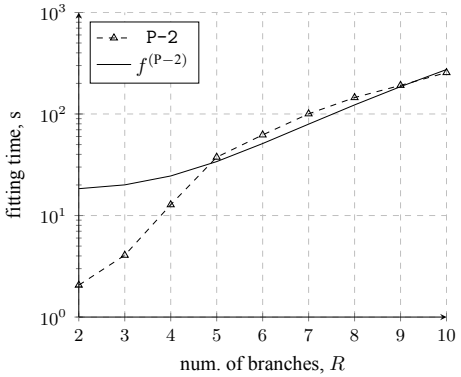

Fig. 3.7. Comparison of P-2 algorithm runtimes with its theoretical complexity for $L = 4L^*$.

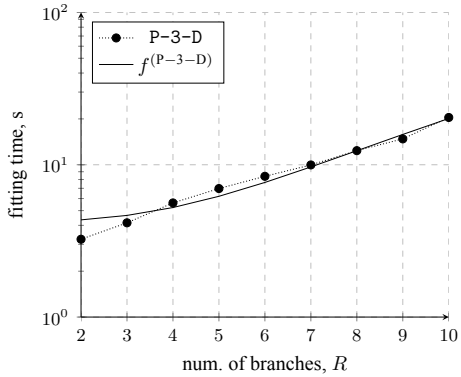

Fig. 3.8. Comparison of P-3-D algorithm runtimes with its theoretical complexity for $L = 4L^*$.

In both cases, as shown in Figures 3.7, 3.8, algorithm P-2, P-3-D runtimes converge to their theoretical complexities. However, in the case of algorithm P-2 for a lower number of branches $R = 2, 3, 4$, runtimes are significantly lower than expected. That reveals yet another advantage of algorithm P-2 lower memory usage. It is possible to allocate frequently used variables (for example, $\boldsymbol{\omega}^{(3)}_{\boldsymbol{i},\boldsymbol{j}}[\ell]$) in the register memory, which is much faster.

In general, algorithms with higher memory consumption are faster, but if T is large, the memory limitation of the hardware can be reached easily. In case the memory size poses a limitation, the more memory efficient procedures need to be applied.

Our primary forcus is on the parallel implementation of EM-based MAP fitting, but thanks to the efficient parallel implementation, we have run several fitting experiments for a reasonably large data set, and we have drawn some conclusions on the fitting properties of the EM method. These properties are very much aligned with the related properties of other EM fitting methods (e.g., EM-based Phase-type distribution fitting [14]). When the initial guess of the EM method was identical with the ER-CHMM which was used to generate the samples, the obtained likelihood value was $-6.4384612 \cdot 10^7$, and the elements of the fitted $\boldsymbol{\lambda}$ and $\boldsymbol{\Pi}$ were identical with the initial ER-CHMM in the first 4 digits. Apart from this artificial experiment, we have also executed fitting experiments with a general initial guess for all ER-CHMM structures with 10 states, i.e., where $\boldsymbol{r}$ is such that $\sum_{i=1}^{R} \boldsymbol{r}_i = 10$ for $R \geq 2$. The initial guess in these experiments was generated by drawing $\boldsymbol{\Pi}$ randomly and then setting $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i^*$ for $i = 1, \ldots, R$, where $\boldsymbol{\lambda}_i^*$ is chosen to fit the sample mean. With this general initial guess, the best likelihood is obtained by the $\boldsymbol{r} = \begin{bmatrix} 4 & 3 & 1 & 1 & 1 \end{bmatrix}$ ER-CHMM structure, and it was $-6.4397824 \cdot 10^7$; the second best likelihood was obtained by the $\boldsymbol{r} = \begin{bmatrix} 4 & 2 & 2 & 1 & 1 \end{bmatrix}$ ER-CHMM structure ($-6.4406064 \cdot 10^7$), and the third best one by the $\boldsymbol{r} = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \end{bmatrix}$ structure ($-6.4406496 \cdot 10^7$). In spite of the close likelihood values, the parameters of this best fitting $\boldsymbol{r} = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \end{bmatrix}$ structure were rather different from the ones used for generating the samples:

$$\boldsymbol{\lambda}^{(fit)} = \begin{bmatrix} 1.007464 & 0.963276 & 1.791007 & 1.301413 & 1.59442 \end{bmatrix},$$

$$\boldsymbol{\Pi}^{(fit)} = \begin{bmatrix} 0.032330 & 0.139179 & 0.395032 & 0.203579 & 0.229877 \\ 0.073530 & 0.017319 & 0.192946 & 0.401569 & 0.314634 \\ 0.121469 & 0.245083 & 0.043730 & 0.441435 & 0.148281 \\ 0.001723 & 0.068507 & 0.326094 & 0.381525 & 0.222148 \\ 0.272118 & 0.218650 & 0.108775 & 0.246126 & 0.154328 \end{bmatrix}.$$

This observation refers to an optimization problem with a flat surface and several local minima, which was also reported in previous EM fitting experiments.

For completeness, we have applied our `P-2` and `P-3-D` algorithms for fitting the Bellcore Aug89 data trace [90, 91] whose measurement is described in [91] and consists of 1000000 samples. This trace is used to investigate the MAP fitting quality using various approches in [92, 64, 93]. Since we are only interested in the fitting runtime, we do not investigate the fitting quality. The randomly generated initial MAP of ER-CHMM structure $\boldsymbol{r} = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \end{bmatrix}$ is updated for 100 iterations.

Table 3.9. The runtimes of Bellcore Aug89 trace fitting by algorithms SERIAL, P-2, P-3-D in seconds.

Algorithm	number of partitions, L	runtime	expected runtime
SERIAL	1	44.041	38.467
P-2	1920	0.831	0.765
	3840	1.112	0.98
	7680	1.457	0.751
P-3-D	1920	0.381	0.348
	3840	0.268	0.204
	7680	0.276	0.14

The expected runtime is obtained by dividing the respective runtimes from Tables 3.6, 3.7, 3.8 by 50. The speed up for algorithm P-2 is about 50 times, and for algorithm P-3-D, it is about 164 times. In all the cases, the log-likelihood of the obtained parameter estimates is 4974352.5.

3.8. Conclusions

We have presented three parallel algorithm variants for EM-based MAP fitting and provided a primary performance evaluation of them. The first two (P-1 and P-2) algorithms are developed based on the Baum-Welch algorithm which is used for searching hidden Markov model parameter estimates. Algorithm P-1 performs matrix computations while P-2 operates with vectors. Therefore, the computational complexity of algorithm P-2 is one order less, i.e., $\mathcal{O}\left(R^5T/L\right)$, compared to one of P-1, which is $\mathcal{O}\left(R^4T/L\right)$. The third algorithm P-3 computes parameter estimates directly from likelihood vectors, and its computational complexity is the lowest, i.e., $\mathcal{O}\left(TR + LR^2\right)$.

The algorithms have been compared from the perspective of memory usage as well. The algorithm P-1 memory usage complexity is $\mathcal{O}\left(T + LR^4\right)$. Algorithm P-2 requires less memory since vectors are used instead of matrices, and its complexity equals $\mathcal{O}\left(T + LR^3\right)$. Algorithm P-3 has the highest memory usage, its complexity is $\mathcal{O}\left(TR + LR^2\right)$.

Algorithm performance depends a lot on the specific hardware platform. Algorithm P-1 traverses trace data only once per iteration, which can be an advantage over other algorithms in case trace data is kept in slow access storage. Otherwise, one might consider P-2 and P-3 algorithms. Based on the complexity analysis, the strength of algorithm P-2 is its low demand for memory, and it can be fast for fitting ER-CHMM process structures with a low number of Erlang branches. Contrary, algorithm P-3 demand for memory is high; it is the fastest one for an arbitrary number

of Erlang branches. One might choose the most appropriate algorithm given hardware constraints.

In order to compare the algorithms from the practical point of view, they have been implemented for execution on GPU. For `SERIAL` algorithm, the summation technique (see Section 3.5) was applied. For all algorithm implementations, in order to avoid numerical overflow/underlow, the scaling technique (see Section 3.5) was applied. Two modifications `P-2-D` and `P-3-D` have been introduced. These modifications use an additional amount of memory for storing precalculated Erlang branch densities. The data trace of 50 000 000 inter-arrivals has been generated and used to evaluate algorithm performance. The following conclusions have been drawn from the obtained results. Algorithm `P-2-D` uses more memory compared to `P-2`, but its performance is quite similar. Meanwhile, Erlang branch density caching in algorithm `P-3-D` provided a significant speed up compared to `P-3`.

4. TRANSIENT MARKOV ARRIVAL PROCESS OF ER-CHMM STRUCTURE FITTING

The thesis in this section is based on the work presented in the conference paper 'Efficient Implementations of the EM-Algorithm for Transient Markovian Arrival Processes'.

The EM method has been successfully used for parameter estimation of several models with background Markov chains, e.g., for PH distributions [14], for PH distributions with structural restriction [13], for MAPs [63], for MAPs with structural restrictions [31, 32]. The experiences from these previous research results indicate that the inherent redundancy of the stochastic models with background Markov chains makes the parameter estimation of the general models inefficient. In this work, we avoid the implementation of the EM based estimation of general TMAPs and immediately apply a similar structural restriction as the one which turned out to be efficient in the case of PH distributions [13] and MAPs [31, 32]. The formulas of the EM method for TMAP fitting show similarities with the ones for MAP fitting in [32], but there are intricate details associated with the handling of the background process termination which require a non-trivial reconsideration of the expectation and the maximization steps of the method.

Apart from the algorithmic description of the EM method for TMAP fitting, we pay attention to efficient implementation for both traditional computing devices (CPU) and the graphics processing unit (GPU). Both platforms required various implementation optimizations for efficient computing of the steps of the fitting method. Together with the fitting results and the related computation times, we present the applied implementation optimization methods and the related considerations.

4.1. Expectation Maximization algorithm

Let us consider a process which terminates after generating a sequence of random values which might be correlated. The length of the generated sequence is of random length as well. We can interpret these values as inter-arrivals of a transient Markov arrival process. However, a single case of observed process realization is not sufficient to determine the parameters since it has partial information about only one (among many possible) evolution of the TMAP process. Thus a number of (say, U) sequences have to be considered

$$\boldsymbol{\mathcal{X}} = \left(\mathcal{X}_1^{(1)}, \mathcal{X}_2^{(1)}, ..., \mathcal{X}_{\mathcal{K}_1}^{(1)}, \ \ldots, \ \mathcal{X}_1^{(U)}, \mathcal{X}_2^{(U)}, \ldots, \mathcal{X}_{\mathcal{K}_U}^{(U)}\right),$$

where $\mathcal{K}_i$ is a random number of arrivals in the i^{th} sequence before reaching the absorbing state. Since the sequence may be short, the influence of initial probability distribution $\boldsymbol{\pi}$ is significant. Therefore, we investigate a non-stationary process (contrary to the MAP fitting) here, which is denoted by parameters $(\boldsymbol{\alpha}, \boldsymbol{D_0}, \boldsymbol{D_1})$.

We begin by assuming that $\boldsymbol{\mathcal{X}}$ is generated by the sequence of continuous time Markov processes

$$\mathcal{J}_t^{(1,1)}, ..., \mathcal{J}_t^{(1,\mathcal{K}_1)}, \ ..., \ \mathcal{J}_t^{(U,1)}, ..., \mathcal{J}_t^{(U,\mathcal{K}_U)}, \tag{71}$$

where, at the time instance $t = \mathcal{X}_1^{(1)} + ... + \mathcal{X}_k^{(u)}$, an arrival occurs, which results in switching from process $\mathcal{J}_t^{(u,k)}$ to $\mathcal{J}_t^{(u,k+1)}$. Similarly, in the case of absorption, process $\mathcal{J}_t^{(u,K_u)}$ is switched to $\mathcal{J}_t^{(u+1,1)}$ process. To simplify further investigation, let us look at (71) as two discrete time processes embedded at the transition time instances. The first such discrete time process denotes the visited state indices

$$\mathcal{I}_0^{(1,1)}, ..., \mathcal{I}_{\mathcal{M}^{(1,1)}-1}^{(1,1)}, ..., \mathcal{I}_m^{(u,k)}, ..., \mathcal{I}_{\mathcal{M}^{(U,\mathcal{K}_U)}-1}^{(U,\mathcal{K}_U)} \tag{72}$$

and the discrete time process of sojourn times spent in every state visited is

$$\mathcal{S}_0^{(1,1)}, ..., \mathcal{S}_{\mathcal{M}^{(1,1)}-1}^{(1,1)}, ..., \mathcal{S}_m^{(u,k)}, ..., \mathcal{S}_{\mathcal{M}^{(U,\mathcal{K}_U)}-1}^{(U,\mathcal{K}_U)}, \tag{73}$$

where $\mathcal{M}^{(u,k)}$ is a random number of transitions in the background process before an arrival event (i.e., transition according to rate matrix $\boldsymbol{D_1}$) occurs.

