Heuristic Representation Optimization for Efficient Generation of PH-distributed Random Variates

Gábor Horváth[3,4], Philipp Reinecke[1], Miklós Telek[3,5], and Katinka Wolter[2]

[1] HP Labs, Bristol
[2] Freie Universität Berlin
Institut für Informatik
[3] Budapest University of Technology and Economics
Department of Telecommunications
[4] MTA-BME Information systems research group
[5] Inter-University Center for Telecommunications and Informatics Debrecen
{philipp.reinecke, katinka.wolter}@fu-berlin.de,
{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-variate generation (PRVG). In this paper we study the problem of efficient representation and efficient generation of PH distributed variates. We introduce various PH representations of different sizes and optimize them according to different cost functions associated with PRVG.
Key words: PH distribution, pseudo random variate generation, monocyclic representation.

1 Introduction

Phase-type (PH) distributions [1] 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. The relations of representations with identical and different sizes are discussed in [2] and in [3],

respectively. In the context of PRVG, the importance of different representations comes from the fact that the computational complexity of PH-distributed random-variate generation depends on the representation [4]. This fact poses the research problem of finding the representation that is optimal for random-variate generation.

In [5] 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 [6]. 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 [6]. The second step is to find the optimal ordering of the diagonal (and the associated sub-diagonal) elements. It is shown in [5] that for APH in bi-diagonal form the optimal representation is achieved by reversing the order of the rates on the diagonals, if the resulting reversed CF-1 form 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 a previous conference version of this paper [7] we generalized the results obtained for the APH class to the PH class using a similar approach. In the first step we transformed the representation to the monocyclic representation proposed by Mocanu and Commault in [8]. The monocyclic representation is a sparse Markovian representation. This 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 structure these matrix blocks are referred to as feedback Erlang (FE) blocks. The second step of the procedure in [7] was to find the optimal ordering of the FE matrix blocks (and the associated sub-diagonal matrix blocks). We showed that due to the complex structure introduced by FE blocks there is no general optimal ordering that holds for all PH distributions and proposed several heuristics for finding optimal orderings.

In this paper we add a new degree of freedom to the considered class of PH representations and optimize over this extended class. The introduced new degree of freedom is the possible transition at the departure from FE blocks to the absorbing state with probability p, referred to as exit probability. At $p = 0$ the obtained PH structure is identical with the one in [7]. We show that by extending the class of representations we can achieve a significant improvement in the efficiency of random-variate generation.

The introduction of nonzero exit probability might result in a non-Markovian representation of the PH distribution, that is, a representation with negative elements in the initial vector, which cannot be used for random-variate generation with play methods. In order to exploit the whole flexibility of the introduced degree of freedom, we check if a further representation transformation can provide a Markovian representation (possibly of larger size).

The method proposed in this paper is composed of two parts, preprocessing and random-variate generation. The computational complexity of the combina-

tion of both steps should be optimized in general. There are obvious extreme solutions for the cases when very few and extremely large numbers of random samples are required. In the first case the preprocessing phase can be omitted and in the second case arbitrarily large look-up tables can be computed during the preprocessing phase. Our proposed solution is between these extremes. We assume $10^6 - 10^{10}$ samples, where the cost of the preprocessing phase of our method is negligible in case of moderate size (< 10 states) PH distributions. The cost of the preprocessing phase increases sub-linearly with the size.

The paper is structured as follows. We first introduce basic concepts of phase-type distributions and the notation used throughout the paper in Section 2. In Section 3 we propose a set of Markovian representations for PH distributions which appear promising for efficient random variate generation. Section 4 presents a procedure for generating PH distributed random variates from monocyclic representations and computes its cost measures. A case study and an exhaustive numerical experiment demonstrate the abilities of the proposed representations (Section 5). Section 6 concludes the paper.

2 Phase-type distributions

A ME distribution of size n is described by an initial row vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ and a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, which define the cumulative distribution function (CDF)

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

where $\mathbb{1}$ is the column vector of ones of appropriate size. We denote this ME distribution by $\mathrm{ME}(\boldsymbol{\alpha}, \mathbf{A})$. The related probability density function (pdf), Laplace transform and moments are

$$f_X(x) = \boldsymbol{\alpha} e^{\mathbf{A}x} (-\mathbf{A}) \mathbb{1}, \tag{2}$$

$$f_X^*(s) = E(e^{-sX}) = \boldsymbol{\alpha} (s\mathbf{I} - \mathbf{A})^{-1} (-\mathbf{A}) \mathbb{1}, \tag{3}$$

$$\mu_n = E(X^n) = n! \boldsymbol{\alpha} (-\mathbf{A})^{-n} \mathbb{1}. \tag{4}$$

Phase-type distributions [1, 9] of size n are a sub-class of matrix-exponential distributions of the same size, where vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ is a probability vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ with entries α_i such that $\alpha_i \geq 0$ and matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a sub-generator matrix with entries a_{ij} such that $a_{ij} \geq 0$ for $i \neq j$ and $a_{ii} \leq -\sum_{j, j \neq i} a_{ij}$. We denote the phase type distribution with vector $\boldsymbol{\alpha}$ and matrix $\mathbf{A}$ by $\mathrm{PH}(\boldsymbol{\alpha}, \mathbf{A})$. In this paper we restrict the attention to the cases with $\boldsymbol{\alpha}\mathbb{1} = 1$, which results in distributions without probability mass at zero.

2.1 Representations

The representation of a phase-type distribution by a vector $\boldsymbol{\alpha}$ and matrix $\mathbf{A}$ (denoted as $(\boldsymbol{\alpha}, \mathbf{A})$) is not unique [6, 10], as summarized in the following theorems.

Theorem 1. *[2] Let ME($\boldsymbol{\alpha}, \mathbf{A}$) of cardinality n and ME($\boldsymbol{\gamma}, \mathbf{G}$) of cardinality n represent two ME distributions with cdf $F_X(x)$ and $F_Y(x)$, respectively. The two distributions are identical if there exists a non-singular matrix $\mathbf{B}$ of cardinality $n \times n$, such that $\boldsymbol{\gamma} = \boldsymbol{\alpha}\mathbf{B}$, $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A}\mathbf{B}$ and $\mathbf{B}\mathbb{1} = \mathbb{1}$. The same relation holds when ME is replaced by PH.*

Proof. If matrix $\mathbf{B}$ is non-singular and $\boldsymbol{\gamma} = \boldsymbol{\alpha}\mathbf{B}$, $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A}\mathbf{B}$, $\mathbf{B}\mathbb{1} = \mathbb{1}$, then

$$F_Y(x) = 1 - \boldsymbol{\gamma} e^{\mathbf{G}x}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathbf{B} e^{\mathbf{B}^{-1}\mathbf{A}\mathbf{B}x}\mathbb{1} = 1 - \boldsymbol{\alpha}\mathbf{B}\mathbf{B}^{-1} e^{\mathbf{A}x}\mathbf{B}\mathbb{1} = 1 - \boldsymbol{\alpha} e^{\mathbf{A}x}\mathbb{1} = F_X(x).$$

We classify the vector matrix pairs which represent a distribution according to (2) by the following important property.

Definition 1. *The vector matrix pair $(\boldsymbol{\gamma}, \mathbf{G})$ with $\boldsymbol{\gamma}\mathbb{1} = 1$ is called* Markovian *if $\forall i : \gamma_i \geq 0$ and $\mathbf{G}_{ii} < 0$, $\mathbf{G}_{ij} \geq 0, \forall i \neq j$. Otherwise it is called* non-Markovian.

The Markovian representation of a PH distribution has a nice stochastic interpretation, which is the time to absorption in a Markov chain with n transient states and one absorbing state. In this case, we refer to $\boldsymbol{\gamma}$ of size n as the initial probability vector and to $\mathbf{G}$ of size $n \times n$ as the sub-generator matrix of the Markov chain. We employ the relation between the representations in Theorem 1 throughout this paper.

Theorem 1 uses the square matrix $\mathbf{B}$ to transform between representations of the same size. This operation is defined as follows:

Definition 2. *The* similarity transformation *of $(\boldsymbol{\alpha}, \mathbf{A})$ with matrix $\mathbf{B}$ is $(\boldsymbol{\alpha}\mathbf{B}, \mathbf{B}^{-1}\mathbf{A}\mathbf{B})$ if $\mathbf{B}$ is invertible and $\mathbf{B}\mathbb{1} = \mathbb{1}$.*

The main properties of a similarity transformation are as follows (cf. [11])

- $(\boldsymbol{\alpha}, \mathbf{A})$ and $(\boldsymbol{\alpha}\mathbf{B}, \mathbf{B}^{-1}\mathbf{A}\mathbf{B})$ have the same size,
- the eigenvalues of $\mathbf{A}$ and $\mathbf{B}^{-1}\mathbf{A}\mathbf{B}$ are identical.
- If $(\boldsymbol{\alpha}, \mathbf{A})$ is Markovian then $(\boldsymbol{\alpha}\mathbf{B}, \mathbf{B}^{-1}\mathbf{A}\mathbf{B})$ can be both Markovian and non-Markovian.

Representations with different sizes can be transformed into each other in a similar manner, using a non-square matrix. This is stated in the following two theorems, which are symmetric to each other.

Theorem 2. *[10, 3] Let ME($\boldsymbol{\alpha}, \mathbf{A}$) of cardinality n and ME($\boldsymbol{\gamma}, \mathbf{G}$) of cardinality m ($m > n$), be two ME distributions with cdf $F_X(x)$ and $F_Y(x)$, respectively. If there exists a matrix $\mathbf{V}$ of cardinality $m \times n$, such that $\boldsymbol{\alpha} = \boldsymbol{\gamma}\mathbf{V}$, $\mathbf{V}\mathbf{A} = \mathbf{G}\mathbf{V}$, $\mathbf{V}\mathbb{1}_n = \mathbb{1}_m$ then ME($\boldsymbol{\alpha}, \mathbf{A}$) $\equiv$ ME($\boldsymbol{\gamma}, \mathbf{G}$).*

Proof. If $\boldsymbol{\alpha} = \boldsymbol{\gamma}\mathbf{V}$, $\mathbf{V}\mathbf{A} = \mathbf{G}\mathbf{V}$, $\mathbf{V}\mathbb{1}_n = \mathbb{1}_m$ then

$$\begin{aligned} F_X(x) &= 1 - \boldsymbol{\alpha} e^{\mathbf{A}x}\mathbb{1}_n = 1 - \boldsymbol{\alpha}\sum_{i=0}^{\infty}\mathbf{A}^i\frac{x^i}{i!}\mathbb{1}_n = 1 - \boldsymbol{\gamma}\mathbf{V}\sum_{i=0}^{\infty}\mathbf{A}^i\frac{x^i}{i!}\mathbb{1}_n = \\ &= 1 - \boldsymbol{\gamma}\sum_{i=0}^{\infty}\mathbf{G}^i\frac{x^i}{i!}\mathbf{V}\mathbb{1}_n = 1 - \boldsymbol{\gamma}\sum_{i=0}^{\infty}\mathbf{G}^i\frac{x^i}{i!}\mathbb{1}_m = 1 - \boldsymbol{\gamma} e^{\mathbf{G}x}\mathbb{1}_m = F_Y(x) \end{aligned} \tag{5}$$

Theorem 3. *[3] Let ME($\boldsymbol{\alpha}, \mathbf{A}$) of cardinality n and ME($\boldsymbol{\gamma}, \mathbf{G}$) of cardinality m ($m > n$), be two ME distributions with cdf $F_X(x)$ and $F_Y(x)$, respectively. If there exists a matrix $\mathbf{W}$ of cardinality $n \times m$, such that $\boldsymbol{\alpha}\mathbf{W} = \boldsymbol{\gamma}$, $\mathbf{AW} = \mathbf{WG}$, $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ then ME($\boldsymbol{\alpha}, \mathbf{A}$) $\equiv$ ME($\boldsymbol{\gamma}, \mathbf{G}$).*

Proof. If $\boldsymbol{\alpha}\mathbf{W} = \boldsymbol{\gamma}$, $\mathbf{AW} = \mathbf{WG}$, $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ then

$$\begin{aligned} F_X(x) = 1 - \boldsymbol{\alpha} e^{\mathbf{A}x}\mathbb{1}_n = 1 - \boldsymbol{\alpha}\sum_{i=0}^{\infty}\mathbf{A}^i\frac{x^i}{i!}\mathbb{1}_n = 1 - \boldsymbol{\alpha}\sum_{i=0}^{\infty}\mathbf{A}^i\frac{x^i}{i!}\mathbf{W}\mathbb{1}_m = \\ = 1 - \boldsymbol{\alpha}\mathbf{W}\sum_{i=0}^{\infty}\mathbf{G}^i\frac{x^i}{i!}\mathbb{1}_m = 1 - \boldsymbol{\gamma}\sum_{i=0}^{\infty}\mathbf{G}^i\frac{x^i}{i!}\mathbb{1}_m = 1 - \boldsymbol{\gamma}e^{\mathbf{G}x}\mathbb{1}_m = F_Y(x) \end{aligned} \tag{6}$$

In both cases of equivalent representations with different sizes we have the following properties.

- The eigenvalues of $\mathbf{A}$ are all eigenvalues of $\mathbf{G}$ with at least the same multiplicity.
- If $(\boldsymbol{\alpha}, \mathbf{A})$ is Markovian then $(\boldsymbol{\gamma}, \mathbf{G})$ can be both Markovian and non-Markovian.

3 Phase-type representations for efficient random variate generation

We recall that the computational complexity of PH-distributed random-variate generation depends on the representation, and that there are examples where the computational cost associated with a representation of larger size is less than the one associated with representations of smaller size [4]. Our goal is to find the Markovian representation with the lowest computational complexity, where we allow the size of the representation to be increased. As the set of representations of a PH distribution is enormously complex, we restrict our attention to a well-defined subset of representations and look for an optimal representation in this subset.

3.1 Monocyclic representation

We define the subset of representations considered here by a procedure. The first element of this procedure is the generation of the monocyclic representation defined in [8]. Here we summarize the main steps of this procedure.

Step 1: Given an $(\boldsymbol{\alpha}, \mathbf{A})$ representation of the PH distribution, compute the eigenvalues of $\mathbf{A}$, denoted by σ_k, with their multiplicities. Choose the dominant eigenvalue (the one with minimal absolute real part) and denote it by σ_1.

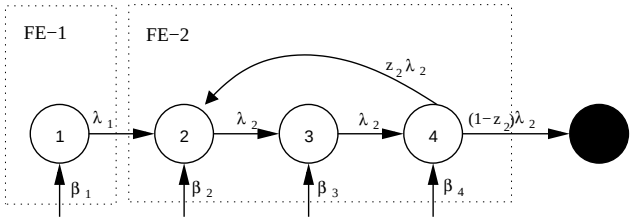

Fig. 1. Monocyclic form of a PH distribution.

Step 2: For each eigenvalue generate a Markovian block representing the eigenvalue. In case of a real eigenvalue the associated Markovian block is composed by a single state whose exit rate is the absolute value of the eigenvalue (c.f. the first block in Figure 1). In case of a pair of complex eigenvalues, $\sigma_k = -a_k \pm c_k \,\mathrm{i}$, the associated Markovian block contains a loop from the last state of the block to the first state (c.f. the second block in Figure 1). The loop is determined by the triple (b_k, λ_k, z_k), where b_k is the number of phases in the loop, λ_k is the common exit-rate of the phases in the loop, and z_k is the feedback probability at the end of the loop. From a_k, c_k and σ_1 (where $a_k, c_k > 0$, σ_1 is real and $\sigma_1 < 0$) the triple (b_k, λ_k, z_k) is computed by

$$b_k = \left\lceil \frac{2\pi}{\pi - 2\arctan\left(\dfrac{c_k}{a_k + \sigma_1}\right)} \right\rceil, \tag{7}$$

$$\lambda_k = \frac{1}{2}\left(2a_k - c_k \tan\frac{\pi}{b_k} + c_k \cot\frac{\pi}{b_k}\right), \tag{8}$$

$$z_k = \left(1 - \frac{a_k - c_k \tan\frac{\pi}{b_k}}{\lambda_k}\right)^{b_k}, \tag{9}$$

where $\lceil x \rceil$ denotes the smallest integer strictly greater than x. These Markovian blocks are referred to as feedback Erlang (FE) blocks in the following. The single phase associated with a real eigenvalue can also be considered as a degenerate FE block, where $(b_k = 1, \lambda_k = -\sigma_k, z_k = 0)$.

Step 3: Compute the dominant eigenvalue of the FE blocks by

$$r_k = -\lambda_k \left(1 - z_k^{\frac{1}{b_k}}\right)$$

and order the FE blocks with non-decreasing absolute r_k. In case of identical r_k values the FE blocks are ordered according to non-decreasing b_k and then non-decreasing λ_k and then nondecreasing z_k.

Step 4: Compose a Markovian transient generator matrix, denoted by $\mathbf{B}$, from the FE blocks such that the FE blocks are repeated as many times as the

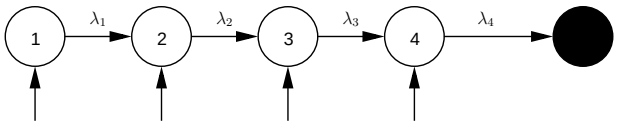

Fig. 2. CF-1 canonical form of Cumani [6].

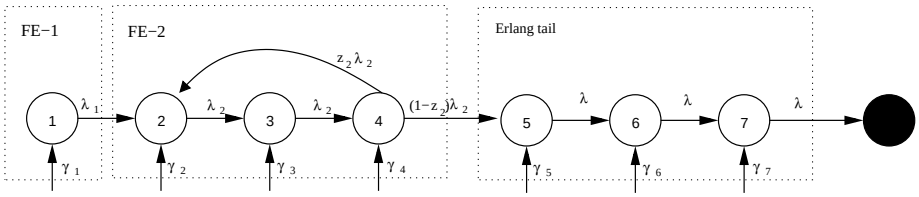

Fig. 3. Monocyclic form of a PH distribution with Erlang tail.

multiplicity of the associated eigenvalue, the exit transition of an FE block goes to the first phase of the next FE block and the FE blocks are ordered. This way the FE block(s) associated with σ_1 is (are) the first one(s).

Note that the case when all eigenvalues are real results in a transient generator matrix which is identical with the CF-1 canonical form of the acyclic PH distributions [6], c.f. Figure 2.

Step 5: Compute the initial vector, $\boldsymbol{\beta}$, such that $(\boldsymbol{\alpha}, \mathbf{A})$ and $(\boldsymbol{\beta}, \mathbf{B})$ are different representations of the same PH distribution. For this computation [8] proposed a method based on the identity of the rational Laplace transforms of $(\boldsymbol{\alpha}, \mathbf{A})$ and $(\boldsymbol{\beta}, \mathbf{B})$. Here we recommend a different "time domain" approach based on [3]: If the size of $\mathbf{B}$ is m then introduce the unknown matrix $\mathbf{W}$ of size $n \times m$, solve the set of linear equations $\mathbf{AW} = \mathbf{WB}$, $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ for $\mathbf{W}$ and compute $\boldsymbol{\beta} = \boldsymbol{\alpha}\mathbf{W}$. Appendix A demonstrates the uniqueness of $\mathbf{W}$ when $\mathbf{A}$ and $\mathbf{B}$ are diagonalizable.

Step 6: Check if $\boldsymbol{\beta}$ is non-negative. If $\boldsymbol{\beta}$ is non-negative then $(\boldsymbol{\beta}, \mathbf{B})$ is the monocyclic representation of the PH distribution. If $\boldsymbol{\beta}$ contains at least one negative element then a further transformation is required to obtain a Markovian representation, which is summarized in Step 7.

Step 7: Extend the monocyclic generator with an Erlang tail according to Figure 3. The obtained transient generator matrix, denoted by $\mathbf{G}$, has the following structure

$$\mathbf{G} = \begin{pmatrix} \mathbf{B} & -\mathbf{B}\mathbb{1} & & & \\ & -\lambda & \lambda & & \\ & & \ddots & \ddots & \\ & & & -\lambda & \lambda \\ & & & & -\lambda \end{pmatrix}.$$

```
Procedure BFMarkovian(β, B):
λ := −σ_1
for i := 1, i < 20, i++ do
   for n := 1, n < 100, n++ do
      (γ, G) := ExtendWithErlangTail(β, B, λ, n)
      if γ ≥ 0 then
         return (λ, n)
      end if
   end for
   λ := 2 ∗ λ
end for
return (0, 0)    % No Markovian representation is found
```

Fig. 4. A brute-force procedure for finding λ and n_λ

This extension has two parameters, the rate of the Erlang tail λ and the number of phases in the Erlang tail n_λ. The initial vector, $\boldsymbol{\gamma}$, which ensures that $(\boldsymbol{\beta}, \mathbf{B})$ and $(\boldsymbol{\gamma}, \mathbf{G})$ are different representations of the same PH distribution is computed according to Step 5. The computation of λ and n_λ such that the associated initial vector $\boldsymbol{\gamma}$ is non-negative is a weakly documented part of [8]. A potential brute-force approach is Procedure `BFMarkovian` in Figure 4. A more sophisticated approach is proposed in [12].
The resulting tuple $(\boldsymbol{\gamma}, \mathbf{G})$ is the monocyclic representation of the PH distribution.

The applicability of this procedure is ensured by the following theorem.

Theorem 4. *[8] Every PH distribution has a monocyclic representation with Markovian initial vector.*

The monocyclic form of a PH distribution can be computed using, e.g., the MOMI tool [13] or the Butools library [14].

Examples in the numerical example section indicate that the monocyclic representation is often more efficient for PRVG than an arbitrary initial representation. In the following subsections we introduce two variants of the monocyclic representation to further enhance the efficiency of PRVG. The structural properties of these representations also follow from [15, 8] due to Theorem 7 in Appendix B.

3.2 Monocyclic representation with non-ordered FE blocks

The first generalization of the considered set of representations is obtained by relaxing the ordering of the FE blocks in the monocyclic representation. The associated representations are obtained by the following procedure.

Step 1-2: Obtain the FE blocks as before.
Step 3: Compute all permutations of the FE blocks where the FE block(s) associated with σ_1 is (are) the first one(s).

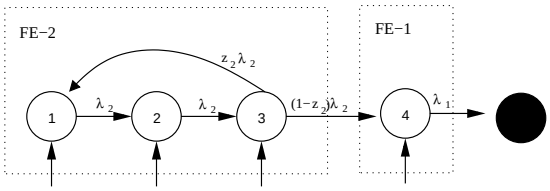

Fig. 5. Monocyclic form of a PH distribution with non-ordered FE blocks.

Step 4-7: Obtain associated representations as before.

Figure 5 illustrates the effect of re-ordering FE blocks in the monocyclic representation.

This procedure has the following properties: The transient generator matrices with different FE block orders have different initial vectors. Consequently, the sizes of the representations are different, in general, due to the fact that Erlang tails of different sizes are required for obtaining the associated non-negative initial vectors. The permutations where the block(s) associated with σ_1 is (are) not the first one(s) violate the structural properties of Theorem 7.

3.3 Monocyclic representation with non-zero exit probability

The second generalization of the considered set of representations is to exit from each FE block with a non-zero probability. In this case, upon exit from an internal FE-block with rate $\lambda_k(1 - z_k)$ the next visited phase is the first phase of the next FE-block with probability $1 - p$ and the absorbing state with probability p. The associated representations are obtained by the following procedure.

Step 1-3: Obtain and order the FE blocks as in the case of the monocyclic representation.

Step 4: Compose a Markovian transient generator matrix, denoted by $\mathbf{B}$, from the FE blocks such that the FE blocks are repeated as many times as the multiplicity of the associated eigenvalue, the exit transition of an FE block goes to the first phase of the next FE block with probability $1 - p$ and to exit with probability p, and the FE blocks are ordered.

Step 5-7: Obtain the Markovian representation as in the case of the monocyclic representation.

Figure 6 shows a representation with non-zero exit probabilities at the internal FE blocks after the computation of the Erlang tail.

This procedure has the following properties: The exit from the last FE block is not affected. With $p = 0$ we obtain the monocyclic representation. The cases when the exit probability is 1 also violate the structural properties in Theorem 7. The sizes of the representations with different p values might be different.

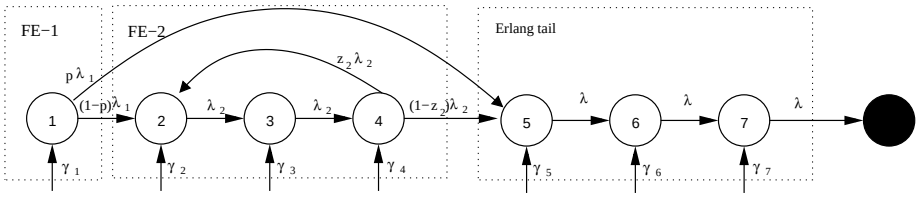

Fig. 6. Monocyclic form of a PH distribution with non-zero exit probability.

3.4 Monocyclic representation with non-ordered FE blocks and non-zero exit probability

The combination of the above two generalizations results in the widest class of representation we consider for efficient PRVG in this paper.

All of the introduced PH representations have a special sparse structure, and consequently the cost of drawing PH-distributed random samples using these representations can be computed in a simple way, as discussed in the next section.

4 Random-variate generation from FE-diagonal representations

In the following we consider the monocyclic representation of a PH distribution with m Feedback-Erlang blocks for the eigenvalues and an Erlang tail of length n_λ and rate λ. Since the Erlang tail is a degenerate FE block, we have a monocyclic representation with $m+1$ FE blocks and overall length $n = \sum_{i=1}^{m+1} b_i$, where the $(m+1)$th block is given by the Erlang tail (if no Erlang tail exists, we assume that the $(m+1)$th block is of length $b_{i+1} = 0$). Given this representation, random variates can be generated by the procedure `Monocyclic`, shown in Figure 7. The algorithm uses the

$$\mathrm{Geo}(q) = \left\lfloor \frac{\ln U}{\ln p} \right\rfloor \tag{10}$$

operation for drawing a random variate from the Geometric distribution with parameter q and support $0, 1, \ldots$; the

$$\mathrm{Erl}(b, \lambda) = -\frac{1}{\lambda} \ln \left(\prod_{i=1}^{b} U_i \right) \tag{11}$$

operator for drawing a random variate from the Erlang-(b, λ) distribution; and the

$$\mathrm{Discrete}(\{p_0, p_1, p_2, \ldots, p_n\}) = \Big\{ i \quad \text{if } \textstyle\sum_{j=0}^{i-1} p_j \le U < \sum_{j=0}^{i} p_j, \tag{12}$$

operator to draw a discrete random variate with distribution $p_0, p_1, p_2, \ldots, p_n$. In all three cases, U denotes a uniformly distributed pseudo-random number on $(0, 1)$.

```
Procedure Monocyclic(γ, [{b_i, λ_i, z_i}, ..., {b_{m+1}, λ_{m+1}, z_{m+1}}]):
x := 0
{j, l} := Discrete(γ)
% The chain starts from block j and inside the block
% it has to traverse l states until the last state of the block
% (e.g., for the left-most state of the jth block, l = b_j).
while j ≤ m do
  if z_j > 0 then
    c := Geo(z_j)
  else
    c := 0
  end if
  x := x + Erl(cb_j + l, λ_j)
  if j < m then
    if p > 0 and Discrete({1 − p, p}) = 1 then
      j := m + 1
    else
      j++
    end if
  else
    j++
  end if
  l := b_j
end while
x := x + Erl(l, λ_j)
Return(x)
```

Fig. 7. The `Monocyclic` procedure

The algorithm works as follows: First, an initial state is chosen according to the initial probability vector $\boldsymbol{\gamma}$. We assume that this state belongs to Feedback-Erlang block $j \in \{1, \ldots, m+1\}$, and there are $1 \leq l \leq b_j$ states to traverse before the chain may enter the next block. Since all rates in the given FE block are equal (λ_j), this corresponds to an Erlang-(l, λ_j) distribution. When the last state of the block is reached, one may either leave the current block or stay in the block by following the feedback-loop to the first state of the block. Note that the random variate corresponding to one loop is Erlang-(b_j, λ_i) distributed and that the number of loops $C_j = 0, 1, \ldots$ within the jth FE block follows a geometric distribution with parameter z_j and support $0, 1, \ldots$. Consequently, for the block entered upon initialization the algorithm draws a random variate from an Erlang-$(C_j b_j + l, \lambda_j)$ distribution. Since all the remaining blocks are entered at the first state, the respective random variates for blocks $j+1, j+2, \ldots, m+1$ are Erlang-$((1 + C_j)b_j, \lambda_j)$-distributed.

Non-zero exit probabilities, as introduced in Section 3.3, extend this behaviour as follows: When leaving any block $j = 1, \ldots, m$, one can go to the start of the Erlang tail (with probability p), or to the next block (with probability

$1-p$). When the chain enters the Erlang tail from the jth block, the remaining blocks $j+1,\ldots,m$ are skipped.

The computational cost of `Monocyclic` is dominated by the number of logarithm operations and the number of uniformly distributed pseudo random numbers that are required. Due to the regular structure of the monocyclic representation the expected numbers of both operations can be computed from the representation in a simple way.

4.1 Logarithm Costs

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.

Let the row vector

$$\boldsymbol{\beta} := \left(\sum_{i=1}^{b_1} \gamma_i, \sum_{i=b_1+1}^{b_1+b_2} \gamma_i, \ldots, \sum_{i=n-b_{m+1}}^{n} \gamma_i \right) \tag{13}$$

of size $m+1$ denote the initial probabilities for each FE block (if $n_\lambda = 0$, $\beta_{m+1} := 0$). Then, define the row vector $\boldsymbol{\nu} \in \mathbb{R}^{m+1}$ with entries

$$\nu_{m+1} := \begin{cases} 1 & n_\lambda > 0 \\ 0 & \text{else} \end{cases} \tag{14}$$

$$\nu_m := \nu_{m+1} + 1 + \mathcal{I}_m \tag{15}$$

$$\nu_j := \nu_{j+1} + 1 + \mathcal{I}_j - p(\nu_{j+1} - \nu_{m+1}) \text{ for } j = m-1,\ldots,1, \tag{16}$$

where $\mathcal{I}_j$ is 0 if $z_j = 0$ and 1 otherwise. (15) represents the special case when $j = m$. The jth entry of $\boldsymbol{\nu}$ gives the number of logarithms when the chain is entered at the jth block: Each traversed block requires one logarithm for drawing the Erlang sample. If the feedback probability z_j is larger than zero, an additional logarithm is needed for drawing a Geometric sample (the logarithm of q in (10) can be precomputed and therefore does not appear in the equation). The last term of (16) accounts for the fact that with probability p the remaining blocks may be skipped. The expected number of logarithm operations is then:

$$n_{Log}(\boldsymbol{\gamma}, \mathbf{G}) = \boldsymbol{\beta}\boldsymbol{\nu}^{\mathsf{T}}. \tag{17}$$

4.2 Uniform Costs

The expected number of uniformly-distributed random variates depends on the number of choices between jumping to the Erlang tail or proceeding, on the number of geometric samples, and on the number of uniforms required for the Erlang samples. Additionally, one uniform is required for selecting the initial state. These considerations yield the following expression:

$$n_{Uni}(\boldsymbol{\gamma}, \mathbf{G}) = n_{Uni}^{(1)}(\boldsymbol{\gamma}, \mathbf{G}) + n_{Uni}^{(2)}(\boldsymbol{\gamma}, \mathbf{G}) + n_{Uni}^{(3)}(\boldsymbol{\gamma}, \mathbf{G}) + 1. \tag{18}$$

We first compute the number of samples required for the jump choices, which depends on the number of visited blocks. Let $\boldsymbol{\omega} \in \mathbb{R}^{m+1}$ denote the row vector with entries

$$\omega_{m+1} = 0 \tag{19}$$
$$\omega_m = 0 \tag{20}$$
$$\omega_j = 1 + (1-p)\omega_{j+1} \text{ for } j = 1, \ldots, m-1. \tag{21}$$

Each entry of $\boldsymbol{\omega}$ represents the expected number of choices that can be made when starting from this block. For $j \in \{m, m+1\}$, the procedure cannot skip towards the Erlang tail. For each block $1 \leq j < m-1$, there is one local choice, and with probability $(1-p)$ the $(j+1)$th block is entered, which then entails further choices. Since each discrete random variate requires one uniform random number, the average number of uniforms in this part is

$$n_{Uni}^{(1)}(\boldsymbol{\gamma}, \mathbf{G}) = \boldsymbol{\beta}\boldsymbol{\omega}^{\mathsf{T}}. \tag{22}$$

Similarly, the mean number of uniforms needed for the geometric samples is obtained by defining the vector $\boldsymbol{\phi}$, whose entries

$$\phi_{m+1} = 0 \tag{23}$$
$$\phi_j = \mathcal{I}_j + (1-p)\phi_{j+1} \text{ for } j = 1, \ldots, m \tag{24}$$

denote the mean number of Geo() operations starting from the ith FE block, and computing

$$n_{Uni}^{(2)}(\boldsymbol{\gamma}, \mathbf{G}) = \boldsymbol{\beta}\boldsymbol{\phi}^{\mathsf{T}}. \tag{25}$$

The number of uniforms needed for the Erlang samples depends on the initial block and on the initial state. For brevity, let $s_j := \sum_{k=1}^{b_j} b_k$ denote the last state of the jth block. As before, we recursively define a vector $\boldsymbol{\psi} \in \mathbb{R}^n$, starting with the entries corresponding to the Erlang tail (if present):

$$\psi_{s_{m+1}} := 1 \tag{26}$$
$$\psi_i := \psi_{i+1} + 1 \text{ for } i = s_{m+1} - 1, \ldots, s_{m+1} - n_\lambda + 1 \tag{27}$$

When starting in states belonging to the mth block, the length of the Erlang distribution (and hence the number of uniform samples) depends on the distance to the feedback point and on the number of feedback loops. Furthermore, when the chain is entered at this block, the Erlang tail has to be traversed completely. Taking into account the length of the Erlang tail, the respective entries in $\boldsymbol{\psi}$ are therefore given by

$$\psi_{s_m} := 1 + n_\lambda + \frac{z_m}{1-z_m} b_m \tag{28}$$
$$\psi_i := \psi_{i+1} + 1 \text{ for } i = s_m - 1, \ldots, s_m - b_m + 1 \tag{29}$$

All the remaining blocks $j = 1, \ldots, m-1$ can jump to the Erlang tail with probability p, and continue with probability $(1-p)$. The corresponding entries

are thus defined as

$$\psi_{s_j} := 1 + pn_\lambda + (1-p)\psi_{s_j+1} + \frac{z_j}{1-z_j} b_j \tag{30}$$

$$\psi_i := \psi_{i+1} + 1 \text{ for } i = s_j - 1, \ldots, s_j - b_j + 1. \tag{31}$$

The number of uniforms for Erlang samples is then

$$n^{(3)}_{Uni}(\boldsymbol{\gamma}, \mathbf{G}) = \boldsymbol{\gamma}\boldsymbol{\psi}^{\mathsf{T}}. \tag{32}$$

4.3 Modification for Exit-Probability Zero

In the form shown in Figure 7, `Monocyclic` supports representations with exit probability $p \geq 0$. If it is known that the exit probability will be zero, a simplified version that does not draw a discrete random variate for the exit arc can be used. With this version, the expression for the expected number of uniforms in (18) simplifies to $n_{Uni} = n^{(2)}_{Uni} + n^{(3)}_{Uni} + 1$. Equation 17 for the average number of logarithms is not affected.

5 Numerical examples

In this section we demonstrate the efficiency improvements obtainable by using optimized representations, compared to the direct approach. The following methods are considered in the comparison:

- The *Play method* is based on the direct simulation of the absorbing Markov chain representing the PH random variable. At each step the sojourn time of the state and the next state are drawn till the absorption. The average speed of the algorithm depends on the mean number of phase transitions occurring before absorption.
- The *SmartPlay method* (proposed in [16]) is the enhanced version of the play method. When drawing a random sample, it first records which states are traversed before absorption and how many times. The time spent in a given state is then considered to be Erlang distributed, which requires only a single logarithm operation to draw a sample from. Thus, the number of logarithm operations needed to generate a PH distributed random sample is less than or equal to the size of the PH distribution.
- The *monocyclic representation* can be beneficial for random number generation as shown in [4].
- Additional improvement can be achieved by *optimizing the order of feedback Erlang blocks* of the monocyclic representation (Section 3.2). At some ordering of the blocks the initial probability vector can be such that on average fewer blocks are traversed till absorption than in the original monocyclic representation [7].
- The simulation cost can also be reduced by *optimizing the exit rates of the feedback Erlang blocks* to reduce the average number of blocks to traverse till absorption (Section 3.3).

– Finally, the combination of the latter two approaches by *optimizing the order of feedback Erlang blocks* and *optimizing the exit rates of the feedback Erlang blocks* at the same time, is considered.

We use a Mathematica implementation of the above methods on a modern PC clocked at 3.4 GHz for our evaluation. Although the implementations might not be fully optimized in terms of execution speed, it becomes obvious that the computational complexity of the methods above is significantly different. Of course, optimizing the representation of the PH distribution is done only once, when the simulation is started, thus the additional overhead of a possibly slow optimization amortizes with the simulation time.

Our findings on the execution speeds of the optimization methods are as follows:

– Drawing PH variables using the play method and its enhanced variant does not need any preliminary preparations at all.
– The transformation to the monocyclic representation is fast. It terminates almost promptly with PHs having tens of phases, and takes a few seconds with a hundred or more phases.
– To find the optimal order of feedback Erlang blocks in terms of the cost function we implemented an exhaustive search algorithm that calculates the costs of all possible orderings of the feedback Erlang blocks and selects the cheapest one. Obviously, this is not a scalable approach due to the combinatorial explosion (i.e., for m FE blocks, $m!$ orderings must be considered). Optimizing the order of 6 feedback Erlang blocks takes 10-20 seconds, and it increases with the size of the problem so fast that it becomes intolerably slow with more than 10 phases. Note that we cannot apply the heuristic optimization algorithms proposed in [7] because we optimize the order on the generally non-Markovian monocyclic representation and compute the Erlang tail afterwards.
– To find the optimal exit probability only a single variable needs to be optimized (between 0 and 1), thus this is a quick and scalable method which takes few seconds with tens of phases.
– The combined optimization is the slowest method. All possible orderings of the feedback Erlang blocks are evaluated one by one and the optimal exit probability is obtained for each ordering. While random-variate generation with this method intuitively can be expected to perform best, it is also obvious that the optimization step is the slowest of all the methods considered here. It needs several minutes to optimize a structure involving as few as 6 feedback Erlang blocks.

During the simulation, the execution-time of generating a single random variate using any of the presented methods is dominated by the elementary operations of drawing a uniformly-distributed pseudo-random number and of computing a logarithm. The exact cost of these operations depends on hardware and software specifics. In the numerical examples the presented methods will be

evaluated using a combined cost function that depends on the ratio L of the costs of these operations:

$$n_{Total} = n_{Uni} + L \cdot n_{Log}. \tag{33}$$

5.1 A Case Study

In this section we provide a detailed case study to show how the presented methods are able to reduce the cost of random variate generation. We use the PH distribution $(\boldsymbol{\alpha}, \mathbf{A})$ with the following parameters:

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.40535 & 0.0914074 & 0.403507 & 0.099735 \end{bmatrix},$$
$$\mathbf{A} = \begin{bmatrix} -8.69773 & 1.1017 & 2.227 & 4.6353 \\ 2.10619 & -10.1553 & 4.61562 & 1.50111 \\ 1.46425 & 4.261 & -12.0265 & 4.40511 \\ 3.85212 & 5.72213 & 0.704262 & -14.1558 \end{bmatrix}.$$

In this PH distribution the average number of state transitions up to absorption is

$$n^* = \boldsymbol{\alpha}(\mathrm{diag}\langle 1/a_{ii}\rangle \mathbf{A})^{-1}\mathbb{1} = 5.54692.$$

According to the *Play method* at each transition an exponentially distributed sojourn time spent in the current phase is generated, followed by drawing the next phase to visit. Generating exponentially distributed random variates involves a logarithm operation and generating a uniform sample. To obtain the next phase (according to the appropriate discrete distribution), an uniform sample is needed. Furthermore, another uniform sample is required to obtain the initial phase of the distribution. Thus, assuming $L = 1$ we have

$$\begin{aligned} n_{Uni} &= 1 + 2n^* = 12.0938, \\ n_{Log} &= n^* = 5.54692, \\ n_{Total} &= n_{Uni} + L \cdot n_{Log} = 17.6408. \end{aligned}$$

According to the *SmartPlay method* at each transition only the next phase is drawn and the number of visits to the current phase is incremented by one. Thus, assuming $L = 1$ we have

$$\begin{aligned} n_{Uni} &= 1 + 2n^* = 12.0938, \\ n_{Log} &= \tilde{n} = 2.85703, \\ n_{Total} &= n_{Uni} + L \cdot n_{Log} = 14.9508, \end{aligned}$$

where $\tilde{n}$ is the average number of visited phases out of the all phases n.

Next, the monocyclic representation of PH(α, A) is constructed (see Figure 8). The number of logarithms required depends on how many states are traversed till the absorption. Note that drawing Erlang distributed random variates needs

only a single logarithm operation, and the geometrically distributed variable representing the number of times the feedback is taken needs an additional logarithm as well. The number of logarithms required is calculated as

$$n_{Log} = 0.527017 \cdot 3 + 0.13063 \cdot 2 + (0.102815 + 0.129809 + 0.109729) \cdot 1 + 1 = 3.18466.$$

Now we calculate the number of uniform samples required. Since we know that $p = 0$, we assume that the simplified version of the procedure `Monocyclic` is used (cf. Section 4.3). One uniform is required for exponential distributions and n uniforms are needed for order-n Erlang distributions. The average number of times the feedback is taken is the mean of a geometric distribution with parameter $0.115691/14.9325 = 0.00774763$, thus it is 0.00781. An additional uniform sample is needed to determine the initial phase of the PH distribution as well. This yields

$$n_{Uni} = 0.527017 \cdot 5 + 0.13063 \cdot 4 + 0.102815 \cdot 3 + 0.129809 \cdot 2 + 0.109729 \cdot 1 + 0.00781 \cdot 3 + 1 = 5.85882.$$

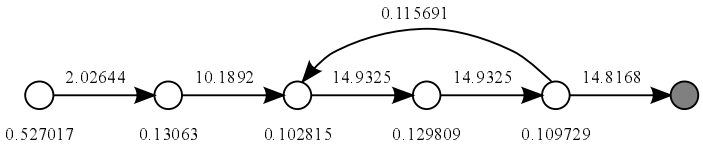

Fig. 8. Monocyclic representation of PH(α, A).

Thus the total cost of the monocyclic representation is $n_{Total} = 9.04349$, which is only half the cost of the play method.

In the hope of reducing the cost even further, we can try to change the order of feedback Erlang blocks. As there are 3 feedback Erlang blocks in this example, there are 6 different orderings to check. Figure 9 depicts the ordering that has been found to be optimal. Note that an Erlang-3 distribution has been appended to maintain the Markovian property of the initial distribution. The cost of this structure is $n_{Total} = 6.06503$. At first sight it seems to be contradictory that the size of the PH is increased to reduce the cost of generating random variates from it, but it can be observed in Figure 9 that the initial probabilities closer to the absorbing state increased, reducing the average cost of the representation.

An alternative way of reducing the cost is to allow absorption when leaving a feedback Erlang block with a given probability. The exit probability that provides the best cost needs to be obtained by optimization. In this particular example the optimal exit probability is $p_e = 0.63904$, yielding a cost of $n_{Total} = 5.92069$ (for $L = 1$). The resulting representation is shown in Figure 10.

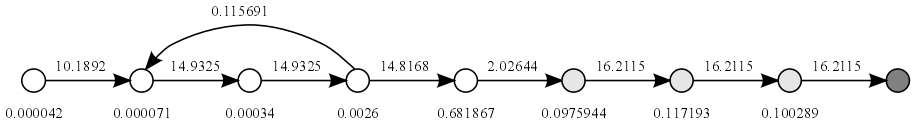

Fig. 9. Representation of PH(α, A) with optimized order of feedback Erlang blocks.

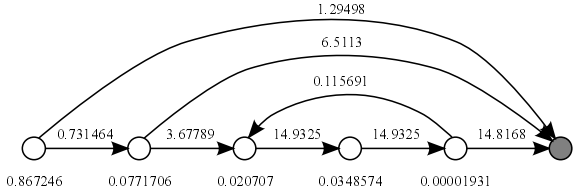

Fig. 10. Representation of PH(α, A) with optimal exit probability $p_k = 0.63904$.

Optimizing both the order of feedback Erlang blocks and the exit probabilities may give even better results in general, but in this particular example it did not improve the results further. Thus we conclude that with the representation of Figure 10 we are able to generate PH(α, A) distributed random variates almost 4 times faster than the play method.

5.2 Evaluating the algorithms on random PH distributions

In this experiment we generate a large number of PH distributions and subject them to optimization, in order to compare the performance of the presented methods. To this end we implemented a random general PH representation generator. For a given size n, it first draws uniformly distributed samples for the initial distribution, which are normalized later. In order to ensure a kind of diversity the elements of the transition matrix $\mathbf{A}$ are drawn as follows. For $i, j \in \{1, \ldots, n\}, i \neq j$ we set $A_{ij} = (i + j)\mathrm{U}$ and for $i \in \{1, \ldots, n\}$ we set $A_{ii} = -\sum_{j,j\neq i} A_{i,j} - ir\mathrm{U}$, where U is a uniformly distributed pseudo random number on $(0, 1)$ and r is referred to as *termination rate*.

We also investigate the effect of the cost ratio L of the elementary operations. The cost ratio can differ considerably, depending on the random-number generator, the implementation of the library routines, and the hardware. For instance, from the data presented in Chapter 2 of [17] it follows that the L factor can vary between 1.65 and 4.82 across machines with different hardware and software versions. The study in [17] considered the simple congruential random-number generator from [18] that was also employed in earlier versions of the OMNeT++ discrete-event simulator. On the other hand, the Mersenne twister random-number generator (used in current OMNeT++ versions) takes almost the same amount of time per sample as the computation of a logarithm on our PC, indicating an L factor of 1. We therefore consider two situations, $L = 5$ and $L = 1$ to study the effect of different configurations.

termination rate	play method	smart play	mono cyclic	optimized		
				block order	exit prob.	both
0.033	862.228	581.053	14.8945	8.98875	5.62649	4.54437
0.1	274.3	188.889	14.6538	8.05776	5.70746	4.41429
0.33	93.2193	67.6334	13.8572	7.25978	5.55492	4.46992
1	32.8511	26.4622	12.1112	6.32547	5.24005	4.26856
3.3	13.6558	12.3872	9.06915	5.29952	4.83196	3.66658
10	7.62302	7.39187	6.95718	4.77055	4.20163	3.56268
33	5.2984	5.27564	5.76659	4.57731	3.96352	3.64796

Table 1. Median of the average simulation costs for order 6 PH distributions based on 200 samples, with $L = 1$

termination rate	play method	smart play	mono cyclic	optimized		
				block order	exit prob.	both
0.033	2010.53	604.659	38.6973	21.7033	12.1318	9.44551
0.1	638.701	211.645	38.0652	20.5111	12.0221	9.27802
0.33	216.178	88.2694	35.9704	18.7376	11.8333	9.38989
1	75.3192	43.2917	32.0196	14.1857	11.2056	8.92384
3.3	30.5302	24.1673	24.4524	12.3403	9.85334	8.04083
10	16.4537	15.2964	18.4818	9.49973	8.57759	7.7933
33	11.0296	10.9113	15.2998	9.41429	7.99007	7.68007

Table 2. Median of the average simulation costs for order 6 PH distributions based on 200 samples, with $L = 5$

For each random general PH representation we first compute the monocyclic representation and then apply the optimization approaches proposed in this paper to find an optimal ordering, an optimal exit probability, and an optimal combination of these.

Table 1 and Table 2 show the cost of generating PH-distributed random variates based on the presented methods with $L = 1$ and $L = 5$, respectively. When the *termination rate* is equal to 1 and $n = 6$ there is a gain of $\sim$65% due to the transformation to the monocyclic representation. By optimizing the monocyclic representation further gain can be obtained. Optimizing only the order of blocks yields an additional $\sim$40%, while optimizing only the exit probability gives an additional $\sim$55% gain. When both the order and the exit probabilities are subject to optimization, the cost given by the monocyclic representation is reduced by $\sim$60%.

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 variate generation with direct simulation varies from 5.27 to 862.23, while the cost of random variate generation with optimized representation varies from 3.56 to 8.99 (for $L = 1$). This pattern also applies to the cost with cost ratio $L = 5$.

5.3 Studying the effect of the shape of the density functions

In Section 5.2 we use randomly generated PH distributions for our evaluation, but do not consider the possible impact of the distributions' properties on the results. In this section we extend the focus of our experiments by studying the impact of the shape of the density function on the effectiveness of our optimization approach. As phase-type distributions are often employed specifically because of their ability to fit arbitrary densities, the question whether our approach can improve the efficiency of random-variate generation independently of the shape of the density is of considerable practical interest.

We proceed as follows: We use a set of randomly-generated PH representations, split into classes by basic properties of their density functions, and perform the comparisons for each class. We split the set of distributions as follows:

- Based on the squared coefficient of variation c_v^2, where we distinguish
  - low c_v^2 distributions: $0 < c_v^2 \leq 0.8$,
  - medium c_v^2 distributions: $0.8 < c_v^2 \leq 2$, and
  - high c_v^2 distributions: $c_v^2 \geq 2$.
- Based on the skewness γ, where we consider
  - low skewness distributions: $0 <$ skewness ≤ 1.7,
  - medium c_v^2 distributions: $1.7 <$ skewness ≤ 3, and
  - high c_v^2 distributions: skewness ≥ 3.
- Based on the shape of the density function, where we have
  - unimodal distributions with mode $= 0$,
  - unimodal distributions with mode > 0,
  - bimodal distributions.

We ensured that there were at least 100 distributions in each class, leading to a total of 491 distributions in the evaluation (some distributions can be in more than one class). The number of random PH distributions belonging to the different classes and their properties are summarized in Table 3.

In order to ensure sufficient diversity of the random PH representations we employ sparse representations, which we can generate more efficiently than non-sparse representations. The representations we use are of size 6, each with a total of 29 zero entries in the vector $\boldsymbol{\alpha}$, matrix $\mathbf{A}$ and closing vector $(-\mathbf{A})\mathbb{1}$.

Table 4 shows the median of the simulation costs corresponding to the classes and methods used in the comparison. According to the results it is clear that the shape of the distribution has only a minor effect on the simulation costs; furthermore, this effect is far less than the effect of the the rate to absorption (see Section 5.2). The methods proposed by us are always better than the play method and its smart variant. Our representation-optimization procedures (monocyclic representation, block re-ordering, exit-rate optimization) are thus effective indeed.

Finally, we want to point out another interesting feature of using special optimized representations in random-variate generation: As illustrated in Table 5, our methods result in lower deviations from the average costs. The table shows the median absolute deviation, computed as the median of the absolute distance

Class	Num. of PHs	Properties
low c_v^2:	122	$\min\{c_v^2\} = 0.38$, $\max\{c_v^2\} = 0.79$, $\text{avg}\{c_v^2\} = 0.633$
medium c_v^2:	245	$\min\{c_v^2\} = 0.801$, $\max\{c_v^2\} = 1.95$, $\text{avg}\{c_v^2\} = 1.24$
high c_v^2:	124	$\min\{c_v^2\} = 2.0$, $\max\{c_v^2\} = 30.0$, $\text{avg}\{c_v^2\} = 3.88$
low skewness:	100	$\min\{\text{skewness}\} = 0.88$, $\max\{\text{skewness}\} = 1.69$, $\text{avg}\{\text{skewness}\} = 1.49$
medium skewness:	291	$\min\{\text{skewness}\} = 1.7$, $\max\{\text{skewness}\} = 2.99$, $\text{avg}\{\text{skewness}\} = 2.17$
high skewness:	100	$\min\{\text{skewness}\} = 3.0$, $\max\{\text{skewness}\} = 17.8$, $\text{avg}\{\text{skewness}\} = 4.42$
unimodal ($= 0$):	178	
unimodal (> 0):	195	
bimodal:	112	

Table 3. Distribution classes used in the numerical experiment and their properties

Distribution category	play method	smart play	mono cyclic	optimized block order	optimized exit prob.	optimized both
low c_v^2	17.1197	15.1901	10.1383	8.06738	7.84758	6.99389
medium c_v^2	19.2366	16.4011	10.6259	8.7675	8.06567	6.85941
high c_v^2	12.8795	11.9807	9.19163	7.61338	6.93164	6.16798
low skewness	16.9059	15.037	10.0735	8.10224	7.84758	6.99389
medium skewness	18.9395	16.1918	10.4715	8.376	8.00979	6.8764
high skewness	12.6723	11.866	9.12945	7.4516	6.81537	6.22411
unimodal ($= 0$)	17.0809	15.1225	9.78639	7.92051	7.48931	6.46098
unimodal (> 0)	16.5704	14.5647	9.99832	8.0776	7.36248	6.73391
bimodal	18.243	15.9309	10.4821	8.80954	8.41663	7.44066

Table 4. Median of the average simulation costs for the different distribution classes, with $L = 1$

of the average cost from the median of the cost, for each set. Irrespective of the shape of the distribution, the play and smart play methods always exhibit significantly higher deviations than our methods, and, typically, optimization reduces the costs further. Thus, our optimization procedures not only reduce the average costs, but also help to reduce the variations in costs that may occur due to different shapes of the density.

5.4 Summary

Comparing the performance of the presented methods we observe that using an optimized monocyclic representation for generating PH distributed random variates improves the computational efficiency. The level of improvement depends on the particular PH representation and the point of reference. Compared to computationally expensive representations and the most naive and inefficient (but probably the most frequently used) *Play Method* the proposed approach can be orders of magnitudes faster. Compared to less expensive representations

Distribution category	play method	smart play	mono cyclic	optimized block order	optimized exit prob.	optimized both
low c_v^2	6.91391	5.36713	1.80913	1.50672	1.66765	1.34934
medium c_v^2	9.40542	6.82684	2.3076	2.15295	2.14065	1.79224
high c_v^2	4.8017	3.97381	1.45068	1.2827	1.77428	1.75371
low skewness	7.38663	5.48416	1.60848	1.50833	1.96546	1.43628
medium skewness	9.18359	6.62589	2.19786	1.89588	2.13394	1.61766
high skewness	4.75207	3.82742	1.41999	1.2801	1.61185	1.57042
unimodal ($= 0$)	8.15012	6.11172	1.96242	1.50907	2.07041	1.7176
unimodal (> 0)	6.26133	4.95313	1.72021	1.59748	1.73314	1.39008
bimodal	9.09796	6.88498	2.41072	2.30431	2.3464	1.76058

Table 5. The median absolute deviations of the simulation costs for the distribution classes

and more advanced methods the proposed method still has computational benefit. If the time to generate the optimal representation is not an issue, and the target PH distribution does not have too many phases (< 10), it is worth to optimize both the block order and the exit probability. In other cases, we recommend optimizing only the exit probability. This approach is fast, scales well with the number of phases, and efficiently reduces the generation cost. Furthermore, by optimizing the representations we are able to reduce the variability of the costs. On the other hand, a potential risk of the proposed method is the use of numerically sensitive calculations like the eigenvalue analysis of matrix $\mathbf{A}$.

6 Conclusion

In this paper we studied the optimization of phase-type distributions for random-variate generation. We exploited the use of FE-blocks based representations together with a nonzero exit probability from the FE-blocks. We optimized the generation costs by changing the ordering of the FE-blocks and by modifying exit-probability values. Our optimization procedures are supplemented with a model transformation step in case of a non-Markovian representation.

The methods proposed in this paper have several desirable properties that help with the use of phase-type distributions in simulation: First, as shown in our numerical evaluations, they reduce the costs of random-variate generation by orders of magnitude, when compared to non-optimized approaches. We could demonstrate that both the ordering of the feedback-Erlang blocks and the introduction of a nonzero exit probability are effective methods, and that the improvements are independent of the shape of the distribution. Second, the optimization procedures reduce the variability of the costs across different distributions considerably. Thus, using our approach, the run-time costs of simulations also become more predictable. Third, our methods ensure that the cost of random-variate generation is independent of the structure of the original PH representation. Thus, arbitrary PH distributions can be used efficiently.

Acknowledgements

This work was partially supported by DFG grants Wo 898/3-1 and Wo 898/5-1, by the European Union TAMOP-4.2.2C-11/1/KONV-2012-0001, the OTKA K101150 and the Research and Technology Innovation Fund EITKIC_12-1-2012-0001 projects, and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. We would also like to thank the anonymous reviewers for their extremely helpful suggestions on improving the paper.

References

1. Neuts, M.: Probability distributions of phase type. In: Liber Amicorum Prof. Emeritus H. Florin, University of Louvain (1975) 173–206
2. Telek, M., Horváth, G.: A minimal representation of markov arrival processes and a moments matching method. Performance Evaluation **64**(9-12) (Aug. 2007) 1153–1168
3. Buchholz, P., Telek, M.: On minimal representation of rational arrival processes. Annals of Operations Research **202**(1) (2013) 35–58
4. 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
5. 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
6. Cumani, A.: On the Canonical Representation of Homogeneous Markov Processes Modelling Failure-time Distributions. Microelectronics and Reliability **22** (1982) 583–602
7. Horváth, G., Reinecke, P., Telek, M., Wolter, K.: Efficient generation of ph-distributed random variates. In: ASMTA. LNCS, Gernoble, France, Springer (june 2012) 271–285
8. Mocanu, S., Commault, C.: Sparse Representations of Phase-type Distributions. Commun. Stat., Stochastic Models **15**(4) (1999) 759 – 778
9. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach. Dover Publications, Inc., New York (1981)
10. O'Cinneide, C.A.: On non-uniqueness of representations of phase-type distributions. Communications in Statistics. Stochastic Models **5**(2) (1989) 247–259
11. Bernstein, D.S.: Matrix Mathematics: Theory, Facts, and Formulas. Princeton University Press (2011) 2nd edition.
12. Reinecke, P., Telek, M.: Validity check of matrix-exponential distributions. Technical report, BME, Budapest, Hungary (2012) submitted for publication.
13. Mocanu, S.: MOMI tool, for mono-cyclic representation of PH distributions. Available at http://webspn.hit.bme.hu/~telek/tools/momi.zip (2003)
14. 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)

15. O'Cinneide, C.A.: Characterization of phase-type distributions. Commun. Stat., Stochastic Models **6** (1990) 1–57
16. Neuts, M.F., Pagano, M.E.: Generating random variates from a distribution of phase type. In: WSC '81: Proceedings of the 13th conference on Winter simulation, Piscataway, NJ, USA, IEEE Press (1981) 381–387
17. Reinecke, P.: Efficient System Evaluation Using Stochastic Models. PhD thesis, Freie Universität Berlin (2012/2013)
18. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley, New York (1991)

Appendix

A Unique solution of the representation transformation method

$(\boldsymbol{\alpha}, \mathbf{A})$ of size n and $(\boldsymbol{\beta}, \mathbf{B})$ of size $m \geq n$ are two representations of a PH distribution, Let $\mathbf{A} = {\boldsymbol{\Gamma}_\mathbf{A}}^{-1}\boldsymbol{\Lambda}_\mathbf{A}\boldsymbol{\Gamma}_\mathbf{A}$ and $\mathbf{B} = {\boldsymbol{\Gamma}_\mathbf{B}}^{-1}\boldsymbol{\Lambda}_\mathbf{B}\boldsymbol{\Gamma}_\mathbf{B}$ be the Jordan decomposition of $\mathbf{A}$ and $\mathbf{B}$, where $\boldsymbol{\Gamma}_\mathbf{A}$ and $\boldsymbol{\Gamma}_\mathbf{B}$ are normalized such that $\boldsymbol{\Gamma}_\mathbf{A}\mathbb{1} = \mathbb{1}$ and $\boldsymbol{\Gamma}_\mathbf{B}\mathbb{1} = \mathbb{1}$. The unique solution of $\mathbf{AW} = \mathbf{WB}$, $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ can be obtained from the combination of the following theorems.

Theorem 5. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *have distinct eigenvalues, the set of eigenvalues of* $\mathbf{A}$ *is a subset of the set of eigenvalue of* $\mathbf{B}$*, and the eigenvalues (diagonal elements) of* $\boldsymbol{\Lambda}_\mathbf{B}$ *are ordered such that the size* n *upper left corner of* $\boldsymbol{\Lambda}_\mathbf{B}$ *is identical with* $\boldsymbol{\Lambda}_\mathbf{A}$ *then the unique solution of* $\mathbf{AW} = \mathbf{WB}$*,* $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ *is*

$$\mathbf{W} = {\boldsymbol{\Gamma}_\mathbf{A}}^{-1} \; [\quad \mathbf{I}_n \quad | \, \mathbf{0}_{n,m-n}] \; \boldsymbol{\Gamma}_\mathbf{B} \; ,$$

where $\mathbf{I}_n$ *is the identity matrix of size* n *and* $\mathbf{0}_{n,m-n}$ *is the zero matrix of size* $n \times m - n$*.*

Proof. Substituting the spectral decomposition into $\mathbf{AW} = \mathbf{WB}$ and multiplying with $\boldsymbol{\Gamma}_\mathbf{A}$ from the left and with ${\boldsymbol{\Gamma}_\mathbf{B}}^{-1}$ from the right gives

$$\boldsymbol{\Lambda}_\mathbf{A} \underbrace{\boldsymbol{\Gamma}_\mathbf{A}\mathbf{W}{\boldsymbol{\Gamma}_\mathbf{B}}^{-1}}_{\bar{\mathbf{W}}} = \underbrace{\boldsymbol{\Gamma}_\mathbf{A}\mathbf{W}{\boldsymbol{\Gamma}_\mathbf{B}}^{-1}}_{\bar{\mathbf{W}}}\boldsymbol{\Lambda}_\mathbf{B} \; . \tag{34}$$

Let $\bar{\mathbf{W}} = [\bar{\mathbf{W}}_\mathbf{1}|\bar{\mathbf{W}}_\mathbf{2}]$ be the partition of $\bar{\mathbf{W}}$ such that $\bar{\mathbf{W}}_\mathbf{1}$ is of size $n \times n$, $\bar{\mathbf{W}}_\mathbf{2}$ is of size $n \times m - n$. Using this partition of $\bar{\mathbf{W}}$ in (34) we have

$$\begin{aligned}
&\boldsymbol{\Lambda}_\mathbf{A}\bar{\mathbf{W}} = \boldsymbol{\Lambda}_\mathbf{A}[\bar{\mathbf{W}}_\mathbf{1}|\bar{\mathbf{W}}_\mathbf{2}] = [\boldsymbol{\Lambda}_\mathbf{A}\bar{\mathbf{W}}_\mathbf{1}|\boldsymbol{\Lambda}_\mathbf{A}\bar{\mathbf{W}}_\mathbf{2}] = \\
&\bar{\mathbf{W}}\boldsymbol{\Lambda}_\mathbf{B} = [\bar{\mathbf{W}}_\mathbf{1}|\bar{\mathbf{W}}_\mathbf{2}]\boldsymbol{\Lambda}_\mathbf{B} = [\bar{\mathbf{W}}_\mathbf{1}|\bar{\mathbf{W}}_\mathbf{2}] \begin{bmatrix} \boldsymbol{\Lambda}_\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_\mathbf{E} \end{bmatrix} = [\bar{\mathbf{W}}_\mathbf{1}\boldsymbol{\Lambda}_\mathbf{A}|\bar{\mathbf{W}}_\mathbf{2}\boldsymbol{\Lambda}_\mathbf{E}] \; .
\end{aligned}$$

where $\boldsymbol{\Lambda}_\mathbf{E}$ of size $m - n \times m - n$ contains the extra eigenvalues of $\mathbf{B}$. Since $\boldsymbol{\Lambda}_\mathbf{A}$ and $\boldsymbol{\Lambda}_\mathbf{E}$ are diagonal matrixes with different eigenvalues the solution of

$\mathbf{\Lambda_A}\bar{\mathbf{W}}_\mathbf{1} = \bar{\mathbf{W}}_\mathbf{1}\mathbf{\Lambda_A}$ is a diagonal matrix and the solution of $\mathbf{\Lambda_A}\bar{\mathbf{W}}_\mathbf{2} = \bar{\mathbf{W}}_\mathbf{2}\mathbf{\Lambda_E}$ is the zero matrix. Considering that the multiplication with $\mathbf{\Gamma_B}$ does not affect the row sum because $\mathbf{\Gamma_B}\mathbb{1} = \mathbb{1}$ we have $\mathbb{1} = \mathbf{W}\mathbb{1} = \bar{\mathbf{W}}\mathbb{1} = \bar{\mathbf{W}}_\mathbf{1}\mathbb{1}$ and consequently diagonal matrix $\bar{\mathbf{W}}_\mathbf{1}$ should be $\bar{\mathbf{W}}_\mathbf{1} = \mathbf{I}$.

Theorem 6. *If the Jordan decomposition of* $\mathbf{A}$ *and* $\mathbf{B}$ *are composed by a single Jordan block with identical eigenvalue, then the unique solution of* $\mathbf{AW} = \mathbf{WB}$, $\mathbf{W}\mathbb{1}_m = \mathbb{1}_n$ *is*

$$\mathbf{W} = \mathbf{\Gamma_A}^{-1} \; [\; \mathbf{0}_{n,m-n} \mid \; \mathbf{I}_n \;] \; \mathbf{\Gamma_B} \;,$$

where $\mathbf{I}_n$ *is the identity matrix of size* n *and* $\mathbf{0}_{n,m-n}$ *is the zero matrix of size* $n \times m - n$.

Proof. The proof follows the same simple substitution pattern as the one of Theorem 5 based on

$$\mathbf{\Lambda_A} \; [\; \mathbf{0}_{n,m-n} \mid \; \mathbf{I}_n \;] = [\; \mathbf{0}_{n,m-n} \mid \; \mathbf{\Lambda_A} \;]$$

and

$$[\; \mathbf{0}_{n,m-n} \mid \; \mathbf{I}_n \;] \; \mathbf{\Lambda_B} = [\; \mathbf{0}_{n,m-n} \mid \; \mathbf{\Lambda_A} \;].$$

B The dominant eigenvector of a monocyclic generator

Theorem 7. *The dominant left eigenvector of a monocyclic generator with exit probability* $p_k < 1$ *and arbitrarily ordered FE blocks such that the FE block associated with the dominant eigenvalue is the first one is strictly positive.*

Proof. The general from of such generator is

$$\mathbf{B} = \begin{pmatrix} -\lambda_1 & \lambda_1 & & & & & & \\ & \ddots & (1\text{–}p)(1\text{–}z_j)\lambda_j & & & & & \\ & & -\lambda_i & \lambda_i & & & & \\ & & & \ddots & \ddots & & & \\ & & & & -\lambda_i & \lambda_i & & \\ & & z_i\lambda_i & & & -\lambda_i & (1\text{–}p)(1\text{–}z_i)\lambda_i & \\ & & & & & & \ddots & \end{pmatrix},$$

where $i, j \in \{2, 3, \ldots, n\}$, n is the number of FE blocks, $0 \leq p < 1$ and for all $i \in \{2, 3, \ldots, n\}$ we have $0 \leq z_i < 1$, $-\lambda_1 > r_i = -\lambda_i \left(1 - z_i^{\frac{1}{b_i}}\right)$. The last inequality ensures that the dominant eigenvalue of the FE block is less than $-\lambda_1$. Reformulating this last inequality gives

$$\left(\frac{\lambda_i - \lambda_1}{\lambda_i}\right)^{b_i} - z_i > 0. \tag{35}$$

The dominant eigenvalue of $\mathbf{B}$ is $-\lambda_1$. Let $\gamma = \max_{i\in\{1,\ldots,n\}} \lambda_i$ then $\mathbf{I} + \mathbf{B}/\gamma$ is a non-negative matrix, whose eigenvectors are identical with the ones of $\mathbf{B}$ and whose dominant eigenvalue is $1 - \lambda_1/\gamma$. According to the Perron–Frobenius theorem the dominant left eigenvector associated with $1 - \lambda_1/\gamma$ is non-negative. It remains to prove that the eigenvector is strictly positive.

We show this by investigating the sign of the eigenvector elements associated with consecutive FE blocks. The dominant left eigenvector of $\mathbf{B}$, denoted by v, is the solution of $v(\mathbf{B} + \lambda_1\mathbf{I}) = 0$. For FE block i it has the following structure:

$$(v_{k-b_i}, v_{k-b_i+1}, \ldots, v_k) \begin{pmatrix} (1\text{–}p)(1\text{–}z_j)\lambda_j & & & & \\ -\lambda_i + \lambda_1 & \lambda_i & & & \\ & \ddots & \ddots & & \\ & & -\lambda_i + \lambda_1 & \lambda_i & \\ z_i\lambda_i & & & -\lambda_i + \lambda_1 \end{pmatrix} = 0,$$

where $v_{k-b_i+1}, \ldots, v_k$ are the eigenvector elements associated with FE block i and v_{k-b_i} is last eigenvector element associated with the previous FE block, FE block j. This linear system gives $v_{j-1} = \left(\frac{\lambda_i-\lambda_1}{\lambda_i}\right) v_j$ for $j \in \{k - b_i + 2, \ldots, k\}$, where $0 < \left(\frac{\lambda_i-\lambda_1}{\lambda_i}\right) < 1$ and for the first column we have

$$v_{k-b_i}(1\text{–}p)(1\text{–}z_j)\lambda_j - v_{k-b_i+1}(\lambda_i - \lambda_1) + v_k z_i \lambda_i = 0,$$

from which

$$v_{k-b_i}(1\text{–}p)(1\text{–}z_j)\lambda_j/\lambda_1 = v_k \left(\frac{\lambda_i - \lambda_1}{\lambda_i}\right)^{b_i} - v_k z_i.$$

That is, if v_k is positive then v_j, $j \in \{k - b_i + 1, \ldots, k - 1\}$ are positive as well and finally v_{k-b_i} is positive because the right-hand side is positive due to (35). If v_k is 0 then $v_j = 0$ for $j \in \{k - b_i, \ldots, k - 1\}$. Recursively applying this relation for the consecutive FE blocks gives the theorem.