The Multi-Branched Method of Moments for Queueing Networks

Giuliano Casale
SAP Research
CEC Belfast, UK
giuliano.casale@sap.com

Abstract

*We propose a new exact solution algorithm for closed multiclass product-form queueing networks that is several orders of magnitude faster and less memory consuming than established methods for multiclass models, such as the Mean Value Analysis (*MVA*) algorithm. The technique generalizes the recently proposed Method of Moments (*MoM*) which, differently from* MVA*, recursively computes higher-order moments of queue lengths instead of mean values.*

The main contribution of this paper is to show that the information used in the MoM *recursion can be increased by considering multiple recursive branches that evaluate models with fewer queues. This reformulation allows to define a simpler matrix difference equation for computing normalizing constants which leads to large computational savings with respect to the* MoM *recursion. Computational analysis shows many cases where the proposed algorithm is between* $1,000$ *and* $10,000$ *times faster and less memory consuming than* MoM*, thus extending the range of multiclass models where exact solutions are feasible.*

1. Introduction

Product-form queueing networks [1] are popular stochastic models used in capacity planning of computer architectures and networks with the purpose of evaluating the effect of resource contention on scalability. In many applications, notably modern multi-tier systems, workloads are best described as multiclass, that is, requests are assigned to different categories according to the statistical characteristics of their demand at the different servers. Yet, multiclass workloads are extremely challenging to analyze in queueing networks even using state-of-the-art solution techniques such as Mean Value Analysis (MVA) [22], the Convolution Algorithm [5, 21], RECAL [12], LBANC [9], or more recent methods based on the generating function approach and Monte Carlo sampling [2, 10, 16, 23]. The main problem is that multiclass models typically involve at least four or five classes, hundreds or thousands of competing requests, and many servers, see e.g. [17] for a recent case study. Yet, established exact solution methods require computational costs which are prohibitive for models of this size, e.g., memory requirements can be of the order of many terabytes. As a result, multiclass networks cannot be usually solved with exact techniques and the focus is on approximation methods [8, 13, 24], which yet cannot return probabilistic measures because they ignore the normalizing constant of the Markov chain underlying the queueing network.

Recently, we have proposed the Method of Moments (MoM) [6, 7], a new exact technique for multiclass models that recursively computes higher-order moments of queue length instead of mean values like in the classic MVA approach. The MoM algorithm computes normalizing constants, thus it can also help in evaluating probabilistic measures that cannot be computed by the MVA algorithm. More importantly, the higher-order moments approach of MoM is much more scalable than the MVA approach, since the computational costs increase at most log-quadratically with the total population in the network, whereas they grow exponentially in existing methods such as MVA, RECAL, or LBANC. Although much more efficient than MVA, the MoM approach becomes infeasible if the number of queue and classes grows simultaneously [6], thus models with many classes and many queues can be hard to analyze even with MoM. In order to address this limitation, we propose in this paper an algorithm that generalizes the MoM approach and has lower computational requirements. The proposed approach is always more efficient than MoM in all cases, yet the largest improvements are achieved on models with several queues and many classes which are infeasible in MoM.

Our idea consists in integrating the recursive equation used in the Convolution Algorithm [5, 21] within the MoM approach. MoM jointly considers in a linear system of equations the exact recursive formulas for normalizing constants used in RECAL [12] and LBANC [9], but not those used in

Convolution. By integrating a new formula into MoM we obtain a new computational scheme which recursively evaluates higher-order moments of queue lengths on a sequence of models that differ for the number of queues and jobs. The main advantage of this approach is that the size of the matrix recurrence equation solved at each step of the recursion is much smaller than the one used in the original MoM recursion. This is a fundamental improvement since linear system solution needed at each step of the recursion grows quadratically or cubically with the coefficient matrix order. In particular, we show that even using a multi-branched recursion on hundreds or thousands of models, the generalized MoM is much more efficient than the original MoM which does not consider models with different number of queues.

The remainder of this paper is organized as follows. After giving background in Section 2, we use in Section 3 a simple multiclass model to illustrate MoM and the principles of the generalization proposed in this paper. The analysis of the effects of the multi-branched recursion on models with different number of queues is derived in Section 4, where we give in Theorem 1 and Theorem 2 the main theoretical results of this paper. Computational complexity of the resulting algorithm is analyzed in Section 5. Finally, Section 6 gives conclusions and outlines possible extensions of this paper.