Thus full information $\boldsymbol{\mathcal{Y}}$ about how inter-arrivals $\boldsymbol{x} = \left(x_1^{(1)}, ..., x_{K_U}^{(U)}\right)$ have been generated is

$$\boldsymbol{y} = \begin{pmatrix} i_0^{(1,1)}, ..., i_{m^{(1,1)}-1}^{(1,1)}, ..., i_m^{(u,k)}, ..., i_{m^{(U,K_U)}-1}^{(U,K_U)}, \\ s_0^{(1,1)}, ..., s_{m^{(1,1)}-1}^{(1,1)}, ..., s_m^{(u,k)}, ..., s_{m^{(U,K_U)}-1}^{(U,K_U)} \end{pmatrix}.$$

The sufficient characteristics of full information $\boldsymbol{\mathcal{Y}}$ for parameter fitting would be the number of transitions from state i to state j, the number of visits in each state, etc., similarly as in case of Phase-type fitting (see Section 1.3).

However, to simplify further derivation, we apply the ER-CHMM [32, 13, 31] structural restriction.

In addition, from process (72), we derive a discrete process

$$\mathcal{G}_1^{(1)}, \ldots, \mathcal{G}_{K_1}^{(1)}, \ldots, \mathcal{G}_k^{(u)}, \ldots, \mathcal{G}_{K_U}^{(U)},$$

where $\mathcal{G}_k^{(u)}$ is the index of an Erlang branch which generates inter-arrival $\mathcal{X}^{(}u)_k$.

If $\boldsymbol{g} = \left(g_1^{(1)}, g_2^{(1)}, ..., g_{K_U}^{(U)}\right)$ were known for the specific inter-arrivals realization $\boldsymbol{x} = \left(x_1^{(1)}, x_2^{(1)}, \ldots, x_{K_U}^{(U)}\right)$, the likelihood function of parameters $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ could be expressed as

$$\mathcal{L}\left(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda} \mid \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{G}}\right) = \prod_{u=1}^{U} \prod_{k=1}^{K_u} f_{g_k^{(u)}}\left(x_k^{(u)}\right),$$

and the maximum likelihood parameter estimates could be found by [94]

$$\boldsymbol{\lambda}_i = \frac{\sum_{u=1}^{U}\sum_{k=1}^{K_u} \boldsymbol{r}_i \mathbb{1}_{\{g_k^{(u)}=i\}}}{\sum_{u=1}^{U}\sum_{k=1}^{K_u} x_k^{(u)} \mathbb{1}_{\{g_k^{(u)}=i\}}},$$

$$\boldsymbol{\Pi}_{i,j} = \frac{\sum_{u=1}^{U}\sum_{k=1}^{K_u-1} \mathbb{1}_{\{g_k^{(u)}=i, g_{k+1}^{(u)}=j\}}}{\sum_{u=1}^{U}\sum_{k=1}^{K_u} \mathbb{1}_{\{g_k^{(u)}=i\}}},$$

$$\boldsymbol{\pi}_i = \frac{1}{U}\sum_{u=1}^{U} \mathbb{1}_{\{g_1^{(u)}=i\}}.$$

However, given only observations $\boldsymbol{x}$, the information about the chosen Erlang branches $\boldsymbol{g} = \left(g_1^{(1)}, \ldots, g_k^{(u)}, \ldots, g_{K_U}^{(U)}\right)$ is unknown. Yet, we can estimate the distribution of the index values of each $\mathcal{G}_k^{(u)}$, which suggests the sufficient characteristics

$$\boldsymbol{\mathcal{H}} = \left\{\mathcal{Q}_i^{(u)}[k], \mathcal{Q}_{i,j}^{(u)}[k]\right\},$$

where $\mathcal{Q}_i^{(u)}[k]$ is an indicator of $\mathcal{X}_k^{(u)}$ being generated by Erlang branch i

$$\mathcal{Q}_i^{(u)}[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(u,k)}=\boldsymbol{s}_i\right\}},$$

$\mathcal{Q}_{i,j}^{(u)}[k]$ is an indicator of choosing Erlang branch j given $\mathcal{X}_k^{(u)}$ is generated by Erlang branch i

$$\mathcal{Q}_{i,j}^{(u)}[k] = \mathbb{1}_{\left\{\mathcal{I}_0^{(u,k)}=\boldsymbol{s}_i,\, \mathcal{I}_0^{(u,k+1)}=\boldsymbol{s}_j\right\}}.$$

Before giving the formulas for computing expectation $\boldsymbol{h}$ of sufficient characteristics $\boldsymbol{\mathcal{H}}$, we introduce likelihood vectors. After an arrival (or initially), only the first states of each Erlang branch can be entered. Therefore, we only need to consider the likelihood of states $\boldsymbol{s} = (s_1, \ldots, s_R)$ being chosen. The forward $\boldsymbol{a}^{(u)}[k]$ and backward $\boldsymbol{b}^{(u)}[k]$ vectors can be expressed as

$$\boldsymbol{a}^{(u)}[k] = \boldsymbol{\pi}\boldsymbol{P}^{(u)}\left[\begin{smallmatrix} k \\ 1 \end{smallmatrix}\right], \quad \boldsymbol{b}^{(u)}[k] = \boldsymbol{P}^{(u)}\left[\begin{smallmatrix} K_u \\ k \end{smallmatrix}\right]\boldsymbol{d}, \tag{74}$$

where $\boldsymbol{P}^{(u)}\left[\begin{smallmatrix} s \\ l \end{smallmatrix}\right]$ is a density matrix of the inter-arrival sequence $x_l^{(u)}, ..., x_s^{(u)}$ defined as

$$\boldsymbol{P}^{(u)}\left[\begin{smallmatrix} s \\ l \end{smallmatrix}\right] = \prod_{k=l}^{s} \boldsymbol{P}^{(u)}[k],$$

where $\boldsymbol{P}^{(u)}[k]$ is a density matrix associated with the observed inter-arrival $x_k^{(u)}$ and is

$$\boldsymbol{P}^{(u)}[k] = \begin{bmatrix} f_1\left(x_k^{(u)}\right)\boldsymbol{\Pi}_{1,1} & \dots & f_1\left(x_k^{(u)}\right)\boldsymbol{\Pi}_{1,R} \\ \vdots & \ddots & \vdots \\ f_R\left(x_k^{(u)}\right)\boldsymbol{\Pi}_{R,1} & \dots & f_R\left(x_k^{(u)}\right)\boldsymbol{\Pi}_{R,R} \end{bmatrix}.$$

Then the likelihood of parameters $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ can be written simply as

$$\mathcal{L}\left(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda} \mid \boldsymbol{\mathcal{X}}\right) = \prod_{u=1}^{U} \boldsymbol{\pi}\left(\prod_{k=1}^{K_u} \boldsymbol{P}^{(u)}[k]\right)\boldsymbol{d}. \tag{75}$$

The expectation of $\boldsymbol{\mathcal{H}}$ (i.e., the E-step in the EM method) can be computed by the following formulas

$$\begin{aligned} q_i^{(u)}[k] &= \mathbb{E}[\mathcal{Q}_i^{(u)}[k] \mid \boldsymbol{\mathcal{X}}] \\ &= \mathbb{P}(\mathcal{I}_0^{(u,k)} = \boldsymbol{s}_i \mid \boldsymbol{\mathcal{X}}) = \frac{\mathbb{P}(\mathcal{I}_0^{(u,k)} = \boldsymbol{s}_i, \boldsymbol{\mathcal{X}})}{\mathbb{P}(\boldsymbol{\mathcal{X}})} \\ &= \frac{\left(\boldsymbol{a}_i^{(u)}[k-1]\boldsymbol{b}_i^{(u)}[k]\right)\prod_{v\neq u}\boldsymbol{\pi}\boldsymbol{b}^{(v)}[1]}{\prod_{v=1}^{U}\boldsymbol{\pi}\boldsymbol{b}^{(v)}[1]} \\ &= \frac{\boldsymbol{a}_i^{(u)}[k-1]\boldsymbol{b}_i^{(u)}[k]}{\boldsymbol{\pi}\boldsymbol{b}^{(u)}[1]}, \\ q_{i,j}^{(u)}[k] &= \mathbb{E}[\mathcal{Q}_{i,j}^{(u)}[k] \mid \boldsymbol{\mathcal{X}}] \\ &= \mathbb{P}(\mathcal{I}_0^{(u,k)} = \boldsymbol{s}_i, \mathcal{I}_0^{(u,k+1)} = \boldsymbol{s}_j \mid \boldsymbol{\mathcal{X}}) \\ &= \frac{\mathbb{P}(\mathcal{I}_0^{(u,k)} = \boldsymbol{s}_i, \mathcal{I}_0^{(u,k+1)} = \boldsymbol{s}_j, \boldsymbol{\mathcal{X}})}{\mathbb{P}(\boldsymbol{\mathcal{X}})} \\ &= \frac{\boldsymbol{a}_i^{(u)}[k-1]f_i\left(x_k\right)\boldsymbol{\Pi}_{i,j}\boldsymbol{b}_j^{(u)}[k+1]}{\boldsymbol{\pi}\boldsymbol{b}^{(u)}[1]}. \end{aligned} \tag{76}$$

Now, given expectations $\boldsymbol{h} = \left\{q_i^{(u)}[k], q_{i,j}^{(u)}[k]\right\}$, which is sufficient to characterize full (unobserved) information $\boldsymbol{y}$, the maximum likelihood parameter estimates

(i.e., the M-step in the EM method) can be found by

$$\boldsymbol{\lambda}_i = \frac{\boldsymbol{r}_i \sum_{u=1}^{U} \sum_{k=1}^{K_u} q_i^{(u)}[k]}{\sum_{u=1}^{U} \sum_{k=1}^{K_u} x_k^{(u)} q_i^{(u)}[k]}, \tag{77}$$

$$\boldsymbol{\Pi}_{i,j} = \frac{\sum_{u=1}^{U} \sum_{k=1}^{K_u - 1} q_{i,j}^{(u)}[k]}{\sum_{u=1}^{U} \sum_{k=1}^{K_u - 1} q_i^{(u)}[k]}, \tag{78}$$

$$\boldsymbol{\pi}_i = \frac{1}{U} \sum_{u=1}^{U} q_i^{(u)}[1]. \tag{79}$$

The transient Markov arrival process parameter estimation while using the EM method is summarized in Algorithm 17.

Algorithm 17 Transient Markov arrival process with ER-CHMM structure parameter estimation by EM method.

procedure TMAP-EM-fitting ($\boldsymbol{x}$, $\boldsymbol{r}$)
    initial parameter $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ of structure $\boldsymbol{r}$ generation by Algorithm 18
    computation of parameters $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ likelihood by (75)
    repeat
        computation of expectation of sufficient characteristics $\boldsymbol{h}$ by (76)
        maximum likelihood estimate parameter $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ computation by (77), (78), (79)
        computation of parameters $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ likelihood by (75)
    until likelihood keeps increasing
    return $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$
end procedure

In this case, we simply set (by Algorithm 18) the initial parameters so that the process mean inter-arrival and mean sequence length match the given ones $\mathbb{E}[\mathcal{X}]$ and $\mathbb{E}[\mathcal{K}]$, respectively.

Algorithm 18 Algorithm for generating initial parameters of non-stationary transient MAP process of ER-CHMM structure.

procedure Transient-ErChmm-parameter-generation ($\boldsymbol{r}$, $\mathbb{E}[\mathcal{X}]$, $\mathbb{E}[\mathcal{K}]$)
    $p = \frac{1}{R}\left(1 - \frac{1}{\mathbb{E}[\mathcal{K}]}\right)$ ▷ computation of probability of generating next inter-arrival in sequence
    $\forall i : \boldsymbol{\lambda}_i = \boldsymbol{r}_i / \mathbb{E}[\mathcal{X}]$ ▷ initial Erlang branch rates
    $\forall i : \boldsymbol{\pi}_i = 1/R$ ▷ initial Erlang branch probabilities
    $\forall i, j : \boldsymbol{\Pi}_{i,j} = p$ ▷ initial Erlang branch switching probability matrix
    return $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$
end procedure

4.2. The parallel algorithm

The serial algorithm can be directly parallelized, i.e., without significantly increasing computational complexity. It can be observed that forward and backward vectors can be computed independently on separate sequences by formulas (74). Also, summations in parameter estimation formulas (77), (78), (79) can be easily parallelized by computing partial sums in separate threads which are summed to obtain new parameter estimates.

The parallel algorithm needs an additional sequences data pre-processing step, in which, each thread is assigned a list of sequences. It is preferred to balance the thread load by assigning sequences in such a way that each thread would have roughly the same number of inter-arrivals to process.

Let us assume that there are L threads available. For each thread ℓ, a list of sequences is assigned

$$\mathbf{\Phi}^{(\ell)} = (u_1, u_2, \dots),$$

where u_k is an index of certain sequences, $1 \leq u_k \leq U$. Then, partial sums can be defined as

$$\begin{aligned} S_i^{(1)}[\ell] &= \sum_{u \in \mathbf{\Phi}^{(\ell)}} \sum_{k=1}^{K_u} q_i^{(u)}[k], \quad S_i^{(2)}[\ell] = \sum_{u \in \mathbf{\Phi}^{(\ell)}} \sum_{k=1}^{K_u} x_k^{(u)} q_i^{(u)}[k], \\ S_{i,j}^{(3)}[\ell] &= \sum_{u \in \mathbf{\Phi}^{(\ell)}} \sum_{k=1}^{K_u - 1} q_{i,j}^{(u)}[k], \quad S_i^{(3)}[\ell] = \sum_{u \in \mathbf{\Phi}^{(\ell)}} \sum_{k=1}^{K_u - 1} q_i^{(u)}[k], \end{aligned} \tag{80}$$

and used for maximum likelihood parameter estimate computation by

$$\boldsymbol{\lambda}_i = \frac{\boldsymbol{r}_i \sum_{\ell=1}^{L} S_i^{(1)}[\ell]}{\boldsymbol{r}_i \sum_{\ell=1}^{L} S_i^{(2)}[\ell]}, \quad \mathbf{\Pi}_{i,j} = \frac{\sum_{\ell=1}^{L} S_{i,j}^{(3)}[\ell]}{\sum_{\ell=1}^{L} S_i^{(3)}[\ell]}. \tag{81}$$

The complete parallel procedure is given in Algorithm 19.

