Approximation of Discrete Phase-Type Distributions

Claudia Isensee, Graham Horton
Computer Science Department, Otto-von-Guericke Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
claudia@sim-md.de (contact), graham@sim-md.de

Abstract

The analysis of discrete stochastic models such as generally distributed stochastic Petri nets can be done using state space-based methods. The behavior of the model is described by a Markov chain that can be solved mathematically. The phase-type distributions that are used to describe non-Markovian distributions have to be approximated. An approach for the fast and accurate approximation of discrete phase-type distributions is presented. This can be a step towards a practical state space-based simulation method, whereas formerly this approach often had to be discarded as unfeasible due to high memory and runtime costs.

Discrete phases also fit in well with current research on proxel-based simulation. They can represent infinite support distribution functions with considerably fewer Markov chain states than proxels. Our hope is that such a combination of both approaches will lead to a competitive simulation algorithm.

1. Introduction

This paper introduces a new approach for the approximation of discrete phase-type distributions, which is based on common optimization methods.

Discrete stochastic models are widely used in Simulation. One way to analyze them is to use state space-based methods in which the model's behavior is represented by a discrete-time Markov chain (DTMC), which can be solved mathematically, as opposed to stochastic event-driven simulation which uses replications to produce confidence intervals. To include non-Markovian distributions in a DTMC, they have to be approximated, for example by discrete phase-type distributions (DPH).

Even though state space-based methods work well in theory, they are not used widely. This is partly due to the fact that the state space of a model can grow exponentially with the order of the representations of the non-Markovian distributions, a behavior called state-space explosion. Most of the previous research in phase-type distributions has concentrated chiefly on the mathematical part of the approximation, but only marginally on the performance, or practical feasibility of the results (see section 1.1). The approach presented here applies general optimization methods, which have already been researched and optimized concerning performance, to the DPH fitting task. Performance and result quality will be analyzed, as well as the influence of some optimization and distribution parameters on them.

In order to be useful for a state space-based simulation algorithm, the approximations of the non-Markovian distributions need to be close to the original distribution functions and of a low order. An algorithm for the fitting of DPH needs to be inexpensive, because this fitting is a pre-processing step to the simulation of a model, and should not take longer than the analysis of the model itself. The goal is to develop an algorithm that fulfills those requirements. If successful, these approximations could then be included into the proxel-based simulation method (section 2.2) to produce a state space-based simulation method which can compete with current simulation systems.

1.1. Previous Work

Phase-type distributions both continuous (CPH) and discrete (DPH) were first described in detail by Neuts in [11]. Most of the research since has concentrated on CPH and their approximation. The EMPht fitting tool described in [12] for example can only fit CPH. Among the work of Bobbio and Telek is also a paper [6] that

conducts a benchmark for an algorithm developed by Bobbio and Cumani [3], which fits CPH of a canonical form. This is one of the few papers concerned with the efficiency of the estimation algorithms; however, the algorithm examined again only fits CPH.

Only recently have DPH become of interest again, and several papers have been published concerning their approximation and parameters. In [5] the influence of the discretization time step when approximating DPH is examined, and in [4] an algorithm is proposed for the fitting of DPH. The algorithm is also tested, but only the quality of the results is evaluated; neither the performance of the algorithm nor the influence of the input size on the result quality is covered. The PhFit Tool recently introduced in [8] can fit CPH and DPH, but is again only evaluated concerning the result quality.

1.2. The Problem

When approximating a continuous distribution function by a DPH, the first step is discretization. The discrete steps d_i are calculated using the original CDF $F()$ and a time step Δt, according to the following formula:

$$d_i = F(i * \Delta t) - F((i-1) * \Delta t)$$

The series $\{d_i\}$ represents the amount of probability within each time interval of the original distribution. The problem to be solved when approximating this discretized distribution function by a DPH is to find a sequence of c_i that is produced by a DPH and minimizes the distance between $\{c_i\}$ and $\{d_i\}$ according to a predefined distance measure.

