Reducing the Cost of Generating APH-distributed Random Numbers

Philipp Reinecke[1], Miklós Telek[2], and Katinka Wolter[1]

[1] Freie Universität Berlin
Institut für Informatik
Takustraße 9
14195 Berlin, Germany
{philipp.reinecke, katinka.wolter}@fu-berlin.de
[2] Budapest University of Technology and Economics
Department of Telecommunications
1521 Budapest, Hungary
telek@webspn.hit.bme.hu

Abstract. Phase-type (PH) distributions are proven to be very powerful tools in modelling and analysis of a wide range of phenomena in computer systems. The use of these distributions in simulation studies requires efficient methods for generating PH-distributed random numbers. In this work, we discuss algorithms for generating random numbers from PH distributions and propose two algorithms for reducing the cost associated with generating random numbers from Acyclic Phase-Type distributions (APH).

1 Introduction

Phase-type (PH) distributions have been widely used in modelling various phenomena such as response times, inter-arrival times and failure times in computer systems [1–3], and several tools that fit phase-type distributions to trace data have been developed [4, 5]. The fact that there are simple and elegant solution techniques available for PH distributions has made them appealing for analytic solutions.

PH distributions can also be employed in simulation studies, where they allow the introduction of realistic response-time distributions obtained from measurements into simulations without modification of the typically Markovian simulation tool. As such simulations often require many random variates and are repeated many times, generating PH-distributed random numbers efficiently is important. In this work we investigate the efficiency of generating random numbers from continuous PH distributions. Due to the fact that the Markovian representation of PH distributions is not unique the key issue to investigate is which representation of a PH distribution is most efficient for random-number generation.

In [6] we posed the following optimisation problem: Starting from a Markovian representation of a PH distribution, find the (not necessarily minimal)

Markovian representation that minimises the cost associated with generating random numbers. In this paper we study this optimisation problem for Acyclic Phase-Type (APH) distributions. We provide a result on the optimal representation and develop two algorithms that transform a given APH representation into a representation with lower simulation cost.

The paper is structured as follows. In the next section we introduce the considered model class and the notation used throughout the paper. We then describe a number of algorithms for generating random numbers from phase-type distributions (Section 3) and derive average costs (Section 4). In Section 5 we study the problem of optimising bi-diagonal representations for random-number generation. Section 6 illustrates the application of our algorithms to several theoretic and fitted phase-type distributions. Finally, in Section 7 we conclude with an outlook on future work.

2 Definitions and Notation

Continuous phase-type (PH) distributions represent the time to absorption in a continuous-time Markov chain with one absorbing state [7]. PH distributions are commonly specified as a tuple $(\boldsymbol{\alpha}, \mathbf{A})$ of the initial probability vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and the transient part $\mathbf{A} = \{a_{ij}\}, 1 \le i, j \le n$ of the generator matrix, also referred to as transient generator matrix. The probability density function, the cumulative distribution function, the Laplace-Stieltjes Transform (LST) of the CDF, and the kth moment, respectively, are [4, 7, 8]:

$$\begin{aligned}
f(x) &= \boldsymbol{\alpha} \mathrm{e}^{\mathbf{A}x} \mathbf{a}, \\
F(x) &= 1 - \boldsymbol{\alpha} \mathrm{e}^{\mathbf{A}x} \mathbb{1}, \\
\tilde{F}(s) &= \alpha_{n+1} + \boldsymbol{\alpha}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{a}, \text{ and} \\
E\left[X^k\right] &= k! \boldsymbol{\alpha}(-\mathbf{A})^{-k} \mathbb{1}.
\end{aligned}$$

where $\mathbf{a} = -\mathbf{A}\mathbb{1}$, $\mathbf{I}$ is the identity matrix, and $\mathbb{1}$ is the column vector of ones, both of appropriate size.

Phase-type distributions have rational LST. It follows that the eigenvalues of the transient generator matrix are the poles of the LST of the distribution [9].

Definition 1. *The $(\boldsymbol{\alpha}, \mathbf{A})$ representation of a phase-type distribution is called* Markovian *if $\boldsymbol{\alpha} \ge 0$, $\boldsymbol{\alpha}\mathbb{1} = 1$, $a_{ij} \ge 0, 1 \le i \ne j \le n$ and $\mathbf{a} = -\mathbf{A}\mathbb{1} \ge 0$. Then, the* generator matrix *of the associated CTMC is*

$$\overline{\mathbf{A}} = \begin{pmatrix} \mathbf{A} & \mathbf{a} \\ \mathbf{0} & 0 \end{pmatrix}.$$

Definition 2. *The* size *of the $(\boldsymbol{\alpha}, \mathbf{A})$ representation is the size of the vector $\boldsymbol{\alpha}$, which is equal to the size of the square matrix $\mathbf{A}$.*

The $(\boldsymbol{\alpha}, \mathbf{A})$ representation is not unique. In particular, another representation of the same size can be obtained by a similarity transformation using a matrix $\mathbf{B}$:

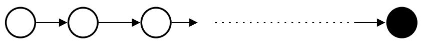

Fig. 1. CTMC for a bi-diagonal representation.