Algorithm 19 An algorithm for computing transient Markov arrival process parameters by EM method in parallel.

procedure TMAP-EM-parallel-fitting ($\boldsymbol{x}$, $\boldsymbol{r}$)
    initial parameter $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ of structure $\boldsymbol{r}$ generation
    parameter $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ likelihood computation by (75)
    repeat
        parallel for $\ell = 1$ **to** L **do**
            estimation of sufficient characteristics $\boldsymbol{h}$ for runs $\boldsymbol{\Phi}^{(\ell)}$ by (76)
        end parallel for
        parallel for $\ell = 1$ **to** L **do**
            computation of parial sums by (80)
        end parallel for
        computation of parameters $\boldsymbol{\Pi}$, $\boldsymbol{\lambda}$ estimates by (81), and $\boldsymbol{\pi}$ by (79)
        parameter $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$ likelihood computation by (75)
    until likelihood keeps increasing
    return $(\boldsymbol{\pi}, \boldsymbol{\Pi}, \boldsymbol{\lambda})$
end procedure

4.3. Analysis of algorithm complexities

Table 4.1. The memory consumption of algorithms measured in floating point values to store.

Data	serial	parallel
parameters $\boldsymbol{r}$, $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$, $\boldsymbol{\Pi}$	$R^2 + 3R$	$R^2 + 3R$
inter-arrival times $\boldsymbol{x}$	T	T
sequences sizes	L	L
sequences start indices	L	L
forward, backward vectors	$2TR(+2T)$	$2TR(+2T)$
branch densities $f_i\left(x_k^{(u)}\right)$	RT	RT
partial sums		$L(R^2 + 2R)$

There is not much memory usage (Table 4.1) overhead for using a parallel algorithm, only additional storage for $L(R^2 + 2R)$ partial sums to store is necessary. The overall memory complexity of both algorithms (serial and parallel) is $\mathcal{O}(TR)$.

Table 4.2. The computational complexity of algorithms measured in floating point multiplications.

Data	serial	parallel
Erlang branch densities $f_i\left(x_k^{(u)}\right)$	RT	RT/L
forward, backward vectors	TR^2	TR^2/L
parameter $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$, $\boldsymbol{\Pi}$ estimation	TR^2	
partial sums		$L(R^2+2R)$
parameter $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$, $\boldsymbol{\Pi}$ estimation using partial sums		LR^2

The serial algorithm execution complexity (Table 4.2) is $\mathcal{O}(TR^2)$ and the complexity per thread of a parallel algorithm is smaller by the factor of the number of threads L, i.e., $\mathcal{O}(TR^2/L)$. It has to be noted that, depending on a particular parallel execution architecture, an additional memory/execution overhead is likely to be present.

4.4. Numerical experiments

We have implemented serial and parallel versions of the algorithm for comparison. The first implementation is serial and uses single precision floating point numbers (i.e., float in the C++ programming language). It follows the presented Algorithm 17; in addition, to cope with the numerical issues, the scaling technique for forward/backward vectors and the summation technique for evaluating (77), (78), (79) are used (see Section 3.5).

The second implementation is parallel according to Algorithm 19, and it uses the CUDA library. It also implements the scaling technique for forward/backward vectors (see Section 3.5). However, in this implementation, the summation technique is not that relevant since the partial summation results are computed separately in each thread. This helps to avoid inaccuracies of summing up values to some extent.

The same hardware settings were used as in the experiments presented in Section 3.7. Also, we repeat the experiments for $L = kL^*$, $k = 1, 2, 4$ threads as we did in the experimental computations with MAP fitting in Section 3.7. We do not investigate the optimal choice number of threads for executing on our hardware because it would depend on the number of branches R and other complex intrinsic details, for example, how a NVidia compiler handles registers.

In the measurement of runtime, we do not include the time to allocate/deallocate memory because these operations are done once and become insignificant with the increasing number of iterations.

The run data distribution among threads is done in advance, thus it is not included in the runtime.

We have generated all the ER-CHMM structures (with $R \geq 2$) of all possible Erlang branch configurations with 10 states.

For testing and demonstration purposes, we have generated sequences data by simulating the TMAP process of the ER-CHMM structure with parameters

$$\boldsymbol{r} = \begin{bmatrix} 3 & 3 & 2 & 1 & 1 \end{bmatrix}, \boldsymbol{\lambda} = \begin{bmatrix} 1 & 3 & 2 & 1 & 3 \end{bmatrix}, \boldsymbol{\Pi} = \frac{1}{10} \begin{bmatrix} 1 & 2 & 1 & 3 & 2 \\ 3 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 1 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 & 2 \end{bmatrix}.$$

The generated data consists of $4\,996\,632$ sequences and $50\,000\,000$ inter-arrivals. The mean sequences size is 1.0, and the mean of inter-arrivals is about 1.308. The log-likelihood value of the ER-CHMM which was used to generate the data is $-7.9212363 \cdot 10^7$.

The fitting procedures have been executed for 100 iterations. The log-likelihood values are given in Table 4.3. Clearly, 100 iterations have not been enough to reach the maximal log-likelihood, but, most importantly, we see a log-likelihood value consistency across serial and parallel implementations.

The procedure runtimes are given in Table 4.4 and are graphically depicted in Figure 4.1. The achieved speed-up is small compared to the number of threads. To obtain the optimal performance, GPU threads within the same warp (i.e., a thread group) have to perform the same operation (with different data) at every moment, and the global memory access has to be coalesced in order to the hide memory operation latency. However, in our case, every thread is assigned sequences of different sizes, which makes the fulfillment of these optimization requirements very hard if possible at all.

Table 4.3. The log-likelihood values of the best solutions obtained by TMAP fitting procedures.

R	serial	parallel, $L = L^*$	parallel, $L = 2L^*$	parallel, $L = 4L^*$
2	-79611116	-79611134	-79611136	-79611134
3	-79452224	-79452257	-79452234	-79452230
4	-79425934	-79425984	-79425953	-79425941
5	-79410651	-79410690	-79410670	-79410664
6	-79443226	-79443274	-79443258	-79443239
7	-79476124	-79476131	-79476120	-79476133
8	-79516724	-79516726	-79516734	-79516719
9	-79516761	-79516762	-79516747	-79516750
10	-79683891	-79683290	-79683406	-79683404

Table 4.4. The runtimes (in seconds) of TMAP fitting procedures.

R	serial	parallel, $L = L^*$	parallel, $L = 2L^*$	parallel, $L = 4L^*$
2	2992.35	239.27	167.06	182.72
3	3779.80	296.15	216.99	295.09
4	4423.85	363.43	295.82	407.54
5	5242.63	414.14	402.48	504.83
6	6239.32	483.69	518.32	605.46
7	7598.72	567.83	634.03	701.60
8	8732.11	667.00	745.96	798.75
9	10404.82	780.53	842.25	894.20
10	12209.27	903.22	947.06	977.08

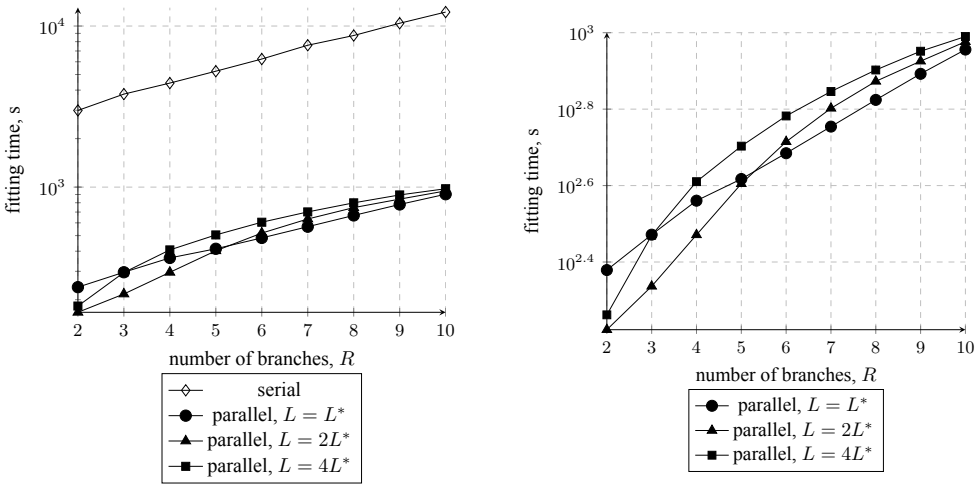

Fig. 4.1. The runtimes of TMAP fitting procedures.

4.5. Conclusions

The EM method for TMAP (of ER-CHMM structure) fitting has been derived. The serial and parallel versions of the algorithm have been implemented and tested with a generated trace data of 50 000 000 inter-arrivals. The following conclusions have been drawn:

- In order to perform a serial algorithm for a huge number of inter-arrivals, a summation technique (see Section 3.5) has to be used.
- The serial and parallel procedures have to use a scaling technique (see Section 3.5) to avoid numerical issues in the computation of forward/backward vectors.

- Our parallel procedure is not effective for execution on GPU because its threads have high divergence, and the global memory access is not coalesced.

However, the parallel procedure can be effectively used on parallel execution systems whose thread performance does not depend on a common operation sheduling.

5. FINITE REQUEST QUEUE MODEL

The work presented in this section is an extension of methodology of modeling presented in our conference paper 'Software relialibility Markovian model based on phase-type distribution'.

We formulate and implement a simple finite request queue model. Then we compute several of its characteristics when the given distributions are approximated by Phase-type distributions. The objective of this section is to compare Phase-type fitting of full and random (with $m = 2n$ transitions) structures in a more practical context.

5.1. Model description and construction

Let us describe the conceptual finite request model. We assume that there are L requests. Yet, initially, these requests are not available. Requests appear (i.e., are received) at random time intervals, which has a distribution $\mathrm{F}^{(a)}$. Once a request has been received, its serving starts and takes a random amount of time of distribution $\mathrm{F}^{(b)}$. If a request is received while the previous one is still being served, it is placed into a waiting queue (first in, first out). The queue model is shown in Figure 5.1.

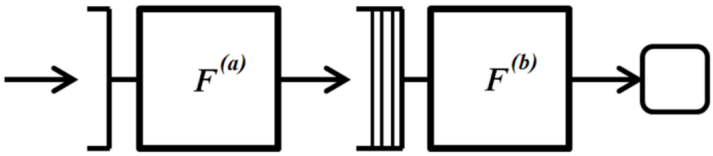

Fig. 5.1. The process of receiving and serving requests.

We approximate $\mathrm{F}^{(a)}$, $\mathrm{F}^{(b)}$ distributions by Phase-type distributions using the full structure (i.e., all the transitions between phases are enabled) and the randomly generated sparse structures (with $m = 2n$ transitions). For more on Phase-type approximation, see Section 2.

In addition, we simulated the process for checking how these results match the ones obtained from the Markovian model approximation.

The queueing system (Figure 5.1) Markovian approximation is obtained by replacing $\mathrm{F}^{(a)}$, $\mathrm{F}^{(b)}$ distributions with their Phase-type distribution approximations. The Markovian queueing system is given in Figure 5.2.

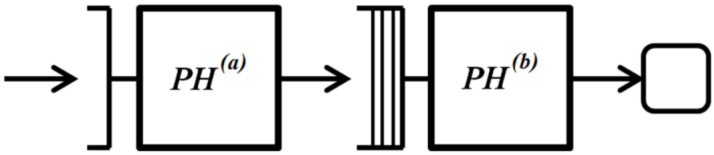

Fig. 5.2. The process of receiving and serving requests using Phase-type distributions.

The construction of the model consists of two phases. First, we generate a set of all the reachable system states, and then generate the transition rate matrix $\boldsymbol{Q}$. The modeling is performed by computing $\boldsymbol{\alpha} e^{\boldsymbol{Q}t}$ vectors, from which, based on the state's definition, desired statistical properties are extracted.

The system state is vector $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 & \boldsymbol{v}_4 \end{bmatrix}$, where $\boldsymbol{v}_1 \in \{1, \ldots, n^{(a)}\}$ denotes the index of an active phase of $\text{PH}^{(a)}$ and $\boldsymbol{v}_1 = 0$ indicates that there is any active phase in $\text{PH}^{(a)}$; $\boldsymbol{v}_2 \in \{0, 1, \ldots, L-1\}$ denotes a number of the received requests which are waiting in the queue; $\boldsymbol{v}_3 \in \{1, \ldots, n^{(b)}\}$ denotes the index of an active phase of $\text{PH}^{(b)}$, $\boldsymbol{v}_3 = 0$ indicates that there is any active phase in $\text{PH}^{(b)}$; $\boldsymbol{v}_4 \in \{0, 1, \ldots, L\}$ denotes a number of served requests. The number of requests in the system state, including the one which is being received, is the function

$$c(\boldsymbol{v}) = \mathbb{1}_{\{\boldsymbol{v}_1 > 0\}} + \boldsymbol{v}_2 + \mathbb{1}_{\{\boldsymbol{v}_3 > 0\}} + \boldsymbol{v}_4.$$

We denote the set of the reachable system states by set $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n_{\boldsymbol{\mathcal{V}}})}\}$. The ordering of the states is not relevant.