The representation of a specific DPH is not unique. To reduce the space of possible solutions and the number of variables needed for the description, a canonical form for the DPH is used (Figure 1.2), which has been adapted from the one presented in [4]. The a_i represent the initial probabilities of the DTMC states and the p_i are the transition probabilities. It was proven in the same paper, that all DPH can be reduced to a representation of the same order that has this structure.

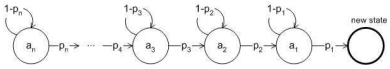

Figure 1. Canonical representation of DPH

A DPH of the current type and order n is then specified by $\Delta t, \vec{a}$ and $\vec{p}$ with the following properties:

$$0 \leq a_i \leq 1, \sum a_i = 1$$

$$0 < p_i \leq 1, \; i = 1 \ldots n$$

The problem is to compute vectors $\vec{a}$ and $\vec{p}$ for a given n and Δt that describe a DPH, which approximates the discretized input distribution function as closely as possible. Finding optimal n and Δt is also important, but was not the primary concern of this paper. Both parameters have a great influence on the result [5], but often one or both parameters might be dictated by the later application of the DPH. Some general guidelines on choosing n and Δt are presented in the experiments section 5.

2. Phases and Proxels

2.1. Phase-Type Distributions

The time-to-absorption in a Markov chain that is a phase-type approximation tries to mimic the behavior of the original distribution function. CPH have been used for a number of years to approximate non-Markovian distribution functions. DPH have only recently become of interest again [8, 4], since they hold some advantages over the continuous type.

Firstly DPH can approximate finite-support distributions functions such as the Uniform distribution fairly accurately, which CPH cannot. For example, the jumps in the Uniform PDF can only be approximated by steep slopes, when using CPH, whereas the discrete nature of DPH allows jumps. Secondly, the goal functions to be satisfied when fitting a DPH are mathematically easier than those of a CPH (e.g. sums instead of integrals), making the implementation of an algorithm easier.

One of the main reasons for choosing DPH for the approximation of non-Markovian distributions was their intended combination with the proxel-based method (see section 2.2). It is also based on discrete-time Markov chains, and in order to use phase-type distributions, these also had to be discrete.

2.2. The Proxel-Based Method

The proxel-based method for the analysis of discrete stochastic systems was introduced in [7] and is based in the method of supplementary variables. The method creates and analyzes the state space of a model on-the-fly. Non-Markovian distribution functions are approximated by using their instantaneous rate functions (IRF). This results in a DTMC of the structure shown in Figure 2.2. In contrast to the usual supplementary variable-based methods, proxels do not require differential equations, but are conceptually simple. For some classes of models the method shows significant runtime and accuracy improvements over discrete event simulation [10].

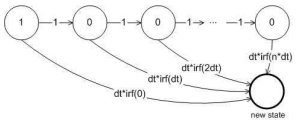

Figure 2. Proxel approximation of non-Markovian distribution function

2.3. Combining Proxels and Phases

Both the method of supplementary variables and DPH are ways to describe a non-Markovian distribution function using a discrete-time Markov chain. This is necessary in state space-based methods, since they mathematically analyze a Markov chain that contains all possible states of the model.

The proxel-based method calculates the approximation at runtime using the IRF, producing one Markov chain state per time step. These runtime calculations can become quite expensive, depending on the type of distribution. The proxel representation of finite support distribution functions is already good. However, if Δt is small, the representation of differentiable and smooth distribution functions such as Weibull can become of high order. By contrast, DPH can often approximate these functions well with just a few phases.

DPH can be integrated into the proxel algorithm easily [9], and reduce the runtime computation cost and the number of Markov chain states used to represent some distributions. Experiments in combining the proxel-based method with DPH show significant runtime advantages; one is shown in the next subsection.

