Phase-Type Fitting Using HyperStar

Philipp Reinecke, Tilman Krauß, and Katinka Wolter

Freie Universität Berlin
Institut für Informatik
Takustraße 9
14195 Berlin, Germany
{philipp.reinecke,tilman.krauss,katinka.wolter}@fu-berlin.de

Abstract. In this paper we provide a hands-on discussion of the use of the HyperStar phase-type fitting tool in common application scenarios. HyperStar allows fitting Hyper-Erlang distributions to empirical data, using a variety of algorithms and operation modes. We describe simple cluster-based fitting, a new graphical method for refining the density approximation, a new command-line interface, and the integration of HyperStar with a Mathematica implementation of a fitting algorithm. Furthermore, we describe the use of Hyper-Erlang distributions in simulation. Throughout our discussion we illustrate the concepts on a data set which has been shown to be difficult to fit with a PH distribution.

Keywords: Phase-type fitting, Tool description, Case-study.

1 Introduction

Phase-type (PH) distributions [1] are a very flexible class of distributions for modelling e.g. failure times or response times. As PH distributions have Markovian representation, they can be used in analytical as well as in simulation approaches to system evaluation.

Phase-type distributions are typically applied to approximate empirical data sets. In recent years several tools have been developed to help with the task of fitting PH distributions to data: EMPHT [2] is a command-line tool that can fit arbitrary phase-type distributions. PhFit [3] fits acyclic phase-type distributions (both discrete and continuous), offering both a graphical user-interface and command-line tools. G-FIT [4] fits Hyper-Erlang distributions and runs on the command-line.

In [5] we proposed a cluster-based fitting approach for fitting mixtures of distributions, and in [6] we presented the HyperStar tool that implements this approach. HyperStar complements the above set of tools in that its intuitive user-interface helps domain-experts to apply PH distributions for data fitting; furthermore, its user-interface can also be applied to existing tools and prototypes, thus fostering the development of new fitting approaches. HyperStar is implemented in Java and is available for download at [7].

M.S. Balsamo, W.J. Knottenbelt, and A. Marin (Eds.): EPEW 2013, LNCS 8168, pp. 164–175, 2013.

In this paper we provide a hands-on discussion of the use of HyperStar in common fitting tasks. HyperStar implements several varieties of the fitting algorithm described in [5] and also offers several operation modes. Our focus here will be on illustrating the application of HyperStar in typical scenarios. We will therefore focus on the simple mode and on the command-line mode, as these are probably the modes that are used most of the time. Expert mode offers a high degree of flexibility in parameterising different variants of the clustering algorithm, and also supports the inclusion of existing tools, such as PhFit and G-FIT for fitting the branch distributions. Although this allows the expert to configure HyperStar in great detail when fitting, we have observed that in typical examples there are only minor improvements. We will therefore only describe the Mathematica interface, as this interface has proved to be helpful in evaluating new fitting algorithms. With the Mathematica interface, the user only has to implement the functionality of the fitting algorithm, which then seamlessly integrates with the dataset manipulation and display of results of HyperStar. Throughout the paper we first discuss the application and then provide some details on the underlying algorithms.

The paper is structured as follows: In Section 2 we briefly introduce some properties of phase-type distributions. We then describe the data set that we use throughout this paper. In Section 4 we describe the simple mode of using HyperStar. This mode enables the user to fit a HyperErlang distribution without requiring expert knowledge of PH distributions. Section 5 describes the new peak-adjustment feature that allows the adjustment of the shape of the fitted density using purely graphical means. In Section 6 we introduce the new command-line mode, which automatically detects peaks before applying cluster-based fitting. In Section 7 we give an example of how HyperStar can be integrated with a Mathematica implementation of a PH-fitting algorithm using the Mathematica interface of the expert mode. We complement our discussion of phase-type fitting by a description of how to use Hyper-Erlang distributions in simulation tools in Section 8, before concluding the paper with an outlook on future work.