The generation of state set $\boldsymbol{\mathcal{V}}$ is given in Algorithm 20. States are generated by two events of: a) request reception (see Algorithm 21), b) request serving (i.e., an event which occurs while a request is being served, see Algorithm 22). We start with the initial state of $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ and apply these two events to it so that to produce the proceeding states which are used to generate the remaining states. At first, in the state generation algorithm, we do not account for the number of phases in $\text{PH}^{(a)}$, $\text{PH}^{(b)}$ distributions (it is assumed to be 1). These states are then used to generate the final states which account for the number of phases in Phase-type distributions and are included into system state set $\boldsymbol{\mathcal{V}}$.

Algorithm 20 Pseudo-code for generating reachable state set $\boldsymbol{\mathcal{V}}$.

procedure System-State-Set-Generation (L, $n^{(a)}$, $n^{(b)}$)
  $\boldsymbol{\mathcal{V}} := \emptyset$
  $\boldsymbol{\mathcal{V}}^* := \{\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\}$
  while $n_{\boldsymbol{\mathcal{V}}^*} > 0$ **do**
    remove one state $\boldsymbol{v}$ from $\boldsymbol{\mathcal{V}}^*$
    invoke Request-Reception-Event (v, $\boldsymbol{\mathcal{V}}^*$)
    invoke Request-Serving-Event (v, $\boldsymbol{\mathcal{V}}^*$)
    ▷ generate the states which account for $n^{(a)}$, $n^{(b)}$
    if $\boldsymbol{v}_1 = 1$ and $\boldsymbol{v}_3 = 0$ **then**
      for $i = 1$ **to** $n^{(a)}$ **do**
        insert $\begin{bmatrix}i & \boldsymbol{v}_2 & 0 & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}$
      end for
    end if
    if $\boldsymbol{v}_1 = 0$ and $\boldsymbol{v}_3 = 1$ **then**
      for $j = 1$ **to** $n^{(b)}$ **do**
        insert $\begin{bmatrix}0 & \boldsymbol{v}_2 & j & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}$
      end for
    end if
    if $\boldsymbol{v}_1 = 1$ and $\boldsymbol{v}_3 = 1$ **then**
      for $i = 1$ **to** $n^{(a)}$ **do**
        for $j = 1$ **to** $n^{(b)}$ **do**
          insert $\begin{bmatrix}i & \boldsymbol{v}_2 & j & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}$
        end for
      end for
    end if
  end while
  return $\boldsymbol{\mathcal{V}}$
end procedure

Algorithm 21 Pseudo-code for generating a reachable state associated with request reception event.

procedure Request-Reception-Event ($\boldsymbol{v}$, $n^{(b)}$, $\boldsymbol{\mathcal{V}}^*$)
  if $\boldsymbol{v}_1 = 1$ **then**
    if $\boldsymbol{v}_3 = 0$ **then** ▷ request serving can be started
      if $c(\boldsymbol{v}) = L$ **then**
        insert $\begin{bmatrix}0 & 0 & 1 & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}^*$
      else
        insert $\begin{bmatrix}1 & 0 & 1 & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}^*$
      end if
    else ▷ the received request is placed into queue
      if $c(\boldsymbol{v}) = L$ **then**
        insert $\begin{bmatrix}0 & \boldsymbol{v}_2 + 1 & 1 & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}^*$
      else
        insert $\begin{bmatrix}1 & \boldsymbol{v}_2 + 1 & 1 & \boldsymbol{v}_4\end{bmatrix}$ into $\boldsymbol{\mathcal{V}}^*$
      end if
    end if
  end if
end procedure

Algorithm 22 Pseudo-code for generating a reachable state associated with request serving event.

```
1: procedure Request-Serving-Event (v, n^(b), V*)
2:     if v_3 = 1 then
3:         if v_2 = 0 then
4:             insert [v_1  0  0  v_4 + 1] into V*
5:         else
6:             insert [v_1  v_2 − 1  1  v_4 + 1] into V*
7:         end if
8:     end if
9: end procedure
```

Next, based on system state set $\boldsymbol{\mathcal{V}}$, we construct the state transition rate matrix $\boldsymbol{Q}$, see Algorithm 23. The generation of the transition rate associated with request reception is given in Algorithm 24. The generation of the transition rate associated with request serving is given in Algorithm 25. The order of the states in the set is not significant, but, for the purpose of consistency, we keep it fixed.

Algorithm 23 Pseudo-code for generating transition rate matrix $\boldsymbol{Q}$.

```
1: procedure Transition-Rate-Matrix-Generation (PH^(a), PH^(b), V)
2:     ∀i, j: Q_{i,j} := 0
3:     for from = 0 to n_V do
4:         invoke Request-Reception-Transitions (PH^(a), v^(from), Q)
5:         invoke Request-Serving-Transitions (PH^(b), v^(from), Q)
6:     end for
7:     return Q
8: end procedure
```

Algorithm 24 Pseudo-code for generating transition rates associated with request reception event.

1: **procedure** Request-Reception-Transitions (PH$^{(a)}$, $\boldsymbol{v}^{(from)}$, $\boldsymbol{Q}$)
2: **if** $\boldsymbol{v}_1^{(from)} > 0$ **then**
3: **for** $i = 1$ **to** $n^{(a)}$ **do**
4: find $\boldsymbol{v}^{(to)} = \begin{bmatrix} i & \boldsymbol{v}_2^{(from)} & \boldsymbol{v}_3^{(from)} & \boldsymbol{v}_4^{(from)} \end{bmatrix}$
5: $\boldsymbol{Q}_{from,to} := \boldsymbol{D}_{\mathbf{0}}^{(a)}{}_{\boldsymbol{v}_1^{(from)},i}$
6: **end for**
7: **if** $c(\boldsymbol{v}^{(from)}) < L$ **then**
8: **if** $\boldsymbol{v}_3^{(from)} > 0$ **then**
9: **for** i = 0 **to** $n^{(a)}$ **do**
10: **for** j = 0 **to** $n^{(b)}$ **do**
11: find $\boldsymbol{v}^{(to)} = \begin{bmatrix} i & 0 & j & \boldsymbol{v}_4^{(from)} \end{bmatrix}$
12: $\boldsymbol{Q}_{from,to} := \boldsymbol{d}_{\mathbf{1}}^{(a)}{}_{\boldsymbol{v}_1^{(from)}} \boldsymbol{\alpha}_i^{(a)} \boldsymbol{\alpha}_j^{(b)}$
13: **end for**
14: **end for**
15: **else**
16: **for** $i = 1$ **to** $n^{(a)}$ **do**
17: find $\boldsymbol{v}^{(to)} = \begin{bmatrix} i & \boldsymbol{v}_2^{(from)} + 1 & \boldsymbol{v}_3^{(from)} & \boldsymbol{v}_4^{(from)} \end{bmatrix}$
18: $\boldsymbol{Q}_{from,to} := \boldsymbol{d}_{\mathbf{1}}^{(a)}{}_{\boldsymbol{v}_1^{(from)}} \boldsymbol{\alpha}_i^{(a)} \boldsymbol{\alpha}_j^{(b)}$
19: **end for**
20: **end if**
21: **else**
22: **if** $\boldsymbol{v}_3^{(from)} = 0$ **then**
23: **for** $j = 1$ **to** $n^{(b)}$ **do**
24: find $\boldsymbol{v}^{(to)} = \begin{bmatrix} 0 & 0 & j & \boldsymbol{v}_4^{(from)} \end{bmatrix}$
25: $\boldsymbol{Q}_{from,to} := \boldsymbol{d}_{\mathbf{1}}^{(a)}{}_{\boldsymbol{v}_1^{(from)}} \boldsymbol{\alpha}_i^{(b)} \boldsymbol{\alpha}_j^{(b)}$
26: **end for**
27: **else**
28: find $\boldsymbol{v}^{(to)} = \begin{bmatrix} 0 & \boldsymbol{v}_2^{(from)} + 1 & \boldsymbol{v}_3^{(from)} & \boldsymbol{v}_4^{(from)} \end{bmatrix}$
29: $\boldsymbol{Q}_{from,to} := \boldsymbol{d}_{\mathbf{1}}^{(a)}{}_{\boldsymbol{v}_1^{(from)}}$
30: **end if**
31: **end if**
32: **end if**
33: **end procedure**

Algorithm 25 Pseudo-code for generating transition rates associated with request serving event.

procedure Request-Serving-Transitions (PH$^{(b)}$, $\boldsymbol{v}^{(from)}$, $\boldsymbol{Q}$)
  if $\boldsymbol{v}_3^{(from)} > 0$ **then**
    for $j = 1$ **to** $n^{(b)}$ **do**
      find $\boldsymbol{v}^{(to)} = \left[\boldsymbol{v}_1^{(from)} \quad \boldsymbol{v}_2^{(from)} \quad j \quad \boldsymbol{v}_4^{(from)}\right]$
      $\boldsymbol{Q}_{from,to} := \boldsymbol{D_0^{(b)}}_{\boldsymbol{v}_3^{(from)},j}$
    end for
    if $\boldsymbol{v}_2^{(from)} > 0$ **then**
      for $j = 1$ **to** $n^{(b)}$ **do**
        find $\boldsymbol{v}^{(to)} = \left[\boldsymbol{v}_1^{(from)} \quad \boldsymbol{v}_2^{(from)} - 1 \quad j \quad \boldsymbol{v}_4^{(from)} + 1\right]$
        $\boldsymbol{Q}_{from,to} := \boldsymbol{d_1^{(b)}}_{\boldsymbol{v}_3^{(from)}} \boldsymbol{\alpha}_j^{(b)}$
      end for
    else
      find $\boldsymbol{v}^{(to)} = \left[v_1^{(from)} \quad 0 \quad 0 \quad \boldsymbol{v}_4^{(from)} + 1\right]$
      $\boldsymbol{Q}_{from,to} := \boldsymbol{d_1^{(b)}}_{\boldsymbol{v}_3^{(from)}}$
    end if
  end if
end procedure

5.2. Numerical experiments (case 1)

In this case, we chose the so-called 'well behaved' distributions (for F$^{(a)}$, F$^{(b)}$) which can be easily approximated by Phase-type distributions. This will help us to validate our modeling approach. In the following section, we will experiment with a distribution which is much more difficult to approximate to better contrast full/random structure Phase-type fitting.

We chose the Weibull distribution, whose distribution function is

$$F(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{\left(\frac{x}{\lambda}\right)^k}, x \geq 0, \\ 0, x \leq 0. \end{cases}$$

For time intervals between request reception distribution F$^{(a)}$, we chose parameters $k^{(a)} = 0.9$, $\lambda^{(a)} = 0.5$ (**W3**). To have more variety, for request serving time distribution F$^{(b)}$, we chose parameters $k^{(b)} = 2$, $\lambda^{(b)} = 0.9$ (**W4**). The latter (**W4**) distribution is less similar to exponential (because value $k^{(b)}$ is further from 1 compared to $k^{(a)}$); thus we approximate it with more phases.

Based on the research results presented in Section 2.6, in order to better benefit from random structure fitting, we introduce the following strategy. We perform the fitting procedure in a number of passes. The pass purpose is to select the most potentially good solutions (i.e., Phase-type distribution representation parameters) for further fitting. In the first pass, a number of randomly generated parameters (by Algorithm 2) are

updated for a fixed number of iterations (with the EM method) and sorted according to their likelihood value. In the second pass, the best (i.e., with the bigger likelihood value) parameters are further updated, and so on. In the final pass, the number of iterations is not limited, and updates are performed until the relative likelihood value increase becomes smaller than 10^{-8}. Such a final pass is ideal for full structure fitting which can benefit from its flexibility. Meanwhile, for randomly generated structures, a lot depends on how good a structure was selected in the first passes. For details on how full/random structure fitting differs, we refer to Section 2.6.

For approximating request reception distribution **W3**, we have chosen to use 4 phases, and for request serving distribution **W4** we use 8 phases. The summary of running passes is presented in Tables 5.1, 5.2.

Table 5.1. Passes summary of fitting **W3** by PH(4).

pass	struct.	inst. count	best llh	RND better by
#1, 1 min, 100 it.	FULL	538	$-3.485439 \cdot 10^{-1}$	$3.699660 \cdot 10^{-4}$
	RND	491	$-3.481739 \cdot 10^{-1}$	
#2, 1 min, 200 it.	FULL	261	$-3.484207 \cdot 10^{-1}$	$2.629414 \cdot 10^{-4}$
	RND	288	$-3.481578 \cdot 10^{-1}$	
#3, 1 min, 300 it.	FULL	162	$-3.483044 \cdot 10^{-1}$	$1.976435 \cdot 10^{-4}$
	RND	184	$-3.481068 \cdot 10^{-1}$	
#4, 3 min	FULL	5	$-3.480386 \cdot 10^{-1}$	$-1.756088 \cdot 10^{-5}$
	RND	23	$-3.480562 \cdot 10^{-1}$	

Table 5.2. Passes summary of fitting **W4** by PH(8).

pass	struct.	inst. count	best llh	RND better by
#1, 1 min, 100 it.	FULL	220	$-4.983716 \cdot 10^{-1}$	$6.101469 \cdot 10^{-3}$
	RND	112	$-4.922701 \cdot 10^{-1}$	
#2, 1 min, 200 it.	FULL	113	$-4.926344 \cdot 10^{-1}$	$1.888646 \cdot 10^{-3}$
	RND	86	$-4.907457 \cdot 10^{-1}$	
#3, 1 min, 300 it.	FULL	76	$-4.915799 \cdot 10^{-1}$	$1.155153 \cdot 10^{-3}$
	RND	57	$-4.904248 \cdot 10^{-1}$	
#4, 3 min	FULL	4	$-4.902161 \cdot 10^{-1}$	$3.048077 \cdot 10^{-5}$
	RND	7	$-4.901857 \cdot 10^{-1}$	

The general tendency (Tables 5.1, 5.2) of the fitting procedure using passes is the following. Sparse structures converge faster, but, when more iterations are given, the full structures start to catch up.