2.4. Experiment

An experiment was conducted to test the possibility of combining proxels and DPH, and to reveal possible advantages. The Petri net shown in Figure 2.4 ($T_{BA} \sim U(1,2),\ T_{AB} \sim W(4,2)$) was simulated using proxels and a combination of proxels and DPH. In the first experiment, both transitions were approximated using the IRF. The second experiment replaced transition T_{AB} with a DPH of order 3. For $\Delta t = 0.01$, the SPD (see section 5.3 Table 2) as error had a value of $1.02e-05$ and the approximation needed less than $1s$ to be computed.

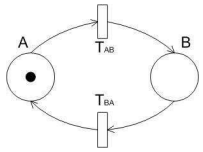

Figure 3. Simple test Petri net

The results of the experiment can be seen in Table 1. "#prox" refers to the total number of proxels (or Markov chain states) processed, which is also a measure of the complexity of the algorithm. While the simulation results themselves are very similar, both the number of states processed and the computation time needed is reduced to less than $\frac{1}{4}$ when phase approximation is used for the Weibull transition.

The main improvement is the drastic reduction of the state space. The number of proxels (= Markov chain states) needed to approximate the Weibull distribution, is equal to the number of the discrete time steps of that distribution, which is in the case of $\Delta t = 0.1$ more than 100. The DPH approximation on the other hand consists of only 3 Markov chain states. This reduces the state space of the model to about $\frac{1}{10}$. This promising result suggests that the combination of proxels and DPH will improve the performance of the proxel-based method and result in an effective algorithm. A more detailed experiment can be found in [9].

Table 1. Experimental results

	$\Delta t = 0.1$	$\Delta t = 0.01$
Proxels	#prox=$42,424$	#prox=$4,211,422$
	time=0.125	time=$16,062$
Combination	#prox=$6,713$	#prox=$589,102$
	time=0.031	time=3.687

3. Optimization Methods

The problem of finding parameters for a DPH that minimizes the distance to the input distribution can be interpreted as an optimization problem. Furthermore, optimization does not require the impose preconditions on the distribution function, and can handle boundary conditions. For given n and Δt, the error function to be minimized is only dependent on the a_i and p_i. The implemented error functions will be described in section 5.3. The resulting objective functions are not differentiable. Therefore, the gradient has to be approximated.

The optimization problem itself is multidimensional, global, nonlinear and has boundary conditions for the DPH parameters. Only few algorithms exist that fulfill all these requirements, therefore some of the chosen algorithms had to be adapted. The implemented methods include gradient descent (GD), Nelder-Mead Simplex (SX), Simulated Annealing (SA), Simultaneous Perturbation Stochastic Approximation (SPSA) and the Augmented Lagrange Penalty Function (ALPF).

GD and SX are local optimization methods, and therefore dependent on the start vector, whereas SA and SPSA are stochastic methods, and thus not as sensitive to the initial parameter values. Of the implemented al-

gorithms only ALPF is designed to handle boundary conditions by including them in the goal function. To enforce these restrictions with the other four, the vectors $\vec{a}$ and $\vec{p}$ are corrected after each iteration.

SX, GD and SA were successfully adapted to the problem, and will be described in this section. They were evaluated based on the experiments described in section 5. We were not successful in finding parameter settings that efficiently produced good results for ALPF and SPSA. They will only be described briefly among the discarded approached in the next section.

3.1. Gradient Descent

Gradient descent methods are standard optimization techniques. The one implemented here successively improves an initial vector by determining the gradient of the goal function in every direction and then shifting the current guess in the direction of steepest descent by a given step size. This is a slight modification to the algorithm described in [1]. An advantage of GD is that the goal function improves with every step, but this strict downward movement can result in getting trapped in a local optimum. The step size is crucial to the success of GD, a small one makes the algorithm converge very slowly, a large one can result in skipping over valleys in the goal function. For that reason the step size starts out at a large value, and is decreased every time no further improvement can be achieved, until a threshold is reached. To increase the speed of GD it was modified to take as many steps in the direction of steepest descent as possible, until no further improvement can be made.