2 Phase-Type Distributions

Continuous phase-type (PH) distributions are defined as the distribution of time to absorption in a Continuous-Time Markov Chain (CTMC) with one absorbing state [1]. PH distributions are commonly represented by a vector-matrix tuple $(\boldsymbol{\alpha}, \mathbf{Q})$, where

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n \text{ and } \mathbf{Q} = \begin{pmatrix} -\lambda_{11} & \cdots & \lambda_{1n} \\ \vdots & \ddots & \vdots \\ \lambda_{n1} & \cdots & -\lambda_{nn} \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{1}$$

with $\lambda_{ij} \geq 0, \lambda_{ii} > 0, \mathbf{Q}\mathbb{1} \leq \mathbf{0}$, $\mathbf{Q}$ is non-singular, and $\boldsymbol{\alpha}\mathbb{1} = 1$, where $\mathbb{1}$ is the column vector of ones of the appropriate size. $\boldsymbol{\alpha}$ is referred to as the initial probability vector, and $\mathbf{Q}$ is the sub-generator matrix of the phase-type distribution.

Definition 1. *If* $(\boldsymbol{\alpha}, \mathbf{Q})$ *is the representation of a phase-type distribution, then the* probability density function (PDF)*, cumulative distribution function (CDF), and* k*th* moment*, respectively, are given by [1,3,8]:*

$$f(t) = \boldsymbol{\alpha} \mathrm{e}^{\mathbf{Q}t}(-\mathbf{Q}\mathbb{1}), \tag{2}$$

$$F(t) = 1 - \boldsymbol{\alpha} \mathrm{e}^{\mathbf{Q}t}\mathbb{1}, \tag{3}$$

$$E\left[X^k\right] = k!\boldsymbol{\alpha}(-\mathbf{Q})^{-k}\mathbb{1}.. \tag{4}$$

HyperStar fits mixtures of m phase-type distributions, that is, the distributions created by HyperStar have branch structure. Each branch has a sub-generator matrix $\mathbf{Q}_1, \ldots, \mathbf{Q}_m$ and an initial probability vector $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$. The mixture is then given by

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{0} & & \\ & \ddots & \ddots & \\ & & & \mathbf{0} \\ & & & \mathbf{Q}_m \end{pmatrix} \text{ and } \boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m). \tag{5}$$

HyperStar is most commonly used to fit Hyper-Erlang distributions, i.e. mixtures of Erlang distributions [4]. With Hyper-Erlang distributions, all block matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_m$ in (5) are of the form

$$\mathbf{Q}_i = \begin{pmatrix} -\lambda_i & \lambda_i & & \\ & \ddots & \ddots & \\ & & & \lambda_i \\ & & & -\lambda_i \end{pmatrix}, \tag{6}$$

and only the first element in each initial probability vector $\boldsymbol{\alpha}_i, i = 1, \ldots, m$ is non-zero.

3 The Data Set

Throughout this paper we use the data set shown in Figure 1. This data set contains samples of the packet-delivery ratio in the DES testbed, a testbed for wireless mesh networks deployed in different buildings across the campus of Freie Universität Berlin [9]. Since the data is for packet-delivery ratios, which are in the interval $[0, 1]$, the density is 0 for samples outside this interval. The histogram of the density shows three peaks, at 0, 0.75, and 1. Note that this data set is very difficult to fit using a phase-type distribution; in particular, the density of 0 for PDR values larger than 1 cannot be fitted exactly with any phase-type distribution. We use this data set as an extreme example for showing the potential of HyperStar for fitting difficult-to-fit data sets. For a more in-depth evaluation and comparison to other fitting tools we refer the reader to [5].