Next, we present some general characteristics of the Phase-type distribution approximations in Tables 5.3, 5.4.

Table 5.3. Statistical properties of **W3** distribution and its PH(4) approximations.

	mean	std.	skewness	llh
dist. **W3**	0.52609	0.58556	2.34497	N/A
trace	0.52602	0.58504	2.32717	N/A
PH(4):FULL	0.52602	0.58469	2.31704	−0.34804
PH(4):RND	0.52602	0.58363	2.28630	−0.34806

Table 5.4. Statistical properties of **W4** distribution and its PH(8) approximations.

	mean	std.	skewness	llh
dist. **W4**	0.79760	0.41693	0.63111	N/A
trace	0.79758	0.41686	0.62953	N/A
PH(8):FULL	0.79758	0.41809	0.66883	−0.49022
PH(8):RND	0.79758	0.41798	0.66656	−0.49019

The density plots of **W3**, **W4** distributions and their respective Phase-type distribution approximations are shown in Figures 5.3, 5.4.

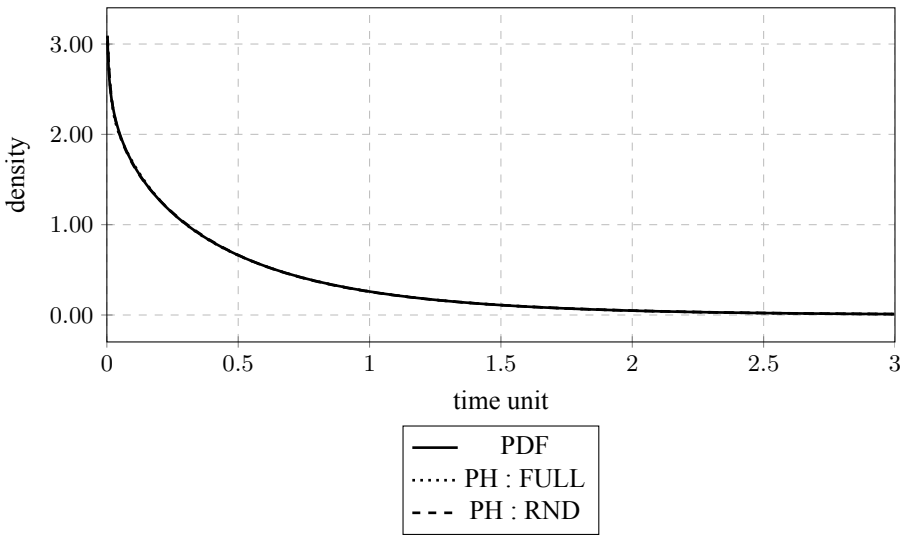

Fig. 5.3. Density graph of **W3** distribution and its Phase-type approximations.

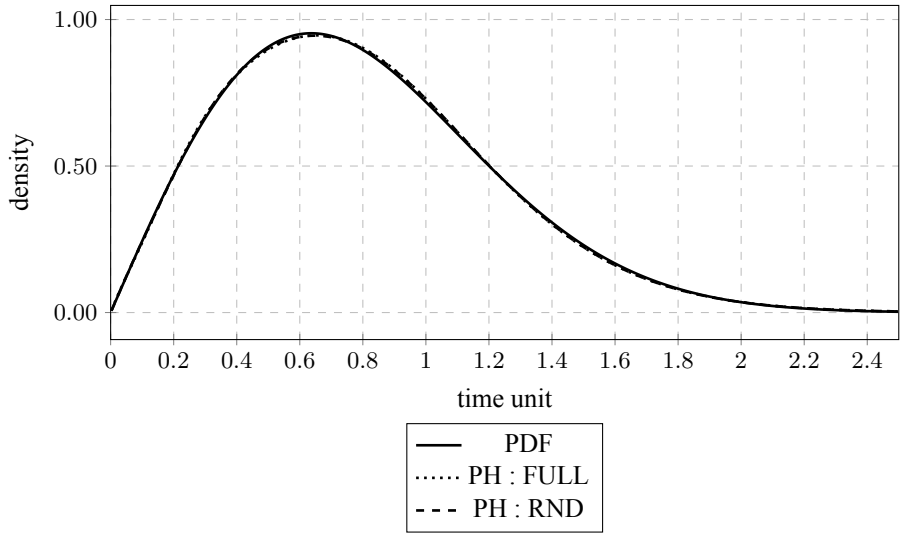

Fig. 5.4. Density graph of **W4** distribution and its Phase-type approximations.

The Phase-type approximations are fitted to distributions **W3**, **W4** fairly well.

In order to validate our Markovian model, we have implemented a simple process simulation routine. A question arises how many instances of process simulation have to be performed in order to get stable results. In order to have a clearer picture here, we have plotted how the total request serving time mean and standard deviation depends on the number of simulated process instances (Figures 5.5, 5.6). The number of total requests is $L = 3$.

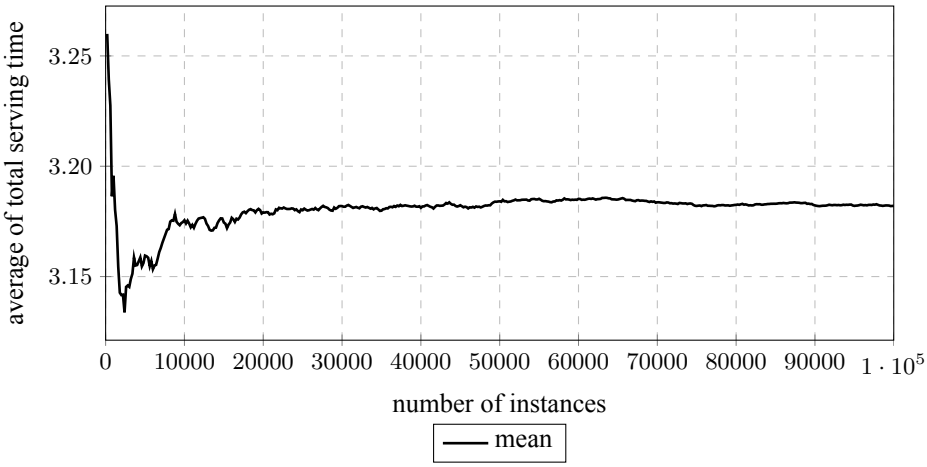

Fig. 5.5. Convergence of total serving time mean for a finite request queue model with distributions **W3**, **W4**.

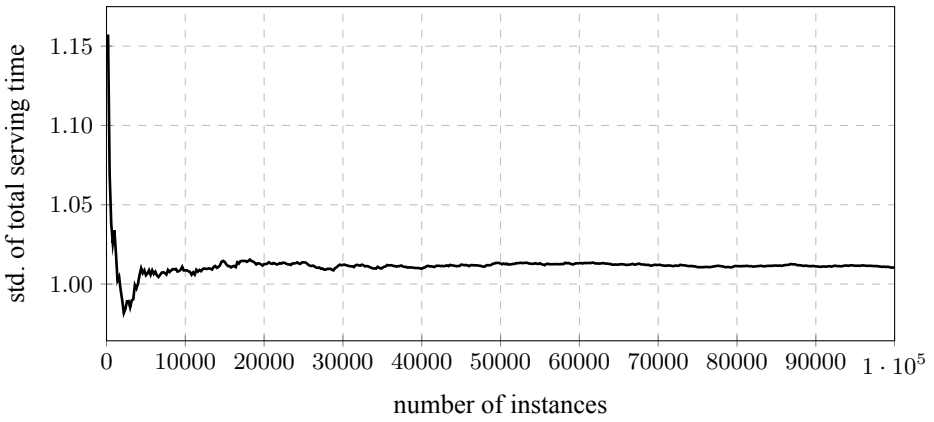

Fig. 5.6. Convergence of total serving time standard deviation for finite request queue model with **W3**, **W4** distributions.

As shown in Figures 5.5, 5.6, the convergence is not stable. This shows the importance of Markovian modeling as, for a more complex model simulation, it might not be feasible due to a large number of necessary instances to reach convergence. Let us choose to have 100000 process simulation instances for further investigation.

First we compare the average number of requests in the queue; the plot is shown in Figure 5.7.

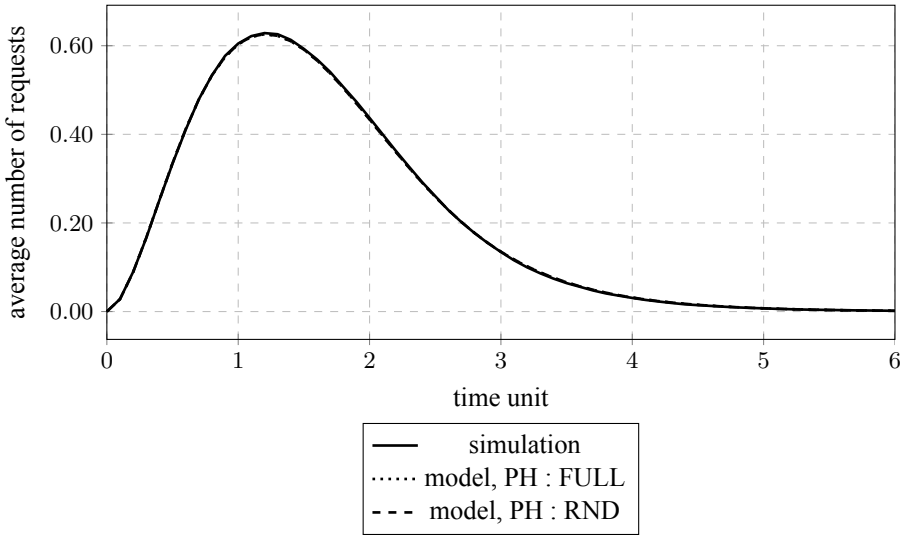

Fig. 5.7. Graph of the average number of requests in the queue for a finite queue model with distributions **W3**, **W4**.

The average simulated queue occupation nicely matches the one modeled by

the Markovian model. Another characteristic is the total time it takes to serve all the requests. These results are presented in Table 5.5.

Table 5.5. Statistical properties of completion time distribution for a finite request model with distributions **W3**, **W4**.

	mean	std.	skewness
simulation	3.18208	1.01053	0.91849
model, PH : FULL	3.18378	1.01482	0.91021
model, PH : RND	3.18368	1.01295	0.89380

Since our Markovian model is terminating CTMC, we can look at it as a huge Phase-type distribution which is a distribution of the total serving time. The density plots of such distributions alongside with the histogram obtained from the simulated data are shown in Figure 5.8.

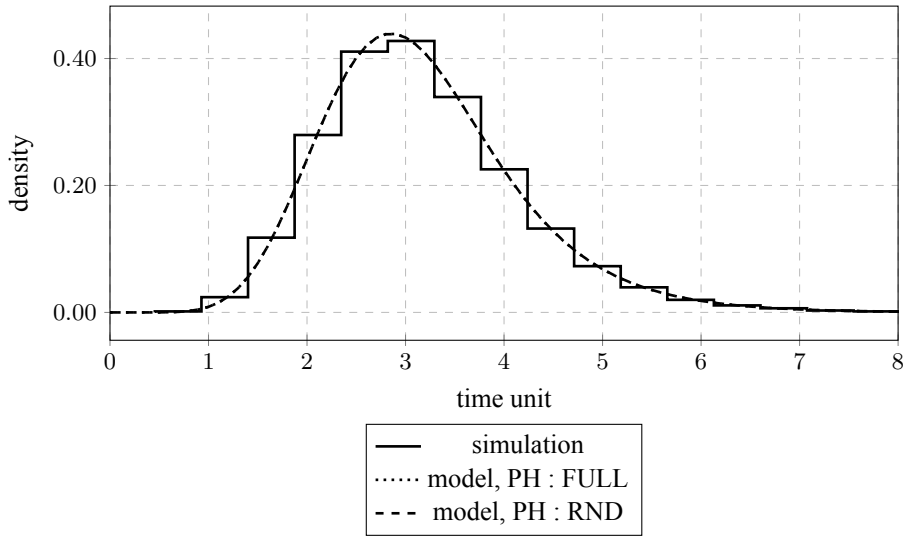

Fig. 5.8. Density of distribution of time it takes to serve all the requests for a finite queue model with distributions **W3**, **W4**.

To conclude this case, we can state that, for small numbers of phases, there is not much difference whether the fitting of full or random structure is performed. The main difference between these fitting approaches is that fitting with a set of sparse structures might help to obtain more likely parameters at first. This might be considered as an option in the situations where precision can be slightly compromised for speed. But, given enough time, the full structure might be able to give more likely parameters in the end.