Definition 3. *When* $\mathbf{B}$ *is invertible and* $\mathbf{B}\mathbb{1} = \mathbb{1}$*, then the* similarity transform

$$(\boldsymbol{\alpha}\mathbf{B}, \mathbf{B}^{-1}\mathbf{A}\mathbf{B})$$

provides another representation of the same distribution, since its CDF is

$$1 - \boldsymbol{\alpha}\mathbf{B}\mathrm{e}^{\mathbf{B}^{-1}\mathbf{A}\mathbf{B}x}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathbf{B}\mathbf{B}^{-1}\mathrm{e}^{\mathbf{A}x}\mathbf{B}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathrm{e}^{\mathbf{A}x}\mathbb{1}.$$

In the following we refer to a PH representation as being *bi-diagonal* (cf. Figure 1) if it meets the following requirements:

Definition 4. *A* bi-diagonal representation *of a PH distribution* $(\boldsymbol{\alpha}, \mathbf{A})$ *has* $a_{ii} < 0, a_{ii+1} = -a_{ii}$ *and* $a_{ij} = 0$ *for* $j < i$ *and* $j > i+1$*. An alternative notation is* $(\boldsymbol{\alpha}, \boldsymbol{\Lambda})$*, with* $\boldsymbol{\Lambda} = (a_1, \ldots, a_n)$ *a row vector of length* n *and* $a_i = -a_{ii}$*.*

Note that for a bi-diagonal representation the eigenvalues of the transient generator matrix $\mathbf{A}$, and thus the poles of the LST of the distribution function, are given by the entries of the diagonal, a_{ii}.

In this paper we are concerned with acyclic phase-type distributions (APH):

Definition 5. *An* acyclic phase-type distribution (APH) *is a phase-type distribution that has an acyclic Markovian representation.*

We make use of the following important bi-diagonal representation:

Definition 6. *[10] The* Canonical Form 1 (CF-1 form) *is a bi-diagonal representation* $(\boldsymbol{\alpha}, \boldsymbol{\Lambda})$ *where* $\boldsymbol{\alpha}$ *is Markovian and the rates* a_i *in* $\boldsymbol{\Lambda}$ *are in increasing order:* $a_1 \leq a_2 \leq \cdots \leq a_n$*.*

The next theorem, which we state without proof, ensures that the distributions we consider have at least one bi-diagonal representation with a Markovian initial vector:

Theorem 1. *[10, 11] Every acyclic phase-type distribution with a representation of size* n *has a unique CF-1 representation of the same size.*

We remark that smaller CF-1 representations may exist if there is redundancy in the original representation [7, 12, 13].

The CF-1 form for an APH given as $(\boldsymbol{\alpha}, \mathbf{A})$ can be obtained by the transformation provided in [12]. It presents a way to construct a similarity matrix $\mathbf{B}$, which transforms $\mathbf{A}$ to a bi-diagonal matrix $\mathbf{G}$ where the entries of the diagonal are the ordered eigenvalues of the matrix $\mathbf{A}$, i.e., $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A}\mathbf{B}$. The same similarity matrix can be used to compute the initial vector $\boldsymbol{\gamma} = \boldsymbol{\alpha}\mathbf{B}$.

Example 1. Consider the phase-type distribution given by

$$\boldsymbol{\alpha} = (0.3, 0.4, 0.3), \ \mathbf{A} = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -2 & 0.5 \\ 0 & 3 & -3 \end{pmatrix}.$$

The eigenvalues of $\mathbf{A}$ are $-0.205168, -1.85629$, and -3.93854. Let $\mathbf{G}$ be the corresponding CF-1 bi-diagonal matrix. From $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A}\mathbf{B}$ we obtain the similarity transformation matrix

$$\mathbf{B} = \begin{pmatrix} 0.931611 & 0.0683893 & 0 \\ 0.740474 & 0.132575 & 0.126951 \\ 0.794832 & 0.205168 & 0 \end{pmatrix}.$$

We compute the initial probability vector as $\boldsymbol{\gamma} = \boldsymbol{\alpha}\mathbf{B}$ and get the CF-1 form $(\boldsymbol{\gamma}, \boldsymbol{\Lambda}_G) = ((0.814123, 0.135097, 0.0507803), (0.205168, 1.85629, 3.93854))$.