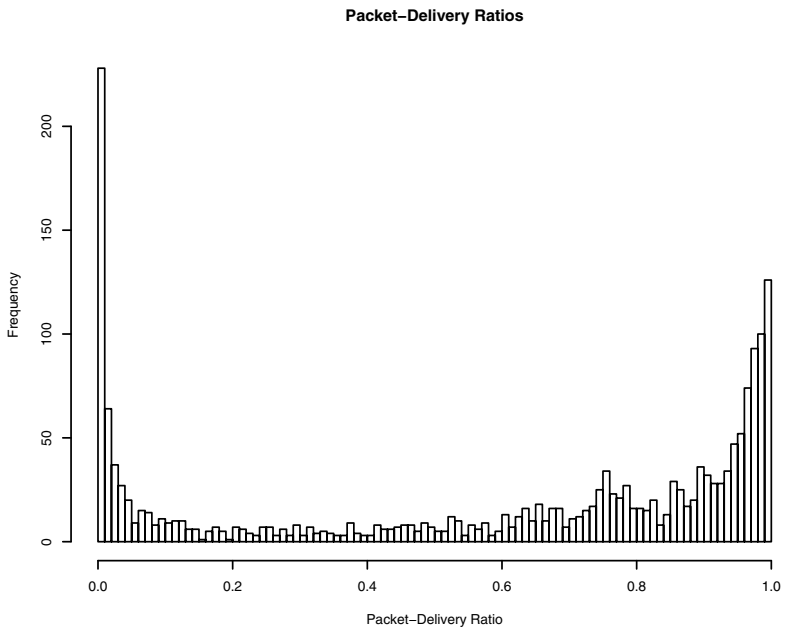

Fig. 1. Histogram of the DES data set

4 Fitting a Data Set in Simple Mode

When started without parameters, HyperStar shows the user interface for simple mode. Simple mode is probably the most commonly used mode of HyperStar and allows the user to quickly and accuratly fit a Hyper-Erlang distribution to a data set.

The user interface for simple mode is shown in Figure 2: The left-hand panel displays the histogram and empirical CDF for the data set and, after fitting, the fitted PDF and CDF. Furthermore, this panel also serves to control the fitting algorithm. The panel on the top right guides the user through the steps necessary to fit a PH distribution. The panel on the bottom right shows default quality measures for the fitted distribution, as defined in [10].

In order to fit a PH distribution, we first have to load the data set from a file. HyperStar expects an ASCII file with one sample per line, given in a common numerical format. In the next step we can adjust the number of bars in the histogram, in order to get a better understanding of the shape of the density. In this example we set the number of bars to 100, and we observe the peaks at 0, 0.75, and 1. We can then mark these peaks by clicking on them. In doing so, we create new clusters for the clustering algorithm and define the initial cluster centres to be at the location of the marker. Each cluster corresponds to an Erlang branch in the final distribution, and thus the choice of cluster centres determines the number of

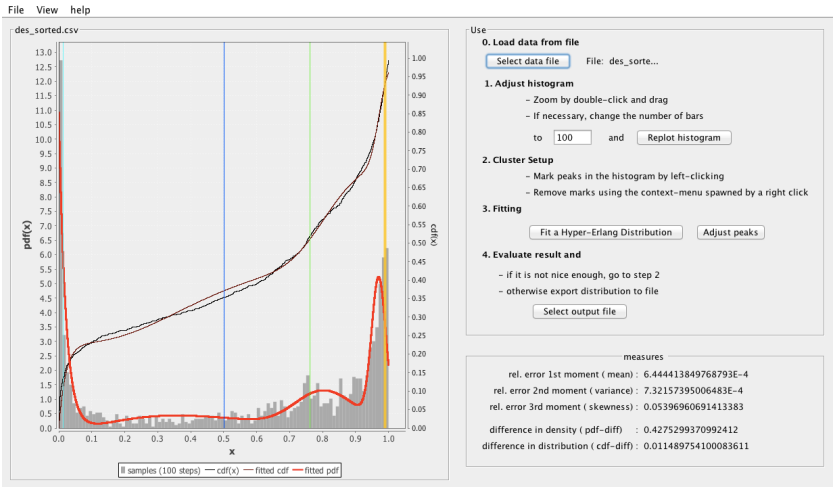

Fig. 2. User interface in Simple Mode, with fitted distribution

branches in the result. We then start the fitting algorithm by clicking on the 'Fit' button. The resulting PDF and CDF are displayed in the left-hand panel, and the bottom right panel shows various quality measures. The distribution can be further refined by adding or removing clusters or by shifting peaks (see Section 5). Once we are satisfied with the results, we save the distribution in a file. In simple mode HyperStar exports the distribution in G-FIT output format, which is a simple text file that is both human-readable and easily parsable for further use [4]. G-FIT files specify the number of branches, the branch lengths, the rates, and the initial branch probabilities, one value per line.

4.1 Algorithm

Simple mode uses the clustering algorithm with probabilistic re-assignment, as described in [5] for fitting Hyper-Erlang distributions. In the first step, the samples in the data set are clustered using the k-means algorithm [11], starting with the cluster centres specified by the user. Clustering aggregates similar samples in the same cluster and thereby identifies the samples that correspond to individual peaks of the density. The algorithm then fits each cluster's samples with an Erlang distribution. The assignment of samples to clusters is refined iteratively until either convergence is reached or a maximum number of 100 rounds has elapsed.