5.3. Numerical experiments (case 2)

In this case, we change request reception distribution **W3** by distribution **ME**. The request serving time distribution remains the same, i.e., **W4**.

First of all, we attempt to approximate distribution **ME** with an arbitrarily chosen number of phases, 17. The summary of the fitting passes is given in Table 5.6.

Table 5.6. Passes summary of fitting **ME** by PH(17).

pass	struct.	inst. count	best llh	RND better by
#1, 10 min, 100 it.	FULL	118	$-9.509980 \cdot 10^{-1}$	$9.124475 \cdot 10^{-2}$
	RND	127	$-8.597532 \cdot 10^{-1}$	
#2, 10 min, 200 it.	FULL	59	$-9.121713 \cdot 10^{-1}$	$6.667418 \cdot 10^{-2}$
	RND	87	$-8.454972 \cdot 10^{-1}$	
#3, 10 min, 300 it.	FULL	39	$-8.782566 \cdot 10^{-1}$	$4.597987 \cdot 10^{-2}$
	RND	53	$-8.322767 \cdot 10^{-1}$	
#4, 30 min	FULL	3	$-8.184421 \cdot 10^{-1}$	$-7.391915 \cdot 10^{-3}$
	RND	6	$-8.258340 \cdot 10^{-1}$	

There (Table 5.6), the tendency observed in the previous section remains – sparse structures find better parameters during the first couple of hundreds of iterations. Yet, the full structure, due to its flexibility, finds better parameters.

The general characteristics of the Phase-type distribution approximation are given in Table 5.7.

Table 5.7. Statistical properties of **ME** distribution and its PH(17) approximations.

	mean	std.	skewness	llh
dist. **ME**	1.04941	0.97623	2.07791	N/A
trace	1.04830	0.97110	2.07447	N/A
PH(17) : FULL	1.04830	0.99608	2.21869	−0.81844
PH(17) : RND	1.04830	1.00003	2.22452	−0.82583

The density plot of **ME** distribution and its respective Phase-type distribution approximations are shown in Figure 5.9.

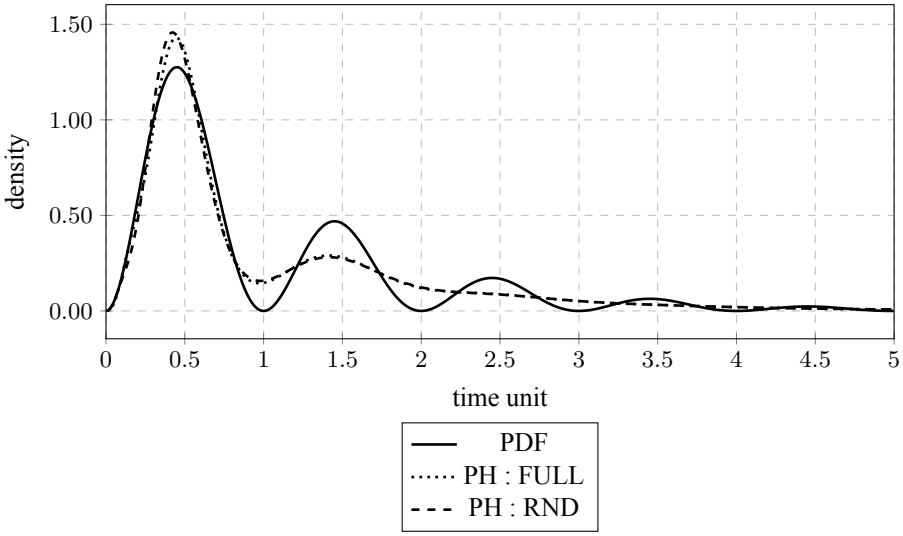

Fig. 5.9. Density graph of **ME** distribution and its Phase-type approximations.

The simulated model convergence of the total serving time mean and the standard deviation are shown in Figures 5.10, 5.11.

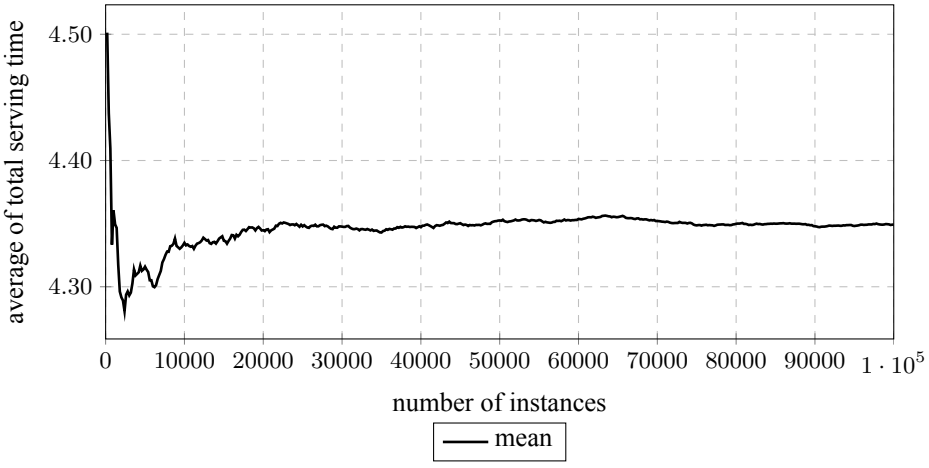

Fig. 5.10. Convergence of the total serving time mean for the finite request queue model with **ME**, **W4** distributions.

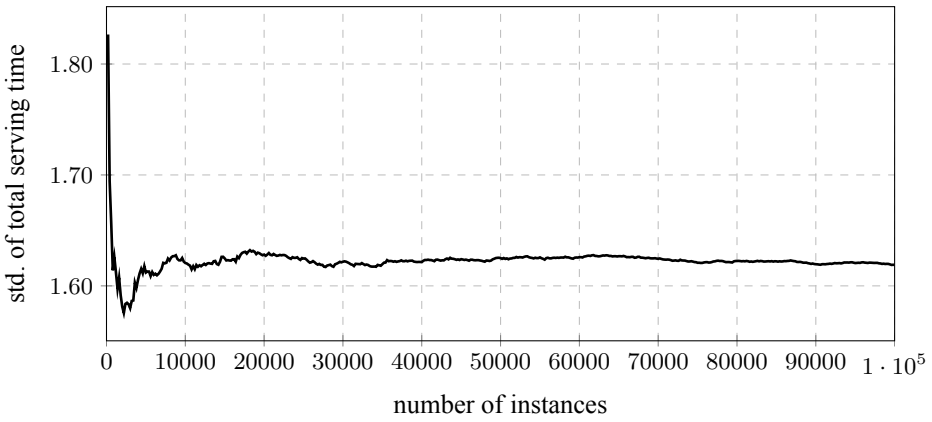

Fig. 5.11. Convergence of the total serving time standard deviation for the finite request queue model with **ME**, **W4** distributions.

The average queue occupation is shown in Figure 5.12. Due to the failure to properly approximate the **ME** distribution, inaccuracies showed up at the peak of the average queue occupation.

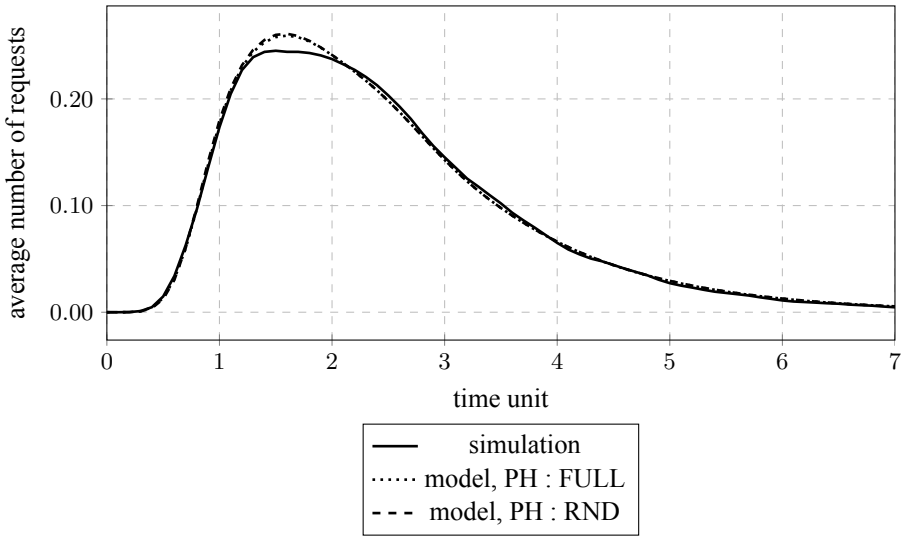

Fig. 5.12. Graph of the average number of requests in the queue for the finite queue model with distributions **ME**, **W4**.

The total serving time characteristics and the desnity plot are given in Table 5.8 and Figure 5.13.

Table 5.8. Statistical properties of the completion time distribution for a finite request model with distributions **ME**, **W4**.

	mean	std.	skewness
simulation	4.34926	1.61894	1.21197
model, PH : FULL	4.35451	1.65742	1.25033
model, PH : RND	4.35612	1.66357	1.25414

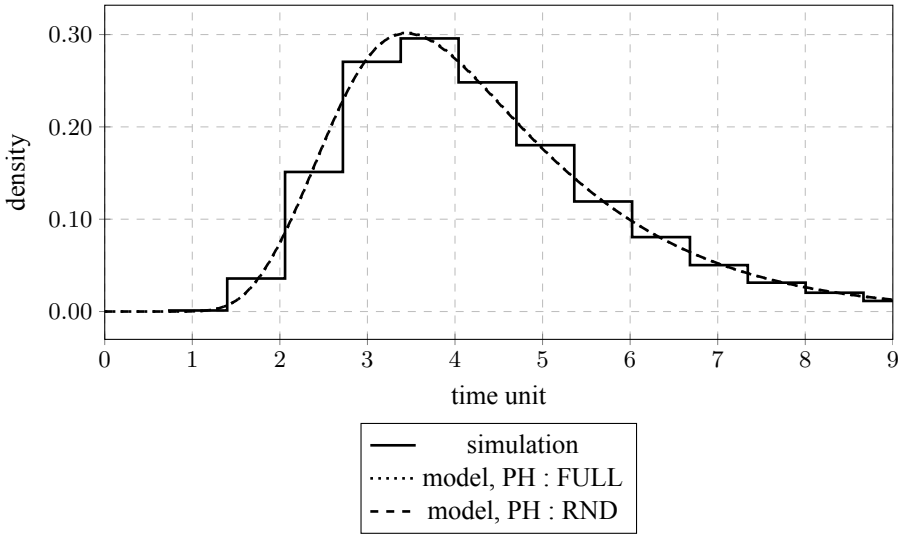

Fig. 5.13. Density of the distribution of time it takes to serve all the requests for a finite request model with distributions **ME**, **W4**.

The characteristics of the total serving time are similar to the ones obtained from simulation. The reason for this could be the fact that even though we failed to approximate the distribution **ME** probability function shape, the basic characteristics (i.e., mean, standard deviation) have been approximated fairly well. That was sufficient (for this relatively simple queueing model) to obtain a reasonable total serving time distribution.

5.4. Conclusions

Phase-type fitting using sparse (randomly generated, with $m = 2n$ transitions) and full structures has been compared in a more practical context. For that, a finite request queue model has been formulated and implemented. The modeling results do not differ much regardless of the structure type. However, two key observations have been developed. First, sparse structures find more likely parameter estimates earlier (which is consistent with the findings in Section 5). Second, despite the fact that a relatively small number of randomly generated structures have been checked (i.e., for

order $n = 17$), the obtained parameter estimates likelihood is not that worse compared to the ones obtained with the full structure. This suggests that the minimal structure set which covers the whole Phase-type distribution space might be comparatively small.

The advantage of fitting with a sparse structure is in its ability to converge faster. However, to use this advantage in a practical context, a methodology to determine the optimal (or near optimal) structure based on trace data should be used. Such methodology, in general, is not known yet.

FINAL CONCLUSIONS

1. A Phase-type distribution matrix form representation structure algorithm has been developed. It is not guaranteed that the generated structure sets are minimal, and the algorithm is not practical for higher order ($n > 4$) structure generation. However, it managed to generate structures of order $n = 4$ for our presented research.
2. The hypothesis that $2n$ transitions are sufficient for Phase-type fitting is validated empirically for the case $n = 4$ using structure sets generated with our algorithm.
3. PH(n) fitting of sparse representations with $2n$ transitions find more likely parameter estimates early. However, a full structure is capable of approaching good parameter estimates, but it requires more computational effort and can be terminated too early due to slow convergence. The convergence of sparse structures depends on the structure choice. In case of a suitable structure, convergence is much faster (compared to the full structure). In order to use an advantage of sparse structure fast convergence, a methodology for a suitable structure should be developed.
4. With additional computational complexity, the EM forward-backward algorithm for MAP fitting can be parallelized. The implementations based on the Baum-Welch algorithm are memory efficient, while an algorithm based on parallel likelihood vector computation is faster.
5. An algorithm based on the Expectation Maximization method for Transient Markov Arrival Process (TMAP) fitting has been formulated and tested.
6. MAP fitting performed 300-500 times faster on our GPU, while simple TMAP parallelization implementation for GPU gave only 10-20 times speed-up. The MAP parallelization is more efficient for the execution on a GPU device (compared to TMAP fitting implementation) because of its uniform nature of computations.