3 Generation of PH-distributed Random Numbers

We now discuss two methods for generating random variates from a general PH distribution given in Markovian form. While one may apply numerical inversion of the distribution [14], we consider approaches based explicitly on the CTMC interpretation. The discussed algorithms all rely on the following elementary operation for drawing an exponentially distributed random number:

$$\mathrm{Exp}(\lambda) = -\frac{1}{\lambda}\ln(U),$$

where U denotes a $[0, 1]$ uniformly distributed pseudo-random number.

The most natural way to obtain a PH-distributed random number is to play the CTMC until absorption. By 'play' we mean to simulate the state transitions of the CTMC according to the following basic steps. Let $\boldsymbol{e_i}$ denote the row vector with 1 at position i, and 0 everywhere else.

```
Procedure Play:

1) clock= 0, draw an $\boldsymbol{\alpha}$-distributed discrete sample for the initial state,
2) the chain is in state $i$
     • draw an $\boldsymbol{e}_i(-\mathrm{diag}\langle 1/a_{ii}, 0\rangle\overline{\mathbf{A}}+\mathbf{I})$-distributed discrete sample for the next state,
     • clock += $\mathrm{Exp}(-a_{ii})$,
     • if the next state is the absorbing one go to 3), otherwise go to 2)
3) return the clock value
```

We point out that `Play` is suited to general PH distributions in an arbitrary form, where each phase may have several successor phases. If our simulation is such that it involves only acyclic-phase type distributions in a bi-diagonal representation (such as the CF-1 form), we can make use of the following structural restriction: For each phase, there is exactly one successor phase; consequently, there is no need to randomly choose the next state. This observation allows the following simplification of `Play`:

```
Procedure SimplePlay:
1) clock= 0, draw an $\boldsymbol{\alpha}$-distributed discrete sample for the initial state.
2) The chain is in state $i$.
    • clock += Exp($-a_{ii}$),
    • $i$ += 1,
    • if the next state is the absorbing state go to 3), otherwise go to 2).
3) Return the clock value.
```

In the next section we discuss costs of random-number generation using these algorithms.

4 Average Costs of Generating PH-Distributed Random Numbers

As we saw in the previous section, PH random number generation requires uniform random variates, both for state selection and for generating exponential random variates. Furthermore, for each exponential random variate a logarithm operation must be performed. Both operations are expensive in terms of computing time and can significantly increase the running-time of simulations that require many random variates. Therefore, we consider the following complexity metrics:

- *#uni,* the number of required uniform random variates, and
- *#ln,* the number of logarithms that need to be computed.

The average cost associated with drawing a random variate from a phase-type distribution depends on the average number n^* of state transitions up to absorption,

$$n^* = \boldsymbol{\alpha}(\text{diag}\langle 1/a_{ii}\rangle \mathbf{A})^{-1}\mathbb{1}.$$

For APH in bi-diagonal form this reduces to

$$n^* = \boldsymbol{\alpha}\boldsymbol{\nu}^{\mathsf{T}},$$

where $\nu = (n, n-1, \ldots, 1)$. Thus $n^* = \sum_{i=1}^{n} \alpha_i(n-i+1)$.

Both procedures require one uniform random variate to choose the initial state. The `Play` procedure then needs two uniform random variates per step, because the next phase is chosen randomly. `Play` therefore requires $\#uni = 2n^* + 1$ uniform random variates, while `SimplePlay` requires only $\#uni = n^* + 1$ uniforms. The number of logarithms required for the `Play` and `SimplePlay` procedures is $\#ln = n^*$, since in each phase an exponentially distributed random variate is drawn.

We can thus conclude that for generating random numbers from an APH efficiently we should transform the distribution into a bi-diagonal representation and then apply the `SimplePlay` procedure.

5 Optimal Representations for APH-PRNG

As illustrated in Section 4, average costs for random-number generation depend mainly on the number of visited states, n^*. In [6] we posed the problem of finding a Markovian representation that minimises n^*.

In the following we tackle this optimisation problem for acyclic phase-type (APH) distributions $(\boldsymbol{\alpha}, \mathbf{A})$ in CF-1 form. We choose the CF-1 form as our starting point because bi-diagonal forms allow efficient random-variate generation, using the procedure `SimplePlay`, and because APHs are commonly given in CF-1 form (e.g. by the phase-type fitting tool PhFit [4]). Furthermore, Theorem 1 ensures that the results for CF-1 are applicable to the whole APH class.

In order to solve the optimisation problem, we try to find a bi-diagonal representation $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ for which the average number of traversed states,

$$n^* = \boldsymbol{\alpha}^* \boldsymbol{\nu} = \sum_{i=1}^{n} \alpha_i^* (n - i + 1). \tag{1}$$

is minimal.

