A Novel Approach for Phase-Type Fitting with the EM Algorithm

Axel Thümmler and Peter Buchholz
University of Dortmund
Department of Computer Science
August-Schmidt-Str. 12
44227 Dortmund, Germany
{thuemmler, buchholz}@ls4.cs.uni-dortmund.de

Miklós Telek
Budapest University of Tech. and Econ.
Department of Telecommunications
Magyar Tudósok krt. 2
1117 Budapest, Hungary
telek@hit.bme.hu

Abstract

The representation of general distributions or measured data by phase-type distributions is an important and non-trivial task in analytical modeling. Although a large number of different methods for fitting parameters of phase-type distributions to data traces exist, many approaches lack efficiency and numerical stability. In this paper, a novel approach is presented that fits a restricted class of phase-type distributions, namely mixtures of Erlang distributions, to trace data. For the parameter fitting an algorithm of the expectation maximization type is developed. The paper shows that these choices result in a very efficient and numerically stable approach which yields phase-type approximations for a wide range of data traces that are as good or better than approximations computed with other less efficient and less stable fitting methods. To illustrate the effectiveness of the proposed fitting algorithm, we present comparative results for our approach and two other methods using six benchmark traces and two real traffic traces as well as quantitative results from queueing analysis.

Keywords:

Performance and dependability assessment/analytical and numerical techniques,

design of tools for performance/dependability assessment,

traffic modeling,

hyper-Erlang distributions.

1 Introduction

The central idea of traffic modeling lies in constructing analytically tractable models that capture the most important statistical properties of an underlying measured data trace. For analytical performance and dependability modeling measured data has to be represented or approximated by phase-type (PH) distributions in several cases [11], [25]. The procedure of estimating the parameters of a phase-type distribution according to some sample data or with respect to some other known distribution is commonly denoted as phase-type fitting.

Among the large number of available fitting methods, expectation-maximization (EM) algorithms [19] are general methods of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data trace when the data is incomplete or has missing values. EM algorithms for phase-type fitting have been available for some time [1], [4], but the application of the basic approach to general PH distributions turns out to be extremely costly and the fitted distribution depends heavily on the initial values [20]. Thus, it seems that fitting general PH distributions is not appropriate if the number of phases increases above 4, which is often the case for small coefficients of variation or traces that cannot be adequately represented by a PH distribution of low order. To overcome these problems the class of PH distributions used for fitting has to be restricted which is in principle possible in the basic EM algorithm by initializing only some elements in the matrix with non-zero values, but it seems to be more appropriate to develop an EM algorithm tailored to specific types of PH distributions. Based on earlier work from [10], El Abdouni Khayari et al. developed an EM algorithm in [8] to fit the parameters of a hyperexponential distribution to values of a data trace. The resulting approach is extremely efficient and yields good fitting results for heavy-tailed distributions with monotonically decreasing density functions. However, the use of hyperexponential distributions restricts the class of distributions, which can be represented. In fact, hyperexponential distributions cannot adequately capture general distributions with increasing and decreasing densities or with a coefficient of variation less than one.

Since the fitting of parameters of a PH distribution is in general a non-linear optimization problem, apart from the EM algorithm also other optimization algorithms can be applied.

However, the optimization problem for general PH distributions is too complex to yield satisfactory results if the number of phases is larger than two or three. As shown in several papers [2], [3], [13], [14], the fitting problem becomes much easier if acyclic instead of general phase-type distributions are used, because for this type of distributions a canonical representation exists which reduces the number of free parameters to 2N compared to $N^2 + N$ for the general case, where N is then number of phases [5]. On the other hand, the restriction to acyclic PH distributions does not seem to limit the flexibility of the approach. However, even in the acyclic case, the resulting optimization is still complex and contains local optima and saddle points. To overcome the problem of convergence to a local optimum, the fitting algorithm is usually started with several initial settings and the best fitting is chosen.

Apart from acyclic phase-type distributions several other restricted classes have been used. For our approach the works of Johnson [15] and Schmickler [24] are most important, since both use mixtures of Erlang distributions, which are also used in our work and will be denoted as hyper-Erlang distributions (HErD) according to [9]. However, in contrast to our approach, the mentioned techniques fit some moments and specific properties of the distribution or density function using nonlinear optimization.

In this paper, an EM algorithm for the fitting of hyper-Erlang distributions is presented. The approach, which will be denoted as G-FIT, extends the fitting procedure of [8] from hyperexponential to hyper-Erlang distributions, which extends the class of representable distributions significantly since mixtures of Erlang distributions of unlimited order are theoretically as powerful as acyclic or general PH distributions (see Theorem 1). However, the class of distributions still allows the realization of a very efficient fitting algorithm. In particular the fitting time is independent of the number of states; it depends only on the number of Erlang branches, which might be significantly lower than the number of states. In fact, for M Erlang branches and a trace with K samples the time complexity of our algorithm is in O(M·K). Thus, distributions with a large number of states can be fitted efficiently. Furthermore, the fitting algorithm is rather stable due to the specific structure of the density function, which yields a fast and reliable convergence of the EM method. Additionally, the fitting of the first three moments using a polynomial of degree 5 is introduced and it is shown how moment fitting can be integrated in the proposed EM algorithm that fits the empirical distribution function.

Apart from the efficiency of the approach, the quality of the approximation for a given number of phases is important. We tested the approach on a set of six benchmark traces [3] and compared it with general PH-fitting [1] and fitting of acyclic PH distributions [14]. As expected, G-FIT is significantly faster than the other two approaches. Additionally, we were able to reach with an identical number of states a similar or better fitting quality than with the other two approaches on almost all examples. This result was not expected, because hyper-Erlang distributions of a given order are in general less flexible than acyclic or general PH distributions of the same order. The practical applicability of G-FIT is demonstrated by fitting a call center trace [21] and a large traffic trace, which was recorded at the Web proxy server at the University of Dortmund in March 2005. The presented EM algorithm is implemented in the software package G-FIT, which is available for download on the Web [12].

The paper is organized as follows. Section 2 introduces the considered class of hyper-Erlang distributions, it studies its relationship to general phase-type distributions, and it introduces the fitting of the first three moments of a hyper-Erlang distribution. Section 3 develops a specialized EM algorithm for fitting the continuous parameters of a hyper-Erlang distribution and Section 4 presents an approach for finding optimal settings of the discrete parameters of the distribution. Experimental results obtained from fitting synthetically generated benchmark traces and two real traffic traces as well as quantitative results from queueing analysis are presented in Section 5. Finally, concluding remarks are given.

2 Hyper-Erlang Distributions and its Properties

2.1 Hyper-Erlang Distributions

We consider a mixture of M mutually independent Erlang distributions weighted with the (initial) probabilities $\alpha_1,\ldots,\alpha_M$ with $\alpha_m \geq 0$ and $\alpha_1+\alpha_2+\ldots+\alpha_M = 1$. The *number of phases* of the m-th Erlang distribution is denoted with r_m. We assume $1 \le r_1 \le \ldots \le r_M$ without loss of generality. Furthermore, let λ_m be the *scale parameter* of the m-th Erlang distribution. Note, that the individual Erlang distributions need not have the same mean. According to [9], we call this mixture of Erlang distributions a *hyper-Erlang distribution (HErD).* The HErD belongs to the class of acyclic phase-type distributions [2]. Besides the Erlang distribution, for M = 1, the hyperexponential distribution is a special case of a HErD with $r_m = 1$ for all m = 1,…,M. Let X be a hyper-Erlang random variable. The probability density function (pdf) for X is given by

$$f_X(x) = \sum_{m=1}^{M} \alpha_m \frac{(\lambda_m x)^{r_m - 1}}{(r_m - 1)!} \lambda_m e^{-\lambda_m x}\ , \tag{1}$$

and the cumulative distribution function (cdf) is given by

$$F_X(x) = 1 - \sum_{m=1}^{M} \alpha_m \sum_{i=0}^{r_m - 1} \frac{(\lambda_m x)^i}{i!} e^{-\lambda_m x}\ . \tag{2}$$

The i-th moment $E[X^i]$ is given by

$$E[X^i] = \sum_{m=1}^{M} \alpha_m \frac{(r_m + i - 1)!}{(r_m - 1)!} \frac{1}{\lambda_m^i}\ . \tag{3}$$

A common measure to characterize the flexibility in approximating a given general distribution function is the range of variability of the squared coefficient of variation c_X^2, which is defined in terms of the first and second moment, i.e.,

$$c_X^2 = \frac{E[X^2]}{E[X]^2} - 1\ . \tag{4}$$

Recall that an Erlang distribution with r phases is defined as the sum of r independent identical exponentially distributed random variables. Thus, the HErD is constructed from a mixture of sums of exponential distributions. The *number of states* of a HErD is the overall number of exponential distributions involved in its construction. Keeping this in mind, the overall number of states of a HErD is given by

$$N = \sum_{m=1}^{M} r_m\ . \tag{5}$$

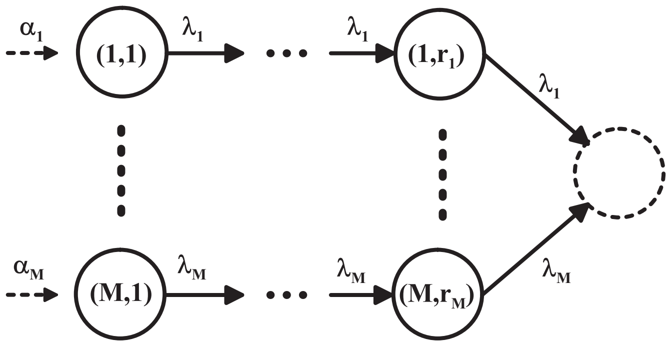

Figure 1. State transition graph of a hyper-Erlang distribution

Figure 1 shows the state transition graph of a HErD, which corresponds to an absorbing continuous-time Markov chain where a state change occurs after an exponentially distributed delay with mean $1/\lambda_m$, m = 1,…,M, and the time until absorption has a HErD. The absorbing state is shown as a dashed circle in Figure 1.

Let f(x; M, **r**, $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$) denote the density function of the HErD with M Erlang branches, where $\mathbf{r} = (r_1, r_2, \dots, r_M) \in \mathbb{N}^M$ is a vector containing the number of phases of each Erlang branch, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_M) \in \mathbb{R}^M$ is a vector with the initial probabilities for each Erlang branch and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_M) \in \mathbb{R}^M$ is a vector with the scaling parameters, respectively. Using the constraint $\sum_{m=1}^{M} \alpha_m = 1$, a HErD with M Erlang branches has 2M−1 continuous parameters given by $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ and M discrete parameters given by the vector **r**. Let $\mathcal{H}_N$ be a set of all HErD with N states, i.e.,

$$\mathcal{H}_N = \left\{ f\left(x; M, \mathbf{r}, \boldsymbol{\alpha}, \boldsymbol{\lambda}\right) \middle| 1 \le M \le N, \lambda_m > 0, \alpha_m \ge 0, r_m \ge 1, \sum_{m=1}^{M} \alpha_m = 1, \sum_{m=1}^{M} r_m = N \right\}. \qquad (6)$$

Note, that the set $\mathcal{H}_N$ contains all HErD distributions having at most N states, since HErD with less than N states are obtained by simply setting some α_m values to zero. The versatility of the HErD in approximating general distributions is shown by the following theorem.

Theorem 1:

(i) Let $\mathcal{F}$ denote the set of all probability density functions of nonnegative random variables, then $\mathcal{H}_\infty$ is a dense set in $\mathcal{F}$. In fact, for every density function $f \in \mathcal{F}$ it is possible to choose a sequence of hyper-Erlang densities $h_{\lambda,M}(x) \in \mathcal{H}_N$ with M Erlang branches each having scale parameter λ, such that

$$\lim_{\lambda\to\infty}\lim_{M\to\infty} h_{\lambda,M}(x) = f(x) \tag{7}$$

for all x at which f(x) is continuous.