2. Background

We consider a closed product-form queueing network with M distinct[1] queues and R service classes. Jobs are routed probabilistically through the queues where they receive service; after completing service, all jobs re-enter the network with a delay of Z_r units of time which depends on the request's service class $r = 1, \ldots, R$. The mean service demand, i.e., the mean service time multiplied by the mean number of visits [14], of class-r jobs at queue k is indicated with $D_{k,r}$. The number of jobs of class r is the integer N_r; we define $\vec{N} = (N_1, N_2, \ldots, N_R)$ as the population vector of the model and $N = N_1 + N_2 + \ldots + N_R$ is the total number of jobs circulating in the network.

We consider the computation of mean performance indices such as the mean throughput $X_r(\vec{N})$ and the mean response time $R_r(\vec{N}) = N_r/X_r(\vec{N}) - Z_r$ of class-r jobs; additionally, for each queue k and class r, we are interested in computing the utilization $U_{k,r}(\vec{N}) = D_{k,r}X_r(\vec{N})$, the mean queue length $Q_{k,r}(\vec{N})$, and the mean residence times $R_{k,r}(\vec{N}) = Q_{k,r}(\vec{N})/X_r(\vec{N})$. These quantities are uniquely determined if one knows how to compute efficiently throughput and mean queue lengths, which are given by the following ratios [19]:

$$X_r(\vec{N}) = \frac{G(\vec{m}, \vec{N} - \vec{1}_r)}{G(\vec{m}, \vec{N})}, \quad (1)$$

$$Q_{k,r}(\vec{N}) = \frac{D_{k,r}G(\vec{m} + \vec{1}_k, \vec{N} - \vec{1}_r)}{G(\vec{m}, \vec{N})}, \quad (2)$$

where $G(\vec{m}, \vec{N})$ denotes the normalizing constant of the equilibrium state probabilities of the Markov chain underlying the queueing network [15], $\vec{1}_l$ indicates a vector composed by all zeros except for a one in the lth position, and $\vec{m} \equiv (m_1, m_2, \ldots, m_M)$ is the multiplicity vector such that the multiplicity m_k is the number of queues in the model with identical service demands $D_{k,1}, D_{k,2}, \ldots, D_{k,R}$. According to these definitions, e.g., $G(\vec{m} + \vec{1}_k, \vec{N} - \vec{1}_r)$ represents the normalizing constant of a model augmented with an additional copy of queue k and with a job of class r removed. Because of the presence of replicated stations, the total number of queues in the model is $M_{tot} = \sum_{k=1}^{M} m_k$, among which only M have distinct demands.

The advantage of working with normalizing constants instead of mean values is that $G(\vec{m}, \vec{N})$ enables the computation of probabilistic measures that provide fine-grain information about the equilibrium state of the network. For instance, for the case $\vec{m} = (1, 1, \ldots, 1)$ where all queues are distinct, the equilibrium state probabilities can be computed as

$$\Pr(\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_M) = \frac{\prod_{k=1}^{M} C(\vec{n}_k) \prod_{r=1}^{R} D_{k,r}^{n_{k,r}}}{G(\vec{m}, \vec{N})}, \quad (3)$$

where $\vec{n}_k = (n_{k,1}, n_{k,2}, \ldots, n_{k,R})$, being $n_{k,r}$ the number of class-r jobs in queue k in the considered state, $C(\vec{n}_k) = n_k! / \prod_r n_{k,r}!$, and $n_k = n_{k,1} + n_{k,2} + \ldots + n_{k,R}$. Note that quantities like (3) cannot be computed either by the MVA algorithm or by local iterative approximations [8, 13, 24], thus the normalization constant approach considered in this paper is inherently more general than these methods.

2.1. Computational Solution

The analysis of queueing networks can be performed efficiently either by approaches that directly evaluate mean queue lengths and throughputs in a recursive fashion, such as the Mean Value Analysis (MVA) algorithm [22], or by computational methods for the normalizing constant (1) [4]. The normalizing constant approach is usually slightly more efficient than MVA, but it can suffer numerical issues that do not apply to the mean value analysis [18]. From a probabilistic perspective, the MVA algorithm and some methods for the normalizing constant, such as the LBANC algorithm [9], can be interpreted as a recursive evaluation

1 We say that two queues are distinct if there exist at least a class r, $1 \leq r \leq R$, which places different service demands at the two queues.

of *mean* queue lengths[2] over models with different population sizes. Yet, [6, 7] note that recursively evaluating a set of *higher-order moments* of queue lengths can be much more efficient computationally than computing mean values, while still returning the exact solution of the model. The Method of Moments (MoM) [6] is an algorithm that implements this higher-order moment approach and that we generalize for increased efficiency in the next sections; thus we give here a brief overview of the method. Due to limited space and thanks to wide availability of material on the subject, we point to the literature for MVA [22], LBANC [9], RECAL [12], and Convolution [5, 21]; comparative analyses can be found in [3, 6].

2.1.1. Method of Moments (MoM) MoM computes the normalizing constant by simultaneously considering into a linear system of equations the following exact formulas for normalizing constants: the *convolution expression* (CE) [9, 19]

$$G(\vec{m}+\vec{1}_k,\vec{N}) = G(\vec{m},\vec{N}) + \sum_{r=1}^{R} D_{k,r} G(\vec{m}+\vec{1}_k,\vec{N}-\vec{1}_r) \tag{4}$$

for all $1 \leq k \leq M$, and the *population constraint* (PC) [6, 12]

$$N_r G(\vec{m},\vec{N}) = Z_r G(\vec{m},\vec{N}-\vec{1}_r) + \sum_{k=1}^{M} m_k D_{k,r} G(\vec{m}+\vec{1}_k,\vec{N}-\vec{1}_r), \tag{5}$$

for all $1 \leq r \leq R$, which are also the fundamental recurrence relations employed in the LBANC and RECAL algorithms. These recursions are subject to the following termination conditions: (i) $G(\vec{m},\vec{N}) = 0$ if any entry in $\vec{N}$ or $\vec{m}$ is negative; (ii) $G(\vec{0},\vec{0}) = 1$, where $\vec{0} = (0, 0, \ldots, 0)$. In classic algorithms, $G(\vec{m},\vec{N})$ is obtained by recursively evaluating either (4) or (5) until termination conditions are met. Following this approach, time and space requirements grow roughly as $O(N^R)$ if (4) is used (e.g., LBANC) and as $O(N^M)$ if (5) is used (e.g., RECAL). In practice, these costs are often prohibitive since in modeling modern systems it is not rare to have N of the order of hundreds or thousands and $\min\{M, R\} \geq 5$ to 6 queues or classes (see [17] for a recent case study), which make the storage requirement of hundreds of gigabytes regardless of the recursion used.

MoM avoids this memory inefficiency by observing that, if one considers a certain subset of normalizing constants $\vec{V}(\vec{N})$, which we call *basis*[3], then this basis can be computed recursively by jointly using (4) and (5) to define the matrix difference equation

$$\mathbf{A}(\vec{N})\vec{V}(\vec{N}) = \mathbf{B}(\vec{N})\vec{V}(\vec{N}-\vec{1}_R), \tag{6}$$

where $\vec{V}(\vec{0})$ is known from the termination conditions of (4)-(5), and the matrices $\mathbf{A}(\vec{N})$ and $\mathbf{B}(\vec{N})$ are square of identical size. The matrices $\mathbf{A}(\vec{N})$ and $\mathbf{B}(\vec{N})$ are defined by the coefficients of the equations (4)-(5) that relate *all and only* the normalizing constants in $\vec{V}(\vec{N})$ with those in $\vec{V}(\vec{N}-\vec{1}_R)$. The basis is:

$$\vec{V}(\vec{N}) = \{G(\vec{m}',\vec{N}), G(\vec{m}',\vec{N}-\vec{1}_1), \ldots, G(\vec{m}',\vec{N}-\vec{1}_{R-1}) \mid \vec{m}' = \vec{m}+(\delta_1,\ldots,\delta_M),\ R-1 \leq \textstyle\sum_{k=1}^{M}\delta_k \leq R, \delta_k \in \{0,1\}\},$$

which is the set of normalizing constants of models where we have increased the elements of the vector $\vec{m}$ by R or $R-1$ units in all possible ways and where the models are evaluated over the populations $\vec{N}$, $\vec{N}-\vec{1}_1$, …, $\vec{N}-\vec{1}_{R-1}$. The multiplicity increase operation is equivalent to adding new queues to the model and, probabilistically, this can be interpreted as computing binomial moments of queue lengths in the original queueing network [6, 7, 12]; hence one concludes that a recursive computation of $\vec{V}(\vec{N})$ is also a recursive evaluation of higher-order moments of queue length. Indeed, the knowledge of $\vec{V}(\vec{N})$ is sufficient to compute all the normalizing constants used in (1), see [6]; thus, computing $\vec{V}(\vec{N})$ is equivalent to solving the model.