5 Refinement Using Peak Adjustment

Simple mode usually produces a PH distribution whose density is very close to the empirical density. In some cases, however, the peaks of the fitted distribution are not located exactly on the peaks of the empirical distribution. Then, the user

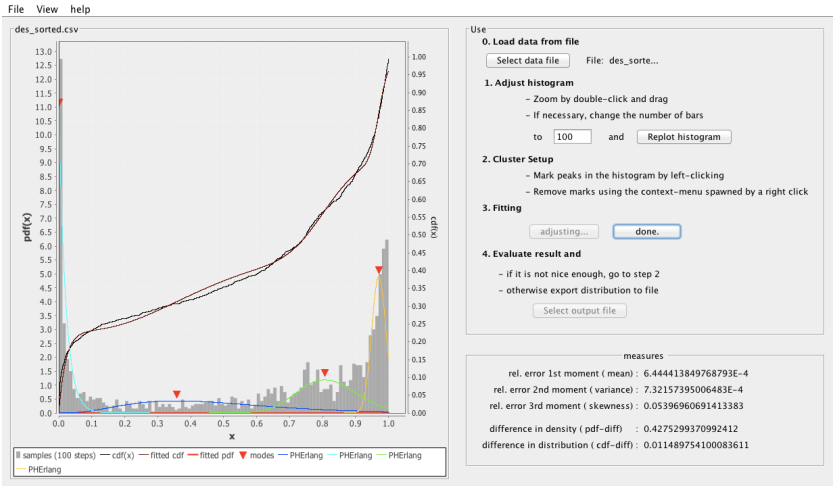

Fig. 3. User interface in Peak-Adjustment Mode

can try to improve the result by manually adjusting the branch distributions to better fit the peaks.

Manual adjustment is performed by switching to 'Peak Adjustment' mode after fitting. In this mode, HyperStar displays individual branch densities and their modes, marked by a triangle atop the mode. By clicking on a triangle and dragging the mouse to a different location, we can move the mode of this branch distribution to a different location, typically to a peak nearby. The density is adjusted accordingly, and a second click places the mode at the new location.

It should be noted that relocating the peaks of the distribution can lead to a worse fit than the automatic fitting, according to the quality measures displayed in the bottom right panel. On the other hand, iterative application of peak adjustment and fitting can improve the results. Furthermore, peak adjustment can be used to explore the impact of changes in the measurements. For instance, with our data set we may be interested not only in a distribution that fits the current peak placement well, but also in the effect of moving the middle peak from 0.75 to other values. In this case, we would relocate this peak in the fitted distribution and use the result in our evaluation.

5.1 Algorithm

After the user has moved the peak, the new mode of the distribution is read from the graphical interface and a new rate λ is computed from the mode, $\frac{1}{\lambda}(k-1)$, as follows:

1. Let x be the new mode of the Erlang distribution with length k and rate λ.
2. Let $\lambda' = \max\left\{\frac{(k-1)}{x}, 1\right\}$
3. Return (k, λ')

6 The Command-Line Interface

For fitting several data sets at once, one typically employs scripts that call the fitting tool on each data set in turn. The new command-line interface (CLI) to HyperStar enables the use of the tool in scripting.

In graphical mode, HyperStar relies on the user providing initial cluster centres. Since there is no user interaction in CLI mode, this mode identifies peaks automatically, using one of two algorithms.

Command-line mode is initiated by specifying the `-cli` option upon startup and providing `-f`, followed by a filename. By default, the fitted distribution is written in G-FIT format to the file `cbhe-result.txt`. A different output filename can be specified using the `-of` option. Table 1 lists additional options to control the behaviour of the fitting algorithm. The default values have been chosen based on experience.

Table 1. Parameters for command-line mode

Parameter	Description (default value)
`-cli`	Use command-line interface
`-f`	Select input file
`-of`	Select output file (`cbhe-result.txt`)
`-bn`	Number of branches to be fitted (10)
`-ip`	Initial Erlang lengths (10)
`-lc`	Convergence threshold (10^{-10})
`-mi`	Maximum number of iterations (10)
`-qu`	Compute quality measures and append them to the output file
`-pd`	Peak-detection algorithm, either `simple` or `hist` (`simple`)
`-b`	Number of bars for `hist` peak detection.