(ii) Let h be a hyper-Erlang density out of the set $\mathcal{H}_N$, with $N \geq 2$. The parameters of h can be tuned such that the squared coefficient of variation of h equals 1/N or takes on an arbitrary value greater or equal to 1/(N–1) with h still being an element of $\mathcal{H}_N$.

Proof: The proof of (i) can be found in [16]. In particular, the hyper-Erlang densities $h_{\lambda,M}(x)$ are to be chosen so that $\lambda_m = \lambda$, $r_m = m$, and $\alpha_m = \left(F(m/\lambda) - F((m-1)/\lambda)\right)/F(M/\lambda)$ for $m = 1,\ldots,M$, where F is the cdf of $f \in \mathcal{F}$. For the proof of (ii) we distinguish three different cases. Let c_0 be the value that should be matched by the squared coefficient of variation of a hyper-Erlang density $h \in \mathcal{H}_N$.

Case 1: $c_0 = 1/N$: It is a well known fact that the squared coefficient of variation of an Erlang distribution with r phases equals 1/r, independent of the scaling parameter λ (see e.g. [25]). Thus, to obtain a squared coefficient of variation $c^2 = c_0 = 1/N$ we simply choose $M = 1$, $r_1 = N$, and $\alpha_1 = 1$.

To find a hyper-Erlang distribution with coefficient of variation that is greater or equal to 1/(N–1) we only have to consider the case M = 2 with $\alpha_1 := \alpha$, $\alpha_2 := 1-\alpha$, $r_1 = 1$, and $r_2 = N-1$, i.e., a mixture of an exponential distribution and an Erlang distribution with N–1 phases. Simplifying Eq. (4) with the help of Eq. (3) the general form of the squared coefficient of variation for this hyper-Erlang distribution is given by

$$c^2 = \frac{(1-\alpha)\cdot(N-1)\cdot N\cdot\lambda_1^2 + 2\alpha\cdot\lambda_2^2}{((1-\alpha)\cdot(N-1)\cdot\lambda_1 + \alpha\cdot\lambda_2)^2} - 1\,. \tag{8}$$

Case 2: $1/(N-1) \leq c_0 < 1$: A possible setting of the scaling parameters and weights is $\lambda_1 = 1$, $\lambda_2 = N-1$, and $\alpha = \left((N-1)\cdot c_0 - 1\right)/\left(N-2\right)$.

Case 3: $c_0 \geq 1$: It is sufficient to consider the case N = 2, i.e., a hyperexponential distribution with 2 phases. Recall, that $\mathcal{H}_2$ is also a subset of $\mathcal{H}_N$. A possible setting of the scaling parameters and weights is $\lambda_1 = 1$, $\lambda_2 = 1/(2c_0)$, and $\alpha = 2(c_0-1)/(2c_0-1)$.

■

Note that Theorem 1 states that any probability density function of a nonnegative random variable can be approximated arbitrarily close by a hyper-Erlang distribution. That is, for every point of continuity of a general density function f, there exist values λ and M such that the finite hyper-Erlang density $h_{\lambda,M}(x)$ is arbitrarily close to f(x).

2.2 Hyper-Erlang Distributions as Subclass of PH

In this section, we intend to shed some light onto the relationship between sub-classes of PH distributions. Let $\mathcal{A}$ and $\mathcal{B}$ be sets of specific PH distributions. We consider three types of relationships between sets $\mathcal{A}$ and $\mathcal{B}$.

(i) $\mathcal{A} < \mathcal{B}$ means that all finite-state distributions of $\mathcal{A}$ can be represented by an appropriately selected finite-state distribution of $\mathcal{B}$ and $\mathcal{B}$ contains at least one distribution that cannot be represented by a distribution of $\mathcal{A}$ even with an infinite number of states.

(ii) $\mathcal{A} \leq_\infty \mathcal{B}$ means that all finite-state distributions of $\mathcal{A}$ can be represented by an appropriately selected finite-state distribution of $\mathcal{B}$ and $\mathcal{B}$ contains at least one distribution that can only be represented by a distribution of $\mathcal{A}$ with an infinite number of states.

(iii) $\mathcal{A} \neq \mathcal{B}$ means that none of the relationships (i) and (ii) hold.

Note, that relationship (i) means that a distribution of $\mathcal{B}$ cannot be approximated arbitrarily close by a distribution of $\mathcal{A}$, whereas in relationship (ii) this approximation is possible. According to this definition, we consider the relationship between some well-known sub-classes of phase-type distributions and their versatility in representing general distributions. In particular we consider exponential distributions (ED), hyperexponential distributions (HED), Erlang distributions (ErD), hyper-Erlang distributions (HErD), hypoexponential distributions (HoED), acyclic phase-type distributions (APHD), and phase-type distributions (PHD). A detailed definition of these distributions as well as the computation of their squared coefficient of variation c_X^2 can be found in standard textbooks (see e.g. [25]). Nevertheless, we briefly introduce the HoED, also called generalized Erlang distribution by some authors.

In fact, the HoED is an Erlang distribution where the scaling parameters are not necessarily identical in each phase. A HoED where scaling parameters λ_i are pairwise different in each phase has the probability density function

$$f_X(x) = \sum_{n=1}^{N} a_n \lambda_n e^{-\lambda_n x}, \qquad \text{where } a_n = \prod_{i=1, i \neq n}^{N} \frac{\lambda_i}{\lambda_i - \lambda_n} \quad \text{and } \lambda_n \neq \lambda_i \text{ for } n \neq i. \tag{9}$$

Similarly the probability density function can be expressed in closed-form if some of the scaling parameters are equal.

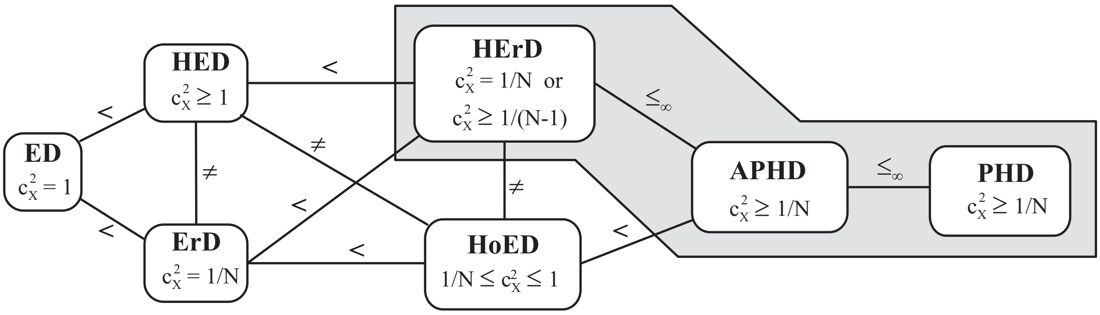

Figure 2. Relationship of sub-classes of phase-type distributions

Figure 2 shows the relationship between the distributions introduced above according to the cases (i) to (iii). The results for c_X^2 for the HErD are presented in Theorem 1. The classes of distributions that are dense in the set of general distributions, i.e., HErD, APHD, and PHD, are combined in the gray shaded area. Most of the relationships can be simply explained by comparing the possible range of the squared coefficient of variation that a distribution can take on. Since the HErD is a generalization of ErD and HED, which are both generalizations of ED, these distributions can be represented by a distribution of HErD with the same number of states. Comparing the possible range of the squared coefficient of variation, we see that the ED is less expressive than HED and ErD, which are again less expressive than HErD. Similarly, the relationship ErD < HoED < APHD can be explained. Furthermore, HED is clearly different from ErD and HoED. The relationship between APHD and PHD is true, since there is a simplification of the distributions from PHD to APHD. Nevertheless, a PHD can be approximated arbitrarily close by an APHD [3].

The relationship between HErD and HoED as well as HErD and APHD needs a further explanation. Comparing the squared coefficient of variation range of HoED and HErD we see that HoED are less expressive than HErD. On the other hand, one might think that an N-state HoED can be represented by an (N+1)-state HErD such that the relationship HoED < HErD holds, but it can be shown by a comparison of the densities (see Eqs. (1) and (9)) that even a two-state HoED with distinct scaling parameters cannot be represented by any finite-state HErD. Nevertheless, due to Theorem 1 all distributions of HoED can be approximated arbitrarily close by a distribution of HErD but not vice versa. Comparing HErD with APHD, we see that HErD is a special case of APHD, but any APHD can be approximated arbitrarily close with a HErD (see Theorem 1), hence the relationship "$\leq_\infty$" holds. On the other hand considering a finite number of states, HErD is less expressive than APHD with the same reason as for HoED. We conclude from this comparison that HErD is the most versatile sub-class of APHD, since HErD also provides full flexibility but can be more efficiently tuned to match general distributions than APHD as shown in Section 3.

2.3 Matching Moments with Hyper-Erlang Distributions

In this section we consider the problem of adjusting the parameters of a hyper-Erlang distribution to match the first three empirical moments $\hat{\mu}_j$, j = 1,2,3, as estimated from trace data. First of all we consider the case M = 2, i.e., a mixture of two Erlang distributions with number of phases r_1 and r_2, respectively. Without loss of generality we assume $r_1 \leq r_2$. The moment matching problem for mixtures of Erlang distributions was extensively studied by Johnson and Taaffe [15], [17], [18]. They provided conditions under which the problem is solvable, but determined the solution only for the case $r_1 = r_2$. Suppose n* is the smallest integer that satisfies the inequality

$$n^* > \max\left\{\frac{\hat{\mu}_1^2}{\hat{\mu}_2 - \hat{\mu}_1^2}, \frac{\hat{\mu}_2^2}{\hat{\mu}_1\hat{\mu}_3 - \hat{\mu}_2^2} - 1\right\}, \tag{10}$$

then the following cases must be distinguished (see [15]):

(i) If r_1, r_2 < n*, then the first three moments cannot be matched exactly.

(ii) If $r_1 < n^* \le r_2$, then the moment matching problem has (at least) one solution.

(iii) If $n^* \le r_1 = r_2$, then the moment matching problem has a unique solution.

(iv) If $n^* \le r_1 < r_2$, then the moment matching problem has (at least) two solutions.

In case (iii) a simple closed-form solution exists (see [17]) whereas in cases (ii) and (iv) the solution can only be determined numerically. In fact, the roots of a polynomial of degree five must be computed. In Appendix A.1 we show how to determine the parameters λ_1, λ_2, and α_1 ($\alpha_2 = 1-\alpha_1$) for cases (ii) to (iv). Our results are in contrast to the results from Schmickler who determined a polynomial of degree six for the matching problem, but did not provide any conditions when it gives feasible solutions [24]. In fact, we were not able to find any correct solution with his polynomial.

Suppose now we have given a hyper-Erlang distribution with M Erlang branches and number of phases of each branch r_m, m = 1,…,M. For matching the first three empirical moments only three of the 2M–1 free continuous parameters are needed. The moments of a HErD with more than two branches can be reduced to the two-branch case by subtracting the contributions of the other branches, if their parameters are known. To be precise, let i_1 and i_2, with $1 \le i_1 < i_2 \le M$, be the indices of the two Erlang branches to be used for the moment matching. Then we define the j-th *reduced moment* $\tilde{\mu}_j$ by

$$\tilde{\mu}_j = \left(\hat{\mu}_j - \sum_{m=1, m \neq i_1, i_2}^{M} \alpha_m \frac{(r_m + j - 1)!}{(r_m - 1)!} \frac{1}{\lambda_m^j} \right) \cdot \frac{1}{\beta}, \tag{11}$$

where $\beta = \alpha_{i_1} + \alpha_{i_2}$ is the portion of the two branches used for matching the moments. Note that the reduced moments are not necessarily moments of a distribution function. A sufficient condition for values $\tilde{\mu}_1$, $\tilde{\mu}_2$, and $\tilde{\mu}_3$ to be moments of a distribution with support on [0, ∞] is

$$\min\left\{ \tilde{\mu}_1, \tilde{\mu}_2 - \tilde{\mu}_1^2, \tilde{\mu}_1 \tilde{\mu}_3 - \tilde{\mu}_2^2 \right\} > 0. \tag{12}$$

If condition (12) holds we can apply the procedure for matching the moments $\tilde{\mu}_1$, $\tilde{\mu}_2$, and $\tilde{\mu}_3$ with a mixture of the two Erlang branches i_1 and i_2. Denote α' and $(1-\alpha')$ the initial probabilities of this two-branch solution. Finally, we have to set the weights α_{i_1} and α_{i_2} properly, i.e., $\alpha_{i_1} = \alpha' \cdot \beta$ and $\alpha_{i_2} = (1-\alpha') \cdot \beta$.

