The Component Queues Method for Quasi-Birth-Death Process Fitting and Analysis

Julianna Bor[1], Giuliano Casale[1], and Evgenia Smirni[2]

[1] Department of Computing, Imperial College London, UK
{jb1316,gcasale}@imperial.ac.uk
[2] Department of Computer Science, William & Mary, USA
exsmir@wm.edu

Abstract. Markovian queueing analysis requires fitting models of arrival and service processes, such as phase-type (PH) distributions, to accurately represent system measurements. However, fitting real-world data is challenging in practice due to model over-parameterization, the large order of the fitting equations, and the limited understanding of the impact of fitting decision on queueing. Motivated by these issues, we present a methodology that uses quasi-birth-death (QBD) processes to fit PH distributions directly from measured queueing data. Core to the proposed method is a new result that reduces the solution of a broad class of QBD processes to the analysis of a collection of simpler queueing systems featuring PH distributions of small order, leading to exact or approximate solutions depending on the chosen arrival process. The proposed method reduces the computational complexity of solving a QBD process from cubic to quadratic for this special class of QBD processes. Numerical results on real traces demonstrate that this feature enables accurate PH fitting from real-world system data.

1 Introduction

An important branch of stochastic modeling focuses on fitting real-world data to the family of distributions compatible with a given queueing model. In particular, phase-type (PH) distributions are routinely used to parameterize queueing systems, as they give rise to an underlying stochastic process that is a continuous-time Markov chain (CTMC) [14]. PH distribution fitting assumes that the empirical distribution obtained from the data can be well approximated by the distribution of passage times in absorbing CTMCs. Unfortunately, this may not hold in practice unless the number of states of the CTMC is large, which introduces difficulties in the fitting process. In fact, existing algorithms struggle to fit general PH models with more than a handful of states, as they incur significant computational and numerical issues, especially when the model is over-parameterized, i.e., it has more parameters than degrees of freedom [3]. However, working with small or medium-sized PH distributions can considerably constrain the range of distributions that can be fitted, as the number of states is linearly proportional to the degrees of freedom available for fitting [16].

Motivated by the above challenges, this paper proposes a novel fitting algorithm to estimate input PH distributions for a class of Markovian queueing systems, focusing on those that can be modeled as quasi-birth-death (QBD) processes. Surveys such as [2] indicate a gap in the literature to estimate arrival and service distributions tailored to QBD processes. Unlike the direct fitting of a PH distribution from traces of inter-arrival or service times, inverse problems involving QBDs fitted from queueing data have rarely been considered [22]. Intuitively, such a development would allow the PH distribution fitting process to capture the salient aspects that drive the observed QBD behavior, filtering out less influential characteristics of the distributions from a queueing analysis perspective. The present work seeks to address this gap by focusing on fitting the QBD service process from the arrival process and the marginal distribution of queue-length probabilities, whether measured or simulated. This is particularly relevant when service times are unavailable, as is often the case with cloud providers.

To make our technique effective, we focus on PH distributions represented as compositional PH models obtained via the Kronecker Product Composition (KPC) operator [16]. We prove the surprising result that parameterizing QBD processes with PH service distributions defined through the KPC operator allows the solution of the QBD process to be *decomposed* into the solutions of independent, but much smaller, QBD processes. Depending on the characteristics of the arrival process, this approach can be exact or approximate. Although divide-and-conquer approaches already exist to explore the space of PH distributions for fitting purposes (e.g., [16]), to the best of our knowledge, this is the first work to develop a PH distribution fitting method of this kind that efficiently searches a space of QBD processes.

Our KPC-based approach reduces the time complexity of calculating marginal queue-length probabilities in a QBD from cubic to quadratic. Leveraging this fast computation of queue-length probabilities alongside numerical search methods, we develop a fitting algorithm that can efficiently explore PH distributions that result in QBD processes that accurately fit the empirical queueing data. Validation experiments on real traces reveal that our method can learn service distributions from queueing data more accurately than existing methods that directly fit the service distributions. This improvement in accuracy also extends to the response time distributions computed from the resulting queueing models. Furthermore, the fitted PH models exhibit low prediction errors when used to parameterize QBD models with different arrival processes, showing low generalization errors.

The remainder of the paper is organized as follows. Section 2 summarizes related work in the area of PH distribution fitting. Section 3 provides basic definitions and relevant background. Section 4 develops the decomposition result for solving QBD processes. Section 5 applies the results of the previous section to the QBD process fitting. In Section 6, we report and discuss our numerical results. Section 7 concludes the paper and outlines future work.

2 Related work

Most fitting algorithms use PH distributions or Markovian arrival processes (MAPs) to fit trace data. PH distributions and MAPs can be readily plugged into various families of performance models, including quasi-birth-death (QBD) processes, queueing networks, stochastic Petri nets and Markovian process algebras [29,5,10,32,13,33]. PH distribution fitting, in particular, has received considerable attention over the years due to its importance in modeling and simulation studies, software reliability, web server modeling, load balancing, and risk theory [24,35,8,30,26,27,18,6,1]. Established fitting approaches include the expectation maximization (EM) algorithm, the cumulative distribution function (CDF) fitting, and moment matching algorithms [34,19,9]. EM algorithms, for which many solutions are available [3,34,7,31], are understood to perform well on shorter traces; however, they are computationally costly and require long-running executions to find accurate PH distributions. Therefore, EM algorithms are overly demanding for fitting the long traces necessary for heavy-tailed distributions. The high computational costs of EM algorithms are overcome by direct fitting of the cumulative distribution function (CDF) [19] or moment matching algorithms [9,16,21]. CDF and moment-based fitting typically require inverting analytical expressions that describe certain statistics of the PH distribution, such as high-order polynomials used to compute the moments of PH distributions. The computational complexity of such expressions is directly proportional to the size of the corresponding PH distribution. Therefore, methods relying on quantities such as moments tend to induce significant restrictions on the size of the PH distribution they attempt to fit. The result is either the fitting of small PH distributions or compositional models constructed based on Kronecker algebra, such as process superposition or Kronecker product composition, which define large distributions in terms of smaller PH distributions. An interesting recent approach uses Bernstein exponentials [20], which form a subclass of acyclic PH distributions, to fit a distribution by sampling its density function.The method allows for tunable accuracy, notably enabling even the fitting of bounded uniform distributions; however, it can converge slowly and require very large PH distributions that may make the following queueing analysis intractable.

3 Preliminaries

In this paper, we consider QBD processes where both inter-arrival and service times follow a PH distribution. A PH distribution is completely determined by a matrix $\boldsymbol{A}_0$ of transition rates between phases and a probability vector $\boldsymbol{\alpha}$ that defines the initial probabilities of starting in each phase. An arrival process with PH inter-arrival times is a Markovian Arrival Process (MAP). Instead of the vector-matrix pair $(\boldsymbol{\alpha}, \boldsymbol{A}_0)$, a PH distribution can also be represented using MAP notation as $\boldsymbol{A} = (\boldsymbol{A}_0, \boldsymbol{A}_1)$ where $\boldsymbol{A}_0 \in \mathbb{R}^{a\times a}$ defines the transition rates between phases and $\boldsymbol{A}_1 \in \mathbb{R}^{a\times a}$ describes external transitions corresponding to arrivals to the system. $\boldsymbol{A}_1$ is a rank one matrix given by $\boldsymbol{A}_1 := -\boldsymbol{A}_0\boldsymbol{1}\boldsymbol{\alpha}$ where

$\boldsymbol{\alpha}$ is the row vector of initial probabilities and $\mathbf{1}$ is a column vector of ones. The infinitesimal generator of the above PH arrival process is $\boldsymbol{Q}_A = \boldsymbol{A}_0 + \boldsymbol{A}_1$. Keeping the matrix notation, the service process is represented by $\boldsymbol{S} = (\boldsymbol{S}_0, \boldsymbol{S}_1)$ where $\boldsymbol{S}_0, \boldsymbol{S}_1 \in \mathbb{R}^{s \times s}$ and the corresponding infinitesimal generator is $\boldsymbol{Q}_S = \boldsymbol{S}_0 + \boldsymbol{S}_1$.