6.1 Algorithm

Since there is no user interaction, the command-line mode requires a different approach for finding initial cluster centres. Cluster centres should ideally be close to peaks in the empirical density. Therefore, CLI mode first detects peaks and then uses their locations as initial cluster centres.

We implemented two algorithms for peak detection. The first one, `simple`, operates directly on the samples:

1. Let $S = \{s_1, \ldots, s_N\}$ be the set of samples, sorted in increasing order.
2. Let m be the number of branches.
3. $d := \frac{N}{m+1}$
4. Return initial cluster centres $s_d, s_{2d}, \ldots, s_{rd}$.

The algorithm simply places equidistant peaks on the sorted data set. If there are peaks in the empirical density, these are characterised by long stretches of similar values and are likely to receive a cluster centre.

Our second approach explicitly detects peaks in the histogram:

1. Let h_i, c_i $(i = 1, \ldots, M)$ be the height and centre of the ith bucket in the histogram with M buckets.
2. Let m be the number of branches.
3. $s_0 := 2h_0 - h_1$
4. $s_{m-1} := 2h_{M-1} - h_{M-2}$
5. for $i = 1, \ldots, M-2$: assign $s_i := 2h_i - h_{i-1} - h_{i+1}$.
6. Pick the m highest values of all of the s_i, and let $i_1, \ldots, i_m$ be their indices.
7. Return initial cluster centres $c_{i_1}, c_{i_2}, \ldots, c_{i_m}$.

The basic assumption underlying this method is that a peak appears in the histogram as a tall bucket surrounded by buckets of much smaller height. For each bucket the algorithm computes the height difference to the surroundings and then picks the buckets with the largest height differences.

There are clearly two application domains of these methods. If nothing is known about the data, the `simple` method is more suitable. Observe that the method does not yield good results if the number of branches m is small. With a high value of m, the probability of guessing a value, which is near to a peak in density, is high. If the dataset is well-known and one can easily detect peaks in the histogram, it is likely that the same peaks are chosen by the `hist` approach.

7 The Mathematica Interface

So far, we have discussed modes of operation that aim at quickly fitting a phase-type distribution to data. The Mathematica interface to HyperStar differs from these in that its goal is mainly to support algorithm development, prototyping and evaluation. It is often beneficial to first implement new PH-fitting algorithms in Mathematica, before writing a dedicated tool, since Mathematica has higher numerical stability and provides a large library of dedicated mathematical functions, which typically leads to more elegant programs than possible in general-purpose languages. With the HyperStar Mathematica integration, the HyperStar GUI can be used as a front-end to such implementations. The user can then focus on the algorithm itself and evaluate the fitting quality using HyperStar's interface. In this mode, HyperStar does not apply clustering; instead, it only forwards data into Mathematica and displays the results.

Mathematica integration requires the implementation of the following three methods in the Mathematica script:

`fit[Samples_]` : Fits a distribution to the sample set contained in the array `Samples`. This function must return a compact representation of the phase-type distribution (e.g. as fixed list of rules) that can be parsed by the `parameters[]` function.

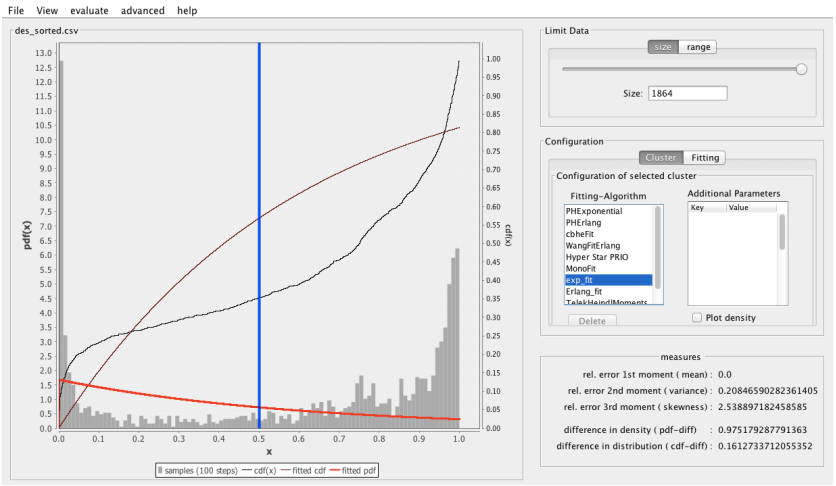

Fig. 4. User interface in Mathematica interface mode

`parameters[D_]` : This method is invoked with the return value of `fit`, in order to convert the representation in `D` (which could be an arbitrary representation used in the fitting algorithm) to a list of key-value pairs. The returned value is the textual representation of the PH distribution.