3 An EM Algorithm for Fitting Hyper-Erlang Distributions

3.1 Fitting Mixture-Densities with the EM Algorithm

The mixture-density parameter estimation problem is probably one of the most widely used applications of the EM algorithm [6]. In this case, we assume the following probabilistic model

$$p(x|\Theta)=\sum_{m=1}^{M}\alpha_m p_m(x|\theta_m), \tag{13}$$

where the parameters are $\Theta = (\alpha_1,\ldots,\alpha_M, \theta_1,\ldots,\theta_M)$ such that $\alpha_1+\alpha_2+\ldots+\alpha_M = 1$ and each p_m is a density function parameterized by θ_m. In other words, we assume that M component densities are mixed using M mixing coefficients α_m. Note that in general θ_m can be a vector of parameters for each density function p_m, but it is a single value in our HErD fitting method.

Let $\mathcal{T} = \{x_1,\ldots,x_K\}$ be a data set of measurements supposedly drawn from the distribution (13). That is, we assume that these data values are drawn from independent and identically distributed random variables with probability density function (13). The log-likelihood expression for this mixture density for the trace $\mathcal{T}$ is given by

$$\log L(\Theta|\mathcal{T})=\log\prod_{k=1}^{K}p(x_k|\Theta)=\sum_{k=1}^{K}\log\left(\sum_{m=1}^{M}\alpha_m p_m(x_k|\theta_m)\right), \tag{14}$$

which is difficult to optimize because it contains the logarithm of a sum. If we consider $\mathcal{T}$ as incomplete data and assume the existence of unobserved data items $y_k \in \{1,\ldots,M\}$, $k=1,\ldots,K$, whose values inform us which component density "generates" each data item of $\mathcal{T}$, the likelihood expression can be significantly simplified. That is, we assume $y_k = m$ if the k-th sample x_k was generated by the m-th mixture component p_m. If we know the values $\mathbf{y} = (y_1,\ldots,y_K)$ the log-likelihood expression of Eq. (14) becomes

$$\log L(\Theta|\mathcal{T},\mathbf{y})=\sum_{k=1}^{K}\log\left(\alpha_{y_k}p_{y_k}(x_k|\theta_{y_k})\right). \tag{15}$$

The problem in dealing with Eq. (15) is, that we do not know the values of y_k. If we assume y_k as random values drawn from a random variable Y, we can derive an expression for the probability mass function (pmf), denoted by q(y), of the unobserved data. First, we guess at parameters for the mixture density, i.e., we guess that $\hat{\Theta} = \left(\hat{\alpha}_1,\ldots,\hat{\alpha}_M,\hat{\theta}_1,\ldots,\hat{\theta}_M\right)$ are the appropriate parameters. Given $\hat{\Theta}$, we can easily compute the mixture components $p_m\left(x_k \middle| \hat{\theta}_m\right)$ for each k and m. Keeping in mind that α_m is the probability of choosing the m-th mixture component we can compute the pmf of the unobserved data given the observed data $\mathcal{T}$ and the estimates $\hat{\Theta}$ using Bayes's rule

$$q\left(y_k \middle| x_k,\hat{\Theta}\right) = \frac{q\left(y_k \middle| \hat{\Theta}\right)\cdot p\left(x_k \middle| y_k,\hat{\Theta}\right)}{p\left(x_k \middle| \hat{\Theta}\right)} = \frac{\hat{\alpha}_{y_k}\cdot p_{y_k}\left(x_k \middle| \hat{\theta}_{y_k}\right)}{\sum\limits_{m=1}^{M}\hat{\alpha}_m\cdot p_m\left(x_k \middle| \hat{\theta}_m\right)} \tag{16}$$

and

$$q\left(\mathbf{y} \middle| \mathcal{T},\hat{\Theta}\right) = \prod_{k=1}^{K} q\left(y_k \middle| x_k,\hat{\Theta}\right), \tag{17}$$

where $\mathbf{y} \in \{1,\ldots,M\}^K$ is an instance of the unobserved data independently drawn from Y. The expected value of the complete-data log-likelihood with respect to the unknown random variable Y given the observed data $\mathcal{T}$ and the current parameter estimates $\hat{\Theta}$, is given by

$$Q\left(\Theta,\hat{\Theta}\right) = E\left[\log L\left(\Theta \middle| \mathcal{T},Y\right) \middle| \mathcal{T},\hat{\Theta}\right] = \sum_{\mathbf{y}\in\{1,\ldots,M\}^K} \log L\left(\Theta \middle| \mathcal{T},\mathbf{y}\right)\cdot q\left(\mathbf{y} \middle| \mathcal{T},\hat{\Theta}\right). \tag{18}$$

Inserting Eqs. (15) and (17) into Eq. (18) we get

$$Q\left(\Theta,\hat{\Theta}\right) = \sum_{\mathbf{y}\in\{1,\ldots,M\}^K} \sum_{k=1}^{K} \log\left(\alpha_{y_k} p_{y_k}\left(x_k \middle| \theta_{y_k}\right)\right)\cdot \prod_{i=1}^{K} q\left(y_i \middle| x_i,\hat{\Theta}\right) \tag{19}$$

and rearranging the sums and the product results in (see Appendix A.2)

$$Q\left(\Theta,\hat{\Theta}\right) = \sum_{m=1}^{M}\sum_{k=1}^{K} \log\left(\alpha_m\right)\cdot q\left(m \middle| x_k,\hat{\Theta}\right) + \sum_{m=1}^{M}\sum_{k=1}^{K} \log\left(p_m\left(x_k \middle| \theta_m\right)\right)\cdot q\left(m \middle| x_k,\hat{\Theta}\right). \tag{20}$$

Note that the computation of the expectation in Eq. (18) constitutes the E-step of the EM algorithm. In general, the main difficulty in computing this expectation is to obtain an expression for the marginal distribution of the unobserved data. However, for the mixture density problem discussed in this section the marginal distribution can be simply computed by Eqs. (16) and (17). The M-step of the EM algorithm is to maximize the expectation computed in the E-step with respect to Θ. To maximize Eq. (20), we can maximize the term containing α_m (first sum in Eq. (20)) and the term containing θ_m (second sum in Eq. (20)) independently since they are not related. According to [19], a Lagrange multiplier can be applied to find the expression for α_m, resulting in

$$\alpha_m = \frac{1}{K}\sum_{k=1}^{K} q\left(m \middle| x_k, \hat{\Theta}\right). \qquad (21)$$

The computation of θ_m depends on the form of the mixture density component p_m and is addressed in the next section for a mixture of Erlang distributions.

3.2 Application to Hyper-Erlang Distributions

In this section we develop the formulas for application of the EM algorithm to the mixture density parameter estimation problem when the m-th mixture component is an Erlang distribution with a fixed number of phases, i.e.,

$$p_m\left(x_k \middle| \lambda_m\right) = \frac{(\lambda_m x_k)^{r_m - 1}}{(r_m - 1)!}\lambda_m e^{-\lambda_m x_k}, \qquad (22)$$

and the mixture distribution is described by the parameter vector $\Theta = (\alpha_1,\ldots,\alpha_M,\lambda_1,\ldots,\lambda_M)$. The parameters α_m, m = 1,…,M, that maximize Eq. (20) are determined according to Eq. (21). In order to determine the parameters λ_m, m = 1,…,M, that maximize Eq. (20) we set the derivatives with respect to λ_m of Eq. (20) equal to zero

$$\sum_{k=1}^{K} q\left(m \middle| x_k, \hat{\Theta}\right) \frac{\partial}{\partial \lambda_m} \log\left(p_m\left(x_k \middle| \lambda_m\right)\right) = 0. \qquad (23)$$

Putting Eq. (22) into Eq. (23) and applying logarithm-rules we get

$$\lambda_m = \frac{r_m \cdot \sum_{k=1}^{K} q\left(m \middle| x_k, \hat{\Theta}\right)}{\sum_{k=1}^{K} q\left(m \middle| x_k, \hat{\Theta}\right) \cdot x_k}. \qquad (24)$$

Note that Eqs. (21) and (24) together with Eq. (16) are simple closed-form expressions for the parameters of a HErD according to a given number of Erlang branches M and a given number of phases r_m per branch.

3.3 Implementation Issues

A high-level pseudo-code representation of the EM algorithm tailored to the parameter estimation of HErD is presented in Figure 3. Note that each iteration (see steps (2) to (7) in Figure 3) is guaranteed to increase the log-likelihood value and the algorithm is guaranteed to converge to a local maximum of the likelihood function [19]. To check whether convergence is reached, we compute in each iteration either

(i) the maximal difference of the values of the parameter vectors of successive iterations,

(ii) the relative difference of the log-likelihood values of successive iterations,

and stop the algorithm when the computed difference is below a predefined ε, e.g. $\varepsilon = 10^{-6}$. The computational complexity of the E-step is O(M·K) when computing the numerator and denominator of the unobserved data pmf separately. The complexity of the M-step is also O(M·K). Thus, the overall computational complexity for one iteration of the EM algorithm is O(M·K). Note that the log-likelihood (see Eq. (14)) can be computed without additional effort during the E-step of the fitting algorithm.

A straightforward computation of the Erlang densities (22) can exhibit numerical difficulties, since for a high number of Erlang phases (e.g. $r > 50$) large factorials and large power values must be computed. To avoid these difficulties, we suggest an evaluation of (22) in logarithmic form, i.e.,

$$p_m\left(x_k \middle| \lambda_m\right) = \lambda_m e^{(r_m-1)\ln(\lambda_m x_k) - \ln(r_m-1)! - \lambda_m x_k}, \tag{25}$$

with pre-computed logarithms of the factorial values, i.e.,

$$\ln r! = \sum_{i=1}^{r} \ln i. \tag{26}$$

On a standard PC with 3 GHz Pentium CPU running the operating system Linux, the EM algorithm requires about 2.4 seconds for 100 iterations when fitting a HErD with $M = 5$ Erlang branches to a trace with $K = 10^4$ samples. The overall number of iterations required to achieve convergence depends on several factors, i.e., the initial setting of α_m and λ_m, the number of Erlang branches M, and the trace data. However, for small values of M (i.e.,

$M \le 10$) the algorithm converges faster than for larger values of M, since fewer parameters have to be optimized. With $M \le 10$ the number of iterations is almost always less then 100 to reach convergence with $\varepsilon = 10^{-6}$.

```
Choose initial parameter estimates $\hat{\Theta} = \left(\hat{\alpha}_1,\ldots,\hat{\alpha}_M,\hat{\lambda}_1,\ldots,\hat{\lambda}_M\right)$
REPEAT
    Compute $p_m\left(x_k \mid \hat{\lambda}_m\right)$ for m=1,…,M and k=1,…,K according to Eq. (25)
    E-step: Compute the pmf of the unobserved data for m=1,…,M and k=1,…,K
        $q\left(m \mid x_k,\hat{\Theta}\right) = \hat{\alpha}_m \cdot p_m\left(x_k \mid \hat{\lambda}_m\right) \Big/ \sum_{i=1}^{M} \hat{\alpha}_i \cdot p_i\left(x_k \mid \hat{\lambda}_i\right)$
    M-step: Compute $\alpha_m$ and $\lambda_m$ that maximize Eq. (20) for m=1,…,M
        $\alpha_m = \frac{1}{K}\sum_{k=1}^{K} q\left(m \mid x_k,\hat{\Theta}\right)$ and $\lambda_m = r_m \cdot \sum_{k=1}^{K} q\left(m \mid x_k,\hat{\Theta}\right) \Big/ \sum_{k=1}^{K} q\left(m \mid x_k,\hat{\Theta}\right) \cdot x_k$
    set $\hat{\Theta} := \Theta$
UNTIL convergence reached according to criterion (i) or (ii)
RETURN optimal parameter vector $\Theta = (\alpha_1,\ldots,\alpha_M,\lambda_1,\ldots,\lambda_M)$
```