Let matrix $\boldsymbol{B}$ denote the service rates (backward transitions), matrix $\boldsymbol{L}$ the local transitions, and matrix $\boldsymbol{F}$ the arrival rates (forward transitions). We focus on QBD processes with the following infinitesimal generator

$$\boldsymbol{Q} := \begin{bmatrix} \boldsymbol{L}_0 & \boldsymbol{F} & \mathbf{0} & \dots \\ \boldsymbol{B} & \boldsymbol{L} & \boldsymbol{F} & \dots \\ \mathbf{0} & \boldsymbol{B} & \boldsymbol{L} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \qquad \begin{aligned} \boldsymbol{B} &:= \boldsymbol{I}_a \otimes \boldsymbol{S}_1 \\ \boldsymbol{L} &:= \boldsymbol{A}_0 \oplus \boldsymbol{S}_0 = \boldsymbol{A}_0 \otimes \boldsymbol{I}_s + \boldsymbol{I}_a \otimes \boldsymbol{S}_0 \\ \boldsymbol{F} &:= \boldsymbol{A}_1 \otimes \boldsymbol{I}_s \end{aligned} \tag{1}$$

where $\boldsymbol{B}, \boldsymbol{L}, \boldsymbol{F} \in \mathbb{R}^{as \times as}$ and $\boldsymbol{I}_a, \boldsymbol{I}_s$ are identity matrices of dimension $a \times a$ and $s \times s$, respectively, and $\otimes$ denotes the Kronecker product operation. Matrix $\boldsymbol{L}_0 \in \mathbb{R}^{as \times as}$ is different to $\boldsymbol{L}$ as only the arrival process has internal phase transition. Hence, $\boldsymbol{L}_0$ is defined as

$$\boldsymbol{L}_0 := \boldsymbol{A}_0 \otimes \boldsymbol{I}_s \tag{2}$$

The state space of the QBD process can be described with triplets (A, S, L), where $A \in \{1, ..., a\}$ and $S \in \{1, ..., s\}$ denote the internal phase of the arrival and service processes, respectively, and $L \in \mathbb{N}_0$ denotes the level of the process. If the stationary distribution exists, it is the unique solution to the equation

$$\boldsymbol{\pi}_l = \boldsymbol{\pi}_0 \boldsymbol{R}^l \tag{3}$$

where $\boldsymbol{\pi}_l = (\pi_{l1}, \pi_{l2}, ..., \pi_{lm})$ denotes the steady state probabilities corresponding to level l and $m = as$ is the number of internal phases within a given level. Therefore, $|\boldsymbol{\pi}_0| = (1 - \rho)$ where ρ is the utilization of the queue, that is, the probability of a busy server, which occurs whenever the queue is non-empty. Furthermore, $\boldsymbol{R} \in \mathbb{R}^{as \times as}$ is defined as

$$\boldsymbol{R} = -\boldsymbol{F}(\boldsymbol{L} + \boldsymbol{F}\boldsymbol{G})^{-1} \tag{4}$$

where $\boldsymbol{G} \in \mathbb{R}^{as \times as}$ is the minimal non-negative solution to the equation

$$\boldsymbol{B} + \boldsymbol{L}\boldsymbol{G} + \boldsymbol{F}\boldsymbol{G}^2 = 0 \tag{5}$$

The matrix $\boldsymbol{G}$ above has a probabilistic interpretation and plays a central role in our proposed fitting algorithm. Given the state space defined above, the elements of $\boldsymbol{G}$ correspond to the following downward first-passage times. Let $X_t \in (A, S, L), t \geq 0$ denote a QBD process and let $i = (i_0, i_1, l)$ and $j = (j_0, j_1, l-1)$. Denoting the elements of $\boldsymbol{G}$ as $\boldsymbol{G}_{ij}$, the elements of $\boldsymbol{G}$ correspond to the following conditional probabilities:

$$\boldsymbol{G}_{ij} = P(T_{l-1} < \infty \wedge X_{T_{l-1}} = j | X_0 = i) \tag{6}$$

where T_{l-1} is the first passage time to level $(l-1)$. Assuming that the QBD process is positive recurrent, the process reaches any level in a finite time with probability 1, regardless of the starting state. Therefore, the rows of $\boldsymbol{G}$ correspond to the conditional probabilities that, given the process starts in state i of level l, the first passage to level $l-1$ hits the state j of level $l-1$. Since the columns of $\boldsymbol{G}$ correspond to all possible states of level $l-1$, they sum to 1, making $\boldsymbol{G}$ a stochastic matrix.

We now introduce a novel characterization result for the PH/PH/1 queue used throughout the paper. A similar result is established for the MAP/G/1 queue in [12, Theorem 12.10]; unlike this earlier result, our characterization allows the service phase to be explicitly included in the definition of the state space, which is essential for the models considered in this paper.

Theorem 1. *Let* $\boldsymbol{\pi}_0 = (\pi_{01}, \ldots, \pi_{0m})$*, then*

$$\boldsymbol{\pi}_0 = (1-\rho)\boldsymbol{x} \tag{7}$$

where $\boldsymbol{x}$ *is the solution to* $\boldsymbol{xG} = \boldsymbol{x}$ *,* $\boldsymbol{x}\mathbf{1}_m = 1$*.*

The proof of Theorem 1 can be found in [11, Section 5.3.4, Chapter 5]. Despite the tractability of the above formulas, it is well-known that a queue with a large PH service distribution, i.e., with many phases, is expensive to solve. However, limiting the size of the PH service distribution would significantly affect the type of data the model could fit. In this paper, we overcome this by focusing on KPC-PH distributions, composed of small PH building blocks [16]. Here, we recall the key KPC-PH definition used throughout the paper.

Definition 1. *Let* $\boldsymbol{N} = (\boldsymbol{N}_0, \boldsymbol{N}_1)$ *represent a general PH distribution and* $\boldsymbol{H} = (\boldsymbol{H}_0, \boldsymbol{H}_1)$ *a hyperexponential distribution. The KPC-PH distribution denoted by* $\boldsymbol{S} = \boldsymbol{N} \otimes \boldsymbol{H} = (\boldsymbol{S}_0, \boldsymbol{S}_1)$ *is defined as* $\boldsymbol{S}_0 = -\boldsymbol{N}_0 \otimes \boldsymbol{H}_0$ *and* $\boldsymbol{S}_1 = \boldsymbol{N}_1 \otimes \boldsymbol{H}_1$*.*

The fact that $\boldsymbol{H}$ corresponds to a hyperexponential distribution ensures that $\boldsymbol{S}$, as defined above, produces a valid PH distribution. Let $X_{\boldsymbol{N}}$ be the random variable denoting the inter-arrival time of process $\boldsymbol{N}$ in Definition 1 and let $X_{\boldsymbol{H}}$ and $X_{\boldsymbol{S}}$ be the corresponding random variables for $\boldsymbol{H}$ and $\boldsymbol{S}$. It can be seen that the following relation holds [17]

$$E[X_{\boldsymbol{S}}^k] = \frac{1}{k!} E[X_{\boldsymbol{N}}^k] E[X_{\boldsymbol{H}}^k] \tag{8}$$

Since the composed PH distributions have independent degrees of freedom, the overall degrees of freedom of the KPC-PH distribution grow linearly with the number of composed PH distributions, allowing larger compositions to be used for fitting more characteristics of the empirical dataset.

4 Obtaining the queue-length distribution

4.1 Introducing component queues

Let $\boldsymbol{A}/\boldsymbol{S}/1$ denote, analogously to Kendall's notation, a KPC-queue with a PH renewal arrival process $\boldsymbol{A}$ and KPC-PH service time distribution $\boldsymbol{S} = \boldsymbol{N} \otimes$

$\boldsymbol{H}_1 \otimes \ldots \otimes \boldsymbol{H}_J$. We follow the convention that scalar multiplication of a process applies to all representation matrices, that is, $c\boldsymbol{A} = (c\boldsymbol{A}_0, c\boldsymbol{A}_1)$ for real c. Our main result is a strategy to decompose the KPC-PH queue into a collection of smaller queueing systems, called *component queues*, enabling a divide-and-conquer solution strategy for the QBD model. We begin by formally defining component queues.

Definition 2 (Component queues). *Consider a KPC queue $\boldsymbol{A}/\boldsymbol{S}/1$ where $\boldsymbol{S} = \boldsymbol{N} \otimes \boldsymbol{H}_1 \otimes \ldots \otimes \boldsymbol{H}_J$ is the Kronecker product composition of a general PH and J hyperexponential distributions, its decomposition into component queues is given by the following set of queueing processes*

1. *$CQ_N := \boldsymbol{A}/\boldsymbol{N}^*/1$ where $\boldsymbol{N}^* := \mu^*_{H_1} \ldots \mu^*_{H_J} \boldsymbol{N}$, and $\mu^*_{H_k}$ is the mean departure rate of process $\boldsymbol{H}_k, k = 1, \ldots, J$.*
2. *$CQ_{H_k} = M/\boldsymbol{H}^*_k/1$, where M is a Poisson process with unit rate and*

$$\boldsymbol{H}^*_k := (\mu^*_N/\lambda^*_A) \left(\prod_{1 \leq i \leq J, i \neq k} \mu^*_{H_i} \right) \boldsymbol{H}_k$$

where μ^*_N is the mean departure rate of process $\boldsymbol{N}$, $\mu^*_{H_i}$ are the mean departure rates of processes $\boldsymbol{H}_i$, and λ^*_A is the mean arrival rate of process $\boldsymbol{A}$.