GD works well for the DPH approximation where the original distribution function is smooth, for example Weibull or Normal. It does not work well for distributions that are not differentiable or have large gradients. The computation time of GD can also be large (up to minutes) because it advances in small steps and only in a direction parallel to one of the axes. The worst case is $2n + 1$ goal function evaluations per iteration. Nevertheless GD yields good solutions for some problems.

3.2. Nelder-Mead Simplex

The Nelder-Mead optimization algorithm is an extension of the well known Simplex algorithm to multiple dimensions. The idea of the algorithm is to take an initial $m + 1$ dimensional simplex, and let it move through the space of possible solutions, until the vertices accumulate around an optimum. Since this optimization problem is not linear it can not be guaranteed that the minimum found is global. For a detailed description of the algorithm and its operations, see [14].

SX is fast and works well for most distribution functions. It needs at most 3 goal function evaluations per iteration, unless the simplex is shrunk; on average this resulted in less than 3 function evaluations per iteration. During the test, SX did not always find the global optimal solution, but in most cases a good one.

3.3. Simulated Annealing

Simulated Annealing algorithms try to mimic the behavior of cooling metal, they start at a high temperature and accept almost any change; as the temperature gradually decreases, they tend more and more to reject changes that do not improve the goal function value, up until only accepting improvements. Through this it covers a larger portion of the space of possible solutions.

The implemented approach is based on the above describe Nelder-Mead Simplex and has been taken from [13, pp.451-455]. The modification is that to every newly computed goal function value a positive random value ($simLOG()$) is added, which is proportional to the temperature; a similar number is subtracted from the reference value. An improvement is thereby always accepted, and occasionally also a deterioration.

SA is also suitable for the approximation of DPH. Owing to its stochastic nature, it has a high probability of finding a global optimum, at the expense of replications. Most of the time SA is slower than SX, but it yields better results.

4. Abandoned Approaches

In this section some approaches to the approximation of DPH will be described, that did not prove to be successful. The evaluations of ALPF and SPSA are based on experiments, which were conducted to try to tune the methods to this particular problem. Another possible solution method considered, was solving a system of nonlinear equations.

ALPF is an optimization method for problems with boundary conditions. A detailed description of the algorithm can be found in [2]. ALPF was slower than the other implemented algorithms and needed several optimization runs to adapt the penalty parameters. It was not possible to find appropriate general parameters for the algorithm within a limited amount of time.

SPSA is a gradient-based optimization technique that has the advantage of only needing two goal function evaluations per iteration, regardless of the number of independent variables. We did not achieve better results with SPSA than with the other methods. We were not able to find a set of generally applicable parameters for the approximation of DPH.

As mentioned, the fitting of a DPH can also be interpreted as the solving of a system of nonlinear equations. The system contains one equation for every time step of the Markov chain. One algorithm for the solution of nonlinear systems of equations is the Newton-Raphson method. The tested implementation was adapted from the one in [13, pp.379-383]. The method did not prove suitable for the approximation of DPH for several reasons: The algorithm is not designed to work with boundary conditions; these had to be artificially enforced within each iteration. Furthermore, square matrices are required and therefore only $2n$ time steps of the distribution can be considered. Through the inversion of a matrix with entries of differing magnitude the method can become numerically unstable. The development of the error was not predictable.

5. Experiments

The experiments described in this section were used to determine the influence of different parameters on the algorithms performance and results. The tested algorithms are gradient-descent (GD), Nelder-Mead Simplex (SX) and Simulated Annealing (SA). The varied parameters are the input distributions, the error functions, the number of phases, and the input size (number or size of time steps). The evaluated performance criteria were computation time (taken on a P4 CPU with 2.6 GHz and 512 MB RAM) and the resulting error.