REFERENCES

1. BASHARIN, G. P., A. N. LANGVILLE and V. A. NAUMOV. The life and work of A.A. Markov. *Linear Algebra and its Applications*. Elsevier Science, 2004, vol. 386, pp. 3–26. doi:10.1016/j.laa.2003.12.041.

2. STEWART, W. J. *Introduction to the numerical solution of Markov chains*. Princeton, NJ: Princeton Univ. Press, 1994. ISBN 0691036993.

3. HORVáTH, A., M. SCARPA and M. TELEK. Phase Type and Matrix Exponential Distributions in Stochastic Modeling. In: *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*. Cham: Springer International Publishing, 2016, pp. 3–25. ISBN 978-3-319-30599-8. doi:10.1007/978-3-319-30599-8_1.

4. NEUTS, M. F. Probability distributions of phase type. In: *Liber Amicorum Prof. Emeritus H. Florin*. 1975, pp. 173–206.

5. NEUTS, M. F. *Matrix Geometric Solutions in Stochastic Models, an Algorithmic Approach*. The Johns Hopkins University Press, 1981.

6. KLEINROCK, L. *Queuing systems, Volume 1: Theory*. New York, NY, USA: Wiley-Interscience, 1975. ISBN 0471491101.

7. KEMENY, J. G. and J. L. SNELL. *Finite Markov Chains: With a New Appendix "Generalization of a Fundamental Matrix"*. 2nd edition. New York Berlin Heidelberg Tokyo: Springer-Verlag, 1976. ISBN 0-387-90192-2, 3-540-90192-2.

8. HORVáTH, G. and M. TELEK. On the Canonical Representation of Phase Type Distributions. *Performance Evaluation*. 2009.

9. TELEK, M. and G. HORVáTH. A minimal representation of Markov arrival processes and a moments matching method. *Performance Evaluation*. 2007.

10. LATOUCHE, G. and V. RAMASWAMI. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. illustrated edition edition. Society for Industrial Mathematics, 1999. ISBN 0898714257,9780898714258.

11. Ó CINNéIDE, C. A. Phase-type distributions: open problems and a few properties. *Communications in Statistics Stochastic Models*. 1999, vol. 15, no. 4, pp. 731–757. doi:10.1080/15326349908807560.

12. HU, L., Y. JIANG, J. ZHU and Y. CHEN. Hybrid of the scatter search, improved adaptive genetic, and expectation maximization algorithms for phase-type distribution fitting. *Applied Mathematics and Computation*. Elsevier Science, 2013, vol. 219, no. 10, pp. 5495–5515. ISSN 0096-3003. doi:10.1016/j.amc.2012.11.019.

13. THüMMLER, A., P. BUCHHOLZ and M. TELEK. A novel approach for Phase-type fitting with the EM algorithm. *IEEE Transactions on Dependable and Secure Computing*. IEEE Computer Society, 2006, vol. 3, no. 3, pp. 245-258. doi:10.1109/TDSC.2006.27.

14. ASMUSSEN, S., O. NERMAN and M. OLSSON. Fitting Phase-Type Distributions via the EM Algorithm. *Scandinavian Journal of Statistics*. Blackwell Publishing on behalf of

Board of the Foundation of the Scandinavian Journal of Statistics, 1996, vol. 23, no. 4, pp. 419-441. ISSN 03036898. doi:10.2307/4616418.

15. BLADT, M. and B. F. NIELSEN. Moment Distributions of Phase Type. *Stochastic Models*. Taylor & Francis Group, LLC, 2011, vol. 27, no. 4, pp. 651-663. doi:10.1080/15326349.2011.614192.

16. Ó CINNéIDE, C. A. On non-uniqueness of representations of phase-type distributions. *Communications in Statistics Stochastic Models*. 1989, vol. 5, no. 2, pp. 247-259. doi:10.1080/15326348908807108.

17. HORVáTH, G. and M. TELEK. A Canonical Representation of Order 3 Phase Type Distributions. In: *Formal Methods and Stochastic Models for Performance Evaluation: Fourth European Performance Engineering Workshop, EPEW 2007, Berlin, Germany, September 27-28, 2007. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 48-62. ISBN 978-3-540-75211-0. doi:10.1007/978-3-540-75211-0_5.

18. HORVáTH, A. and M. TELEK. Matching More Than Three Moments with Acyclic Phase Type Distributions. *Stochastic Models*. Taylor and Francis Group, 2007, vol. 23, no. 2, pp. 167–194. doi:10.1080/15326340701300712.

19. CUMANI, A. . *Microelectronics Reliability*. 1982.

20. ALDOUS, D. and L. SHEPP. The least variable phase type distribution is Erlang. *Stochastic Models*. 1987.

21. FANG, Y. Hyper-Erlang Distribution Model and its Application in Wireless Mobile Networks. *Wireless Networks*. Springer, 2001, vol. 7, no. 3, pp. 211–219. doi:10.1023/a:1016617904269.

22. COMMAULT, C. and S. MOCANU. Phase-type distributions and representations: some results and open problems for system theory. *International Journal of Control*. 2003, vol. 76, no. 6, pp. 566-580.

23. DEMPSTER, A. P., N. M. LAIRD and D. B. RUBIN. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B (Methodological)*. 1977.

24. ALBERT, A. Estimating the infinitesimal generator of a continuous time, finite state markov process. *Ann. Math. Statist.* The Institute of Mathematical Statistics, 1962, vol. 33, no. 2, p. 727–753. doi:10.1214/aoms/1177704594.

25. OKAMURA, H., T. DOHI and K. S. TRIVEDI. A refined EM algorithm for PH distributions. *Perform. Eval.* Elsevier, 2011, vol. 68, no. 10, pp. 353-381. ISSN 0166-5316. doi:10.1016/j.peva.2011.04.001.

26. FANG, Y. and I. CHLAMTAC. Teletraffic analysis and mobility modeling of PCS networks. *IEEE Transactions on Communications*. IEEE, 1999, vol. 47, no. 7, pp. 1062–1072. doi:10.1109/26.774856.

27. EZHOV, I. and A. SKOROKHOD. Markov Processes with Homogeneous Second Component. I. *Theory of Probability & Its Applications*. 1969, vol. 14, no. 1, pp. 1-13. doi:10.1137/1114001.

28. LUCANTONI, D. M., K. S. MEIER-HELLSTERN and M. F. NEUTS. A single-server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*. Applied Probability Trust, 1990, vol. 22, no. 03, pp. 676–705. doi: 10.1017/s0001867800019947.

29. BASHARIN, G. P., V. A. NAUMOV and K. SAMOUYLOV. On Markovian modelling of arrival processes. *Statistical Papers*. 2018, vol. 59, no. 4, pp. 1533–1540. ISSN 1613-9798. doi:10.1007/s00362-018-1042-9.

30. BODROG, L., A. HEINDL, G. HORVáTH and M. TELEK. A Markovian canonical form of second-order matrix-exponential processes. *European Journal of Operational Research*. Elsevier, 2008, vol. 190, no. 2, pp. 459-477. ISSN 0377-2217. doi:10.1016/j.ejor.2007.06.020.

31. OKAMURA, H. and T. DOHI. Faster Maximum Likelihood Estimation Algorithms for Markovian Arrival Processes. In: *Sixth International Conference on the Quantitative Evaluation of Systems*. Budapest, Hungary: IEEE, 2009, pp. 73-82. ISBN 978-0-7695-3808-2. doi:10.1109/QEST.2009.28.

32. HORVáTH, G. and H. OKAMURA. A Fast EM Algorithm for Fitting Marked Markovian Arrival Processes with a New Special Structure. In: *Computer Performance Engineering: 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 119-133. ISBN 978-3-642-40725-3. doi:10.1007/978-3-642-40725-3_10.

33. JOHNSON, M. A. and M. R. TAAFFE. An investigation of phase-distribution moment-matching algorithms for use in queueing models. *Queueing Systems*. Springer US, 1991, vol. 8, no. 1, pp. 129–147. doi:10.1007/bf02412246.

34. FADDY, M. J. Phase-type Distributions for Failure Times. *Math. Comput. Model.* Elsevier Science Publishers B. V., 1995, vol. 22, no. 10-12, pp. 63-70. ISSN 0895-7177. doi:10.1016/0895-7177(95)00181-Z.

35. BLADT, M. A Review on Phase-type Distributions and their Use in Risk Theory. *Astin Bulletin*. Peeters Publishers, 2005, vol. 35, no. 01, pp. 145–161. doi:10.1017/s0515036100014100.

36. BUCHHOLZ, P., J. KRIEGE and I. FELKO. *Input Modeling with Phase-Type Distributions and Markov Models - Theory and Applications*. 1st edition. Springer International Publishing, 2014. ISBN 978-3-319-06673-8,978-3-319-06674-5. doi:10.1007/978-3-319-06674-5.

37. BOUNTZIS, P., E. PAPADIMITRIOU and G. TSAKLIDIS. Estimating the earthquake occurrence rates in Corinth Gulf (Greece) through Markovian arrival process modeling. *Journal of Applied Statistics*. Taylor & Francis, 2019, vol. 46, no. 6, pp. 995-1020. doi: 10.1080/02664763.2018.1531977.

38. HAUTPHENNE, S. and G. LATOUCHE. The Markovian binary tree applied to demography. *Journal of Mathematical Biology*. Springer, 2012, vol. 64, no. 7, pp. 1109–1135. doi:10.1007/s00285-011-0437-1.

39. KIM, K. On the Relation between Phase-Type Distributions and Positive Systems. *Abstract*

and Applied Analysis. Hindawi Publishing Corporation, 2015, vol. 2015, no. 731261, pp. 1–7. doi:10.1155/2015/731261.

40. HE, Q.-M. and H. ZHANG. PH-Invariant Polytopes and Coxian Representations of Phase Type Distributions. *Stochastic Models*. Taylor and Francis Group, 2006, vol. 22, no. 3, pp. 383–409. doi:10.1080/15326340600820349.

41. AHLSTRöM, L., M. OLSSON and O. NERMAN. A Parametric Estimation Procedure for Relapse Time Distributions. *Lifetime Data Analysis*. Springer, 1999, vol. 5, no. 2, pp. 113–132. doi:10.1023/a:1009697311405.

42. AHN, S. and V. RAMASWAMI. Bilateral Phase Type Distributions. *Stochastic Models*. Taylor and Francis Group, 2005, vol. 21, no. 2-3, pp. 239–259. doi:10.1081/STM-200056029.

43. HORVáTH, A. and M. TELEK. On the Properties of Acyclic Bilateral Phase Type Distributions. In: *Proceedings of the 2Nd International Conference on Performance Evaluation Methodologies and Tools*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 79:1–79:8. ISBN 978-963-9799-00-4.

44. RAMASWAMI, V. and N. C. VISWANATH. Phase Type Distributions with Finite Support. *Stochastic Models*. Taylor and Francis Group, 2014, vol. 30, no. 4, pp. 576–597. doi:10.1080/15326349.2014.956226.

45. OSOGAMI, T. and M. HARCHOL-BALTER. Necessary and Sufficient Conditions for Representing General Distributions by Coxians. In: *Computer Performance Evaluation. Modelling Techniques and Tools: 13th International Conference, TOOLS 2003, Urbana, IL, USA, September 2-5, 2003. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 182-199. ISBN 978-3-540-45232-4. doi:10.1007/978-3-540-45232-4_12.

46. HE, Q.-M. and H. ZHANG. Spectral Polynomial Algorithms for Computing Bi-Diagonal Representations for Phase Type Distributions and Matrix-Exponential Distributions. *Stochastic Models*. Taylor & Francis, 2006, vol. 22, no. 2, pp. 289-317. doi:10.1080/15326340600649045.

47. BOSCH, P. M. V., D. C. DIETZ and E. A. POHL. Moment matching using a family of phase-type distributions. *Stochastic Models*. Taylor & Francis Group, LLC, 2000, vol. 16, no. 3-4, pp. 391-398. doi:10.1080/15326340008807595.

48. OSOGAMI, T. and M. HARCHOL-BALTER. Closed form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation*. Elsevier Science, 2006, vol. 63, no. 6, pp. 524–552. doi:10.1016/j.peva.2005.06.002.

49. HE, Q.-M. and H. ZHANG. A Note on Unicyclic Representations of Phase Type Distributions. *Stochastic Models*. Taylor & Francis, 2005, vol. 21, no. 2-3, pp. 465-483. doi:10.1081/STM-200057131.

50. BOBBIO, A., G. HORVáTH and M. TELEK. Matching Three Moments with Minimal Acyclic Phase Type Distributions. *Stochastic models*. 2005.

51. BUCHHOLZ, P. and A. PANCHENKO. An EM Algorithm for Fitting of Real Traffic Traces to PH-Distribution. In: *2004 International Conference on Parallel Computing*

in Electrical Engineering (PARELEC 2004), 7-10 September 2004, Dresden, Germany. Dresden, Germany: IEEE Computer Society, 2004, pp. 283-288. ISBN 0-7695-2080-4. doi:10.1109/PCEE.2004.12.

52. WATANABE, R., H. OKAMURA and T. DOHI. An efficient MCMC algorithm for continuous PH distributions. In: *Winter Simulation Conference, WSC '12, Berlin, Germany, December 9-12, 2012*. WSC, 2012, pp. 426:1–426:12. ISBN 978-1-4673-4779-2. doi: 10.1109/WSC.2012.6465313.