The component queues CQ_N and $CQ_{H_k}, 1 \leq k \leq J$ defined above are themselves QBD processes. Moreover, each component queue has the same utilization as the original $\boldsymbol{A}/\boldsymbol{S}/1$ queue. The latter property can be readily verified by the definitions, using the recursive expression of the Kronecker product composition moments in (8) for $k = 1$, as the service rates are reciprocals of the mean service times. Thus, if the $\boldsymbol{A}/\boldsymbol{S}/1$ queue is stable, so are its component queues.

4.2 Approximating the matrix $\boldsymbol{G}$

We approximate the matrix $\boldsymbol{G}$ of the original queue via the matrices $\boldsymbol{G}_N$ and $\boldsymbol{G}_{H_k}, k = 1, \ldots, J$, which are analogously defined as $\boldsymbol{G}$, but for the component queues CQ_N and CQ_{H_k}. Let $\widetilde{\boldsymbol{G}}$ denote the approximation of the matrix $\boldsymbol{G}$. In the case of Poisson arrivals, $\widetilde{\boldsymbol{G}}$ is the same as $\boldsymbol{G}$, so the approximation holds without error. Formally, we have

Proposition 1. *Let $M/\boldsymbol{N} \otimes \boldsymbol{H}_1 \otimes \ldots \otimes \boldsymbol{H}_J/1$ be a KPC-PH queue with Poisson arrivals and let $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{G}_{H_J}$. Then $\widetilde{\boldsymbol{G}} = \boldsymbol{G} = \mathbf{1}\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the embedded distribution of the service process phase at departure instants.*

The proof of Proposition 1 is given in [11, Section 5.3.2, Chapter 5]. While it gives an exact decomposition of the $\boldsymbol{G}$ matrix for queues with Poisson inputs, if the arrival process is an arbitrary PH then $\boldsymbol{G}$ and $\widetilde{\boldsymbol{G}}$ are no longer identical and our method provides an approximation. The following result outlines for this case a relationship between $\boldsymbol{G}$ and its approximation $\widetilde{\boldsymbol{G}}$.

Theorem 2. *Consider a queue with PH arrival process $\boldsymbol{A}$ and KPC-PH service process $\boldsymbol{S} = \boldsymbol{N} \otimes \boldsymbol{H}_1 \otimes \ldots \otimes \boldsymbol{H}_J$, where $\boldsymbol{H}_k, k = 1, \ldots, J$ are hyperexponential. Defining $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{G}_{H_J}$, we have the following relationship between $\widetilde{\boldsymbol{G}}$ and $\boldsymbol{G}$*

$$\widetilde{\boldsymbol{G}} \approx (\boldsymbol{I}_{\boldsymbol{an}} \otimes \boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{G}_{H_J})\, \boldsymbol{G} \tag{9}$$

where a and n denote the number of phases of process $\boldsymbol{A}$ and $\boldsymbol{N}$, respectively, and $\boldsymbol{I}_{\boldsymbol{an}}$ is the identity matrix with dimension $an \times an$.

An approximate lumping argument supporting Theorem 2 is given in [11, Section 5.3.2, Chapter 5]. As we explain in the following sections, the above result is fundamental, as it allows us to estimate steady-state performance metrics for the QBD process in terms of the $\boldsymbol{G}_N$ and $\boldsymbol{G}_{H_k}$ matrices of the individual component queues CQ_N and CQ_{H_k}. As these processes are significantly smaller than the original queue, the net result is a large reduction in computational cost of the QBD analysis.

The key result in Theorem 2 is interpreted as follows. Matrix $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{G}_{H_J}$, which is efficiently constructed with a Kronecker product of matrices, each of which is obtained by solving the smaller component queues, may be seen as an approximation of $\boldsymbol{G}$ by means of aggregation. The aggregation is applied by the premultiplying factors on the right-hand side of (9). Compared to the exact model, relying on $\widetilde{\boldsymbol{G}}$ thus loses some information about the internal phases of process $\boldsymbol{H}$, but its computation can be done much more efficiently than computing $\boldsymbol{G}$. This provides a novel approximation for the solution of the QBD, which becomes exact in the case of Poisson arrivals.

4.3 Example

To illustrate (9), we now show it in a small example. Let the number of hyperexponential components in the KPC service be $J = 1$, and the degree of the arrival process $\boldsymbol{A}$ and the service components $\boldsymbol{N}, \boldsymbol{H}$ be 2, so that $an = 4$ and let $\boldsymbol{\alpha}_H = [\alpha_1, \alpha_2]$ be the embedded distribution of the hyperexponential component. From Proposition 1, we know that $\boldsymbol{G}_H = \boldsymbol{1}\boldsymbol{\alpha}_H$. Multiplying $\boldsymbol{I_4} \otimes \boldsymbol{G}_H$ with $\boldsymbol{G} = (g_{ij})_{1 \le i,j, \le m}$ implies that the $(2i-1)$th and $2i$th rows of $\widetilde{\boldsymbol{G}}$ are identical. Furthermore, they are computed from $\boldsymbol{G}$ by averaging these specific rows, weighted according to the embedded distribution $\boldsymbol{\alpha}_H$ of $\boldsymbol{H}$. That is,

$$\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_H = (\boldsymbol{I_4} \otimes \boldsymbol{G}_H)\boldsymbol{G} = \begin{bmatrix} \alpha_1 g_{11} + \alpha_2 g_{21} & \alpha_1 g_{12} + \alpha_2 g_{22} & \ldots & \alpha_1 g_{18} + \alpha_2 g_{28} \\ \alpha_1 g_{11} + \alpha_2 g_{21} & \alpha_1 g_{12} + \alpha_2 g_{22} & \ldots & \alpha_1 g_{18} + \alpha_2 g_{28} \\ \alpha_1 g_{31} + \alpha_2 g_{41} & \alpha_1 g_{32} + \alpha_2 g_{42} & \ldots & \alpha_1 g_{38} + \alpha_2 g_{48} \\ \alpha_1 g_{31} + \alpha_2 g_{41} & \alpha_1 g_{32} + \alpha_2 g_{42} & \ldots & \alpha_1 g_{38} + \alpha_2 g_{48} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1 g_{71} + \alpha_2 g_{81} & \alpha_1 g_{72} + \alpha_2 g_{82} & \ldots & \alpha_1 g_{78} + \alpha_2 g_{88} \end{bmatrix}$$

which shows the aforementioned aggregation property. We also provide a numerical example to further illustrate the aggregation process. Consider a queue

with the following hyperexponential arrival process with mean 3 and squared coefficient of variation (SCV) 11/9

$$\boldsymbol{A}_0 = \begin{bmatrix} -0.25 & 0 \\ 0 & -0.5 \end{bmatrix} \qquad \boldsymbol{A}_1 = \begin{bmatrix} 0.125 & 0.125 \\ 0.25 & 0.25 \end{bmatrix}$$

and a KPC service process where the first component is Erlang-2 with unit mean and the second component is a 2-phase hyperexponential distribution with the following parameters

$$\boldsymbol{H}_0 = \begin{bmatrix} -10 & 0 \\ 0 & -20 \end{bmatrix} \qquad \boldsymbol{H}_1 = \begin{bmatrix} 5 & 5 \\ 5 & 15 \end{bmatrix}$$

corresponding to mean 1/15 and SCV 5/4. Below, we can see how closely $\boldsymbol{G}$ aligns with the aggregated matrix $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_H$,

$$\boldsymbol{G} = \begin{bmatrix} 0.488 & 0.488 & 0 & 0 & 0.012 & 0.012 & 0 & 0 \\ 0.494 & 0.494 & 0 & 0 & 0.006 & 0.006 & 0 & 0 \\ 0.494 & 0.494 & 0 & 0 & 0.006 & 0.006 & 0 & 0 \\ 0.497 & 0.497 & 0 & 0 & 0.003 & 0.003 & 0 & 0 \\ 0.024 & 0.024 & 0 & 0 & 0.476 & 0.476 & 0 & 0 \\ 0.012 & 0.012 & 0 & 0 & 0.488 & 0.488 & 0 & 0 \\ 0.012 & 0.012 & 0 & 0 & 0.488 & 0.488 & 0 & 0 \\ 0.006 & 0.006 & 0 & 0 & 0.494 & 0.494 & 0 & 0 \end{bmatrix} \quad \widetilde{\boldsymbol{G}} = \begin{bmatrix} 0.491 & 0.491 & 0 & 0 & 0.009 & 0.009 & 0 & 0 \\ 0.491 & 0.491 & 0 & 0 & 0.009 & 0.009 & 0 & 0 \\ 0.496 & 0.496 & 0 & 0 & 0.005 & 0.005 & 0 & 0 \\ 0.496 & 0.496 & 0 & 0 & 0.005 & 0.005 & 0 & 0 \\ 0.018 & 0.018 & 0 & 0 & 0.482 & 0.482 & 0 & 0 \\ 0.018 & 0.018 & 0 & 0 & 0.482 & 0.482 & 0 & 0 \\ 0.009 & 0.009 & 0 & 0 & 0.491 & 0.491 & 0 & 0 \\ 0.009 & 0.009 & 0 & 0 & 0.491 & 0.491 & 0 & 0 \end{bmatrix}$$

As visible, the proposed approximation based on component queues effectively provides a local weighted sum of neighboring elements of $\boldsymbol{G}$. The benefit is that the expression on the right can be computed efficiently via Kronecker products, whereas the original $\boldsymbol{G}$ matrix on the left cannot.

4.4 Approximating the matrix $\boldsymbol{R}$

The formula (3), which is used for calculating the steady-state queue-length probability vectors, requires the calculation of matrix $\boldsymbol{R}$. In this subsection, $\widetilde{\boldsymbol{G}}$, previously derived, is used to obtain an estimate for $\boldsymbol{R}$. Using (4) and substituting the approximation $\widetilde{\boldsymbol{G}}$ for $\boldsymbol{G}$, we propose the approximation

$$\boldsymbol{R} \approx -\boldsymbol{F}(\boldsymbol{L} + \boldsymbol{F}\widetilde{\boldsymbol{G}})^{-1} \tag{10}$$

In the above formula, the computational bottleneck is the inverse calculation, which, as we now show, exhibits special properties for the class of QBDs we consider. First, we have found through numerical experimentation that the inversion of matrix $\boldsymbol{L}$ is very efficient due to its sparsity. Furthermore, we prove in the following result that $\boldsymbol{F}\widetilde{\boldsymbol{G}}$ is a low-rank matrix.

Theorem 3. *Given a KPC queue with PH arrival process $\boldsymbol{A} = (\boldsymbol{A}_0, \boldsymbol{A}_1)$ and service process $\boldsymbol{S} = \boldsymbol{N} \otimes \boldsymbol{H}_1 \otimes \ldots \otimes \boldsymbol{H}_J$, where $\boldsymbol{H}_k, k = 1, \ldots, J$ are hyperexponential and $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{G}_{H_J}$ as before, $rank(\boldsymbol{F}\widetilde{\boldsymbol{G}}) \leq n$, where n is the number of phases of process $\boldsymbol{N}$.*

The proof of Theorem 3 is given in [11, Section 5.3.3, Chapter 5]. The low-rank of $\boldsymbol{F}\widetilde{\boldsymbol{G}}$ is an important property of our method, which allows us to efficiently compute the inverse of the sum in (10) using the Woodbury matrix identity:

$$\begin{aligned}\boldsymbol{R} \approx -\boldsymbol{F}(\boldsymbol{L}+\boldsymbol{F}\widetilde{\boldsymbol{G}})^{-1} &= -\boldsymbol{F}(\boldsymbol{L}+\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V})^{-1} \\ &= -\boldsymbol{F}(\boldsymbol{L}^{-1}-\boldsymbol{L}^{-1}\boldsymbol{U}(\boldsymbol{\Sigma}^{-1}+\boldsymbol{V}\boldsymbol{L}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{L}^{-1}) \quad (11)\end{aligned}$$

where $\boldsymbol{F}, \boldsymbol{L}, \widetilde{\boldsymbol{G}} \in \mathbb{R}^{m\times m}, \boldsymbol{U} \in \mathbb{R}^{m\times k}, \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}, \boldsymbol{V} \in \mathbb{R}^{k\times m}$, and $k = rank(\boldsymbol{F}\widetilde{\boldsymbol{G}})$. The Woodbury matrix identity relies on the decomposition of $\boldsymbol{F}\widetilde{\boldsymbol{G}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$, where $\boldsymbol{\Sigma}$ is invertible and hence $rank(\boldsymbol{\Sigma}) = rank(\boldsymbol{F}\widetilde{\boldsymbol{G}})$. A straightforward approach to obtain $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ is to use the Singular Value Decomposition (SVD) of $\boldsymbol{F}\widetilde{\boldsymbol{G}}$. The computational complexity of this operation for a matrix with dimension $m \times m$ is typically $\mathcal{O}(m^3)$ in general [4]. However, we can improve that by exploiting the special Kronecker-form structure of $\boldsymbol{F}\widetilde{\boldsymbol{G}}$. We know

$$\boldsymbol{F}_{H_k}\boldsymbol{G}_{H_k} = \boldsymbol{I}_{h_k}(\boldsymbol{1}_{h_k}\boldsymbol{\alpha}_{H_k}) = \boldsymbol{1}_{h_k}\boldsymbol{\alpha}_{H_k}, k = 1, \ldots, J$$

from Proposition 1. Let $\boldsymbol{F}_N\boldsymbol{G}_N = \boldsymbol{U}_N\boldsymbol{\Sigma}\boldsymbol{V}_N$ be the SVD of matrix $\boldsymbol{F}_N\boldsymbol{G}_N$. Define $\boldsymbol{U}$ and $\boldsymbol{V}$ as follows

$$\begin{aligned}\boldsymbol{F}\widetilde{\boldsymbol{G}} &= \boldsymbol{F}_N\boldsymbol{G}_N \otimes \boldsymbol{F}_{H_1}\boldsymbol{G}_{H_1} \otimes \ldots \otimes \boldsymbol{F}_{H_J}\boldsymbol{G}_{H_J} \\ &= (\boldsymbol{U}_N\boldsymbol{\Sigma}\boldsymbol{V}_N) \otimes (\boldsymbol{1}_{h_1}\boldsymbol{\alpha}_{H_1}) \otimes \ldots \otimes (\boldsymbol{1}_{h_J}\boldsymbol{\alpha}_{H_J}) \\ &= (\boldsymbol{U}_N \otimes \boldsymbol{1}_{h_1} \otimes \ldots \otimes \boldsymbol{1}_{h_J})\boldsymbol{\Sigma}(\boldsymbol{V}_N \otimes \boldsymbol{\alpha}_{H_1} \ldots \otimes \boldsymbol{\alpha}_{H_J}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V} \quad (12)\end{aligned}$$

Since $rank(\boldsymbol{F}\widetilde{\boldsymbol{G}}) = rank(\boldsymbol{F}_N\boldsymbol{G}_N) = rank(\boldsymbol{\Sigma})$, this decomposition has the required properties. Moreover, it is significantly faster to calculate as it only requires the SVD of a much smaller matrix $\boldsymbol{F}_N\boldsymbol{G}_N$. The SVD of $\boldsymbol{F}_N\boldsymbol{G}_N$ takes $\mathcal{O}(n^3)$ time and calculating the Kronecker products to obtain $\boldsymbol{U}$ and $\boldsymbol{V}$ takes $\mathcal{O}(m^2)$ time. Therefore, the entire decomposition runs in $\mathcal{O}(n^3 + m^2)$ time. Recall that the size of $\boldsymbol{F}_N\boldsymbol{G}_N$ is $n \times n$ and $\boldsymbol{F}\widetilde{\boldsymbol{G}}$ is a $m \times m$ matrix. Under the assumption that $n \ll m$, $\mathcal{O}(n^3 + m^2)$ is dominated by the m^2 factor.

4.5 Approximating the queue-length probabilities

With the efficient approximation of matrix $\boldsymbol{R}$ in (3), we can now compute the steady-state probability vectors $\widetilde{\boldsymbol{\pi}}_l = (\pi_{l1}, \ldots, \pi_{lm})$ for $l \in \mathbb{N}_0$, where m is the number of phases in each level. As stated in the background section, to calculate $\boldsymbol{\pi}_0$, one must first solve the associated system of equations for $\boldsymbol{x}$. It is known that if a solution $\boldsymbol{x}$ exists, then it is unique for the system. Using our approximate $\widetilde{\boldsymbol{G}} = \boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \cdots \otimes \boldsymbol{G}_{H_1}$, we have similarly to (7) that $\boldsymbol{x}\widetilde{\boldsymbol{G}} = \boldsymbol{x}$, $\boldsymbol{x}\boldsymbol{1}_s = 1$. Let $\boldsymbol{x}_N$ and $\boldsymbol{x}_{H_k}, k = 1, \ldots, J$ be the solutions of the above system of equation where $\widetilde{\boldsymbol{G}}$ is replaced by $\boldsymbol{G}_N$ and $\boldsymbol{G}_{H_k}, k = 1, \ldots, J$, respectively.

$$\begin{aligned}(\boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J})\widetilde{\boldsymbol{G}} &= (\boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J})(\boldsymbol{G}_N \otimes \boldsymbol{G}_{H_1} \otimes \cdots \otimes \boldsymbol{G}_{H_J}) \\ &= \boldsymbol{x}_N\boldsymbol{G}_N \otimes \boldsymbol{x}_{H_1}\boldsymbol{G}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J}\boldsymbol{G}_{H_J} \\ &= \boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J}\end{aligned}$$

where the last equation holds because of the definition of the vectors $\boldsymbol{x}_i$. Furthermore,

$$\begin{aligned}(\boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J})\mathbf{1}_s &= (\boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J})(\mathbf{1}_n \otimes \mathbf{1}_{h_1} \otimes \cdots \otimes \mathbf{1}_{h_J}) \\ &= \boldsymbol{x}_N\mathbf{1}_n \otimes \boldsymbol{x}_{H_1}\mathbf{1}_{h_1} \otimes \cdots \otimes \boldsymbol{x}_{H_J}\mathbf{1}_{h_J} \\ &= 1 \otimes 1 \otimes \cdots \otimes 1 = 1\end{aligned}$$

Therefore, $\boldsymbol{x}_N \otimes \boldsymbol{x}_{H_1} \otimes \cdots \otimes \boldsymbol{x}_{H_k}$ is the unique solution to the approximate system of equations $\boldsymbol{x}\widetilde{\boldsymbol{G}} = \boldsymbol{x}$, $\boldsymbol{x}\mathbf{1}_s = 1$. Furthermore, Proposition 1 states that $\boldsymbol{G}_{H_k} = \mathbf{1}_{h_k}\boldsymbol{\alpha}_{H_k}$. This implies that $\boldsymbol{x}_{H_k} = \boldsymbol{\alpha}_{H_k}$, since $\boldsymbol{\alpha}_{H_k}$ is a probability vector and $\boldsymbol{\alpha}_{H_k}\boldsymbol{G}_{H_k} = \boldsymbol{\alpha}_{H_k}\mathbf{1}_{h_k}\boldsymbol{\alpha}_{H_k} = 1\boldsymbol{\alpha}_{H_k} = \boldsymbol{\alpha}_{H_k}$.
Let $\widetilde{\boldsymbol{R}}$ be the approximation of $\boldsymbol{R}$ given by (11). Based on the previous results, $\widetilde{\boldsymbol{\pi}}_l$ can be computed as

$$\widetilde{\boldsymbol{\pi}}_0 = (1-\rho)(\boldsymbol{x}_N \otimes \boldsymbol{\alpha}_{H_1} \otimes \ldots \otimes \boldsymbol{\alpha}_{H_J}) \tag{13}$$

$$\widetilde{\boldsymbol{\pi}}_l = \widetilde{\boldsymbol{\pi}}_0\widetilde{\boldsymbol{R}}^l \qquad \text{for} \quad l > 0 \tag{14}$$

5 Fitting algorithm

In this section, the theoretical results established in the previous section are combined with a search algorithm to find a QBD process that matches a given set of queue-length probabilities. Our main contribution is the optimization problem formulation. The user may choose different numerical solvers based on their preferred cost and convergence properties.

5.1 Decision variables

The decision variables are a set of vector parameters used to construct the components of the KPC-PH distribution. Initially, we describe the construction of the hyperexponential PH components, denoted as $\boldsymbol{H}_i = (\boldsymbol{H}_{i0}, \boldsymbol{H}_{i1}), i = 1, \ldots, J$, specified by the parameter vectors $\boldsymbol{\alpha}_{H_i} \in \mathbb{R}^{h_i - 1}$ and $\boldsymbol{\lambda}_{H_i} \in \mathbb{R}^{h_i}$. Note, that $\boldsymbol{\alpha}_{H_i}$ has one dimension less than the number of phases in process $\boldsymbol{H}$, with the last element chosen to satisfy the constraint that the elements sum to 1. Therefore, the actual probability vector is formulated as $\boldsymbol{\alpha}'_{H_i} = (\boldsymbol{\alpha}_{H_i}, 1 - \boldsymbol{\alpha}_{H_i}\mathbf{1}_{h_i-1})$.
To ensure $\boldsymbol{\alpha}'_{H_i}$ is a valid probability vector, the following constraints apply to the decision variables: $\mathbf{0} \leq \boldsymbol{\alpha}_{H_i}$ and $\boldsymbol{\alpha}_{H_i}\mathbf{1} \leq 1$ for $i = 1, \ldots, J$. The diagonal elements of $\boldsymbol{H}_{i0}$ are stored in the decision vector $\mathbf{0} \leq \boldsymbol{\lambda}_{H_i}, i = 1, \ldots, J$, so that $\boldsymbol{H}_{i0} = \mathrm{diag}(-\boldsymbol{\lambda}_{H_i})$ and $\boldsymbol{H}_{i1} = \boldsymbol{\lambda}^{\top}_{H_i}\boldsymbol{\alpha}'_{H_i}$.
Constructing $\boldsymbol{N} = (\boldsymbol{N}_0, \boldsymbol{N}_1)$ requires additional effort as the matrices $\boldsymbol{N}_0, \boldsymbol{N}_1$ are no longer diagonal. The decision vector $\boldsymbol{\alpha}_N \in \mathbb{R}^{n-1}$ is analogous to $\boldsymbol{\alpha}_{H_k}$ and thus subject to the same constraints: $\mathbf{0} \leq \boldsymbol{\alpha}_N$ and $\boldsymbol{\alpha}_N\mathbf{1} \leq 1$. Finally, $\mathbf{0} \leq \boldsymbol{\lambda}_N \in \mathbb{R}^{n^2}$ records both the off-diagonal elements of $\boldsymbol{N}_0$ and the exit rates $\boldsymbol{t}$. Here, $\boldsymbol{N}_1 = \boldsymbol{t}\boldsymbol{\alpha}_N$, and the diagonal elements of $\boldsymbol{N}_0$ are set to ensure the row sums of $\boldsymbol{Q}_N = \boldsymbol{N}_0 + \boldsymbol{N}_1$ sum to 0 as required.

Algorithm 1 Calculating the approximate queue-length probabilities

Input: $\boldsymbol{\alpha}_N, \boldsymbol{\lambda}_N, \boldsymbol{\alpha}_{H_1}, \boldsymbol{\lambda}_{H_1} \ldots, \boldsymbol{\alpha}_{H_J}, \boldsymbol{\lambda}_{H_J}, \boldsymbol{A}$

1: Construct the PH distributions from $\boldsymbol{\alpha}_i, \boldsymbol{\lambda}_i$ parameters as shown in Section 5.1
2: Calculate $\boldsymbol{F}$ and $\boldsymbol{L}$ from the components (1)
3: Solve CQ_N with cyclic reduction to get $\boldsymbol{G}_N$ ▷ $\mathcal{O}(L_{CR}n^3)$
4: Calculate the SVD decomposition of $\boldsymbol{F}_N\boldsymbol{G}_N = \boldsymbol{U}_N\boldsymbol{\Sigma}\boldsymbol{V}_N$ ▷ $\mathcal{O}(n^3)$
5: Calculate $\boldsymbol{U}$ and $\boldsymbol{V}$ for decomposition $\boldsymbol{F}\widetilde{\boldsymbol{G}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ (12) ▷ $\mathcal{O}(m^2)$
6: Calculate $\widetilde{\boldsymbol{\pi}}_0$ from the components (4.5)
7: Run Algorithm 2 with input $\boldsymbol{F}, \boldsymbol{L}, \widetilde{\boldsymbol{\pi}}_0, \boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ to get $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_t$ ▷ $\mathcal{O}(m^2t)$
8: Calculate marginal probability for level 0 as $\widetilde{\pi}_0 = \widetilde{\boldsymbol{\pi}}_0\mathbf{1}_m$
9: **return** $\widetilde{\boldsymbol{\pi}} = (\widetilde{\pi}_0, \ldots, \widetilde{\pi}_t)$

Algorithm 2 Calculating $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_t$

Input: $\boldsymbol{F}, \boldsymbol{L}, \widetilde{\boldsymbol{\pi}}_0, \boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$,

1: $\boldsymbol{\Sigma}^{-1} = \text{inv}(\boldsymbol{\Sigma})$ ▷ $\mathcal{O}(k)$ as $\boldsymbol{\Sigma}$ is diagonal
2: $\boldsymbol{L}^{-1} = \text{inv}(\boldsymbol{L})$ ▷ $\mathcal{O}(m^2)$ when $\boldsymbol{L}$ is upper triangular
3: $\boldsymbol{A} = \boldsymbol{V}L^{-1}$ ▷ $\mathcal{O}(km^2)$ as $\boldsymbol{V}$ is rectangular
4: $\boldsymbol{B} = \boldsymbol{\Sigma}^{-1} + \boldsymbol{A}\boldsymbol{U}$ ▷ $\mathcal{O}(k^2m)$ as $\boldsymbol{U}$ is rectangular
5: $\widetilde{\pi}_0 = (1-\rho), \quad \widetilde{\boldsymbol{\pi}}_{l-1} = \widetilde{\boldsymbol{\pi}}_0$
6: **for** $l = 1, \ldots, t$ **do**
7: $\quad \boldsymbol{u}_0 = \widetilde{\boldsymbol{\pi}}_{l-1}(-\boldsymbol{F})\boldsymbol{L}^{-1}$ ▷ $\mathcal{O}(m^2)$
8: $\quad \boldsymbol{u}_1 = \boldsymbol{u}_0\boldsymbol{U}$ ▷ $\mathcal{O}(km)$ as $\boldsymbol{U}$ is rectangular
9: $\quad \boldsymbol{u}_2 = (\boldsymbol{B}^\top \backslash \boldsymbol{u}_1^\top)^\top$ ▷ $\mathcal{O}(k^3)$ to solve system of linear equations
10: $\quad \boldsymbol{u}_2 = \boldsymbol{u}_2\boldsymbol{V}$ ▷ $\mathcal{O}(km)$ as $\boldsymbol{V}$ is rectangular
11: $\quad \boldsymbol{u}_1 = \boldsymbol{u}_2\boldsymbol{L}^{-1}$ ▷ $\mathcal{O}(m^2)$
12: $\quad \widetilde{\boldsymbol{\pi}}_l = \boldsymbol{u}_0 - \boldsymbol{u}_1, \quad \widetilde{\pi}_l = \widetilde{\boldsymbol{\pi}}_l\mathbf{1}_m, \quad \widetilde{\boldsymbol{\pi}}_{l-1} = \widetilde{\boldsymbol{\pi}}_l$
13: **end for** ▷ $\mathcal{O}((m^2 + k^3)t)$
14: **return** $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_t$

5.2 Outline of the fitting algorithm

During the fitting process, the assumption is that we have information on the arrival process $\boldsymbol{A}$ and a finite number t of marginal queue-length probabilities $\boldsymbol{\pi} = (\pi_0, \pi_1, \ldots, \pi_t)$. The marginal queue length probabilities are calculated from the queue-length probabilities by summing the probability vectors at each level, that is $\pi_0 = \boldsymbol{\pi}_0\mathbf{1}_m$ and $\pi_1 = \boldsymbol{\pi}_1\mathbf{1}_m$. We start by implementing the equations of Section 4 in Algorithm 1, to calculate the approximate marginal queue-length distribution from the decision variables $\boldsymbol{\alpha}_N, \boldsymbol{\lambda}_N, \boldsymbol{\alpha}_{H_1}, \boldsymbol{\lambda}_{H_1} \ldots, \boldsymbol{\alpha}_{H_J}, \boldsymbol{\lambda}_{H_J}$, and the arrival process $\boldsymbol{A}$. The time complexity of the main steps is noted inline and discussed in detail in [11, Section 5.4.5, Chapter 5]. Once the relevant matrices and $\widetilde{\boldsymbol{\pi}}_0$ are calculated, Algorithm 1 calls Algorithm 2 to calculate the remaining marginal probabilities iteratively. To reduce the time complexity of the computation, we do not compute matrix $\boldsymbol{R}$ directly. Instead, the Woodbury identity is applied at every iteration, and matrix-matrix multiplication is entirely avoided. We point to [11, Section 5.4.5, Chapter 5] for a detailed analysis of the time complexity corresponding to the steps in Algorithm 2.

The optimization problem to find a QBD process with a KPC-PH service distribution, that has the given arrival process ($\boldsymbol{A}$), and that when solved has the given marginal queue length probabilities (π_l for $l = 0, \ldots, t$), can be written as follows

$$\begin{aligned} &\min \textstyle\sum_{l=0}^{t} |\widetilde{\pi}_l - \pi_l|/\pi_l \\ &\text{s.t. } \boldsymbol{\alpha}_i \mathbf{1} \leq 1,\ i \in I := \{N, H_1, \ldots, H_J\} \\ &\quad \boldsymbol{\alpha}_i, \boldsymbol{\lambda}_i \geq \mathbf{0},\ i \in I \end{aligned} \tag{15}$$

where $\boldsymbol{\alpha}_N \in \mathbb{R}^{n-1}, \boldsymbol{\alpha}_{H_i} \in \mathbb{R}^{h_i - 1}, \boldsymbol{\lambda}_N \in \mathbb{R}^{n \times n}, \boldsymbol{\lambda}_{H_i} \in \mathbb{R}^{h_i}$ with $i = 1, \ldots, J$ are the decision variables, and $\widetilde{\boldsymbol{\pi}} = f(\boldsymbol{\alpha}_N, \boldsymbol{\lambda}_N, \boldsymbol{\alpha}_{H_1}, \boldsymbol{\lambda}_{H_1} \ldots, \boldsymbol{\alpha}_{H_J}, \boldsymbol{\lambda}_{H_J} | \boldsymbol{A})$ where function f is the implementation of Algorithm 1.

The vector $\boldsymbol{\pi}$ and the arrival process $\boldsymbol{A}$ are inputs to the problem and remain unchanged throughout. In the case of an irreducible Markov chain, $\boldsymbol{\pi} > 0$; hence, division by π_l is always allowed up to machine precision. Thus, we have $V = n - 1 + n^2 + \sum_{i=1}^{J}(2h_i - 1)$ variables and $C = V + n - 1 + \sum_{i=1}^{J}(h_i - 1)$ constraints. The objective function calculates the error between the input $\boldsymbol{\pi}$ and approximate $\widetilde{\boldsymbol{\pi}} = (\widetilde{\pi}_0, \ldots, \widetilde{\pi}_t)$. Normalizing the difference forces the fitting algorithm to correctly capture the tail of the distribution. This is necessary because the probabilities corresponding to higher levels are very close to zero; therefore, otherwise, the difference between the approximate and the input probabilities would be negligible.

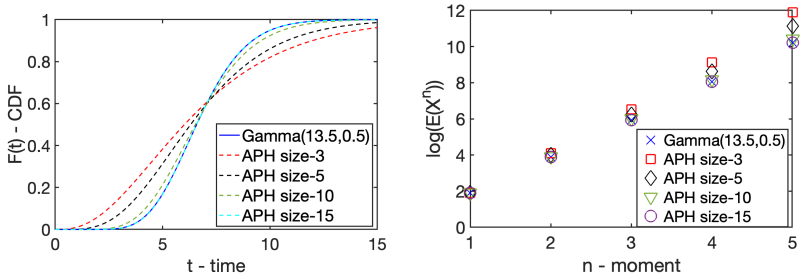
5.3 Using a gamma distribution as $\boldsymbol{N}$

As expected, the general PH distribution $\boldsymbol{N}$ requires more parameters than the hyperexponential components $\boldsymbol{H}_k$. To reduce the number of parameters and therefore the dimension of the optimization problem, we note the following. The SCV of $\boldsymbol{N}$ can be chosen to be $SCV_N \leq 1$ without loss of generality. This is because a PH distribution with $SCV > 1$ is hyperexponential. Setting $SCV_N > 1$ would mean the KPC-PH distribution is hyperexponential [25]. However, setting $SCV_N = 1$ gives a hyperexponential KPC-PH as well. In this case, the first component is lost, but the number of hyperexponential components can be increased from J to $J + 1$ to counteract the lost component.

During the optimization, we reduce the dimension of the search problem by using a gamma distribution, which requires only two parameters. At each iteration, the gamma distribution is approximated with an acyclic phase-type (APH) distribution. This approach is natural since the gamma distribution generalizes the Erlang distribution by allowing a non-integer shape parameter.