From the right side of (1) it is immediately obvious that, in order to reduce n^*, probability mass must be shifted to the higher indices of the initial probability vector $\boldsymbol{\alpha}$. Formally, the new probability vector $\boldsymbol{\alpha}'$ must be stochastically larger than $\boldsymbol{\alpha}$:

Definition 7. *The* stochastic ordering *[15] on the set of stochastic vectors of size n is defined as follows:*

$$\boldsymbol{\alpha} \leq_{st} \boldsymbol{\alpha}' \Leftrightarrow 1 - \Pr\{\boldsymbol{\alpha} \leq k\} \leq 1 - \Pr\{\boldsymbol{\alpha}' \leq k\} \text{ for } k = 1, \ldots, n,$$

where

$$\Pr\{\boldsymbol{\alpha} \leq k\} := \sum_{i=1}^{k} \alpha_i.$$

At the same time, $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ must represent the same distribution as $(\boldsymbol{\alpha}, \mathbf{A})$, and hence the LST of its distribution function must have the same poles. This implies that the matrices $\mathbf{A}$ and $\mathbf{A}^*$ must have the same eigenvalues. In the bi-diagonal form the eigenvalues are the entries of the diagonals. Changing the order of the diagonal elements does not change the eigenvalues, hence a representation where $\mathbf{A}^*$ is obtained by re-ordering the diagonal elements is guaranteed to have the same poles as $(\boldsymbol{\alpha}, \mathbf{A})$. Consequently, we consider shifting probability mass to the right by modifying the order of the rates along the diagonals. We propose the following operator:

Definition 8. *The* $\texttt{Swap}(\boldsymbol{\alpha}, \mathbf{A}, i)$ operator *exchanges the ith rate with the $(i+1)$th rate $(1 \leq i \leq n-1)$ on the diagonals in a bi-diagonal representation by swapping the ith and $(i+1)$th entry in the vector $\boldsymbol{\Lambda}$. The associated similarity* *transformation matrix* $\mathbf{B}$ *has the form*

$$\mathbf{B} = \begin{pmatrix} \ddots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & b_{i+1,i} & b_{i+1,i+1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots \end{pmatrix}$$

where

$$b_{i+1,i} = \frac{a_i - a_{i+1}}{a_i}, \text{ and } b_{i+1,i+1} = \frac{a_{i+1}}{a_i} \text{ for } 1 \leq i \leq n-1.$$

Where appropriate, we also use the $\texttt{Swap}(\boldsymbol{\alpha}, \boldsymbol{\Lambda}, i)$ notation to denote the same operator.

Remark 1. $\texttt{Swap}(\boldsymbol{\alpha}, \boldsymbol{\Lambda}, i)$ only involves the ith and $(i+1)$th entries of $\boldsymbol{\alpha}$ and $\boldsymbol{\Lambda}$, which allows for the analytically tractable expressions employed in the following. Operators that produce more complex permutations in a single step do not have this property. Furthermore, the Swap operator is sufficiently powerful to generate all permutations of a list [16].

Let $(\boldsymbol{\alpha}', \mathbf{A}')$ denote the result of applying $\texttt{Swap}(\boldsymbol{\alpha}, \mathbf{A}, i)$ on $(\boldsymbol{\alpha}, \mathbf{A})$. Because $(\boldsymbol{\alpha}', \mathbf{A}')$ is derived by applying a similarity transformation to $(\boldsymbol{\alpha}, \mathbf{A})$, both tuples represent the same distribution. Recall from Definition 3 that

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha}\mathbf{B}.$$

From this equation and the definition of $\mathbf{B}$ we immediately get the following properties of the result of the Swap operation:

$$\forall j \notin \{i, i+1\} : \alpha'_j = \alpha_j \tag{2}$$

$$\alpha'_i = \alpha_i + \alpha_{i+1}\frac{a_i - a_{i+1}}{a_i} = \alpha_i + \alpha_{i+1}(1 - \frac{a_{i+1}}{a_i}) \tag{3}$$

$$\alpha'_{i+1} = \alpha_{i+1}\frac{a_{i+1}}{a_i}. \tag{4}$$

Putting $(\boldsymbol{\alpha}', A')$ into (1) we obtain

$$n^*(\boldsymbol{\alpha}', \mathbf{A}') = n^*(\boldsymbol{\alpha}, \mathbf{A}) + \alpha_{i+1}(1 - \frac{a_{i+1}}{a_i}). \tag{5}$$

Equations (2)–(5) are valid for Markovian ($\boldsymbol{\alpha} \geq 0$) and non-Markovian ($\boldsymbol{\alpha} \in \mathbb{R}^n$) bi-diagonal representations.