5.1. Different Standard Distributions

The following standard distributions tested were taken from [6], where they were used for a similar benchmark test. The discretization values for Δt and t_{max} were taken if not otherwise stated:

			Δt	t_{max}
• W1	$\sim$	Weibull(1,1.5)	0.1	4
• W2	$\sim$	Weibull(1,0.5)	0.1	4
• L1	$\sim$	Lognormal(1,1.8)	0.05	2.5
• L2	$\sim$	Lognormal(1,0.8)	0.05	2.5
• L3	$\sim$	Lognormal(1,0.2)	0.05	2.5
• U1	$\sim$	Uniform(0,1)	0.05	2
• U2	$\sim$	Uniform(1,2)	0.05	3
• SE	$\sim$	Shifted Exponential	0.05	3
• ME	$\sim$	Matrix Exponential	0.05	3

Because of the sensitivity of the results to n, for each algorithm three cases were tested: 2, 4 and 8 phases - sometimes also 16 and 20. The best fits can be seen in Figure 4. The Weibull and two of the Lognormal distributions are accurately approximated using only a small number of phases. L3 needs 16 phases to be approximated accurately. U1 can be approximated exactly by

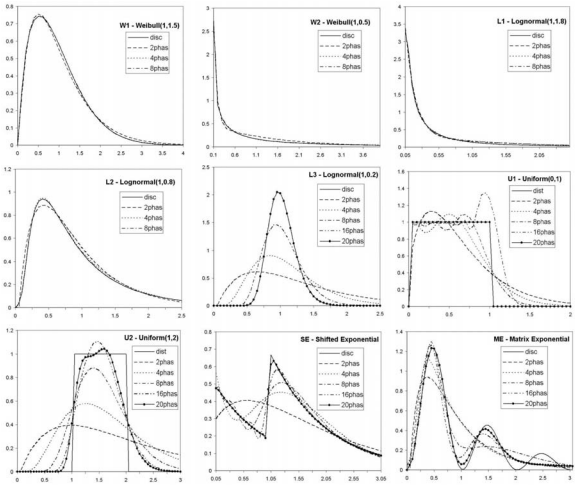

Figure 4. Example distributions and fits

20 phases, since all of the probability is contained in the first 20 time steps. The approximations of U2, SE and ME gets close to the original distribution function with an increasing number of phases, even if it contains non-differentiable segments. With a sufficient number of phases, the DPH approximation can get arbitrarily close to the original distribution. The trivial solution is taking as many phases as time steps, which is what proxels do (see section 2.2).

When comparing the results to [6], the DPH fits are comparably for Weibull and Lognormal, where CPH performed well. Better fits could be obtained for distributions, where CPH did not perform as well. In [4] the same distributions were used to test a fitting algorithm for DPH. The plotted results are very similar to the ones in figure 4.

5.2. Different Optimization Methods

The different methods implemented were also tested on all nine distributions, using 2, 4 and 8 phases. When approximating a distribution using SX and just 2 phases, the algorithm did not terminate, therefore that configuration is not depicted in the diagrams.

GD (see figure 5) performs very well on all distributions, when just 2 phases are used. When using higher order DPH, computation costs rise sharply up to $1000s$, which is not feasible for most distributions. Irregular distributions, like SE, ME or Uniform do not show such a drastic negative effect (see section 5.4).

SX (see figure 6) is much faster than GD. SX works well for fairly smooth distribution functions like Weibull or Lognormal. The computation time differs and is not easily predictable, but always less than $1s$.

SA (figure 7) is slower than SX, but the computations time is still within a few seconds and SA is more

predictable. It finds an adequate solution for DPH with 4 or more phases and at higher orders even better solutions than GD or SX. When the number of independent variables increases, their interdependencies grow more complex, and the stochastic nature of SA explores the space of possible solutions most efficiently.

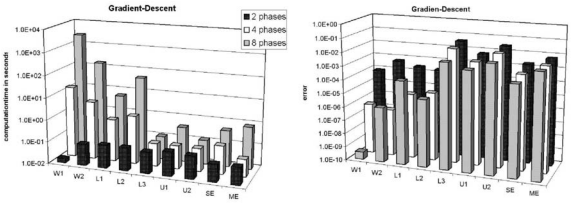

Figure 5. Runtime and error of Gradient-descent method (logarithmic scaling)