Figure 3. Pseudo-code of the EM algorithm tailored to hyper-Erlang distributions

4 Finding the Best Hyper-Erlang Distribution

4.1 Optimizing the Discrete Parameters of a Hyper-Erlang Distribution

With the EM algorithm presented in Section 3.3 we can optimize the continuous parameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ of a HErD for a predefined setting of the number of Erlang branches M and number of phases of each Erlang branch r_m, m = 1,…,M. However, in order to find the "best" N-state HErD we have to consider all HErD out of the set $\mathcal{H}_N$ as candidates. In other words, we have to determine a setting of the number of phases $r_1,\ldots,r_M$ that maximizes the log-likelihood. Due to the efficiency of the algorithm it is feasible to enumerate all possible settings of M and $r_1,\ldots,r_M$ and to fit for each such setting a HErD, if N is small (i.e, $N \le 10$) and K is not too large (i.e., $K \le 10^6$). Comparing the fitted HErD according to their log-likelihood values and choosing the one with the maximum log-likelihood value gives the best HErD in this case. Formally, we denote the *discrete parameter setting* of a HErD f(x; M, **r**, $\boldsymbol{\alpha}$, λ) by the tuple (M, **r**). The following lemma provides a recursive formula to compute the overall number of settings of an N-state HErD, denoted by S_N. This recursion can also be used to enumerate all possible settings in algorithmic fashion.

Lemma 2: The overall number of different N-state settings, S_N, is given by $\varphi_N(N,0)$, where

$$\varphi_m(n,j)=\sum_{i=j}^{\lfloor n/m\rfloor}\varphi_{m-1}(n-i,i)\quad\text{and}\quad \varphi_1(n,j)=\begin{cases}0, & \text{if } j>n,\\ 1, & \text{if } j\le n.\end{cases} \tag{27}$$

Proof: If we allow Erlang branches with zero phases, we can assume N Erlang branches where N–M branches have zero phases. To compute S_N, we have to count the number of possibilities to choose values $r_n \in \{0,1,\ldots,N\}$, $n = 1,\ldots,N$, such that $0 \le r_1 \le\ldots \le r_N$ and $r_1+\ldots+r_N = N$. A first observation is that $r_1 = 0$ or $r_1 = 1$, otherwise the sum of the r_n-values would be larger than N. Assume now $r_1 = \ldots = r_{N-M} = 0$ then it must hold $r_{N-M+1} \le N/M$, otherwise the sum of the r_n-values would be larger than N. Thus, for every possible choice of $r_{N-M+1} = i$ with $i \in \{0,\ldots,\lfloor N/M\rfloor\}$, we have to count the number of possibilities for choosing $r_{N-M+2},\ldots,r_N$ such that $i \le r_{N-M+2} \le \ldots \le r_N$ and $r_{N-M+2}+\ldots+r_N = N–i$. This is exactly what the recursive function $\varphi_m(n,j)$ computes, i.e., it counts the number of possibilities to choose m values $r_{N-m+1},\ldots,r_N$ such that $j \le r_{N-m+1} \le\ldots \le r_N$ and $r_{N-m+1}+\ldots+r_N = n$. To obtain the recursion in Eq. (27) we have to sum up the number of possibilities to choose m–1 values for every possible choice of r_{N-m+1}, i.e., $r_{N-m+1} = j + i$ with $i = 0,\ldots,\lfloor n/m\rfloor$. ■

In fact, for N = 5, 6, 7, 8, 9, 10 only S_N = 7, 11, 15, 22, 30, 42 settings exist. Unfortunately, for larger N the number of settings grows exponentially, e.g., for N = 20 we have 627 different settings. Thus, for large values of N, i.e., N > 10, it is not feasible to apply the EM algorithm for every possible setting. The same holds when fitting even one setting takes some time, which may be the case for very large traces (e.g. $K > 10^7$ samples). In these cases we recommend using one of the following strategies:

(i) **Progressive pre-selection:** In a first round enumerate all possible settings and apply the EM algorithm until convergence with $\varepsilon = 10^{-3}$ is reached. This requires only a few iterations for each setting. Select the settings with the best log-likelihood values and put them into a priority queue. We commonly consider at least 5 and at most 50 settings in this round. Then start a second round with continuing iteration of the selected HErD until convergence with $\varepsilon = 10^{-4}$ is reached. Finally, start a third round with the 50% best of the priority queue until $\varepsilon = 10^{-6}$. Experimental results when applying this strategy are presented in Section 5.

(ii) **Special structures:** If the empirical distribution of the trace has a small squared coefficient of variation (i.e., $c^2 < 1$) we recommend to fit the HErD only with one, two, or three Erlang branches, i.e., $M = 1$, $M = 2$, or $M = 3$. Note that the number of N-state settings with $M \leq M_{max}$ Erlang branches is $\varphi_{M_{max}}(N,0)$, e.g., for $N = 50$ and $M_{max} = 2$ only 26 settings must be fitted. For monotonically decreasing empirical distributions with large squared coefficient of variation (i.e., $c^2 > 1$) we recommend to fit the HErD only with N, N–1, or N–2 Erlang branches. Note that $M = N$ corresponds to a hyperexponential distribution, which was shown to fit heavy-tailed distributions with large squared coefficient of variation quite well in [10].

(iii) **Body/tail fitting:** Fit the body of a distribution, i.e., the part of the distribution that contains most samples, with a (say) 10-state HErD with $M = 1$, 2, and 3 Erlang branches, which requires only 1+5+8 = 14 runs of the EM algorithm. Fit the tail of distribution with a (say) 5-state hyperexponential distribution, i.e., $M = 5$ and $r_1 = \ldots = r_M = 1$. Apply this combined body/tail fitting on a 15-state HErD. Thus, an overall number of 14 settings must be evaluated. A good application example for this approach is the Pareto-II distribution discussed in Section 5.

The first approach works automatically, but requires additional effort, which is not required if the other two variants are used. In practice especially variant (ii) works well, but requires some pre-analysis of the data and an experienced user to decide about a good range for the discrete parameters. The separate fitting of body and tail, as suggested in (iii), is often used for heavy-tailed distributions (e.g., [23]), but requires an appropriate definition of body and tail, and the number of states used for their approximation. Additionally, the low complexity of the presented HErD fitting method allows us to optimize the body and the tail fitting parameters together, which is preformed separately in [14]. The results from [8] seem to indicate that a common fitting algorithm might yield excellent results for fitting heavy-tailed distributions.

4.2 Combining Moment Matching with Likelihood Maximization

With the moment matching procedure described in Section 2.3 we can match the first three moments of a HErD with M ≥ 2 Erlang branches by only adjusting two of the M Erlang branches, i.e., the M–2 remaining Erlang branches (scale parameters and initial probabilities) remain unchanged. There are M·(M–1)/2 possibilities to choose two Erlang branches for matching. To integrate moment matching and likelihood maximization, we propose to try all possibilities, which can be done rather efficiently, and to finally choose the one with the largest log-likelihood. For the combination of moment matching with the EM algorithm for log-likelihood maximization two strategies are considered:

(i) Apply the moment matching for a fitted HErD, i.e., first use G-FIT for maximum likelihood estimation and then adjust the moments in the result that G-FIT found. We tried this approach with quite good results, i.e., the log-likelihood value decreases only marginal when the first three moments are matched. It is also reasonable to iterate between G-FIT and moment matching, i.e., first use G-FIT then moment matching and then G-FIT again with initial distribution obtained from the previous moment matching. We applied this approach for the Pareto-II distribution discussed in Section 5.

(ii) To fit heavy-tailed distributions with a (say) 15-state HerD, a good strategy is to first apply G-FIT for finding the best (say) 10-state HErD as described in Section 4.1. Then we suggest to add a new Erlang branch with one phase, zero initial probability and arbitrary scale parameter and to use this branch together with one of the remaining branches for moment matching. Indeed, one of the remaining branches should be chosen such that the log-likelihood value is maximized. After matching the moments we apply G-FIT where we use the 11-state HErD as initial distribution. Repeating this approach five times, we generate the 15-state HErD as desired (including a 5-states hyperexponential distribution responsible for the tail). Usually the repeated use of G-

FIT is rather efficient, since only a small number of iterations is required after the distribution has been modified due to moment matching.

The second strategy is quite useful for heavy-tailed distributions, since our experience is that the log-likelihood measure is mainly influenced by the body of a distribution and not by the tail (see example in Section 5.2). Therefore, the EM algorithm tends to fit the body perfectly for the cost of neglecting the tail in some sense. In contrast, the moments matching approach captures the tail behavior of a heavy-tailed distribution much better, since heavy-tailed distributions are characterized by their low order moments, which differ in orders of magnitude.

Recall, that the EM algorithm converges to a local maximum of the log-likelihood function. Thus, the vector $\Theta = (\alpha_1,\ldots,\alpha_M, \lambda_1,\ldots,\lambda_M)$ of the continuous parameters of the HErD returned by the EM algorithm depends on the initial estimates $\hat{\alpha}_1,\ldots,\hat{\alpha}_M$ and $\hat{\lambda}_1,\ldots,\hat{\lambda}_M$. Our experience shows, that reinitializing the EM algorithm with a HErD that matches the first three moments exactly, forces the EM algorithm to converge to a local maximum with a better tail fitting.

5 Experimental Results

5.1 Fitting Hyper-Erlang Distributions to Synthetically Generated Traces

In the experiments a hyper-Erlang distribution (HErD) and an acyclic phase-type distribution (APHD) are fitted for given traces with 10^4 samples drawn from known distributions. In particular we consider two Weibull distributions with scale parameter $\eta = 1.0$ and shape parameter $\beta = 0.5$ and $\beta = 5.0$, respectively, and a uniform distribution with left and right boundary equal to 0.5 and 1.5. In addition we consider a Pareto-like distribution with heavy-tail index $\alpha = 1.5$ and $b = 2.0$. This distribution was previously used in [13] as an example of a heavy-tailed distribution, which is not monotonically decreasing. According to [13] it is denoted a Pareto-II distribution. Furthermore, we consider the shifted exponential distribution as well as the matrix exponential distribution, which are part of a set of benchmark distributions for PH fitting algorithms defined in [3]. The non-standard density functions used for the experiments are summarized in Figure 4.

Weibull(η,β):	$f_{\text{weibull}}(x;\eta,\beta)=\frac{\beta}{\eta}\cdot\left(\frac{x}{\eta}\right)^{\beta-1}\cdot e^{-\left(\frac{x}{\eta}\right)^{\beta}}$
Pareto-II(α,b):	$f_{\text{paretoII}}(x;\alpha,b)=\frac{b^{\alpha}e^{-b/x}}{\Gamma(\alpha)}x^{-\alpha-1}$
Shifted exponential (SE):	$f_{SE}(x)=\begin{cases}\frac{1}{2}e^{-x} & ,0\le x<1\\ \frac{1}{2}e^{-x}+\frac{1}{2}e^{-(x-1)} & ,x\ge 1\end{cases}$
Matrix exponential (ME):	$f_{ME}(x)=\left(1+\frac{1}{(2\pi)^2}\right)\cdot\left(1-\cos(2\pi x)\right)\cdot e^{-x}$

Figure 4. Probability density functions of considered distributions

With respect to the set of distributions we compared the fitting quality of the HErD found by G-FIT with the quality of the APHD found by the tool PH-FIT [14]. The PH-FIT tool approximates the optimal parameter set of an APHD by minimizing a predefined distance measure with a non-linear optimization algorithm. This algorithm uses an iterative linearization method based on numerical computation of partial derivatives and the simplex method to determine the direction in which the distance measure decreases most. In the presented comparison the fitting parameters of the PH-FIT tool were the following: only body fitting is applied (i.e., no separate tail fitting) and the distance of the original and the approximate distributions is calculated up to the largest sample value. Furthermore, we run PH-FIT with 3 rounds (i.e., starting from 3 different initial guesses) and at most 200 modifications in each round.

Figure 5 shows the empirical density functions for the six traces as well as the density functions for the fitted HErD and APHD with N = 5 states and N = 10 states, respectively. For some of the distributions also results when fitting a HErD with 50 states are plotted. Densities of traces are approximated by histograms with intervals of width 0.05. The results for G-FIT are obtained by fitting a HErD with the algorithm of Figure 3 for all possible discrete parameter settings. Recall, that for N = 5 only 7 settings and for N = 10 only 42 settings are considered. The EM algorithm stops when convergence is reached according to the log-likelihood criterion with ε = 10^{-6} (see criterion (ii) in Section 3.3). From the curves of Figure 5 we conclude that the fitting quality of HErD is almost always as good as the quality for APHD. Moreover, in some cases the results for HErD are better than for APHD, e.g. when fitting the uniform distribution with 5 states. The reason why PH-FIT did not find the best solution in these cases is that the optimization process got stuck in a local optimum.

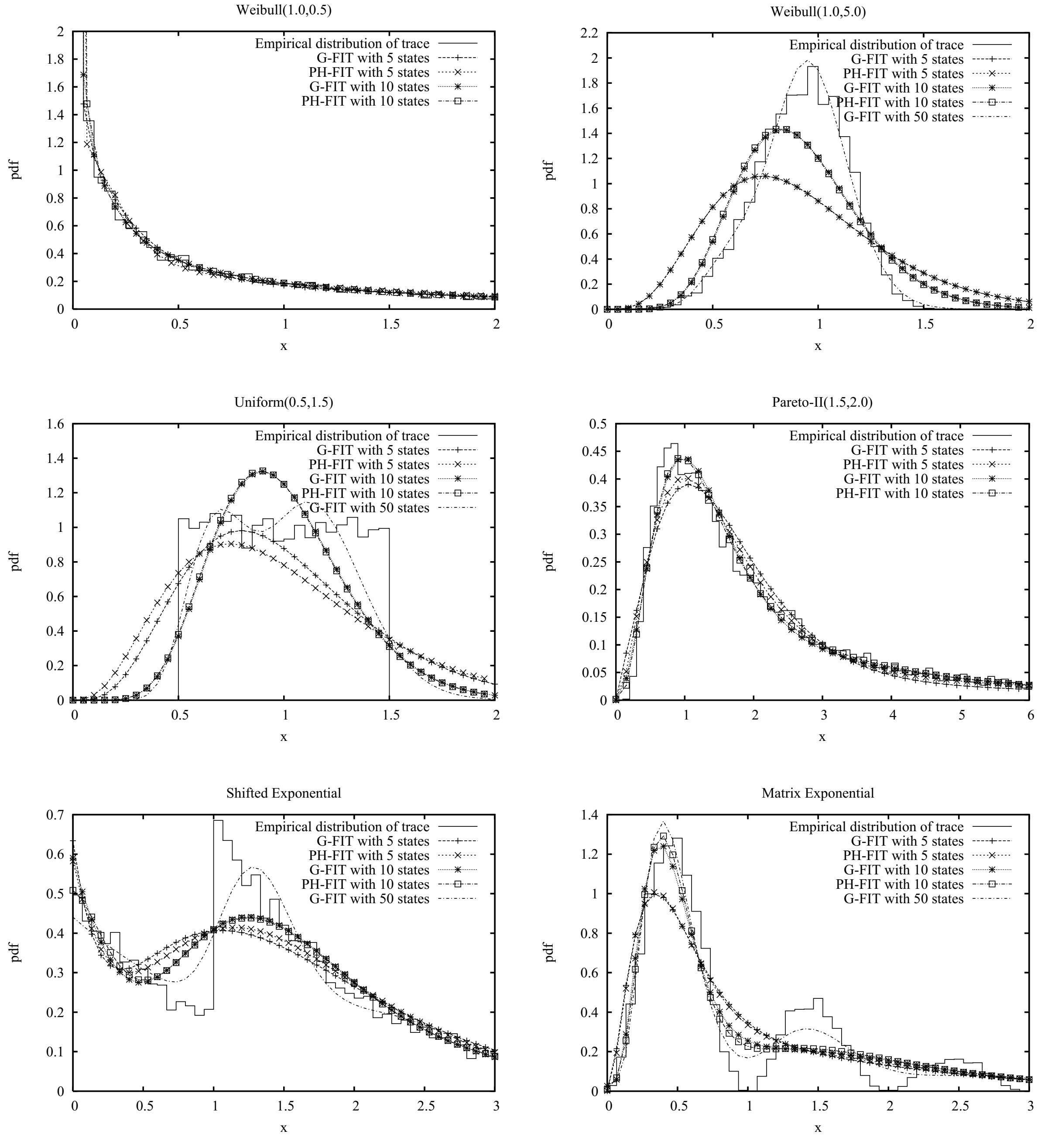

Figure 5. Densities of fitted HErD and APHD for synthetically generated traces

		Trace	5 states		10 states		20 states	Special cases
			G-FIT	PH-FIT	G-FIT	PH-FIT	G-FIT	G-FIT
Weibull(1.0,0.5)	1. Moment	1.99	1.99 (0.0%)	1.98 (0.5%)	1.99 (0.0%)	1.91 (4.3%)	1.99 (0.0%)	1.99 (0.0%)
	2. Moment	25.61	22.47 (12.3%)	20.98 (18.1%)	23.78 (7.1%)	17.77 (30.6%)	24.26 (5.3%)	25.76 (0.6%)
	3. Moment	995.13	512.55 (48.5%)	422.44 (57.5%)	638.04 (35.9%)	300.78 (69.8%)	701.43 (29.5%)	1156.57 (16.2%)
	Log-likelihood		-11068.87	-11364.43	-10998.60	-11258.91	-10986.43	-10992.01
	CPU time [sec]		13 (12)	249	170 (72)	461	268	55
	Phase lengths		1,1,1,1,1		1,1,1,1,1,1,1,1,1,1		1,...,1,2,2	1,1,1,1,1,1,1,1
Weibull(1.0,5.0)	1. Moment	0.92	0.92 (0.0%)	0.92 (0.1%)	0.92 (0.0%)	0.92 (0.0%)	0.92 (0.0%)	0.92 (0.0%)
	2. Moment	0.89	1.02 (14.1%)	1.02 (13.9%)	0.93 (4.6%)	0.93 (4.5%)	0.90 (1.1%)	0.89 (0.1%)
	3. Moment	0.90	1.31 (45.6%)	1.31 (45.3%)	1.03 (14.4%)	1.03 (14.4%)	0.93 (3.6%)	0.91 (0.4%)
	Log-likelihood		-1673.19	-1674.49	394.68	391.10	1092.56	1402.30
	CPU time [sec]		2 (1)	217	27 (14)	434	139	34
	Phase lengths		5		10		1,2,17	16,34
Uniform(0.5,1.5)	1. Moment	1.00	1.00 (0.0%)	1.00 (0.1%)	1.00 (0.0%)	1.00 (0.0%)	1.00 (0.0%)	1.00 (0.0%)
	2. Moment	1.08	1.19 (10.6%)	1.24 (15.1%)	1.09 (1.4%)	1.09 (1.4%)	1.08 (0.6%)	1.08 (0.7%)
	3. Moment	1.24	1.66 (33.9%)	1.85 (49.2%)	1.30 (5.2%)	1.31 (5.3%)	1.27 (2.7%)	1.27 (2.2%)
	Log-likelihood		-3105.56	-3804.21	-1852.51	-1855.00	-1827.56	-1166.51
	CPU time [sec]		2 (1)	182	28 (13)	386	137	13
	Phase lengths		5		10		2,2,2,3,11	21,29
Pareto-II(1.5,2.0)	1. Moment	4.34	4.34 (0.0%)	3.89 (10.3%)	4.34 (0.0%)	3.78 (12.9%)	4.34 (0.0%)	4.34 (0.0%)
	2. Moment	1057.62	323.6 (69.4%)	117.6 (88.9%)	340.7 (67.8%)	105.9 (90.0%)	911.7 (13.8%)	978.0 (7.5%)
	3. Moment	1768568	100147 (94.3%)	12348 (99.3%)	114715 (93.5%)	12579 (99.3%)	1389436 (21.4%)	1674493 (5.3%)
	Log-likelihood		-20340.91	-20236.15	-20084.11	-20108.91	-19980.79	-20093.74
	CPU time [sec]		2 (2)	246	62 (23)	423	156	29
	Phase lengths		1,1,3		1,2,3,4		1,1,2,3,6,7	1,1,1,1,2,4
Shifted Exponential	1. Moment	1.51	1.51 (0.0%)	1.51 (0.1%)	1.51 (0.0%)	1.51 (0.0%)	1.51 (0.0%)	1.51 (0.0%)
	2. Moment	3.58	3.61 (1.0%)	3.59 (0.3%)	3.61 (0.8%)	3.61 (1.0%)	3.59 (0.3%)	3.53 (1.5%)
	3. Moment	11.55	11.96 (3.5%)	11.69 (1.2%)	11.79 (2.1%)	12.47 (7.9%)	11.59 (0.3%)	11.01 (4.7%)
	Log-likelihood		-13376.14	-13327.48	-13265.52	-13253.28	-13184.11	-13148.89
	CPU time [sec]		5 (3)	140	90 (20)*	205	116	309
	Phase lengths		1,1,3		1,3,6		1,1,2,4,12	1,13,17,19
Matrix Exponential	1. Moment	1.06	1.06 (0.0%)	1.06 (0.0%)	1.06 (0.0%)	1.06 (0.2%)	1.06 (0.0%)	1.06 (0.0%)
	2. Moment	2.12	2.15 (1.7%)	2.10 (0.7%)	2.16 (1.9%)	2.10 (1.1%)	2.13 (0.4%)	2.13 (0.5%)
	3. Moment	6.57	6.66 (1.2%)	6.20 (5.6%)	7.24 (10.2%)	6.11 (7.0%)	6.59 (0.2%)	6.53 (0.7%)
	Log-likelihood		-9278.27	-9230.53	-8895.56	-8778.50	-8748.19	-8581.07
	CPU time [sec]		4 (2)	142	60 (17)	285	107	174
	Phase lengths		2,3		1,4,5		1,5,6,8	5,12,15,18

Table 1. Quality indices of fitted HErD and APHD for synthetically generated traces

Table 1 presents several quality indices for the considered distributions. In particular, the first three moments for each of the six traces as well as the fitted HErD and APHD are presented. Relative errors of the fitted distributions are presented in brackets behind the absolute values. Furthermore, Table 1 contains for each trace the log-likelihood value and the CPU time required by G-FIT and PH-FIT. In the last row of each distribution the optimal number of phases of each Erlang branch found by G-FIT is shown. Recall, that for N = 5 and N = 10 the G-FIT results are found from the best fit when fitting a HErD for all possible discrete parameter settings. Applying the progressive pre-selection (see strategy (i) in Section 4.1) yields almost always the same results but requires less CPU time (see numbers in brackets in the rows with CPU time in Table 1). In fact, only for the shifted exponential trace (log-likelihood –13280.73) the results are not as good as in the general case (indicated with an asterisk behind the brackets in Table 1). The reason for this is that with progressive pre-selection some settings may be canceled out of the priority queue in the first or second round, which would get better when running the EM algorithm until convergence with $\varepsilon = 10^{-6}$.