If we restrict our attention to the Markovian bi-diagonal representations then we observe the following: The Swap operation with adjacent rates $a_i < a_{i+1}$, results in $n^*(\boldsymbol{\alpha}', A') \leq n^*(\boldsymbol{\alpha}, A)$ according to (5), because in this case $1 < \frac{a_{i+1}}{a_i}$, and α_{i+1} is non-negative. Consequently, by repeatedly exchanging adjacent rates $a_i < a_{i+1}$ such that each resulting representation is Markovian until no such operations are possible anymore, we can obtain a representation that has minimal costs n^*.

On the other hand, we note that the `Swap` operator will result in a non-stochastic vector $\boldsymbol{\alpha}'$ if $\alpha_i < \alpha_{i+1}(1 - \frac{a_{i+1}}{a_i})$, since then the resulting $\alpha_i' < 0$. In this case $(\boldsymbol{\alpha}', \mathbf{A})$ is a non-Markovian representation of the original phase-type distribution. This representation is not suitable as input for the random-number generation algorithms discussed in Section 3. Furthermore, both the stochastic ordering and n^* are only defined for (sub-)stochastic $\boldsymbol{\alpha}$. We must therefore avoid `Swap` operations that will result in non-stochastic $\boldsymbol{\alpha}'$. Based on these observations we propose the following

Lemma 1. *Given a Markovian representation $(\boldsymbol{\alpha}, \mathbf{A})$ in CF-1 form, the representation $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ that reverses the order of the rates is optimal with respect to n^* if $\boldsymbol{\alpha}^*$ is a stochastic vector. In this case, all bi-diagonal representations constructed by the `Swap` operator are Markovian.*

Proof. *The proof is composed by two steps. First we show that all bi-diagonal representations are Markovian if $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ is Markovian. In the second step we show by contradiction that $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ is optimal with respect to n^*.*

According to property (3), a `Swap` operation applied to a Markovian representation can result in a non-Markovian representation only if a larger rate a_{i+1} is exchanged for a smaller rate a_i. Starting from $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$, all representations can be obtained by a series of `Swap` operations in which a smaller rate a_{i+1} is exchanged for a larger rate a_i. If $(\boldsymbol{\alpha}^*, \mathbf{A}^*)$ is Markovian, none of these `Swap` operations can result in a non-Markovian representation.

To prove the first part of the lemma, assume that $(\boldsymbol{\alpha}', \mathbf{A}')$ with rates

$$a_1', \ldots, a_i', a_{i+1}', \ldots, a_n'$$

ordered such that $a_i' < a_{i+1}'$ is optimal. Then from (5) it follows that by exchanging a_i', a_{i+1}' using the `Swap` operator we can obtain a representation $(\boldsymbol{\alpha}'', \mathbf{A}'') =$ `Swap`$(\boldsymbol{\alpha}', \mathbf{A}', i)$ for which $n^*(\boldsymbol{\alpha}'', \mathbf{A}'') < n^*(\boldsymbol{\alpha}', \mathbf{A}')$. □

5.1 Heuristic Algorithms for Computing Optimal APH Representations

Lemma 1 states that if the reversed CF-1 form is Markovian, then it is optimal with respect to n^*. This optimal representation is easily computed by applying a similarity transformation, as illustrated in Example 1. In the following we develop algorithms for finding a Markovian APH representation that is close to optimal (with respect to n^*) when the reversed CF-1 form is non-Markovian.

The algorithms are best thought of as operating on the graph of all permutations of the rate vector $\boldsymbol{\Lambda}$. From each permutation exactly $n - 1$ other permutations can be reached by applying the `Swap` operation. If the reversed CF-1 is non-Markovian, then some of the permutations have non-Markovian initial vectors.

The most obvious approach proceeds by exploring the complete graph. This is equivalent to generating all permutations of $\boldsymbol{\Lambda}$ and minimising n^* over the subset of permutations whose initial vector is stochastic. The approach is easily implemented, e.g. as a modification to the Steinhaus-Johnson-Trotter algorithm for enumerating permutations [16] and is guaranteed to find the optimum. Unfortunately, since this method explores all $n!$ permutations for an APH of size n, it is infeasible for larger APH.

Lemma 1 provides the basis for the two computationally less expensive algorithms presented in this section. The underlying intuition is as follows: The optimal representation with respect to n^* is somewhere on the Markovian side of the boundary between the Markovian and the non-Markovian representations. Lemma 1 implies that the Markovian optimum is along one of the paths from the CF-1 to the reversed CF-1. We thus need to find the (Markovian) point where the path between CF-1 and reversed CF-1 crosses the boundary between the Markovian and non-Markovian representations.