53. BOBBIO, A. and A. CUMANI. ML estimation of the parameters of a PH distribution in triangular canonical form. *Computer Performance Evaluation*. 1992, pp. 33-46.

54. FELDMAN, A. and W. WHITT. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models . In: *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*. IEEE, 1997, pp. 1096-1104. ISBN 0-8186-7780-5. doi: 10.1109/INFCOM.1997.631130.

55. EL ABDOUNI KHAYARI, R., B. R. HAVERKORT and R. SADRE. Fitting world-wide web request traces with the EM-algorithm. *Performance Evaluation*. 2003.

56. RISKA, A., V. DIEV and E. SMIRNI. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation*. Elsevier Science, 2004, vol. 55, no. 1-2, pp. 147–164. doi:10.1016/s0166-5316(03)00101-9.

57. OKAMURA, H. and T. DOHI. Fitting Phase-Type Distributions and Markovian Arrival Processes: Algorithms and Tools. In: *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*. Cham: Springer International Publishing, 2016, pp. 49-75. ISBN 978-3-319-30599-8. doi:10.1007/978-3-319-30599-8_1.
URL `http://dx.doi.org/10.1007/978-3-319-30599-8_1`

58. BOBBIO, A. and M. TELEK. A benchmark for PH estimation algorithms: results for acyclic-PH. *Stochastic Models*. 1994.

59. HORVáTH, A. and M. TELEK. Approximating Heavy Tailed Behaviour With Phase Type Distributions. In: *3rd International Conference on Matrix-Analitic Methods in Stochastic models, MAM3*. Neshanic Station, NJ: Notable Publications Inc., 2000, pp. 191-214.

60. RISKA, A., V. DIEV and E. SMIRNI. Efficient Fitting of Long-tailed Data Sets into Phase-type Distributions. *SIGMETRICS Perform. Eval. Rev.* ACM, 2002, vol. 30, no. 3, pp. 6-8. ISSN 0163-5999. doi:10.1145/605521.605525.

61. HORVáTH, G. Explicit Phase-type Distribution Fitting in Laplace Transform Domain. In: *Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*. European Council for Modelling and Simulation, 2007, pp. 1-6.

62. MARSHALL, A. H. and M. ZENGA. Recent developements in fitting Coxian Phase-type distributions in healthcare. In: *The XIII International Conference "Applied Stochastic Models and Data Analysis" (ASMDA-2009)*. Vilniaus Gediminas Technical University, 2009, pp. 482-485. ISBN 978-9955-28-463-5.

63. BUCHHOLZ, P. An EM-Algorithm for MAP Fitting from Real Traffic Data. In: *Computer Performance Evaluation. Modelling Techniques and Tools: 13th International Conference, TOOLS 2003, Urbana, IL, USA, September 2-5, 2003. Proceedings*. 2003.

64. CASALE, G., E. Z. ZHANG and E. SMIRNI. KPC-Toolbox: Best recipes for automatic trace fitting using Markovian Arrival Processes. *Performance Evaluation*. Elsevier Science, 2010, vol. 67, no. 9, pp. 873–896. doi:10.1016/j.peva.2009.12.003.

65. MéSZáROS, A. and M. TELEK. Canonical Form of Order-2 Non-stationary Markov Arrival Processes. In: *Computer Performance Engineering: 12th European Workshop, EPEW 2015, Madrid, Spain, August 31 - September 1, 2015, Proceedings*. Cham, 2015, pp. 163-176. ISBN 978-3-319-23267-6. doi:10.1007/978-3-319-23267-6_11.

66. BODROG, L., A. HORVáTH and M. TELEK. Moment characterization of matrix exponential and Markovian arrival processes. *Annals of Operations Research*. Springer, 2008, vol. 160, no. 1, pp. 51–68. ISSN 1572-9338. doi:10.1007/s10479-007-0296-8.

67. MéSZáROS, A., G. HORVáTH and M. TELEK. Representation Transformations for Finding Markovian Representations. In: *Analytical and Stochastic Modeling Techniques and Applications: 20th International Conference, ASMTA 2013, Ghent, Belgium, July 8-10, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 277-291. ISBN 978-3-642-39408-9. doi:10.1007/978-3-642-39408-9_20.

68. TELEK, M. Some Structural Properties of Markov and Rational Arrival Processes. *Stochastic Models*. Taylor and Francis Group, 2011, vol. 27, no. 4, pp. 723–746. doi: 10.1080/15326349.2011.614481.

69. CASALE, G., E. Z. ZHANG and E. SMIRNI. Interarrival Times Characterization and Fitting for Markovian Traffic Analysis. In: *Numerical Methods for Structured Markov Chains*. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008, , no. 07461.

70. HEINDL, A., G. HORVáTH and K. GROSS. Explicit Inverse Characterizations of Acyclic MAPs of Second Order. In: *Proc. 3rd European Performance Engineering Workshop*. Budapest, Hungary, 2006, vol. 4054 of LNCS, pp. 108-122.

71. KIM, S. The characteristic polynomial and the Laplace representations of MAP(2)s. *Stochastic Models*. Taylor and Francis Group, 2017, vol. 33, no. 1, pp. 30–47. doi: 10.1080/15326349.2016.1199282.

72. BODROG, L., P. BUCHHOLZ, J. KRIEGE and M. TELEK. Canonical Form Based MAP(2) Fitting. In: *Seventh International Conference on the Quantitative Evaluation of Systems (QEST)*. Williamsburg, VA, USA, 2010, pp. 107–116. ISBN 978-1-4244-8082-1. doi:10.1109/QEST.2010.22.

73. BUCHHOLZ, P., G. HORVáTH and M. TELEK. A MAP fitting approach with independent approximation of the inter-arrival time distribution and the lag correlation. In: *Proceedings. Second International Conference on the Quantitative Evaluation of Systems*. 2005.

74. BUCHHOLZ, P. and J. KRIEGE. A Heuristic Approach for Fitting MAPs to Moments and Joint Moments. In: *Proc. of the 6th International Conference on Quantitative Evaluation of SysTems (QEST 2009)*. 2009.

75. MéSZáROS, A. and M. TELEK. A Two-phase MAP Fitting Method with APH Interarrival Time Distribution. In: *Proceedings of the Winter Simulation Conference*. Berlin, Germany: Winter Simulation Conference, 2012, pp. 3939-3950.

76. BUCHHOLZ, P., I. FELKO and J. KRIEGE. Transformation of Acyclic Phase Type Distributions for Correlation Fitting. In: *Analytical and Stochastic Modeling Techniques and Applications: 20th International Conference, ASMTA 2013, Ghent, Belgium, July 8-10, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 96-111. ISBN 978-3-642-39408-9. doi:10.1007/978-3-642-39408-9_8.

77. BAUSE, F. and G. HORVáTH. Fitting Markovian Arrival Processes by Incorporating Correlation into Phase Type Renewal Processes. In: *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*. 2010.

78. KRIEGE, J. and P. BUCHHOLZ. An Empirical Comparison of MAP Fitting Algorithms. In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance: 15th International GI/ITG Conference, MMB&DFT 2010, Essen, Germany, March 15-17, 2010. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 259-273. ISBN 978-3-642-12104-3. doi:10.1007/978-3-642-12104-3_20.

79. OKAMURA, H., H. KISHIKAWA and T. DOHI. Application of deterministic annealing EM algorithm to MAP/PH parameter estimation. *Telecommunication Systems*. Springer US, 2013, vol. 54, no. 1, pp. 79–90. doi:10.1007/s11235-013-9717-y.

80. KRIEGE, J. and P. BUCHHOLZ. PH and MAP Fitting with Aggregated Traffic Traces. In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance: 17th International GI/ITG Conference. Proceedings*. Cham: Springer International Publishing, 2014, pp. 1-15. ISBN 978-3-319-05359-2. doi:10.1007/978-3-319-05359-2_1.

81. OKAMURA, H., T. DOHI and K. S. TRIVEDI. Markovian Arrival Process Parameter Estimation With Group Data. *IEEE/ACM Transactions on Networking*. IEEE, 2009, vol. 17, no. 4, pp. 1326–1339. doi:10.1109/tnet.2008.2008750.

82. DALEY, D. J. and D. VERE-JONES. *[An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. 2nd edition. Springer-Verlag New York, 2003. ISBN 978-0-387-95541-4,978-0-387-21564-8.

83. LATOUCHE, G., M.-A. REMICHE and P. TAYLOR. Transient Markov Arrival Processes. *The Annals of Applied Probability*. Institute of Mathematical Statistics, 2003, vol. 13, no. 2, pp. 628–640. doi:10.2307/1193162.

84. HAUTPHENNE, S. and M. TELEK. Extension of some MAP results to transient MAPs and Markovian binary trees. *Performance Evaluation*. Elsevier Science, 2016, vol. 70, no. 9, pp. 607–622. doi:10.1016/j.peva.2013.05.003.

85. BODROG, L., A. HEINDL, G. HORVáTH, M. TELEK and A. HORVáTH. Current results and open questions on PH and MAP characterization. In: *Numerical Methods for Structured Markov Chains*. Dagstuhl, Germany: Internationales Begegnungs, 2008, , no. 07461.

86. COX, D. R. A use of complex probabilities in the theory of stochastic processes. *Mathem-*

atical Proceedings of the Cambridge Philosophical Society. Cambridge University Press, 1955, vol. 51, no. 2, pp. 313-319. doi:10.1017/s0305004100030231.

87. TURIN, W. Unidirectional and parallel Baum-Welch algorithms. *IEEE Trans. Speech and Audio Processing*. IEEE Signal Processing Society, 1998, vol. 6, no. 6, pp. 516-523. ISSN 1063-6676. doi:10.1109/89.725318.

88. LI, L., B. FU and C. FALOUTSOS. Efficient Parallel Learning of Hidden Markov Chain Models on SMPs. *IEICE Transactions on Information and Systems*. Oxford University Press, 2010, vol. E93-D, no. 6, pp. 1330–1342. doi:10.1587/transinf.e93.d.1330.

89. NIELSEN, J. and A. SAND. Algorithms for a Parallel Implementation of Hidden Markov Models with a Small State Space. In: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. 2011, pp. 452-459. doi:10.1109/IPDPS.2011.181.

90. FOWLER, H. J. and W. E. LELAND. Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management. *IEEE Journal on Selected Areas in Communications*. 1991, vol. 9, pp. 1139–1149.

91. LELAND, W. E. and D. V. WILSON. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In: *IEEE INFCOM '91. The conference on Computer Communications. Tenth Annual Joint Comference of the IEEE Computer and Communications Societies Proceedings*. 1991, vol. 3, pp. 1360-1366. doi:10.1109/INFCOM.1991.147663.

92. CASALE, G., E. Z. ZHANG and E. SMIRNI. KPC-Toolbox: Simple Yet Effective Trace Fitting Using Markovian Arrival Processes. In: *Proceedings of 5th international conference on quantitative evaluation of systems (QEST2008)*. St. Malo, France: IEEE, 2008, p. 83–92. ISBN 978-0-7695-3360-5. doi:10.1109/QEST.2008.33.

93. CASALE, G., E. Z. ZHANG and E. SMIRNI. Trace data characterization and fitting for Markov modeling. *Performance Evaluation*. Elsevier Science, 2010, vol. 67, no. 2, pp. 61–79. ISSN 0166-5316. doi:10.1016/j.peva.2009.09.003.

94. ANDERSON, T. W. and L. A. GOODMAN. Statistical Inference about Markov Chains. *The Annals of Mathematical Statistics*. 1957, vol. 28, no. 1. doi:10.1214/aoms/1177707039.

LIST OF SCIENTIFIC PUBLICATIONS ON THE THEME OF THE DISSERTATION

Papers in Master List Journals of the Institute of Scientific Information (ISI)

1. BRAŽĖNAS, Mindaugas; HORVATH, Gabor; TELEK, Miklos. Parallel algorithms for fitting Markov arrival processes. Performance Evaluation. 2018, vol. 123-124, pp. 50-67, ISSN: 0166-5316, doi: 10.1016/j.peva.2018.05.001
2. VALAKEVIČIUS, Eimutis; BRAŽĖNAS, Mindaugas. On Structured Initial Solution Generation for Phase-Type Fitting with EM Method. Information Technology and Control. ISSN: 1392-124X. 2018, vol. 47, iss. 2, pp. 197-208, ISSN: 1392-124X, doi: 10.5755/j01.itc.47.2.18169

Papers in Proceedings List of the Institute of Scientific Information

3. BRAŽĖNAS, Mindaugas; HORVÁTH, Gábor; TELEK, Miklós. Efficient implementations of the EM-algorithm for transient Markovian arrival processes. In: Analytical and stochastic modelling techniques and applications: proceedings of the 23rd international conference, ASMTA 2016, Cardiff, UK, August 24-26, 2016. Editors: Sabine Wittevrongel, Tuan Phung-Duc. Cham: Springer, 2016, pp. 107-122. ISBN 9783319439037. eISBN 9783319439044. doi: 10.1007/978-3-319-43904-4_8
4. BRAŽĖNAS, Mindaugas; VALAKEVIČIUS, Eimutis. Software relialibility Markovian model based on phase-type distribution. In: ICSEA 2014: the Ninth International Conference on Software Engineering Advances, October 12-16, 2014, Nice, France. Nice: IARIA, 2014, pp. 591-597. ISBN 9781612083674.

SL344. 2019-05-02, 18,5 leidyb. apsk. l. Tiražas 14 egz. Užsakymas 101.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas