Efficient Generation of PH-distributed Random Variates

Gábor Horváth[2], 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
{hgabor, telek}@webspn.hit.bme.hu

Abstract. Phase-type (PH) distributions are being used to model a wide range of phenomena in performance and dependability evaluation. The resulting models may be employed in analytical as well as in simulation-driven approaches. Simulations require the efficient generation of random variates from PH distributions. PH distributions have different representations and different associated computational costs for pseudo random number generation (PRNG). In this paper we study the problem of efficient representation and efficient generation of PH distributed variates.
Key words: PH distribution, pseudo random number generation.

1 Introduction

Phase-type (PH) distributions [**?**] are very useful in modelling interarrival times, failure times, and other phenomena in computer systems. They can be employed in analytical approaches as well as in simulation-based evaluations. When PH distributions are used in simulations, often large sets of random variates must be generated, and thus efficiency of random-variate generation from PH distributions is important. We consider algorithms that 'play' the underlying Markov chain. These algorithms provide high accuracy, because they represent each PH sample as a sum of exponential samples, directly following the definition of PH distributions.

PH distributions have different Markovian representations. In [3] we observed that the computational complexity of PH-distributed random-variate generation depends on the representation. This fact poses the research problem of finding the representation that is optimal for random-variate generation.

In [2] we addressed the question by considering the sub-class of Acyclic Phase-type (APH) distributions. For APH distributions the optimal representation is obtained as follows: Starting from any representation the first step is to transform the representation to the CF-1 canonical form defined in [4]. An APH distribution is in CF-1 form if the generator matrix has a bi-diagonal structure and the transition rates are non-decreasing towards the absorbing state. Transformation to the CF-1 form is always possible because all APH distributions have a CF-1 representation [4]. The second step is to find the optimal ordering of the diagonal (and the associated sub-diagonal) elements. It is shown in [2] that for APH in bi-diagonal form the optimal representation is the reversed CF-1 form if it is Markovian. For the case when the reversed CF-1 form is not Markovian, heuristic search algorithms are proposed to find the optimal ordering of the diagonal elements.

In this paper we generalize the results obtained for the APH class to the PH class. We propose to follow a similar approach. In the first step we transform the representation to a sparse Markovian representation. To the best of our knowledge, the only representation with these properties which is available for the whole PH class is the monocyclic representations proposed by Mocanu and Commault [5]. The monocyclic representation is a natural extension of the CF-1 form, in the sense that the generator matrix remains bi-diagonal, but on the matrix block level. Due to their structures these matrix blocks are referred to as feedback Erlang (FE) blocks. The second step of the proposed procedure is to find the optimal ordering of the FE matrix blocks (and the associated sub-diagonal matrix blocks). We are going to show that in contrast to the APH case the optimal ordering of the FE blocks cannot be predicted in a simple way (e.g., by the associated dominant eigenvalue). As a result, finding the optimal representation composed by FE blocks is based on the use of exhaustive or heuristic search algorithms over the set of possible ordering of the FE blocks.

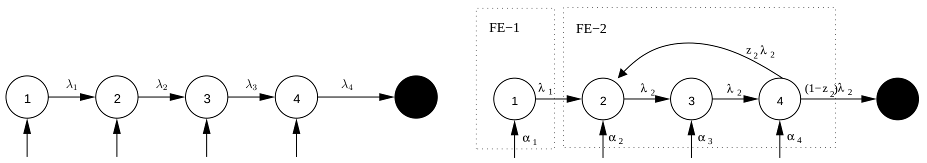
The paper is structured as follows. We first describe the notation used throughout the paper and briefly recall the results from [2] in Section 2. In Section 3 we propose an algorithm for generating random variates from general PH distributions with monocyclic representation and discuss the associated computational cost. Section 4 presents the transformation of reordering the FE blocks in the monocyclic representation and discusses its properties. A counterexample is presented to show that the nice ordering rules of APH representations are not applicable in case of general PH distributions. In Section 5 we provide heuristics for efficient search of optimal representation and Section 6 studies the efficiency of the proposed procedures.

2 Notation and Previous Results

A PH distribution of size n is described by an initial probability vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ and a sub-generator matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ with entries q_{ij} such that $q_{ii} < 0$ and $q_{ij} \geq 0$ for $i \neq j$. In this case the cumulative distribution function (CDF) is

$$F(x) = 1 - \boldsymbol{\alpha} e^{\mathbf{Q}x} \mathbb{1},$$

where $\mathbb{1}$ is the column vector of ones of appropriate size. The representation $(\boldsymbol{\alpha}, \mathbf{Q})$ is not unique.



(a) Bi-diagonal form for APH distributions. (b) FE-diagonal form for general PH distributions.

Fig. 1. Comparison of bi-diagonal and FE-diagonal forms for PH distributions.

Definition 1. *If* $\mathbf{P}$ *is invertible and* $\mathbf{P}\mathbb{1} = \mathbb{1}$*, then the* similarity transformation $(\boldsymbol{\alpha}\mathbf{P}, \mathbf{P}^{-1}\mathbf{Q}\mathbf{P})$ *provides another representation of the same distribution, since its CDF is*

$$1 - \boldsymbol{\alpha}\mathbf{P}\mathrm{e}^{\mathbf{P}^{-1}\mathbf{Q}\mathbf{P}t}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathbf{P}\mathbf{P}^{-1}\mathrm{e}^{\mathbf{Q}t}\mathbf{P}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathrm{e}^{\mathbf{Q}t}\mathbb{1}.$$

The representation $(\boldsymbol{\alpha}, \mathbf{Q})$ is called *Markovian* if $\forall i : \alpha_i \geq 0$ and $\boldsymbol{\alpha}\mathbb{1} = 1$ and $\mathbf{Q}_{ii} < 0$, $\mathbf{Q}_{ij} \geq 0, i \neq j$. With a Markovian representation, we refer to $\boldsymbol{\alpha}$ as the initial probability vector and to $\mathbf{Q}$ as the sub-generator matrix.

The computational complexity of PH-distributed random-variate generation depends on the representation [3]. Our goal is to find the representation with the lowest computational complexity. The optimization of APH representations is based on bi-diagonal representations of APH distributions in [2]. In these representations, the only non-zero entries of the sub-generator matrix $\mathbf{Q}$ are on the diagonal and on the upper diagonal, with $q_{i,i+1} = -q_{ii}$. Bi-diagonal representations can be conveniently specified by the vector $\boldsymbol{\Lambda} = (\lambda_1, \ldots, \lambda_n)$, where $\lambda_i = -q_{ii}$ for $i = 1, \ldots, n$. Figure 1(a) shows the CTMC of an APH distribution with size $n = 4$ in bi-diagonal form. All APH distributions have at least one bi-diagonal representation with Markovian initialisation vector [4]. The bi-diagonal representation with non-decreasing $\boldsymbol{\Lambda}$ is referred to as CF-1 form.

Bi-diagonal Markovian representations are only available if all eigenvalues of the sub-generator matrix $\mathbf{Q}$ (i.e. all poles of the Laplace-Stieltjes Transform of the distribution) are real. As general PH distributions may have complex poles, [5] proposed the use of Feedback-Erlang (FE) blocks to represent pairs of complex eigenvalues:

Definition 2. *[5] A* Feedback-Erlang (FE) block *with parameters* (b, q, z) *is a chain of* b *states with transition rate* q *and one transition from the* b*th state to the first state, with rate* zq*. The probability* $z \in [0, 1)$ *is called the feedback probability.*

The dominant eigenvalue of the FE block with parameters (b, q, z) is always real and given by $r = -q\left(1 - z^{1/b}\right)$ [5]. Feedback-Erlang blocks with length $b = 1$ or feedback probability $z = 0$ are called *degenerate* FE blocks. Note that an FE block (b, q, z) with length $b = 1$ corresponds to an exponential distribution with rate q, while $z = 0$ gives the Erlang-b distribution with rate q (the sum of b independent exponentially distributed random variables with parameter q). In both cases, the dominant eigenvalue is $-q$. Analogously to the approach for the APH class, we consider representations with a diagonal structure of the sub-generator:

Definition 3. *An* FE-diagonal representation *consisting of m Feedback-Erlang blocks $(b_i, q_i, z_i), i = 1, \ldots, m$ has a sub-generator matrix $\mathbf{Q}$ where the only non-zero entries are in the FE blocks along the diagonal and the transition rates from the last state of a Feedback-Erlang block to the first state of the next one. The size of the representation is $n = \sum_{i=1}^{m} b_i$.*

Where appropriate, we also use the vector notation

$$\boldsymbol{\Upsilon} = \{(b_1, q_1, z_1), \ldots, (b_m, q_m, z_m)\},$$

to describe the structure of the sub-generator matrix or we defined $\mathbf{Q}$. Figure 1(b) shows an example of the CTMC of a general PH distribution in FE-diagonal form. In this representation there are two FE blocks, one of length $b_1 = 1$ with rate $q_1 = \lambda_1$, and one of length $b_2 = 3$ with rate $q_2 = \lambda_2$ and feedback probability z_2. The following theorem, restated from [5], ensures that every PH distribution has at least one Markovian FE-diagonal representation:

Theorem 1. *[5] Every PH distribution has a Monocyclic representation with Markovian initial vector. A Monocyclic representation is an FE-diagonal representation*

$$\boldsymbol{\Upsilon} = \{(b_1, q_1, z_1), \ldots, (b_m, q_m, z_m)\}$$

such that the dominant eigenvalues of the FE blocks are ordered by increasing absolute value, $|r_i| \leq |r_j|$, for $1 \leq i \leq j \leq m$.

The Monocyclic form can be computed using, e.g., the implementations available in the Butools library [6]. Note that for APH distributions the feedback probabilities are zero for all blocks, i.e. FE-diagonal forms for APH distributions consist of degenerate FE blocks. For the APH class the FE diagonal form thus corresponds to the bi-diagonal form defined in [2], and the Monocyclic form is equivalent to the CF-1 form. The next section discusses an algorithm for generating random variates from a general PH distribution in FE-diagonal form.

3 Random-Variate Generation from FE-diagonal Representations

Given a PH distribution with representation $(\boldsymbol{\alpha}, \mathbf{Q})$ in FE-diagonal form with FE-block vector $\boldsymbol{\Upsilon} = \{(b_1, q_1, z_1), \ldots, (b_m, q_m, z_m)\}$, random variates can be generated by `FE-diagonal` in Figure 2. This algorithm was proposed as `Procedure Monocyclic` in [3], but can of course also be applied to FE-diagonal representations. The algorithm uses the

$$\mathrm{Geo}(p) = \left\lfloor \frac{\ln U}{\ln p} \right\rfloor \tag{1}$$

```
Procedure FE-diagonal:
Let $x := 0$
Draw an $\boldsymbol{\alpha}$-distributed discrete sample for the initial state.
The chain is in block $i$ and has to traverse $l$ states until the block may be left (e.g.,
for the left-most state of the $i$th block, $l = b_i$).
while $i \leq m$ do
    $c = \mathrm{Geo}(z_i)$
    $x+ = \mathrm{Erl}(cb_i + l, q_i)$
    $i++$
    $l = b_i$
end while
Return($x$)
```

Fig. 2. The FE-diagonal procedure

operation for drawing a random variate from the Geometric distribution with parameter p and support $0, 1, \dots,$ and the

$$\mathrm{Erl}(b, q) = -\frac{1}{q} \ln \left(\prod_{i=1}^{b} U_i \right) \tag{2}$$

operator for drawing a random variate from the Erlang-b with rate q. In both cases, U denotes a uniformly distributed pseudo random number on $(0, 1)$.

The algorithm works as follows: In the initial step, an initial state is chosen according to the initial probability vector $\boldsymbol{\alpha}$. We assume that this state belongs to FE block i, and there are $1 \leq l \leq b_i$ states to traverse before the chain may enter the next block. Since all rates in the given FE block are equal (q_i), this corresponds to an Erlang-l distribution with rate q_i. When the last state of the block is reached, one may either enter the next block or follow the feedback-loop to the first state of the current block. The number of loops $c = 0, 1, \dots$ within the ith block follows a geometric distribution with parameter z_i. The random variate corresponding to the loop is Erlang-c-distributed, again with rate q_i. Consequently, for the block entered upon initialisation the algorithm draws a random variate from an Erlang-$(cb_i + l)$ distribution. All the remaining blocks until absorption are entered at the first state, and thus the respective random variates for the jth block $(j = i+1, \dots, m)$ are distributed according to $(\mathbf{e}_1, \mathbf{F}_j)$ distributions, where $\mathbf{e}_1$ is the row vector with 1 at position 1 and zero everywhere else and $\mathbf{F}_j$ is the sub-generator corresponding to the jth FE block. Following the argument for the initial step, random variates from these distributions are generated from Erlang-$(c_j b_j + l_j)$ distributions, where c_j is the number of loops and $l_j = b_j$ (since each block is entered at the beginning and has to be traversed at least once).

In an initial measurement study we observed that the computational cost of computing logarithm operations dominates the cost of generating PH distributed random numbers. Further performance measures have been considered in [3]. Although the exact cost ratio of the logarithm operations depends on hardware and software specifics, optimisation for the number of logarithm operations appears to be an effective approach. The number of logarithm operations depends on the distribution of the initial probability mass over the Feedback-Erlang blocks and is independent of both the distribution within the blocks and the length of the blocks. Computation of the expected number of logarithms is straightforward:

$$n^*(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}) = 3\boldsymbol{\beta}\boldsymbol{\nu}^{\mathsf{T}},$$

where vector $\boldsymbol{\beta} = \left(\sum_{i=1}^{b_1} \alpha_i, \sum_{i=b_1+1}^{b_1+b_2} \alpha_i, \dots, \sum_{i=n-b_m+1}^{n} \alpha_i\right)$ is the vector of initial probabilities for each FE block, and the entries of $\boldsymbol{\nu} = (m, m-1, \dots, 1)$ give the number of blocks to traverse when entering the ith FE block. The result is multiplied by 3, because 3 logarithm operations are required for the computation of (1) and (2).

Usually, the `FE-diagonal` algorithm is more efficient than the algorithms that simply 'play' the CTMC by selecting a sequence of states, due to the special block bi-diagonal structure of the representation. When a general representation is allowed, the random selection of the next state is required in each step. This is eliminated in the `FE-diagonal` algorithm, since the next FE block is uniquely determined by the chain structure of the representation. Therefore, we consider the optimisation of representations in FE-diagonal form, and we present numerical results about the computational gain of the `FE-diagonal` algorithm in Section 6.

4 Optimisation for FE-diagonal Representations of the PH Class

The optimal representation of an APH for random-variate generation is the reversed CF-1 form (non-increasing $\boldsymbol{\Lambda}$), if the reversed CF-1 is Markovian [2]. This result was obtained by an analysis of the properties of the `Swap` operator, which exchanges two adjacent entries of the vector $\boldsymbol{\Lambda}$. The Monocyclic representation of the PH class is the generalisation of the CF-1 form used for the APH class. The obvious similarity between both forms raises the question whether the result obtained for the APH class can be generalised to the PH class in FE diagonal representation.

To answer this question we introduce the similarity transformation matrix $\mathbf{P}$ that swaps two adjacent FE blocks and produces the new initial probability vector $\boldsymbol{\alpha}' = \boldsymbol{\alpha}\mathbf{P}$.

Definition 4. *The* `GSwap(`$\boldsymbol{\alpha}, \mathbf{A}, i$`)` operator *exchanges the ith FE block with the $(i+1)$th FE block $(1 \le i \le m-1)$ on the diagonal in a block-bi-diagonal representation by swapping the ith and $(i+1)$th entry in the vector $\boldsymbol{\Upsilon}$ (or, equivalently, by swapping the block matrices associated with the FE blocks in the sub-generator). The associated similarity transformation matrix $\mathbf{P}$ has the form*

$$\mathbf{P} = \begin{pmatrix} \mathbf{I}_{\nu\times\nu} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{\mu\times\mu} \end{pmatrix}, \quad \textit{where} \quad \nu = \sum_{k=1}^{i-1} b_k,\ \mu = \sum_{k=i+2}^{m} b_k.$$

ν and μ are the number of states in front of and behind the two blocks, respectively, and $\hat{\mathbf{P}} \in \mathbb{R}^{(b_i+b_{i+1})\times(b_i+b_{i+1})}$ is the solution of

$$\begin{pmatrix} \mathbf{F}_i & -\mathbf{F}_i \mathbb{1}\mathbf{e}_1 \\ \mathbf{0} & \mathbf{F}_{i+1} \end{pmatrix} \hat{\mathbf{P}} = \hat{\mathbf{P}} \begin{pmatrix} \mathbf{F}_{i+1} & -\mathbf{F}_{i+1} \mathbb{1}\mathbf{e}_1 \\ \mathbf{0} & \mathbf{F}_i \end{pmatrix} \tag{3}$$

$$\hat{\mathbf{P}}\mathbb{1} = \mathbb{1}. \tag{4}$$

$\hat{\mathbf{P}}$ is the similarity matrix that swaps the order of the ith and $(i+1)$th FE blocks. Due to the block upper-triangular structure of the coefficient matrices in (3), $\hat{\mathbf{P}}$ is a block lower-triangular matrix. We use the $\texttt{GSwap}(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i)$ notation to denote the swap operation of the ith and $(i+1)$th FE blocks.

We summarise the properties of the `GSwap` operator in the following remarks. While the `GSwap` operator is a generalisation of the `Swap` operator, the fact that the Monocyclic form is block-bi-diagonal with block matrices potentially having size larger than 1 results in one important difference, described in Remark 2.

Remark 1. The structure of $\mathbf{P}$, which describes the `GSwap` operator (i.e. $\mathbf{Q}' = \mathbf{P}^{-1}Q\mathbf{P}$) ensures that `GSwap` has only local effects on the initial probability vector, i.e. it only affects the entries of the initial probability vector that belong to the two FE blocks being swapped.

Remark 2. Given two Feedback-Erlang blocks (b_i, q_i, z_i) and $(b_{i+1}, q_{i+1}, z_{i+1})$ with corresponding sub-generator matrix

$$\mathbf{Q} = \left(\begin{array}{c|ccc|ccc|cc}
\ddots & \ddots & & & & & & & \\
\hline
 & -q_i & q_i & & & & & & \\
 & & \ddots & \ddots & & & & & \\
 & z_i q_i & & -q_i & (1-z_i)q_i & & & & \\
\hline
 & & & & -q_{i+1} & q_{i+1} & & & \\
 & & & & & \ddots & \ddots & & \\
 & & & & z_{i+1}q_{i+1} & & -q_{i+1} & (1-z_{i+1})q_{i+1} & \\
\hline
 & & & & & & & \ddots & \ddots
\end{array}\right),$$

application of the `GSwap` operator exchanges the entire block matrices corresponding to the FE blocks resulting in

$$\mathbf{Q}' = \left(\begin{array}{c|ccc|ccc|cc}
\ddots & \ddots & & & & & & & \\
\hline
 & -q_{i+1} & q_{i+1} & & & & & & \\
 & & \ddots & \ddots & & & & & \\
 & z_{i+1}q_{i+1} & & -q_{i+1} & (1-z_{i+1})q_{i+1} & & & & \\
\hline
 & & & & -q_i & q_i & & & \\
 & & & & & \ddots & \ddots & & \\
 & & & & z_i q_i & & -q_i & (1-z_i)q_i & \\
\hline
 & & & & & & & \ddots & \ddots
\end{array}\right).$$

It means that the `GSwap` operator maintains the proper exit rates from the FE blocks.

Remark 3. The Steinhaus-Johnson-Trotter theorem [7] ensures that all permutations of FE blocks can be obtained by repeated application of the `GSwap` operator.

The `GSwap`$(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i)$ operator only involves the entries of the initial probability vector $\boldsymbol{\alpha}$ that are associated with the ith and $(i+1)$th Feedback-Erlang block. We denote the vector of these entries by $\hat{\boldsymbol{\alpha}}$. Remark 1 implies that $\hat{\boldsymbol{\alpha}}' = \hat{\boldsymbol{\alpha}}\hat{\mathbf{P}}$ and the rest of the initial probability vector remains unchanged by `GSwap`.

According to Lemma 1 of [2], for APH distributions in bi-diagonal form any swap of transition rates that moves a phase with a higher rate away from the absorbing state and maintains a Markovian initialisation vector results in a more efficient representation for simulation. This means that in case of APH distributions the direction of search for the optimal representation is known solely from the properties of the sub-generator matrix and it is independent of the initial probability vector. Unfortunately the corresponding statement does not hold for PH distributions in FE-diagonal form, as we demonstrate in the following counterexample:

Example 1. Let $\mathbf{Q}$ denote the Monocyclic sub-generator matrix defined by $\boldsymbol{\Upsilon} = ((1, 0.1, 0), (b_1, q_1, z_1), (b_2, q_2, z_2))$, where

$$b_1 = 3, q_1 = 1.5, z_1 = 0.5 \text{ and } b_2 = 3, q_2 = 1, z_2 = 0.$$

Exchanging the second and third block results in $\boldsymbol{\Upsilon}' = ((1, 0.1, 0), (b_2, q_2, z_2), (b_1, q_1, z_1))$ and the associated sub-generator matrix $\mathbf{Q}'$. Let

$$\mathbf{P} = \begin{pmatrix} 1 & \\ & \hat{\mathbf{P}} \end{pmatrix} \quad \text{such that} \quad \mathbf{Q}' = \mathbf{P}^{-1}\mathbf{Q}\mathbf{P} \tag{5}$$

denote the similarity transformation that describes the `GSwap` operation for this case, i.e. $\hat{\mathbf{P}}$ is the solution of Equations 3 and 4:

$$\hat{\mathbf{P}} = \begin{pmatrix} 1 & & & & & \\ 0.3333 & 0.6667 & & & & \\ 0.1111 & 0.4444 & 0.4444 & & & \\ -0.925926 & 0.4444 & 0.8889 & 0.592593 & & \\ 0 & -0.925926 & 0.4444 & 0.592593 & 0.8889 & \\ 0 & 0 & -0.925926 & 0.148148 & 0.4444 & 1.3333 \end{pmatrix} \tag{6}$$

Now consider the two initial vectors

$$\boldsymbol{\alpha}_1 = (0.09 \mid 0.1, 0.3, 0.31 \mid 0.1, 0.1, 0),$$

and

$$\boldsymbol{\alpha}_2 = (0.09 \mid 0.1, 0.3, 0.31 \mid 0.2, 0, 0)$$

whose only difference is the distribution of the probability mass assigned to the third FE block. The costs for random-variate generation from $(\boldsymbol{\alpha}_1, \mathbf{Q})$ and $(\boldsymbol{\alpha}_2, \mathbf{Q})$, are

$$n^*(\boldsymbol{\alpha}_1, \mathbf{Q}) = n^*(\boldsymbol{\alpha}_2, \mathbf{Q}) = 3 \cdot (0.09 \cdot 3 + 0.71 \cdot 2 + 0.2) = 5.67.$$

After swapping the two blocks using $\mathbf{P}$ the resulting initial probability vectors are

$$\boldsymbol{\alpha}_1' = \boldsymbol{\alpha}_1 \mathbf{P} = (0.09 \mid 0.141852, 0.28963, 0.271111 \mid 0.118519, 0.0888889, 0)$$

and

$$\boldsymbol{\alpha}_2' = \boldsymbol{\alpha}_2 \mathbf{P} = (0.09 \mid 0.0492593, 0.426667, 0.315556 \mid 0.118519, 0, 0)$$

respectively. Note that in $\boldsymbol{\alpha}_1'$ the initial probability mass assigned to the third FE block increased from 0.2 to 0.207407, while in $\boldsymbol{\alpha}_2'$ the probability mass decreased from 0.2 to 0.118519. The costs of random-variate generation changed as follows:

$$n^*(\boldsymbol{\alpha}_1', \mathbf{Q}') = 3 \cdot 1.8825939 = 5.6477817,$$
$$n^*(\boldsymbol{\alpha}_2', \mathbf{Q}') = 3 \cdot 1.9714836 = 5.9144508.$$

That is, with $\boldsymbol{\alpha}_1$ swapping the blocks resulted in a cost decrease, while with $\boldsymbol{\alpha}_2$ costs increased.

The example illustrates that with true FE-diagonal representations (i.e. those with non-degenerate FE blocks) the effect of swapping two consecutive FE blocks may depend not only on the properties of the sub-generator of the distribution, but also on the distribution in the initial probability vector. Consequently, the procedures proposed for finding the optimal representation of APH distributions in [2] cannot be used for the PH class.

5 Algorithms for Monocyclic Optimisation

Example 1 implies that the reversed Monocyclic form of a true PH distribution is not guaranteed to be optimal, even if the initialisation vector is Markovian. Hence the efficient optimisation methods developed for the APH class cannot be applied to the PH class, since, first, there is no general optimum that only needs to be checked for non-negativity of the initialisation vector, and, second, the direction in which to search for the optimum cannot be derived from the sub-generator alone. On the other hand, Example 1 is not just bad news, since it also shows that a cost reduction by swapping adjacent FE blocks is indeed possible.

The optimum for the FE block form representation of a PH distribution can be found by an optimisation of the costs over the set of all permutations of the FE blocks. This exhaustive approach is guaranteed to find the optimum with respect to all permutations, but involves generating and checking $m!$ representations.

```
Algorithm GBubbleSortOptimise($\boldsymbol{\alpha}, \boldsymbol{\Upsilon}$):
for $i = 1, \dots, m-1$ do
  for $j = 1, \dots, m-1$ do
    $(\boldsymbol{\alpha}', \boldsymbol{\Upsilon}')$ :=GSwap($\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i$)
    if ComparisonHeuristic($\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, j$) $= \text{true} \wedge \boldsymbol{\alpha}' \geq \mathbf{0}$ then
      $(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}) := (\boldsymbol{\alpha}', \boldsymbol{\Upsilon}')$
    else
      break
    end if
  end for
end for
return $(\boldsymbol{\alpha}, \boldsymbol{\Upsilon})$
```

Fig. 3. `GBubbleSortOptimise` attempts to re-order phases such that the global ordering imposed by `ComparisonHeuristic` is generated.

The approach may be feasible if neither the number of FE blocks m nor the block lengths are too large, but running-times become prohibitive for large m or large FE blocks. Large m require a large number of permutations to be checked, while large FE blocks imply large $\hat{\mathbf{P}}$, and thus higher costs for solving (3) and (4).

Therefore, we propose extensions of the algorithms for efficient APH optimisation. In case of FE block representation of PH distribution we need to replace the strict ordering criterion available for the bi-diagonal form of APH distributions with efficient heuristics. The frame of the algorithms `GBubbleSortOptimise` in Figure 3 and `GFindMarkovian` in Figure 4 are identical with the ones for APH representation optimization. In both algorithms, the comparison of two adjacent rates has been replaced by a call to the generic routine `ComparisonHeuristic`, which returns `true` or `false`, depending on whether the two Feedback-Erlang blocks given as its arguments are in the order imposed by the heuristic. The `GBubbleSortOptimise` algorithm is guaranteed to always find a Markovian representation, since it does not leave the region of orderings with Markovian initialisation vectors. The `GFindMarkovian` algorithm, on the other hand, may terminate without finding a Markovian representation. It is guaranteed to terminate with a Markovian representation only with the eigenvalue heuristic discussed below.

In the following we discuss four heuristics that can be used as `Comparison-Heuristic` in either algorithm. The heuristics presented here have been derived based on the following argument: In Lemma 1 of [2] we showed that the optimal ordering for the APH case is obtained if the ordering of the elements of the diagonal of the CF-1 form is reversed (provided that this ordering is Markovian). Due to the simplicity of the CF-1 form, this re-ordering can be seen equivalently as a re-ordering of the dominant eigenvalues, of the means, and of the exit rates. Furthermore, property (3) in [2] corresponds to the determinant of the swap matrix being larger than 1. These four criteria differ from each other if true Feedback-Erlang blocks are compared.

```
Algorithm GFindMarkovian($\boldsymbol{\alpha}, \boldsymbol{\Upsilon}$):
Let $(\boldsymbol{\alpha}', \boldsymbol{\Upsilon}')$ be the reversed Monocyclic form of $(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}')$
r:=0
while $\neg(\boldsymbol{\alpha}' \geq \mathbf{0})$ do
   $i := \mathrm{argmin}_i \{\alpha'_i < 0\}$
   $i := \max\{2, i\}$
   while $\neg(\boldsymbol{\alpha}' \geq \mathbf{0}) \wedge \exists k$ : ComparisonHeuristic($\boldsymbol{\Upsilon}[k], \boldsymbol{\Upsilon}[k+1]$) = false do
      $k := \mathrm{argmin}_j \{j \mid i-1 \leq j \leq m-1 \wedge \boldsymbol{\Upsilon}[j] \geq \boldsymbol{\Upsilon}[j+1]\}$
      $(\boldsymbol{\alpha}', \boldsymbol{\Upsilon}') :=$ GSwap($\boldsymbol{\alpha}', \boldsymbol{\Upsilon}', k$)
      if $(\boldsymbol{\alpha}', \boldsymbol{\Upsilon}')$ is a new representation then
         $r++$
      end if
      if $r = m!$ then
         goto END
      end if
   end while
end while
END:
return $(\boldsymbol{\alpha}', \boldsymbol{\Upsilon}')$
```

Fig. 4. GFindMarkovian starts from the possibly non-Markovian reversed Monocyclic form and searches for a Markovian representation by re-ordering phases such that the reversed ordering imposed by ComparisonHeuristic is produced.

Eigenvalues Heuristic. The eigenvalues heuristic relates to the CF-1 case most directly. Recall that the Monocyclic representation is defined such that the Feedback Erlang blocks are ordered along the diagonal according to increasing absolute value of their dominant eigenvalues. The eigenvalues heuristic directly applies the observation from the CF-1 case that swapping two blocks may move probability mass to the right iff the eigenvalue of the right phase is larger than that of the left phase. Equivalently, in the Monocyclic case probability mass may be moved to the right if the dominant eigenvalue of the right FE block is larger than that of the left FE block. We define the Eigenvalues heuristic as follows:

$$\texttt{EigenvaluesHeuristic}(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i) = \begin{cases} \text{true} & |r_i| < |r_{i+1}|\,, \\ \text{false} & \text{else}, \end{cases}$$

that is, the heuristic returns true if the absolute value of the dominant eigenvalue of the ith block is larger than that of the $(i+1)$th block.

Mean Heuristic. The mean heuristic stems from the following observation for the CF-1 case: Re-ordering phases directly relates to re-ordering the means of the associated distributions. I.e., swapping two phases such that the one with higher rate is moved to the left is equivalent to swapping them such that the one with the lower mean moves to the left. This idea can be applied to the Monocyclic case as follows. First, we have to assign a mean to each FE block. While this is unambiguous in the CF-1 case (where FE blocks are of length 1), in the Monocyclic case probability mass may be assigned to all phases of a block, rendering the mean dependent on the distribution of the mass. The most straightforward approach is then to assign probability mass of 1 to the first entry of the block. The mean of a Feedback-Erlang block (b_i, q_i, z_i) with sub-generator matrix $\mathbf{F}_i$ and probability mass 1 at the first entry is $\hat{M}_i = \mathbf{e}_1(-\mathbf{F}_i)^{-1}\mathbb{1}$. The mean heuristic is then defined as

$$\texttt{MeanHeuristic}(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i) = \begin{cases} \text{true} & \hat{M}_i > \hat{M}_{i+1}, \\ \text{false} & \text{else.} \end{cases}$$

Exit-Rates Heuristic The exit-rates heuristic compares the exit rates $(1 - z_i)q_i, (1 - z_{i+1})q_{i+1}$ of neighbouring FE blocks. Based on the result for the CF-1 case, optimisation then consists in re-ordering blocks such that the highest exit rates (i.e. largest rates q_i in the CF-1 case) move to the left. The heuristic is defined as follows:

$$\texttt{ExitRatesHeuristic}(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i) := \begin{cases} \text{true} & (1 - z_i)q_i < (1 - z_{i+1})q_{i+1}, \\ \text{false} & \text{else.} \end{cases}$$

Determinant Heuristic A heuristic based on the determinant of the transformation matrix $\hat{\mathbf{P}}$ can be obtained by the following reasoning: First, let $b = b_i + b_{i+1}$ be the size of $\hat{\mathbf{P}}$. Now consider the parallelepiped $\mathcal{P}$ spanned by the vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{b-1}, \ldots, \mathbf{e}_b\}$ (e.g. for $b = 3$, $\mathcal{P}$ can be thought of as a cuboid with length 1, width 1, and height 1). The volume of $\mathcal{P}$ is $V = (\mathit{\Pi}_{j=1}^{b} 1) = 1$. Now consider the parallelepiped $\mathcal{P}'$, which results from the application of $\hat{\mathbf{P}}$ to $\mathcal{P}$. $\mathcal{P}'$ is spanned by the vectors $\{\mathbf{e}_1\hat{\mathbf{P}}, \mathbf{e}_2\hat{\mathbf{P}} \ldots \mathbf{e}_{b-1}\hat{\mathbf{P}}, \mathbf{e}_b\hat{\mathbf{P}}\}$. The volume V' of $\mathcal{P}'$ is $V' = |\hat{\mathbf{P}}|$. Consequently, if $V' > V \Leftrightarrow |\hat{\mathbf{P}}| > 1$, holds. Due to the fact that $\hat{\mathbf{P}}$ is a block lower-triangular matrix we assume that $\hat{\mathbf{P}}$ moves the probability to the left if $|\hat{\mathbf{P}}| > 1$, from which the determinant comparison heuristic is defined as

$$\texttt{DeterminantHeuristic}(\boldsymbol{\alpha}, \boldsymbol{\Upsilon}, i) := \begin{cases} \text{true} & |\hat{\mathbf{P}}| > 1, \\ \text{false} & \text{else.} \end{cases}$$

Note that this heuristic simplifies to $|\hat{\mathbf{P}}| = \frac{\lambda_{i+1}}{\lambda_i}$, in case of bi-diagonal representation, in which case the swap of $\lambda_i < \lambda_{i+1}$ improves the representation [2].

5.1 Discussion

While these heuristics are exact for degenerate FE blocks, they may be misleading with non-degenerate FE blocks, as can be illustrated using Example 1. Table 1 shows the relevant properties considered by the eigenvalues, mean, and exit rates heuristics. The determinant of the swap matrix $\hat{\mathbf{P}}$ is $|\hat{\mathbf{P}}| = 0.208$. Observe that the eigenvalues, mean, and exit rates heuristics would recommend to swap the two blocks. As we saw in the counterexample, this is correct for $\boldsymbol{\alpha}_1$, but incorrect for $\boldsymbol{\alpha}_2$. Likewise, the prediction by the determinant heuristic that swapping would not move probability mass to the right is wrong for $\boldsymbol{\alpha}_1$, but correct for $\boldsymbol{\alpha}_2$.

	$\mathbf{F}_1$	$\mathbf{F}_2$
Dominant eigenvalue	$r_2 = -0.3095$	$r_1 = -1$
Mean with mass 1 at first state	$\hat{M}_2 = 4$	$\hat{M}_1 = 3$
Mean with normalised mass ($\boldsymbol{\alpha}_1$)	$M_2 = 4$	$M_1 = 1.7042$
Mean with normalised mass ($\boldsymbol{\alpha}_2$)	$M_2 = 2.5$	$M_1 = 1.7042$
Exit rate	0.75	1

Table 1. Properties of the FE blocks in Example 1.

6 Application of the Algorithms

We have implemented the proposed representation optimization methods in Mathematica. To test their efficiency we also implemented a random general PH representation generator. For a given size n, first it draws uniformly distributed samples for the initial distribution, which are normalized later, then it draws uniformly distributed samples for the off-diagonal elements of the generator matrix and for the transition rates to the absorbing state of the PH distribution. The mean ratio of the off-diagonal elements of the generator matrix and the transition rates to the absorbing state has a significant effect on the cost of the random number generation. This ratio is referred to as *termination rate* below.

Having a random general PH representation we first compute the monocyclic representation of the same distribution and then we optimize the representation using the introduced heuristic approaches and an exhaustive search method by interchanging the FE block of the representation. The exhaustive search method evaluates all permutations of the FE blocks. The computational complexity of the exhaustive search method becomes significant at $n > 6$. The heuristic search algorithms perform a negligible number of `GSwap` operations compared to the exhaustive search method and find suboptimal representations in the majority of the cases. Table 2 indicates the cost of generating PH distributed random number based on the obtained representations. When the *termination rate* equals to 1 and $n = 6$ there is a gain of $\sim$60% due to the transformation to the monocyclic representation. A further $\sim$40% gain comes from heuristic optimisation of the FE blocks. The results of the exhaustive search method indicates that the suboptimal representation of the eigenvalue, the mean and the exit rate heuristics are very close to the global optimum obtained by the exhaustive search method.

The rows of Table 2 demonstrate the effect of the *termination rate*. The higher the transition rate to the absorbing state, the lower the number of state transitions before absorbtion. In case of fast transitions to the absorbing state the simple simulation which ‘plays’ the transitions of the Markov chain until absorption is an efficient simulation methods due to the low number of state transitions. In this case the transformation to the monocyclic representation and the additional representation optimization methods cannot reduce the cost of random number generation. Both, in case of order 6 (Table 2) and order 10 (Table 3) the turning point is around *termination rate* ~ 10, for higher *termination rate* the direct simulation is more efficient and it is the way around for lower *termination rate*.

termination rate	random PH	mono cyclic	eigenvalue	mean	exitrate	determinant	exhaustive search
			heuristic				
0.033	332.008	5.07253	3.06384	3.06234	3.12738	5.07174	3.02422
0.1	102.696	5.03166	3.02877	3.05235	3.11337	5.02784	3.01783
0.33	33.308	4.98996	3.08534	3.09713	3.15197	4.98222	3.01082
1	11.6818	4.24592	2.6476	2.61502	2.77029	4.2132	2.53818
3.3	4.53882	3.38355	2.23615	2.18118	2.25545	3.37198	2.11419
10	2.2624	2.7238	1.9441	1.92582	1.9441	2.72091	1.85605
33	1.50076	2.39407	1.91512	1.91488	1.91512	2.39407	1.83054

Table 2. Average simulation costs (number of logarithms) for order 6 PH distribution based on 100 samples

termination rate	random PH	mono cyclic	eigenvalue	mean	exitrate	determinant
			heuristic			
0.033	561.077	8.64065	8.62078	8.62078	8.62078	8.64065
0.1	186.376	8.18222	8.17221	8.17221	8.17221	8.18222
0.33	58.8492	7.97052	7.9417	7.94135	7.94203	7.97014
1	19.4483	6.52973	6.47238	6.47238	6.47238	6.52973
3.3	7.07673	5.45818	5.38195	5.38257	5.38397	5.45661
10	3.15833	4.98105	4.96477	4.96336	4.96477	4.98105
33	1.77828	3.06237	3.05572	3.05572	3.05572	3.06237

Table 3. Average simulation costs (number of logarithms) for order 10 PH distribution based on 100 samples

It is interesting to see how the proposed transformation reduces the dynamics of the cost. In the evaluated range of *termination rate*, $(0.033, 33)$, the cost of random number generation with direct simulation varies from 1.5 to 332, while the cost of random number generation with optimized representation varies from 1.8 to 3, and a bit larger dynamics reduction appears in Table 3.

Comparing the performance of heuristic optimization methods we obtain that the eigenvalue and mean heuristics perform better than the exit-rate and determinant heuristics. Based on the average performance in Tables 2 and 3, the order of the heuristics is eigenvalue, mean, exit rate and determinant, and there are only a few cases where the mean heuristic performs better than the eigenvalue heuristic. The results of this section are computed by the `GBubbleSortOptimise` procedure, which always terminates with Markovian initial vector.

7 Conclusion

In this paper we considered the optimisation of phase-type distributions for random-variate generation. We propose to use the FE-blocks based representations. We studied optimisation of the costs involved with random-variate generation and showed by a counterexample that the nice ordering property of the APH class does not generalise to the PH class with FE-diagonal form. We developed different heuristic algorithms for optimisation of the PH representation and studied the quality of the heuristics compared to the exhaustive search. The structure of the original PH representation affects the gain of representation optimisation. In a wide range of cases (*termination rate*< 10) the proposed procedure reduces the cost of random-variate generation. Additionally, the proposed procedure significantly reduces the dependence of the cost on the structure of the original PH representation.

8 Acknowledgements

This work was supported by DFG grants Wo 898/3-1 and Wo 898/5-1, and OTKA grant no. K-101150.

References

1. M. Neuts. Probability distributions of phase type. In *Liber Amicorum Prof. Emeritus H. Florin*, pages 173–206. University of Louvain, 1975.
2. Reinecke, P., Telek, M., Wolter, K.: Reducing the costs of generating aph-distributed random numbers. In Müller-Clostermann, B., Echtle, K., Rathgeb, E., eds.: MMB & DFT 2010. Number 5987 in LNCS, Springer-Verlag Berlin Heidelberg (2010) 274–286
3. 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) 16–20
4. Cumani, A.: On the Canonical Representation of Homogeneous Markov Processes Modelling Failure-time Distributions. Microelectronics and Reliability **22** (1982) 583–602
5. Mocanu, S., Commault, C.: Sparse Representations of Phase-type Distributions. Commun. Stat., Stochastic Models **15**(4) (1999) 759 – 778
6. 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. http://webspn.hit.bme.hu/˜butools (October 2011)
7. Johnson, S.M.: Generation of Permutations by Adjacent Transposition. Mathematics of Computation **17**(83) (July 1963) 282–285