Results presented in Table 1 when applying G-FIT with 20 states are computed with progressive pre-selection. Note, that not all of the $\varphi_{20}(20,0) = 627$ settings are evaluated, but only a part of them which seems to be reasonable (see strategy (ii) in Section 4.1). In fact, for the Weibull(1.0,0.5) trace we considered all settings with $M \geq 12$ Erlang branches (67 settings), for the Weibull(1.0,5.0) trace and the matrix exponential trace we considered all settings with $M \leq 5$ Erlang branches (192 settings), and for the Pareto-II trace and the shifted exponential trace we fitted all settings with $M \leq 6$ Erlang branches (282 settings). Comparing the fitted HErD and APHD, it can be observed that for all distributions the fitting quality of the 20-state HErD is much better than that of the 10-state APHD. Moreover, the fitting process for the 20-state HErD is less time consuming as for the 10-state APHD, although the number of states is doubled.

The last column in Table 1 shows results for G-FIT for some special cases. In particular, for the Weibull(1.0,0.5) trace results for an 8-state hyperexponential distribution running the EM algorithm until convergence with $\varepsilon = 10^{-8}$ are presented. It can be observed that due to the longer iteration time the moments are matched much better than in the case when stopping the iteration with $\varepsilon = 10^{-6}$. For the Weibull(1.0,5.0) trace and the uniform trace we fitted all 50-state settings with at most two Erlang branches (26 settings) until convergence is reached with $\varepsilon = 10^{-16}$. The time requirements for the fitting process are still very small as can be observed from Table 1. For the shifted exponential trace and the matrix exponential trace we applied progressive pre-selection with 50 states and $M \leq 4$ Erlang branches (1154 settings). Finally, for the Pareto-II trace we applied the body/tail fitting approach (see strategy (iii) in Section 4.1). We used 6 states for the body and 4 states for the tail. For the body we fitted all settings with two Erlang branches (3 settings) until convergence with $\varepsilon = 10^{-10}$. As expected, the first three moments are fitted very well with this approach and the tail-behavior of the distribution is captured much better, although the log-likelihood value is worse compared to the best 10-state HErD.

Figure 6 shows the probability density function and complementary cdf (ccdf) for the Pareto-II distribution when fitting a trace with 10^6 samples. We considered the 10-state HErD distribution found by progressive pre-selection, denoted by G-FIT(1,2,3,4), the HErD distribution determined with the body/tail fitting approach, denoted with G-FIT(1,1,1,1,2,4), and a HErD distribution obtained from additionally applying the moment matching as described in strategy (i) in Section 4.2, denoted with MM+G-FIT(1,1,1,1,2,4). Comparing the results we observe that the body is fitted quite similar in all cases, whereas the tail fitting differs. In fact, the combined moment matching and likelihood maximization gives the best tail fitting, even if the log-likelihood is slightly worse, i.e., G-FIT(1,2,3,4) gives log-likelihood –1991435 and MM+G-FIT(1,1,1,1,2,4) gives log-likelihood –1993697. The tail plot of Figure 6 indicates that the likelihood function is less sensitive to the tail fitting than the three moments matching also in this case.

In a final experiment, we compared the results obtained by G-FIT with results from the PH fitting tool EMpht [1]. Similar to G-FIT the tool EMpht applies the EM algorithm for distribution fitting, but with no specialization to a sub-class of PH distributions. Throughout all experiments G-FIT outperforms EMpht in terms of CPU time requirements. Furthermore,

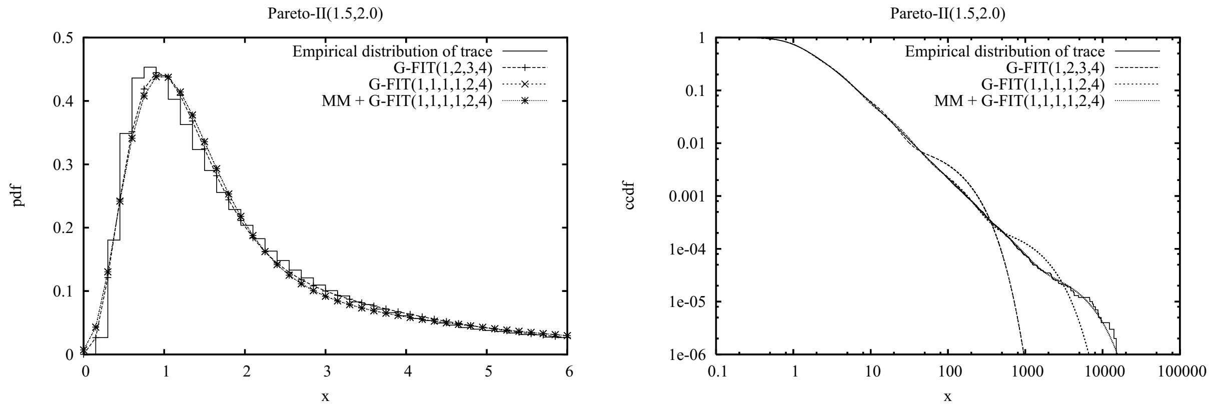

Figure 6. Densities and complementary cdf of fitted HErD for the Pareto-II trace

EMpht converges much slower to optimal parameter values than G-FIT. For example, for the Weibull(1.0,0.5) trace EMpht required 260 seconds CPU time for 1000 iterations on a 5-state PH distribution and reached only a log-likelihood value of –11481.29, which is worse than that for G-FIT and also PH-FIT (see Table 1). For the uniform trace EMpht required 230 seconds CPU time for 1000 iterations on a 10-state PH distribution and reached a log-likelihood value of –2034.95, which is also worse than that for G-FIT and PH-FIT. Fitting the Pareto-II trace with a 10-state distribution seems to be not practicable since already 100 iterations take more than 270 seconds of CPU time with log-likelihood values still far away from the optimum.

The results presented in this section underline the flexibility of the class of HErD for fitting general distributions as theoretically shown in Section 2. Furthermore, we conclude from the experiments that HErD can be fitted much more efficiently and in most cases more accurately than APHD with the proposed EM algorithm. We believe that this is essentially due to the more restricted structure of the HErD class which practically does not reduces its flexibility on fitting. We think that other fitting algorithms over the HErD class would result in similar fitting quality but are less efficient in terms of CPU time requirements.

5.2 Fitting Hyper-Erlang Distributions to Real Traffic Traces

To study an example with a real data traffic trace we used the call center data trace provided by Avishai Mandelbaum [21]. The data archives all calls handled by the call center of one of Israel’s banks over a period of 12 month from January 1999 till December 1999. For every month about 20,000 to 30,000 calls are recorded. For every call the traces contain several attributes, from which we used the service times as given in the traces for our study. Furthermore, service times are scaled to have mean 1.0.

Figure 7 shows the empirical density functions for the traces of January and December as well as the density functions for the fitted HErD with 5, 10, and 20 states, respectively. Service times for January exhibited a quick-hang phenomena [21], i.e., there is a high

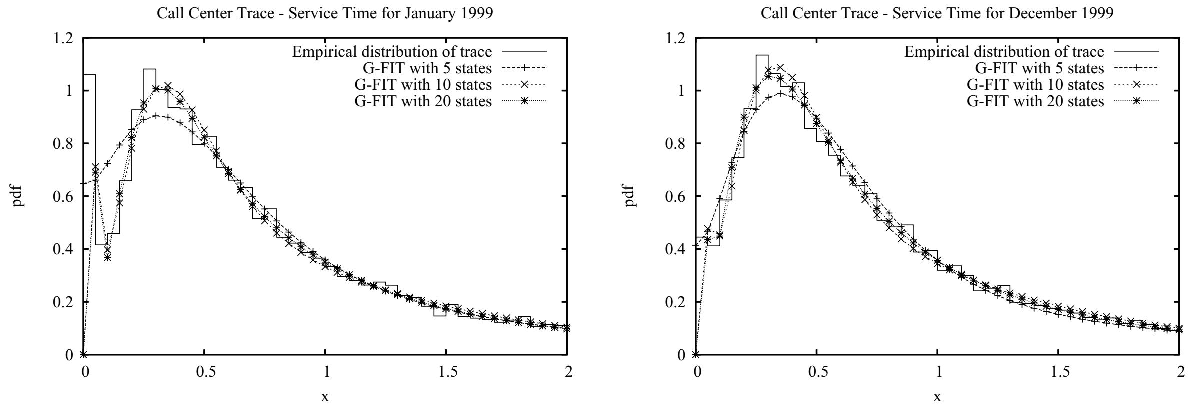

Figure 7. Densities of fitted HErD for call center traces

percentage of calls with very short service times, while service times of December are free of this problem. From Figure 7 we conclude that the traces are fitted very well by a HErD with 10 or 20 states whereas 5 states seem to be not sufficient to adequately represent the traces. Furthermore, Figure 7 shows that the quick-hang phenomena can also be represented by the HErD with 10 states or 20 states. The relative difference between the moments of the trace and the moments of the 20-state HErD is at most 1%, which underlines the high accuracy of the fitted distribution.

To provide a second example we considered a log-file from the Squid proxy server at the University of Dortmund, which was recorded in March 2005. The considered trace contains about $9 \cdot 10^6$ elements and shows heavy-tailed behavior. We used the requested data sizes at the proxy server for fitting and scaled the trace to have mean 1.0. The empirical distribution

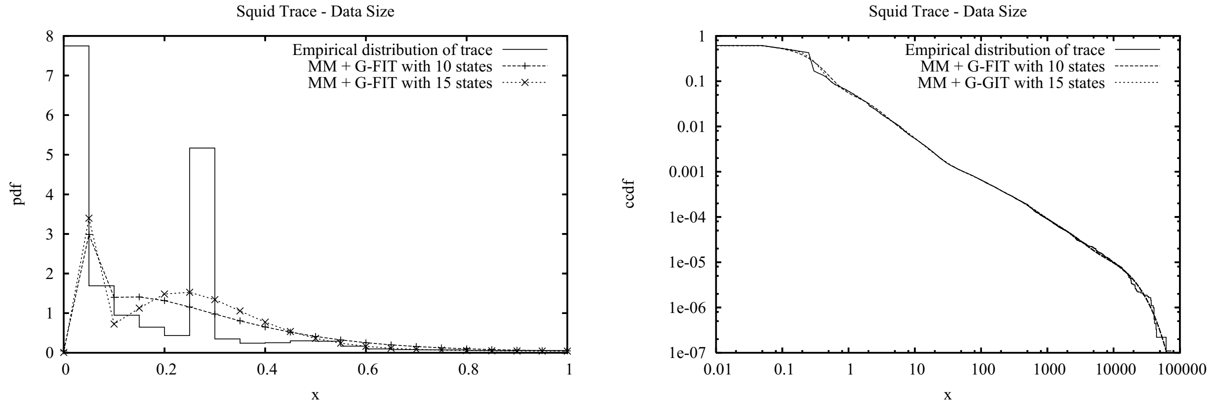

Figure 8. Densities and complementary cdf of fitted HErD for the Squid trace

of the trace and the fitted HErD with 10 and 15 states are presented in Figure 8. For fitting we applied the combined moment matching and likelihood maximization approach as proposed in strategy (ii) in Section 4.2. In fact, we used a 5-state hyperexponential distribution for the tail and the remaining states for the body.

The experiments in Figure 8 show, that the combined moment matching and likelihood maximization yields excellent fitting results for the tail. The body of the distribution contains two peaks in the intervals [0.0, 0.05] and [0.25, 0.3], which result from a high percentage of short file sizes and a very large number of files with size close to 512 bytes. From Figure 8 we observe that such peaks can be better approximated by HErD when increasing the number of states. Furthermore, we observe a significant difference in the log-likelihood values. The 10-state HErD gives log-likelihood 4945880 and the 15-state HErD gives log-likelihood 5332334. Ignoring the tail-fitting completely gives log-likelihood values of 4898814 and 5321959, respectively. This is in accordance with the discussion at the end of Section 4.2 where we stated that the body of the distribution has a stronger impact on the log-likelihood measure. From the results presented in this section we conclude that even very large traces (i.e., 10^6 – 10^8 samples) can be fitted efficiently and accurately with the proposed method.