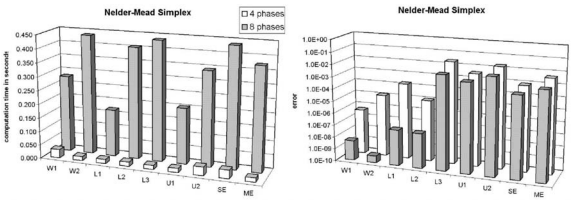

Figure 6. Runtime and error (logarithmic scaling) of Simplex method

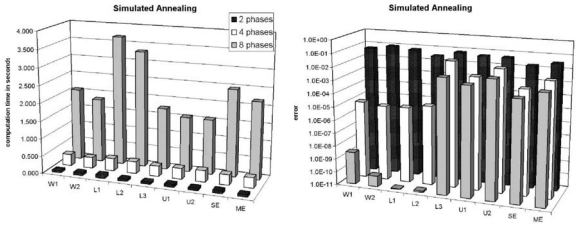

Figure 7. Runtime and error (logarithmic scaling) of Simulated Annealing method

In general, GD works well with small phase numbers at acceptable computation cost of up to $10s$. SX yields adequate solutions for higher phase numbers within $1s$ and the slower SA can find better approximations in most cases at the expense of higher computation cost of a couple of seconds.

5.3. Different Error Functions

The error functions that were implemented are the squared PDF difference (SPD), the squared CDF difference (SCD) and the absolute PDF difference (APD) (Table 2). There are other distance measures that can be considered, like the relative entropy, and the selection does not claim to be exhaustive. However, the chosen distance measures are examples of ones that are much easier to compute for DPH, than for CPH.

Table 2. Error functions

SPD	$\sum \left(f_i - \hat{f}_i\right)^2 = \sum \left(d_i - c_i\right)^2$
APD	$\sum \left\lvert f_i - \hat{f}_i \right\rvert = \sum \lvert d_i - c_i \rvert$
SCD	$\Delta t * \sum \left(F_i - \hat{F}_i\right)^2$

Figure 5.3 shows the four distributions with the most significant difference in fits for the error functions. SPD seems to mimic the shape of the curve closest, and is the default error function. Further practical application of the resulting DPH might change that choice.

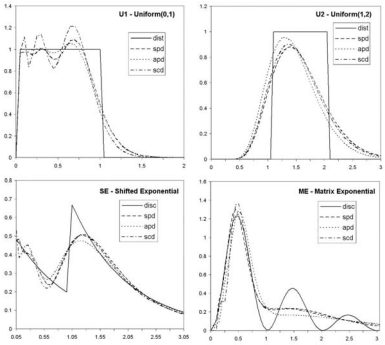

Figure 8. Fits with different error functions

The error development with increasing number of time steps is shown in figure 9. SCD and APD are not dependent on the input size. SPD decreases with increasing input size, since through the squaring of the individual values, larger ones are amplified. The absolute values with less time steps are necessarily larger, and therefore the sum of the squares is larger. Since the number of time steps is much higher than n, actual differences in the fits do not play a role in this statistic.

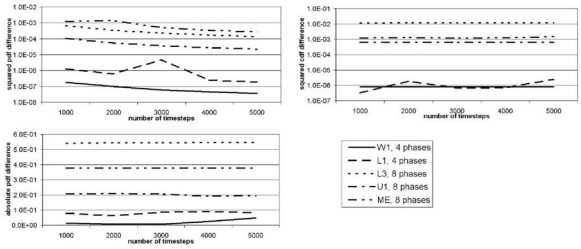

Figure 9. Error with increasing number of discretization steps

5.4. Number of Phases

Another parameter varied was the order of the resulting DPH n. A small value for n is desirable, because

the fitting algorithm is much faster and the state space of the model does not grow as fast. Figure 4 shows that the larger the value of n, the better the approximation. The optimum value is always a trade-off and depends largely on the requirements of the intended application, and can therefore be different for each distribution function and parameter setting.