`ph[D_]` : Returns the (α, A) representation of the phase-type distribution in the arbitrary description `D`. All methods (like moment- or density-computation) within HyperStar use this representation. This method should return a list of two rules, where $\boldsymbol{\alpha}$ yields the vector-representation of the initial distribution and $\mathbf{Q}$ yields the sub-generator matrix. Note that this method must return a valid ph-distribution.

HyperStar redirects all output of the `Print` statement in Mathematica to the standard output, in order to help with debugging the Mathematica code.

The following Mathematica code illustrates how to implement these functions for a very simple fitting algorithm:

```
fit[Samples_] := lambda->1/Mean[Samples];
parameters[D_] := {{"lambda", lambda}} /. D;
ph[D_] := {alpha->{1.0}, A->{{-lambda}}}/. D;
```

The algorithm simply fits the mean of the data set using an exponential distribution (i.e. a phase-type distribution of size 1). The result is shown in Figure 4. As expected, the mean is fitted well, while the all other measures and the shapes of the distribution and the density show large errors. For practical application, the reader might want to implement a more sophisticated algorithm.

7.1 Technical Configuration

In order to configure HyperStar to use Mathematica mode, the following steps have to be taken:

1. The `KernelCommand` entry in the file `config.prop` must be set to the full path to the `MathKernel` executable. `MathKernel` is typically located in Mathematica's root folder or in one of its sub-folders, depending on the operating system.
2. The `JLink.jar` file must be added to the classpath for HyperStar.
3. The Mathematica script containing the algorithm must be made known to HyperStar, as follows:
   (a) Append the algorithm name (e.g. `exp_fit`) to the property `Algorithms`.
   (b) Introduce three new properties for the functions `fit`, `ph` and `parameters` (e.g. `exp_fit.fit`), which point to the respective functions in the script
   (c) Include the filename by setting the property `[algorithm].source` (e.g. `exp_fit.source`) to the full path of the Mathematica script.

HyperStar can then be started with the `-c` option. When started with this option, HyperStar will also test the configured algorithms with a fixed trace before displaying the graphical user interface.

8 Simulation

The Hyper-Erlang distributions created with HyperStar are especially useful to introduce phenomena of real systems in simulations without modelling the underlying systems in detail. Unfortunately, common simulation tools such as OMNeT++ or NS-2 [12,13] do not support Hyper-Erlang distributions as part of their toolkits for random variates. In this section we describe the Libherd library for generating random variates from Hyper-Erlang distributions.

The Libherd library has been developed as a simpler alternative to the Libphprng library in the Butools package [14,15]. It provides the same mechanisms to interface to simulation tools, but is focussed on Hyper-Erlang distributions. In contrast to Libphprng, Libherd does not support other classes of phase-type distributions and does not provide the advanced techniques for optimising phase-type distributions for efficient random-variate generation that are implemented in Libphprng. These restrictions to the functionality resulted in a very small codebase, which is often easier to integrate with specific simulation tools than Libphprng.

Libherd is implemented in C++ as a shared library that must be linked to the simulation framework, e.g. OMNeT++ or NS-2. The library provides the class `HerdGen`. Each instance of this class generates random variates from one Hyper-Erlang distribution specified by the user. In order to generate random variates, the class requires a source of uniform random numbers in the range $(0, 1)$. Libherd uses the random-number generators that are provided by the simulation tool. This requires the user to write a class implementing the

`RandomSourceWrapper` interface and registering an instance of this class with the `HerdGen` instance. The `Libherd` distribution includes wrappers for several simulation frameworks and is available from the main HyperStar page [7].

In the following we illustrate how Libherd can be used to apply Hyper-Erlang distributions fitted with HyperStar in discrete-event simulation. We assume that we have fitted a Hyper-Erlang distribution to the DES data set and saved it in the file `des.gfit`. We want to use this distribution to generate links with typical packet-delivery ratios in a large network.