Starting with the CF-1 form (i.e. inside the Markovian representations), we know from (5) that each exchange of two adjacent rates such that after the exchange the larger rate is moved to the left constructs a new element of the path that has lower n^*, provided the new representation is Markovian. Properties (3) and (4) thus define the direction along which to search for the optimal Markovian representation without enumerating all permutations and without explicitly computing n^* for the new representation. Our first algorithm follows from this intuition. It is a modified version of the Bubblesort algorithm [17] that attempts to re-order the rates into the reversed CF-1 form:

```
Algorithm BubblesortOptimise($\boldsymbol{\alpha}, \boldsymbol{\Lambda}$):
For $i = 1, \ldots, n-1$ do
    For $j = 1, \ldots, n-1$ do
        If $\boldsymbol{\Lambda}[j] < \boldsymbol{\Lambda}[j+1] \wedge (\boldsymbol{\alpha}', \boldsymbol{\Lambda}') :=$ Swap($\boldsymbol{\alpha}, \boldsymbol{\Lambda}, i$) is Markovian then
           $(\boldsymbol{\alpha}, \boldsymbol{\Lambda}) := (\boldsymbol{\alpha}', \boldsymbol{\Lambda}')$
        Else
           break
    done
done
Return $(\boldsymbol{\alpha}, \boldsymbol{\Lambda})$
```

Note that the algorithm does not perform `Swap` operations whose result would be non-Markovian, i.e. it does not cross the boundary between both types of representations. The algorithm terminates once either the reversed CF-1 form is reached or there are no re-orderings left that would result in a Markovian representation with lower cost n^*. While this may mean that the algorithm does not find Markovian representations hidden 'behind' non-Markovian ones, it is necessary because n^* has no meaning for non-Markovian representations.

Our second algorithm starts from the reversed CF-1 form and searches for the point where the path towards the CF-1 first crosses the border to the Markovian representations. The path is constructed by swapping pairs of rates such that in the result the higher rate is to the right (which means that the result is closer to the CF-1). The algorithm stops when it encounters a Markovian representation. Termination is ensured by the fact that the CF-1 is Markovian:

```
Algorithm FindMarkovian:
Let $(\boldsymbol{\alpha}', \mathbf{A}')$ be the reversed CF-1 of $(\boldsymbol{\alpha}, \mathbf{A})$.
While $\exists i \in \{1, \ldots, n-1\} : \boldsymbol{\alpha}'_i < 0$
    $i := \operatorname{argmin}_i \{\alpha'_i < 0\}$
    $i := \max(2, i)$
    While $\boldsymbol{\alpha}'$ is not Markovian $\wedge \exists k : \boldsymbol{\Lambda}[k] \geq \boldsymbol{\Lambda}[k+1]$
        $k := \operatorname{argmin}_j \{i-1 \leq j \leq n-1 \ : \ \boldsymbol{\Lambda}[j] \geq \boldsymbol{\Lambda}[j+1]\}$
        $(\boldsymbol{\alpha}', \boldsymbol{\Lambda}') :=$ Swap$(\boldsymbol{\alpha}', \boldsymbol{\Lambda}', k)$
    end
end
Return $(\boldsymbol{\alpha}', \boldsymbol{\Lambda}')$
```

Note that `FindMarkovian` is also not guaranteed to find the optimum, since it stops when it finds the first Markovian representation.

6 Illustrative Examples

We will now illustrate our results on several APH distributions.

Example 2. Consider the generalised Erlang distribution with $\boldsymbol{\Lambda} = (1, 2, 3, 4)$ and $\boldsymbol{\alpha} = (1, 0, 0, 0)$. For this distribution, every order of rates in $\boldsymbol{\Lambda}$ has costs $n^* = 4$, since no probability mass can be shifted to the right. As expected, both `BubblesortOptimise` and `FindMarkovian` identify $((1, 0, 0, 0), (4, 3, 2, 1))$ as the optimal representation.

Example 3. Let $\boldsymbol{\Lambda} = (1, 2, 3, 4)$, as before, and the initial probability vector $\boldsymbol{\alpha} = (0.7, 0.15, 0.09, 0.06)$. Then, the average number of visited states is

$$n^*(\boldsymbol{\alpha}, \boldsymbol{\Lambda}) = 3.49.$$

Application of `BubblesortOptimise` results in the reversed CF-1 form with $\boldsymbol{\Lambda}' = (4, 3, 2, 1)$, $\boldsymbol{\alpha}' = (0.46, 0.12, 0.18, 0.24)$ and costs

$$n^*(\boldsymbol{\alpha}', \boldsymbol{\Lambda}') = 2.8.$$

Since the reversed CF-1 is Markovian, `FindMarkovian` gives the same result. We observe that probability mass in the initial probability vector has been shifted towards higher indices.