Figure 1 shows that a gamma distribution with an SCV as low as .074 is closely matched by an APH distribution of size 10, and is matched perfectly by an APH distribution of size 15. The updated version of (15) that uses a two-parameter gamma distribution is as follows:

$$\begin{aligned} &\min \textstyle\sum_{l=1}^{t} |\widetilde{\pi}_l - \pi_l|/\pi_l \\ &\text{s.t. } \boldsymbol{\alpha}_i \mathbf{1} \leq 1,\ i \in I := \{H_1, \ldots, H_J\} \\ &\quad k, \theta, \boldsymbol{\alpha}_i, \boldsymbol{\lambda}_i \geq \mathbf{0},\ i \in I \end{aligned} \tag{16}$$



(a) CDF (b) Logmoments

Fig. 1. Gamma distribution with $SCV = 0.074$ approximated by APH distributions of increasing size.

where $\widetilde{\boldsymbol{\pi}} = f(k, \theta, \boldsymbol{\alpha}_{H_1}, \boldsymbol{\lambda}_{H_1} \ldots, \boldsymbol{\alpha}_{H_J}, \boldsymbol{\lambda}_{H_J} | \boldsymbol{A})$ and function f is updated to include the conversion of the gamma distribution into an APH distribution before the queue-length probabilities are computed. The parameters (k, θ) correspond to the gamma distribution.

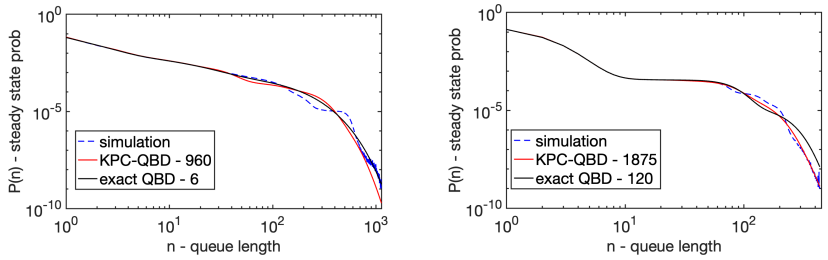
6 Numerical results

We now conduct numerical tests to evaluate the proposed divide-and-conquer fitting algorithm. In our implementation, we used MATLAB and its built-in solver fmincon [23] to solve the optimization problem 16. The numerical tests were run on an Ubuntu machine with an Intel Xeon E5540 CPU having 16 virtual cores and 32GB of RAM.

6.1 Evaluating the fitting algorithm

In this section, we use two real-life Microsoft production server traces, `Radius Authentication` and `BuildServer00`, collected from the RADIUS authentication server and a Microsoft build server, respectively, to evaluate the fitting algorithm outlined in (15) and implemented in MATLAB. In the upcoming figures, we refer to our method as *KPC-QBD*. We first compare *KPC-QBD* to an alternative queue-based fitting method, which we term *exact QBD*. The alternative approach involves using optimization problem 15; however, instead of the divide-and-conquer algorithm we developed, it employs the cyclic-reduction algorithm implemented in QMAM [28] to compute the queue-length probabilities. We also employ other well-established fitting algorithms as benchmarks, all of which directly fit traces using moments of the trace. These fitting algorithms are implemented in the KPC toolbox [16] and include:

– Prony's method, which utilizes characteristic polynomials to exactly match moments of the trace (see, [17, Theorem 1], and [15]).



(a) `Radius Authentication` (b) `Buildserver00`

Fig. 2. Queueing results at 25% utilization when using KPC-QBD and exact QBD methods to fit queueing probabilities.

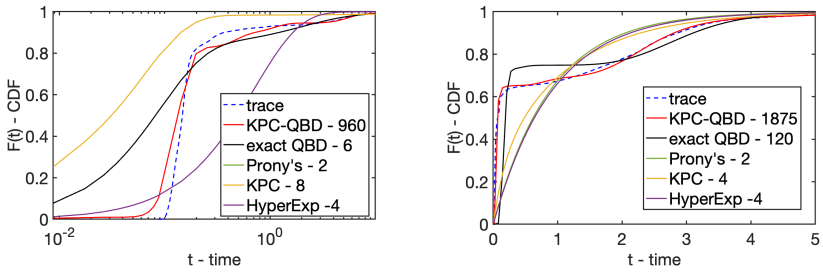
- Hyperexponential parameter fitting, an approximate moment-matching algorithm that uses a non-linear optimization problem to find the hyperexponential distribution that best approximates the first three moments of the empirical trace.
- *KPC-PH* algorithm [16] that uses a divide-and-conquer approach to directly match moments of a trace.

For all Kronecker-composition methods (*KPC-PH*, *KPC-QBD*, and *exact QBD*), the number J of hyperexponential building blocks was experimentally selected by evaluating multiple values of J.

Initially, examining the marginal queue-length probabilities in Figure 2, the difference between *KPC-QBD* and *exact QBD* appears small. Fitting is performed at the 25% utilization level where some improvement is noticeable. We have observed similar result at 50% utilization. However, a significant difference in fitting accuracy becomes evident upon inspecting the empirical Cumulative Distribution Function (CDF) of the traces directly in Figure 3. *KPC-QBD* not only outperforms the exact QBD method but also surpasses all other benchmark methods that directly fit the traces. Noticeably, the sizes of the PH distributions fitted by the *exact QBD* method are significantly smaller compared to those produced by *KPC-QBD*. This size difference is an anticipated limitation of *exact QBD*, attributable to the slow computation of cyclic reduction at each iteration of the optimization problem while evaluating its objective function.

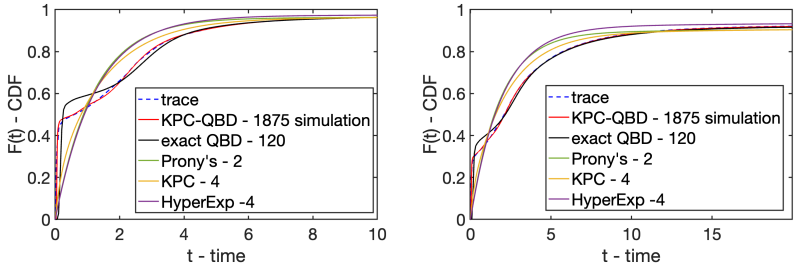
6.2 Response time distributions

To further validate the proposed *KPC-QBD* algorithm, we calculate the response time CDFs at 25% and 50% utilization for both the `Buildserver00` and the `Radius Authentication` traces. There are toolboxes that provide solvers for response time calculations such as [28]. In Figures 4 and 5, the response time CDFs are computed, except for *KPC-QBD* distributions, which are simulated due to their large number of states. It is apparent from these figures that the



(a) `Radius Authentication` (b) `Buildserver00`

Fig. 3. Empirical CDF of traces and the corresponding fitted distributions.

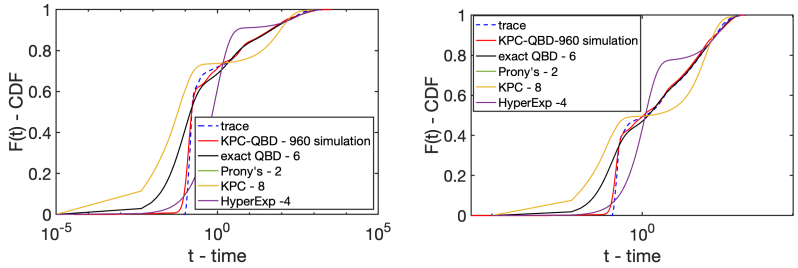


(a) $\rho = 0.25$ (b) $\rho = 0.50$

Fig. 4. `Buildserver00` trace CDF of response time, KPC-QBD and benchmark methods.

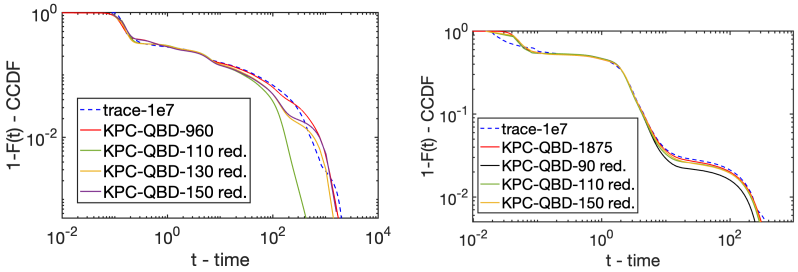
KPC-QBD distributions offer a much closer fit compared to the other methods, including the *exact QBD* distributions.