The increase in runtime is dependent on the distribution being approximated. Paradoxically, the distributions that can be approximated close enough to reach the error threshold also have the sharpest increase in computation cost (W1, W2, L2) (see figures 5.4 and 5.4). This is probably due to the fact, that there is almost always room for improvement before the threshold is reached. On the other hand, the distributions that can only be approximated to a certain point, reach that limit more quickly (SE, U1, U2).

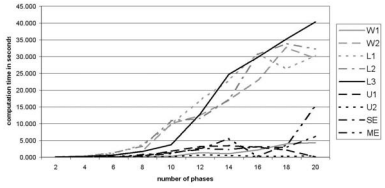

Figure 10. Runtime for different distributions with increasing number of phases

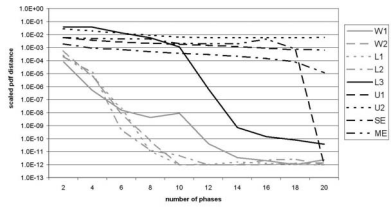

Figure 11. Error for different distributions with increasing number of phases

5.5. Number of Time Steps

The size of the input has an influence on the performance of the algorithm and on the quality of the result. To vary the input size, the step size Δt was adapted with unchanged t_{max} to produce the desired number of time steps. To get comparable results, the number of iterations for GD was limited .

Figure 12, Figure 13 and Figure 14 show the runtime and error for W1, L3 and ME, which exemplify three different algorithm behaviors. In all three cases the best error function value for a parameter configuration slightly increases when the number of time steps decreases, which is explained in section 5.3.

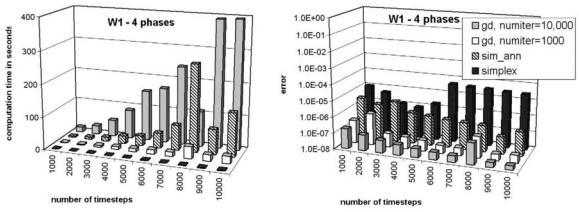

Figure 12. Runtime and error (logarithmic scaling) for W1 with increasing input size

GD usually finds the optimal DPH, but for W1 with input size 10000 at high costs ($400s$). The runtime of SA is proportional to the input size. SX has a short runtime of only a few seconds, but does not find a good solution for half of the cases. GD with 1000 iterations finds the best results at computation cost up to $30s$.

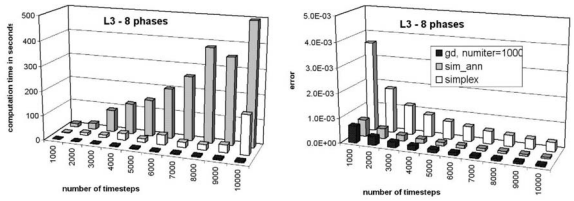

Figure 13. Runtime and error for L3 with increasing input size

For L3 the error results for SX and GD are equally good, and GD only needs 1000 iterations. SA is slower and needs up to $500s$, since it performs at least five replications. The lower computation costs for SX do not pay off, since the results are worse. The optimal method here would again be GD with 1000 iterations.

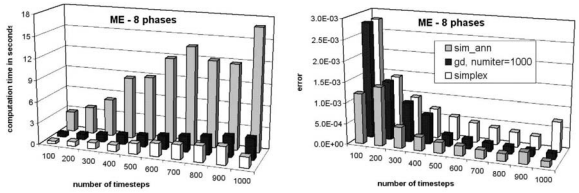

Figure 14. Runtime and error for ME with increasing input size

For ME, SA finds the best results; this is due to the irregular shape of the distribution function. GD has low computation cost ($< 3s$) and finds good solutions. SX reaches comparable results for smaller input sizes, but has consistently low computation costs.

When limiting the number of iterations for GD, the runtime of all three methods is usually proportional to the input size. The error of SX is not acceptable for larger input sizes. SA produces good results for almost all input sizes.