Example 4. We study $(\boldsymbol{\alpha}, \boldsymbol{\Lambda})$ with $\boldsymbol{\alpha} = (0.5, 0.4, 0.05, 0.05)$ and again $\boldsymbol{\Lambda} = (1, 2, 3, 4)$. This representation has costs

$$n^*(\boldsymbol{\alpha}, \boldsymbol{\Lambda}) = 3.35.$$

The initial vector for the reversed CF-1 is $(-0.6, 1.4, 0, 0.2)$, and hence the reversed CF-1 form is non-Markovian. Applying the `BubblesortOptimise` algorithm to the CF-1 form provides us with a representation $(\boldsymbol{\alpha}', \boldsymbol{\Lambda}')$ with $\boldsymbol{\Lambda}' = (2, 4, 3, 1)$ and $\boldsymbol{\alpha}' = (0.1, 0.7, 0, 0.2)$, for which

$$n^*(\boldsymbol{\alpha}', \boldsymbol{\Lambda}') = 2.7.$$

`FindMarkovian` starts on the non-Markovian reversed CF-1 representation and generates the Markovian representation $\boldsymbol{\Lambda}'' = (2, 3, 4, 1)$ and $\boldsymbol{\alpha}'' = (0.1, 0.7, 0, 0.2)$, which has the same costs of 2.7. A complete enumeration of all permutations shows that both orderings are optimal with respect to n^*.

Example 5. As the last example, we fit an APH(8) to the `loss1-50-opc-1` data-set from [3] using the PhFit tool [4]. This data set contains response-time measurements from a SOA system under high load and with network packet loss. The resulting APH has initial probability vector $\boldsymbol{\alpha} = (0.019, 0.006, 0.069, 0.104, 0.164, 0.371, 0.216, 0.051)$ and rate vector

$$\begin{aligned}\boldsymbol{\Lambda} = (&7.181 \cdot 10^{-05}, 2.4280 \cdot 10^{-04}, 5.854 \cdot 10^{-04}, 5.863 \cdot 10^{-04},\\ &5.956 \cdot 10^{-04}, 5.965 \cdot 10^{-04}, 6.178 \cdot 10^{-04}, 6.332 \cdot 10^{-04}).\end{aligned}$$

For this representation,

$$n^*(\boldsymbol{\alpha}, \boldsymbol{\Lambda}) = 3.38.$$

Again, the reversed CF-1 for this representation has negative entries in the initial vector. Application of `BubblesortOptimise` results in $(\boldsymbol{\alpha}', \boldsymbol{\Lambda}')$ with initial probability vector $\boldsymbol{\alpha}' = (0.0047, 0.0203, 0.0614, 0.0929, 0.1327, 0.3911, 0.2417, 0.0552)$ and

$$\begin{aligned}\boldsymbol{\Lambda}' = (&2.4280 \cdot 10^{-04}, 7.181 \cdot 10^{-05}, 6.332 \cdot 10^{-04}, 6.178 \cdot 10^{-04},\\ &5.965 \cdot 10^{-04}, 5.956 \cdot 10^{-04}, 5.863 \cdot 10^{-04}, 5.854 \cdot 10^{-04}),\end{aligned}$$

which has $n^*(\boldsymbol{\alpha}', \boldsymbol{\Lambda}') = 3.256$. According to a complete enumeration, this is also the Markovian optimum. `FindMarkovian` returns $\boldsymbol{\alpha}'' = (0.0047, 0.0203, 0.069, 0.104, 0.164, 0.371, 0.216, 0.051)$ and

$$\begin{aligned}\boldsymbol{\Lambda}'' = (&2.4280 \cdot 10^{-04}, 7.181 \cdot 10^{-05}, 5.854 \cdot 10^{-04}, 5.863 \cdot 10^{-04},\\ &5.956 \cdot 10^{-04}, 5.965 \cdot 10^{-04}, 6.178 \cdot 10^{-04}, 6.332 \cdot 10^{-04}),\end{aligned}$$

for which $n^*(\boldsymbol{\alpha}'', \boldsymbol{\Lambda}'') = 3.366$.

6.1 Discussion

In general, our examples indicate that there are phase-type distributions for which re-ordering of rates results in a cost reduction. The highest reduction was observed in Example 2 (20%), while for the fitted distribution in Example 5 the reduction was 3.6%. In Monte-Carlo simulations with many simulation runs, these reductions lead to significant time-savings.