5.3 Comparison of Queueing Performance Measures

To evaluate the fitting quality, apart from measures directly related to the empirical distributions, also performance measures for a queueing system with interarrival times drawn from the distributions may be used. In the experiments presented in Figure 9 a Weibull(1.0,5.0) trace and a Pareto-II(1.5,2.0) trace each having 10^6 samples is used as arrival process for a G/M/1/K queue with mean service time 0.8. Both traces were scaled to have a mean arrival rate 1.0. Applying a trace-driven simulation we determined the mean queue length and the probability of blocking an arriving customer for different queue capacities. The performance measures were determined from 100 replicated simulations, i.e., an overall number of 10^8 arrival events were simulated. The width of 99% confidence intervals was almost always below 1%.

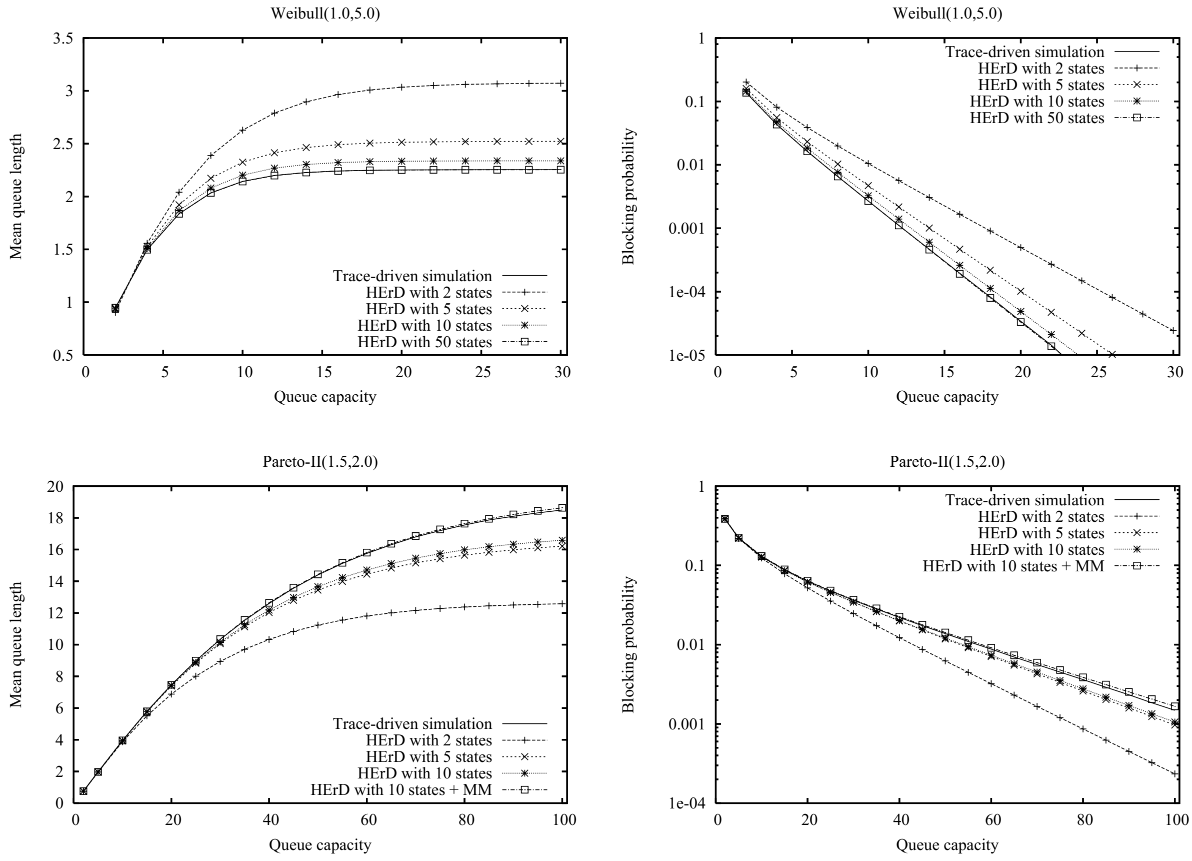

Figure 9. Results for the G/M/1/K queuing system and its approximations

The same performance measures are determined for a HErD/M/1/K queue with appropriately fitted HErD as arrival process. Steady-state performance measures for the HErD/M/1/K queue are computed from numerical analysis of the underlying continues time Markov chain. For both examples we observe that increasing the number of states for the HErD increases the accuracy of performance measures. In fact, using only a 2-state HErD overestimates the mean queue length for the Weibull trace and underestimated it for the Pareto-II trace. Applying the 50-state approximation for the Weibull trace from Table 1, performance measures are exactly matched. For the Pareto-II trace we considered two different 10-state HErD from Figure 6, i.e., the HErD with maximum log-likelihood, G-FIT(1,2,3,4), and the HErD with best tail fitting, MM+G-FIT(1,1,1,1,2,4). The results in Figure 9 indicate that an appropriate tail fitting of the distribution is more important for accurate queueing behavior than using a distribution with slightly better body fitting, i.e., the HErD with maximum log-likelihood.

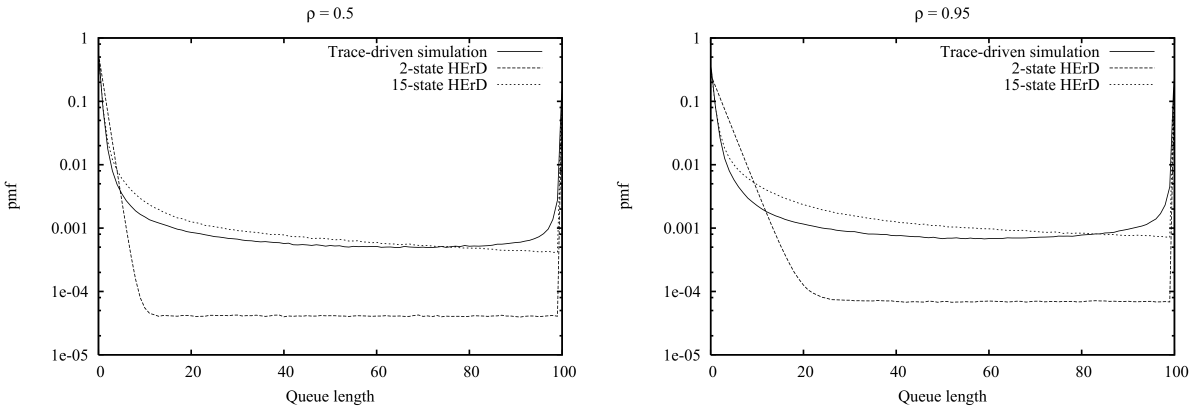

Figure 10. Distribution of M/HErD/1/100 queue length for different utilizations

In a final experiment we evaluate the behavior of an M/G/1/K queue that models the service process at a proxy server. The original Squid trace and the fitted 15-state HErD are used to simulate the service time distribution at the proxy server under different load conditions. Varying the arrival rate we set server utilization ρ to several values between 0.1 and 0.95. Note that an adequate traffic model has to capture the autocorrelation dependencies of the trace, if they are present. The importance of autocorrelation dependencies can be estimated applying a statistical test for data independence. One test method is to plot the autocorrelation function r(k) for various lags k and a 95% confidence interval around 0 for r(k), a so-called magic window. If more than 95% of autocorrelation values are within this interval, then the data trace may be considered as independent [22]. In the autocorrelation function of the Squid trace only 4.2% of r(k) values are outside the 95% interval. Therefore to model the statistical behavior of this trace the autocorrelation dependencies can be neglected.

In Figure 10, we compared the queue length distribution for the fitted HErD and the Squid trace. In all experiments the queue capacity was limited to K = 100. Using the Squid trace, the mean queue length is 24.9 for ρ = 0.5 and 41.6 for ρ = 0.95. For the 15-state HErD we obtain a mean queue length of 22.1 and 38.3, respectively. This corresponds to about 11.4% and 8% relative error. For the 2-state case we observe a relative error of 44.7% and 41.6%, respectively.

Note that for an M/G/1 system with unlimited buffer the mean queue length depends only on the first two moments of the service time distribution (Pollaczek-Khinchin's mean-value formula, see e.g. [25]). To demonstrate that Pollaczek-Khinchin's formula is not applicable in the limited buffer case considered in Figure 10, we also present the mean queue length distribution obtained from a 2-state HErD which matches the first three moments exactly. This shows that matching the first three moments only is not sufficient to yield accurate queueing behavior. Comparing the queue length distributions obtained from the original trace with that computed with the 15-state HErD trace, we conclude that our workload model is quite accurate for a broad range of server utilizations.

6 Conclusions

We presented a novel approach that fits a restricted class of phase-type distributions to trace data. For the parameter fitting we developed an EM algorithm, which is tailored to the special structure of a hyper-Erlang distribution. One of the crucial ideas behind the fitting method presented in this paper is the use of the smallest class of phase-type distributions, which is still sufficiently general to approximate any non-negative distribution (see Theorem 1 and Figure 2). The empirical experiences confirm the expectation that searching for best fitting in a smaller class of distributions is numerically more effective and stable.

The effectiveness of the proposed fitting method is demonstrated by a comparison with two other methods using six benchmark traces and two real traffic traces as well as quantitative results from queueing analysis. We conclude from this comparison that hyper-Erlang distributions are the most versatile sub-class of phase-type distributions, since hyper-Erlang distributions provide practically the full flexibility of the PH class and can be efficiently tuned to match general distributions.

References

[1] S. Asmussen, O. Nerman, and M. Olsson, Fitting Phase-type Distributions via the EM Algorithm, *Scandinavian Journal of Statistics* **23**, 419-441, 1996. <www.maths.lth.se/matstat/staff/asmus/pspapers.html>.

[2] A. Bobbio, A. Horváth, M. Scarpa, and M. Telek, Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm, *Performance Evaluation* **54**, 1-32, 2003.

[3] A. Bobbio and M. Telek, A Benchmark for Estimation Algorithms: Results for Acyclic-PH, *Stochastic Models* **10**, 661-677, 1994.

[4] P. Buchholz, An EM-Algorithm for MAP Fitting from Real Traffic Data, in: P. Kemper and W.H. Sanders (Eds.)*, 13th Int. Conf. on Computer Performance Evaluation - Modelling Techniques and Tools (TOOLS 2003), Urbana, IL, USA, LNCS* **2794**, 218-236, Springer, 2003.

[5] A. Cumani, On the Canonical Representation of Homogeneous Markov Processes Modeling Failure – Time Distributions, *Journal on Microelectronics and Reliability* **22**, 583-602, 1982.

[6] A.P. Dempster, N.M. Laird, and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society* **39**, 1-38, 1977.

[7] A. Edelman and H. Murakami, Polynomial roots from companion matrix eigenvalues, *Mathematics of Computation* **64**, 763-776, 1995.

[8] R. El Abdouni Khayari, R. Sadre, and B.R. Haverkort, Fitting world-wide web request traces with the EM-algorithm, *Performance Evaluation* **52**, 175-191, 2003.

[9] Y. Fang, Hyper-Erlang Distribution Model and its Application in Wireless Mobile Networks, *Wireless Networks* **7**, 211-219, 2001.

[10] A. Feldmann and W. Whitt, Fitting mixtures of exponentials to long-tail distributes to analyze network performance models, *Performance Evaluation* **31**, 245-258, 1998.

[11] B.R. Haverkort, Markovian Models for Performance and Dependability Modeling, in: E. Brinksma, H. Hermanns, and J.-P. Katoen (Eds.), *Formal Methods and Performance Analysis (FMPA 2000), LNCS* **2090**, 38–83, Springer 2001.

[12] G-FIT, <ls4-www.cs.uni-dortmund.de/home/thummler/pubs.html>.

[13] A. Horváth and M. Telek, Markovian Modeling of Real Data Traffic: Heuristic Phase Type and MAP Fitting of Heavy Tailed and Fractal Like Samples, in: M.C. Calzarossa

and S. Tucci (Eds.), *Performance Evaluation of Complex Systems: Techniques and Tools (Performance 2002), Rome, Italy*, *LNCS* **2459**, 405-434, Springer 2002.