This requires the following steps:

1. Creation of a new `HerdGen` instance to store the distribution:
   `PhGen * prng = new HerdGen("des.gfit");`
2. Creation of a wrapper object for the random-number stream from the simulation. Assuming that we have a wrapper class called `RSWrapper`, we create an instance of this class as
   `RSWrapper * ursw = new RSWrapper();`
3. Registration of the uniform random number stream with the `HerdGen` object:
   `prng->setUniformRandomSource(ursw);`
4. Drawing of random variates from the distribution:
   ```
   double x;
   while ((x = prng->getVariate()) > 1) {};
   ```
   Note that with our example we need to ensure that the packet-delivery ratios stay within the range $[0, 1]$, and therefore we truncate the distribution to this range.

9 Conclusion

In this paper we have illustrated the application of HyperStar in common fitting tasks. We have introduced the manual peak adjustment and the command-line interface as new features for HyperStar. These features required new algorithms for the adjustment of the peaks and for the automatic detection of peaks in a data set. Although HyperStar already gives good results in many cases, further improvement of the fitting algorithms is certainly possible and will be studied as part of future work. Furthermore, we are exploring the use of HyperStar, or a similar approach, in fitting other stochastic processes.

Acknowledgements. We would like to thank Chris Guenther for his valuable reports on various bugs and usability issues with HyperStar.

References

1. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach. Dover Publications, Inc., New York (1981)
2. Asmussen, S., Nerman, O., Olsson, M.: Fitting Phase-Type Distribution Via the EM Algorithm. Scand. J. Statist. 23, 419–441 (1996)

3. Horváth, A., Telek, M.: PhFit: A General Phase-Type Fitting Tool. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 82–91. Springer, Heidelberg (2002)
4. Thümmler, A., Buchholz, P., Telek, M.: A Novel Approach for Phase-Type Fitting with the EM Algorithm. IEEE Trans. Dependable Secur. Comput. 3(3), 245–258 (2006)
5. Reinecke, P., Krauß, T., Wolter, K.: Cluster-based fitting of phase-type distributions to empirical data. Computers & Mathematics with Applications 64(12), 3840–3851 (2012); Special Issue on Theory and Practice of Stochastic Modeling
6. Reinecke, P., Krauß, T., Wolter, K.: HyperStar: Phase-Type Fitting Made Easy. In: 9th International Conference on the Quantitative Evaluation of Systems (QEST) 2012, pp. 201–202 (September 2012); Tool Presentation
7. Reinecke, P., Wolter, K., Krauß, T.: HyperStar Homepage (2013), `http://www.mi.fu-berlin.de/inf/groups/ag-tech/projects/HyperStar`
8. Telek, M., Heindl, A.: Matching Moments for Acyclic Discrete and Continous Phase-Type Distributions of Second Order. International Journal of Simulation Systems, Science & Technology 3(3-4), 47–57 (2002)
9. Blywis, B., Günes, M., Juraschek, F., Hahm, O., Schmittberger, N.: Properties and Topology of the DES-Testbed (2nd Extended Revision). Technical Report TR-B-11-04, Freie Universität Berlin (July 2011)
10. Lang, A., Arthur, J.: Parameter Approximation for Phase-Type Distributions. Matrix-Analytic Methods in Stocastic Modells 183, 151–206 (1996)
11. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory 28(2), 129–136 (1982)
12. Varga, A.: The OMNeT++ Discrete Event Simulation System. In: Proceedings of the European Simulation Multiconference, ESM 2001 (June 2001)
13. Various contributors: The Network Simulator ns-2, `http://www.isi.edu/nsnam/ns/` (last seen May 11, 2010)
14. Reinecke, P., Horváth, G.: Phase-type Distributions for Realistic Modelling in Discrete-Event Simulation. In: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, SIMUTOOLS 2012, Brussels, Belgium, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 283–290 (2012)
15. Bodrog, L., Buchholz, P., Heindl, A., Horváth, A., Horváth, G., Kolossváry, I., Németh, Z., Reinecke, P., Telek, M., Vécsei, M.: Butools: Program packages for computations with PH, ME distributions and MAP, RAP processes (October 2011), `http://websp n.hit.bme.hu/~butools`