Motivated by the accuracy of the *KPC-QBD* method, we next reduce the size of its distributions to make response time calculations numerically feasible while preserving accuracy. The reduction is performed using the algorithm described in [11, Algorithm 3, Chapter 5]. A reduced PH distribution size ensures a smoother fit. This smoothing effect occurs because the reduced distribution has fewer degrees of freedom, mitigating any overfitting that may have occurred due to overparameterization. We run simulations of varying lengths to compute the CCDF of response times for the `Buildserver00` and `Radius Authentication` traces. Achieving 99% convergence requires 10^6 iterations, taking about 38 seconds. These results are compared with the original and reduced KPC-QBD distributions in Figure 6. The CCDF emphasizes tail behavior, where reduction errors are most pronounced. For `Buildserver00`, a reduced size of 110 maintains high accuracy and computes the PH response time distribution in 10 seconds, a four times speedup over simulation. For `Radius Authentication`, a size of 130 provides a reasonable estimate in about 27 seconds.



(a) $\rho = 0.25$ (b) $\rho = 0.50$

Fig. 5. `Radius Authentication` trace CDF of response time, KPC-QBD and benchmark methods.



(a) `Radius Authentication` (b) `Buildserver00`

Fig. 6. Response time CCDF at 25% utilization, comparing empirical simulation with original and reduced KPC-QBD distributions.

7 Conclusion

In this paper, we present theoretical results for the efficient approximation of marginal queue-length probabilities for a QBD process with PH-distributed arrival times and compositional PH-distributed service times. We formulate an optimization problem that utilizes this approximation of queue-length probabilities in its objective function and aims to find a QBD process with a KPC-PH service distribution, that closely matches a given set of marginal queue-length probabilities.

Future work could examine the extension of the presented results to a broader class of Markovian processes, such as matrix exponential distributions, Markovian arrival processes, or rational arrival processes. Queueing systems associated with these processes can be studied using matrix analytic methods. Therefore, our results may also be used to fit and analyze these processes.

Acknowledgement

This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101299128. Evgenia Smirni is partially supported by the U.S. National Science Foundation (NSF) grant #2402942. The work was done in part during a visit of the first author to William & Mary, VA, US. We are grateful to Andreas Stathopoulos for constructive comments.

References

1. Ali, A., Pinciroli, R., Yan, F., Smirni, E.: Opt. inference serving on serverless platforms. Proc. VLDB Endow. **15**(10) (2022)
2. Asanjarani, A., Nazarathy, Y., Taylor, P.: A survey of param. and state estimation in queues. Queueing Sys. **97** (2021)
3. Asmussen, S., Nerman, O., Olsson, M.: Fitting phase-type distributions via the EM algorithm. Scan. J. of Stat. (1996)
4. Baglama, J., Reichel, L.: Augmented implicitly restarted lanczos bidiagonalization methods. SIAM Journal on Scientific Computing **27**(1), 19–42 (2005). `https://doi.org/10.1137/04060593X`
5. Banik, A.D., Chaudhry, M.L.: Efficient computational analysis of stationary probabilities for the queueing system BMAP/G/1/N with or without vacation(s). INFORMS J. Comput. **29**(1), 140–151 (2017)
6. Bladt, M.: A review on PH dist. and their use in risk theory. ASTIN Bulletin: The Jour. of the IAA **35**(1), 145–161 (2005)
7. Bladt, M., Esparza, L.J.R., Nielsen, B.F.: Fisher information and statistical inference for phase-type distributions. Journal of Applied Probability **48**(A), 277–293 (2011)
8. Bladt, M., Nielsen, B.F.: Matrix-exponential distributions in applied probability, vol. 81. Springer (2017)
9. Bobbio, A., Horváth, A., Telek, M.: Matching three moments with minimal acyclic phase type distributions. Stochastic models **21**(2-3), 303–326 (2005)
10. Bolch, G., Greiner, S., De Meer, H., Trivedi, K.S.: Queueing networks and Markov chains: modeling and performance evaluation with computer science applications. John Wiley & Sons (2006)
11. Bor, J.: Analytical Methods and Distribution Fitting for Response Times in Markovian Queueing Systems. Ph.D. thesis, Imperial College London, London, UK (2025)
12. Breuer, L., Baum, D.: An Introduction to Queueing Theory and Matrix-Analytic Methods. Springer-Verlag, Berlin, Heidelberg (2006)
13. Buchholz, P.: On a Markovian process algebra. Citeseer (1994)
14. Buchholz, P., Kriege, J., Felko, I.: Input modeling with phase-type distributions and Markov models: theory and applications. Springer (2014)
15. Casale, G., Smirni, E.: KPC-toolbox: fitting markovian arrival processes and phase-type distributions with matlab. SIGMETRICS Perform. Eval. Rev. **39**(4), 47 (apr 2012)
16. Casale, G., Zhang, E.Z., Smirni, E.: KPC-toolbox: Best recipes for automatic trace fitting using Markovian Arrival Processes. Performance Evaluation **67**(9), 873–896 (2010)

17. Casale, G., Zhang, E.Z., Smirni, E.: Trace data characterization and fitting for Markov modeling. Performance Evaluation **67**(2), 61–79 (2010)
18. Ciardo, G., Riska, A., Smirni, E.: Equiload: a load balancing policy for clustered web servers. PEVA **46**(2-3) (2001)
19. Feldmann, A., Whitt, W.: Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. Performance evaluation **31**(3-4), 245–279 (1998)
20. Horváth, A., Vicario, E.: Construction of PH dist. by Bernstein Exponentials. In: Computer Performance Engineering and Stochastic Modelling. pp. 201–215. Springer Nature Switzerland, Cham (2023)
21. Horváth, G.: Moment matching-based distribution fitting with generalized hyper-Erlang distributions. In: Analytical and Stoch. Modeling Tech. and Appl.: 20th Int. Conf., ASMTA 2013, Ghent, Belgium. pp. 232–246. Springer (2013)
22. Kumaran, J., Mitchell, K., Liefvoort, A.: The waiting time distribution of an MEP/MEP/1 queue. In: Proceedings of the 19th International Teletraffic Congress (ITC19). pp. 687–696 (2005)
23. MathWorks: fmincon Documentation Home. MathWorks (2024), `https://uk.mathworks.com/help/optim/ug/fmincon.html`
24. Nasr, W.W., Taaffe, M.R.: Fitting the $\underline{\mathrm{Ph_t}}/\underline{\mathrm{M_t}}/\underline{\mathrm{s}}/\underline{\mathrm{c}}$ time-dependent departure process for use in tandem queueing networks. INFORMS J. Comput. **25**(4), 758–773 (2013)
25. Neuts, M.F.: Matrix-geometric solutions in stochastic models: an algorithmic approach. Courier Corporation (1994)
26. Okamura, H., Dohi, T., Trivedi, K.S.: Improvement of expectation–maximization algorithm for phase-type distributions with grouped and truncated data. Applied Stochastic Models in Business and Industry **29**(2), 141–156 (2013)
27. Pacheco-Sanchez, S., Casale, G., Scotney, B., McClean, S., Parr, G., Dawson, S.: Markovian workload characterization for QoS prediction in the cloud. In: IEEE CLOUD 2011. pp. 147–154. IEEE (2011)
28. Pérez, J.F., Van Velthoven, J., Van Houdt, B.: Q-mam: A tool for solving infinite queues using matrix-analytic methods. In: Proc. of the 3rd Int. Conf. on Perf. Eval. Method. and Tools. p. 9. Brussels, BEL (2008)
29. Rastpour, A., Ingolfsson, A., Sandikçi, B.: Algorithms for queueing systems with reneging and priorities modeled as quasi-birth-death processes. INFORMS J. Comput. **34**(3), 1693–1710 (2022)
30. Reinecke, P., Horváth, G.: PH dists for realistic modelling in discrete-event sim. In: SimuTools. pp. 283–290 (2012)
31. Riska, A., Diev, V., Smirni, E.: An EM-based technique for approximating long-tailed data sets with PH distributions. Performance Evaluation **55**(1), 147–164 (2004)
32. Riska, A., Smirni, E.: M/G/1-type Markov processes: A tutorial. In: Perf. Eval. of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures. Lecture Notes in Comp. Sci., vol. 2459, pp. 36–63. Springer (2002)
33. Riska, A., Smirni, E.: ETAQA solutions for infinite markov processes with repetitive structure. IJOC **19**(2) (2007)
34. Thümmler, A., Buchholz, P., Telek, M.: A novel approach for PH fitting with the EM algorithm. IEEE Trans. Dependable Secur. Comput. **3**(3), 245–258 (2006)
35. Yu, K., Huang, M., Brill, P.H.: An algorithm for fitting heavy-tailed distributions via generalized hyperexponentials. INFORMS J. Comput. **24**(1), 42–52 (2012). `https://doi.org/10.1287/IJOC.1100.0443`, `https://doi.org/10.1287/ijoc.1100.0443`