[14] A. Horváth and M. Telek, Phfit: A general phase-type fitting tool, in: T. Field, P.G. Harrison, J. Bradley, and U. Harder (Eds.), *12th Int. Conf. on Computer Performance Evaluation - Modelling Techniques and Tools (TOOLS 2002), London, UK, LNCS* **2324**, 82-91, Springer 2002. <webspn.hit.bme.hu/~telek/tools.htm>.

[15] M.A. Johnson, Selecting Parameters of Phase Distributions: Combining Nonlinear Programming, Heuristics, and Erlang Distributions, *ORSA Journal on Computing* **5**, 69-83, 1993.

[16] M.A. Johnson and M.R. Taaffe, The Denseness of Phase Distributions, *Purdue School of Industrial Engineering Research Memoranda* **88-20**, 1988.

[17] M.A. Johnson and M.R. Taaffe, Matching Moments to Phase Distributions: Mixtures of Erlang Distributions of Common Order, *Stochastic Models* **5**, 711-743, 1989.

[18] M.A. Johnson and M.R. Taaffe, Matching Moments to Phase Distributions: Nonlinear Programming Approaches, *Stochastic Models* **6**, 259-281, 1990.

[19] T. Krishnan and G.J. McLachlan, *The EM Algorithm and Extensions*, John Wiley & Sons, 1997.

[20] A. Lang and J. L. Arthur, Parameter Approximation for Phase-Type Distributions. Matrix Analytical Methods in Stochastic Models, *Lecture Notes in Pure and Applied Mathematics* **183**, 151-206, Marcel Dekker, 1997.

[21] A. Mandelbaum, A. Sakov, and S. Zeltyn, Empirical analysis of a call center, *Tech. Rep., Technion, Israel Institute of Technology*, 2000. <iew3.technion.ac.il/serveng/callcenterdata>.

[22] S. Resnick, Modeling Data Networks, *TR-1345, School of Operations Research and Industrial Engineering, Cornell University*, 2002.

[23] A. Riska, V. Diev, and E. Smirni, An EM-based technique for approximating long-tailed data sets with PH distributions, *Performance Evaluation* **55**, 147-164, 2004.

[24] L. Schmickler, MEDA: Mixed Erlang Distributions as Phase-Type Representations of Empirical Distribution Functions, *Stochastic Models* **8**, 131-156, 1992.

[25] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd Edition, John Wiley & Sons, 2002.

Appendix

A.1 Matching Moments with a Mixture of Two Erlang Distributions

We consider a mixture of two mutually independent Erlang distributions with initial probabilities α_1 and $\alpha_2 := 1-\alpha_1$ with $0 < \alpha_1 < 1$ and scale parameters λ_1 and λ_2, respectively. The number of phases of the two mixtures is denoted with r_1 and r_2, respectively. Without loss of generality we assume $r_1 \le r_2$. Let μ_1, μ_2, and μ_3 be the moments to be matched by the mixture of the two Erlang distributions. In the following we show how to compute the solution for λ_1, λ_2, and α_1 for the moment matching problem according to the cases (ii) to (iv) considered in Section 2.3. If case (iii) applies the unique solution can be computed in closed form [17] by

$$\lambda_1 = \frac{2D_1}{D_2 + \sqrt{D_2^2 - 4D_1D_3}}, \quad \lambda_2 = \frac{2D_1}{D_2 - \sqrt{D_2^2 - 4D_1D_3}}, \quad \alpha_1 = \lambda_1 \frac{1-\lambda_2 A}{\lambda_1 - \lambda_2}, \tag{28}$$

with

$$A = \frac{\mu_1}{r_1}, \quad B = \frac{\mu_2}{r_1(r_1+1)}, \quad C = \frac{\mu_3}{r_1(r_1+1)(r_1+2)}, \text{ and} \tag{29}$$

$$D_1 = A^2 - B, \quad D_2 = AB - C, \quad D_3 = B^2 - AC. \tag{30}$$

In cases (ii) or (iv) the moment matching problem can only be solved numerically. Using Eq. (3) the first three moments depend on λ_1, λ_2, and α_1 according to

$$\mu_1 = \alpha_1 r_1 \lambda_1^{-1} + \alpha_2 r_2 \lambda_2^{-1}, \tag{31}$$

$$\mu_2 = \alpha_1 r_1 (r_1+1) \lambda_1^{-2} + \alpha_2 r_2 (r_2+1) \lambda_2^{-2}, \tag{32}$$

$$\mu_3 = \alpha_1 r_1 (r_1+1)(r_1+2) \lambda_1^{-3} + \alpha_2 r_2 (r_2+1)(r_2+2) \lambda_2^{-3}. \tag{33}$$

The system of Eqs. (31) to (33) can be transformed into a polynomial equation of degree five depending only on the reciprocal x := 1/λ₂ of the scale parameter of the second branch,

$$c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0 = 0\,, \tag{34}$$

with coefficients

$$\begin{aligned}
c_0 &= \mu_3 r_1 \left(\mu_2^2 (r_1+2) - \mu_1 \mu_3 (r_1+1)\right),\\
c_1 &= \mu_3^2 r_1 r_2 (r_1+1) + 2\mu_1\mu_2\mu_3 r_1 (r_1+1)(r_2+1) - \mu_2^3 (r_1+2)(2r_1+2r_2+3r_1r_2)\,,\\
c_2 &= 2r_2(r_2+1)\left(3\mu_1\mu_2^2(r_1+1)(r_1+2) - \mu_1^2\mu_3 r_1 (r_1+1) - \mu_2\mu_3 r_1 (2r_1+3)\right),\\
c_3 &= 2r_2(r_2+1)\left(\mu_2^2 r_1 (r_1+2)(r_2+2) - 3\mu_1^2\mu_2 (r_1+1)(r_1+2)(r_2+1) + \mu_1\mu_3 r_1 (r_1+1)(2r_2+1)\right),\\
c_4 &= r_2(r_2+1)^2 \left(\mu_1^3 (r_1+1)(4+4r_1+4r_2+3r_1r_2) - 2\mu_1\mu_2 r_1 (r_1+1)(r_2+2) - \mu_3 r_1^2 r_2\right),\\
c_5 &= r_1 r_2 (r_2+1)(r_2+2)\left(\mu_2 r_1 r_2 (r_2+1) - \mu_1^2 r_2 (r_1+1)(r_2+1)\right).
\end{aligned}$$

The roots of polynomial equations cannot be found analytically beyond the special cases of polynomials with degree less then five. Nevertheless, determining the roots of a polynomial is a standard topic in numerical analysis and can be solved quite fast and numerically stable using balanced-QR reduction of the companion matrix of (34) (see e.g. [7]). The roots of this polynomial give five (real and complex) solutions for λ_2 from which we can compute λ_1 and α_1 and subsequently find the feasible solutions. The system of Eqs. (31) to (33) can be transformed to a unique solution for λ_1 depending only on λ_2, which is given by

$$\lambda_1 = \frac{r_1 r_2 (r_2+1)\lambda_2^{-2} - 2\mu_1 (r_1+1)(r_2+1)\lambda_2^{-1} + \mu_2 (r_1+2)}{\mu_1 r_2 (r_2+1)\lambda_2^{-2} - 2\mu_2 (r_2+1)\lambda_2^{-1} + \mu_3}\,, \tag{35}$$

and finally we can determine a unique solution for α_1 from Eq. (31) given by

$$\alpha_1 = \frac{\mu_1 - r_2\lambda_2^{-1}}{r_1\lambda_1^{-1} - r_2\lambda_2^{-1}}\,. \tag{36}$$

Note that Eqs. (34), (35), and (36) give five solutions for λ_1, λ_2, and α_1 fulfilling the system (31) to (33). From these solutions we have to find the feasible solutions, i.e., the real-valued solutions that fulfill the conditions $\lambda_1 > 0$, $\lambda_2 > 0$, and $0 < \alpha_1 < 1$. As a result from our experience and as also motivated in [18], the matching problem has a unique solution in case (ii) and in case (iv) it has exactly two solutions though not yet proved.

A.2 Simplification of the Expectation of the Complete-Data Log-Likelihood

We start with the expectation of the complete-data log-likelihood function as provided in Eq. (19), i.e.,

$$Q\left(\Theta,\hat{\Theta}\right)=\sum_{\mathbf{y}\in\{1,\ldots,M\}^K}\sum_{k=1}^{K}\log\left(\alpha_{y_k}p_{y_k}\left(x_k|\theta_{y_k}\right)\right)\cdot\prod_{i=1}^{K}q\left(y_i|x_i,\hat{\Theta}\right). \tag{37}$$

Introducing the indicator function $\delta_{x,y}$ with $\delta_{x,y} = 1$ if x = y and 0 otherwise we can rewrite Eq. (37) as

$$\begin{aligned}Q\left(\Theta,\hat{\Theta}\right)&=\sum_{\mathbf{y}\in\{1,\ldots,M\}^K}\sum_{k=1}^{K}\sum_{m=1}^{M}\delta_{m,y_k}\log\left(\alpha_m p_m\left(x_k|\theta_m\right)\right)\cdot\prod_{i=1}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\\&=\sum_{m=1}^{M}\sum_{k=1}^{K}\log\left(\alpha_m p_m\left(x_k|\theta_m\right)\right)\sum_{\mathbf{y}\in\{1,\ldots,M\}^K}\delta_{m,y_k}\cdot\prod_{i=1}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\end{aligned} \tag{38}$$

Now, for every m ∈ {1,…,M} and k ∈ {1,…,K} the sum over the vector **y** in Eq. (38) can be simplified, i.e.,

$$\begin{aligned}&\sum_{\mathbf{y}\in\{1,\ldots,M\}^K}\delta_{m,y_k}\cdot\prod_{i=1}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\\&=\sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_K=1}^{M}\delta_{m,y_k}\cdot\prod_{i=1}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\\&=\sum_{y_k=1}^{M}\delta_{m,y_k}\cdot q\left(y_k|x_k,\hat{\Theta}\right)\cdot\sum_{y_1=1}^{M}\cdots\sum_{y_{k-1}=1}^{M}\sum_{y_{k+1}=1}^{M}\cdots\sum_{y_K=1}^{M}\prod_{i=1,i\neq k}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\\&=q\left(m|x_k,\hat{\Theta}\right)\cdot\sum_{y_1=1}^{M}\cdots\sum_{y_{k-1}=1}^{M}\sum_{y_{k+1}=1}^{M}\cdots\sum_{y_K=1}^{M}\prod_{i=1,i\neq k}^{K}q\left(y_i|x_i,\hat{\Theta}\right)\\&=q\left(m|x_k,\hat{\Theta}\right)\cdot\prod_{i=1,i\neq k}^{K}\sum_{y_i=1}^{M}q\left(y_i|x_i,\hat{\Theta}\right)=q\left(m|x_k,\hat{\Theta}\right)\end{aligned} \tag{39}$$

since $\sum_{m=1}^{M} q\left(m|x_i,\hat{\Theta}\right)=1$ for all i = 1,…,K. Using Eq. (39) we can rewrite Eq. (38) as

$$\begin{aligned}Q\left(\Theta,\hat{\Theta}\right)&=\sum_{m=1}^{M}\sum_{k=1}^{K}\log\left(\alpha_m p_m\left(x_k|\theta_m\right)\right)\cdot q\left(m|x_k,\hat{\Theta}\right)\\&=\sum_{m=1}^{M}\sum_{k=1}^{K}\log\left(\alpha_m\right)\cdot q\left(m|x_k,\hat{\Theta}\right)+\sum_{m=1}^{M}\sum_{k=1}^{K}\log\left(p_m\left(x_k|\theta_m\right)\right)\cdot q\left(m|x_k,\hat{\Theta}\right)\end{aligned} \tag{40}$$

which is exactly Eq. (20).