On the other hand, we also observe that the effectiveness of the algorithms depends strongly on the initial representation. Representations with (generalised) Erlang structure (Example 2) are invariant to re-ordering of rates. The same holds within blocks of subsequent phases with zero initial probability. For representations where the probability mass is already concentrated at the higher indices in the CF-1, there is also little room for improvement.

We can thus identify (generalised) Erlang structure and large probability mass at the higher indices as two properties of representations that are not susceptible to the proposed optimisation. However, so far we have not been able to find more formal criteria for when and why the optimisation procedures fail. Such criteria would not only help in improving the optimisation algorithms, but may also enable the development of specialised PH-fitting methods that give APH distributions suited for efficient random-number generation.

7 Conclusion and Future Work

In this paper we considered the complexity of generating random numbers from acyclic phase-type distributions. Our focus lay on bi-diagonal representations of APH distributions, whose structural limitations enable the `SimplePlay` procedure which is more effective than the more general `Play`. By re-ordering rates along the diagonal we undertook a first attempt at optimising the bi-diagonal representation for efficient random-number generation. We presented a limited result for the optimal ordering and proposed two algorithms to optimise the representation, given an APH in CF-1 form.

We note that the effectiveness of our approach depends on the given APH. While we can provide a number of intuitive guidelines, formal criteria for deciding when re-ordering rates may offer an advantage are still future work. Furthermore, in the near future we will extend our approach to eliminate the limitations of our result, and we will apply the approach to general phase-type distributions in Monocyclic form [18].

Acknowledgements

This work was supported by DFG grants Wo 898/2-1 and Wo 898/3-1, and OTKA grant no. K-61709.

References

1. Fallahi, A., Hossain, E.: Distributed and Energy-Aware MAC for Differentiated Services Wireless Packet Networks: A General Queuing Analytical Framework. IEEE Transactions on Mobile Computing **6**(4) (2007) 381–394
2. Reinecke, P., Wolter, K.: Phase-Type Approximations for Message Transmission Times in Web Services Reliable Messaging. In Kounev, S., Gorton, I., Sachs, K., eds.: Performance Evaluation – Metrics, Models and Benchmarks. Volume 5119 of Lecture Notes in Computer Science., Springer (June 2008) 191–207
3. Reinecke, P., Wittkowski, S., Wolter, K.: Response-time Measurements Using the Sun Java Adventure Builder. In: QUASOSS '09: Proceedings of the 1st International Workshop on Quality of Service-oriented Software Systems, New York, NY, USA, ACM (2009) 11–18
4. Horváth, A., Telek, M.: PhFit: A General Phase-Type Fitting Tool. In: TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools, London, UK, Springer-Verlag (2002) 82–91
5. Thümmler, A., Buchholz, P., Telek, M.: A Novel Approach for Phase-Type Fitting with the EM Algorithm. IEEE Trans. Dependable Secur. Comput. **3**(3) (2006) 245–258
6. Reinecke, P., Wolter, K., Bodrog, L., Telek, M.: On the Cost of Generating PH-distributed Random Numbers. In Horváth, G., Joshi, K., Heindl, A., eds.: Proceedings of the Ninth International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-9), Eger, Hungary (September 17–18, 2009 2009)
7. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach. Dover Publications, Inc., New York (1981)
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) (December 2002) 47–57
9. O'Cinneide, C.A.: Characterization of Phase-Type Distributions. Stochastic Models **6** (1990) 1–57
10. Cumani, A.: On the Canonical Representation of Homogeneous Markov Processes Modelling Failure-time Distributions. Microelectronics and Reliability **22** (1982) 583–602
11. O'Cinneide, C.A.: Phase-Type Distributions and Invariant Polytopes. Advances in Applied Probability **23**(3) (1991) 515–535
12. He, Q.M., Zhang, H.: Spectral Polynomial Algorithms for Computing Bi-Diagonal Representations for Phase Type Distributions and Matrix-Exponential Distributions. Stochastic Models **22** (2006) 289–317
13. Pulungan, R.: Reduction of Acyclic Phase-Type Representations. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany (2009)
14. Brown, E., Place, J., de Liefvoort, A.V.: Generating Matrix Exponential Random Variates. Simulation **70** (April 1998) 224–230
15. Szekli, R.: Stochastic Ordering and Dependence in Applied Probability. Springer Verlag (1995)
16. Johnson, S.M.: Generation of Permutations by Adjacent Transposition. Mathematics of Computation **17**(83) (July 1963) 282–285
17. Knuth, D.E.: The Art of Computer Programming. Volume 3. Addison-Wesley (1997)

18. Mocanu, S., Commault, C.: Sparse Representations of Phase-type Distributions. Commun. Stat., Stochastic Models **15**(4) (1999) 759 – 778