5.6. Experiments Summary

The experiments were conducted to investigate the behavior of the fitting methods with different parameter settings. DPH can approximate the test distribution functions equally well or better than CPH. Gradient descent yields good results within a few seconds for small n. Simplex works for larger n, and terminates usually in under $1s$. The computation costs of Simulated Annealing are typically a couple of seconds, but it usually finds very good solutions. All tested error functions resulted in good fits, with slightly different shapes for irregular distribution functions. A large number of phases results in a better fit, but also in higher computation cost, which is most pronounced on fairly smooth distribution functions. A larger input size (smaller time step) up to 10000 caused higher computation cost and worse fits.

6. Summary & Outlook

This paper presented an approach to the approximation of DPH. In contrast to former fitting algorithms the problem was solved using common optimization methods, and the algorithm's performance and the practical applicability of the results were evaluated. One advantage of this general approach is, that any kind of distribution function can be approximated that is defined for $t > 0$, there are no other preconditions that have to be fulfilled. Experiments were conducted to determine the influence of distribution and optimization parameters.

The time needed to approximate a non-Markovian distribution function with reasonable accuracy using the proposed method is usually under $10s$. Considering that the simulation time of a real system usually exceeds a few seconds, it now seems feasible to turn a simulation model into a Markov chain and then solve it. DPH can extend and improve of the proxel-based simulation method. The developed fitting method is another step towards a state space-based simulation algorithm that can compete with common event-driven simulation systems regarding runtime and accuracy, and which is not susceptible to the uncertainty of stochastics.

6.1. Future Work

A desirable feature would be the automatic detection of optimal values for Δt and n; so far one or both variables were dictated by the application of the DPH. Part of our current research is comparing the runtime and results of the developed algorithm with EMPHT [12], which is a CPH fitting tool. A comparison with a fitting tool for DPH has yet to be conducted.

The cheap approximation of DPH is a step towards a generalization of the proxel-based method that includes DPH. A goal of our future research is to combine proxels and DPH to form a general state space-based simulation method. The intended algorithm should detect whether a preprocessing of a DPH or a runtime approximation via proxels is more suitable for a distribution. We believe that such an algorithm can be fast, flexible and a competitor to currently used simulation systems.

References

[1] K. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 1989.

[2] M. A. Bhatti. *Practical Optimization Methods*, chapter Augmented Lagrange Penalty Function, pages 590–608. Springer Verlag, 2000.

[3] A. Bobbio and A. Cumani. Ml estimation of the parameters of a ph distribution in triangular canonical form. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*. Elsevier Science Publishers, 1992.

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

[5] A. Bobbio, A. Horváth, and M. Telek. The scale factor: A new degree of freedom in phase type approximation. In *Proc. of 3rd International Performance & Dependability Symposium (IPDS '02)*, June 2002.

[6] A. Bobbio and M. Telek. A benchmark for ph estimation algorithms: Results for acyclic-ph. stochastic models. *Stochastic Models*, 10:661–667, 1994.

[7] G. Horton. A new paradigm for the numerical simulation of stochastic petri nets with general firing times. In *Proceedings of the European Simulation Symposium 2002*. SCS European Publishing House, 2002.

[8] A. Horváth and M. Telek. Phfit: A general phase-type fitting tool. In *Proc. of 12th Performance TOOLS*, April 2002. Lecture Notes in Computer Science 2324.

[9] C. Isensee, S. Lazarova-Molnar, and G. Horton. Combining proxels and discrete phases. In *Proceedings of ICMSAO*, February 2005.

[10] S. Lazarova-Molnar and G. Horton. Proxel-based simulation of stochastic petri nets. In *Simulation und Visualisierung 2004*. SCS European Publishing House, 2004.

[11] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. The John Hopkins University Press, Baltimore, 1981.

[12] M. Olsson. The empht-programme. Technical report, Department of Mathematics, Chalmers University of Technology, and Göteborg University, Sweden, 1998.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C, Second Edition*. Cambridge University Press, 1992.

[14] H. Shekarforoush, M. Berthod, and J. Zerubia. Direct search generalized simplex algorithm for optimizing nonlinear functions. Technical report, INRIA, France, 1995.