The interest in (6) is that the matrix recursion is linear and does not branch exponentially like (4)-(5), since we can progressively remove the elements of $\vec{N}$ without increasing the size of the $\vec{V}(\cdot)$ vectors until the termination condition $\vec{V}(\vec{0})$ is reached. If the linear system (6) is non-singular, one can compute $\vec{V}(\vec{N}) = \mathbf{A}^{-1}(\vec{N})\mathbf{B}(\vec{N})\vec{V}(\vec{N}-1_R)$ by an exact solution technique, like exact Gaussian elimination or the Wiedemann algorithm[4] which prevent the critical effects of round-off error accumulation when the recursion is evaluated hundreds or thousands of times and also avoid numerical issues arising in normalizing constant computations [7]. If the Wiedemann algorithm is used, the computational cost of linear system solution grows quadratically with the basis size and as $O(N^2 \log N)$ with respect to the total population, which is typically much less than the $O(N^R)$ and $O(N^M)$ of classic methods. An example illustrating the MoM algorithm is given below, together with intuition on the MoM generalization proposed in this work.

2 For normalizing constant methods such as LBANC, the computation focuses on *un-normalized* mean queue lengths [19].

3 The name "basis" stresses that $\vec{V}(\vec{N})$ carries the minimum amount of information needed to perform a recursion that is linear in the total population of the network.

4 See, e.g., the LinBox open source library (`http://www.linalg.org`) for a free implementation of the Wiedemann algorithm, exact Gaussian elimination, and other exact methods that can be used to solve the MoM matrix difference equation.

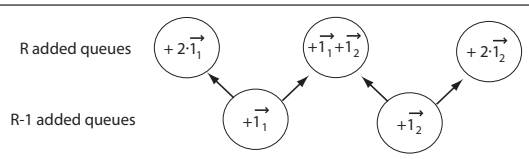


Figure 1. ***Basis of normalizing constants $\vec{V}(\vec{N})$ for a model with $M = 2$ queues and $R = 2$ classes. Each circle represents a group of R normalizing constants of models with populations $\vec{N}$ and $\vec{N}-1_1$. Labels indicate the increase of the multiplicity vector $\vec{m}$ relatively to that subset of normalizing constants.***

3. Motivating Example

We begin by illustrating the structure of (6) on a simple queueing network with $M = 2$ queues, $R = 2$ classes, a population $\vec{N} = (N_1, N_2)$, and where $\vec{m} = (1, 1)$, i.e., all queues are distinct. In order to compact notation, we denote $d_{z,k,s} = (m_k + z) \cdot D_{k,s}$ and $G^{+a,b}_{c,d} = G(\vec{m} + \vec{1}_a + \vec{1}_b, \vec{N} - \vec{1}_c - \vec{1}_d)$. Notations of the type, e.g., $G^{+a,b,d}_{c} = G(\vec{m} + \vec{1}_a + \vec{1}_b + \vec{1}_d, \vec{N} - \vec{1}_c)$ or $G^{+a} = G(\vec{m} + \vec{1}_a, \vec{N})$ are defined similarly and used throughout the paper. From the above definitions, (6) has the following structure:

$$\underbrace{\left[\begin{array}{cccccccccc}
1 & -D_{1,1} & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & -D_{1,1} & \cdot & \cdot & \cdot & \cdot & -1 & \cdot \\
\hline
\cdot & \cdot & 1 & -D_{2,1} & \cdot & \cdot & -1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & -D_{2,1} & \cdot & \cdot & -1 & \cdot \\
\hline
\cdot & -d_{1,1,1} & \cdot & -d_{0,2,1} & \cdot & \cdot & N_1 & -Z_1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & -d_{0,1,1} & \cdot & -d_{1,2,1} & \cdot & \cdot & N_1 & -Z_1 \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & N_2 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & N_2 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & N_2 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & N_2
\end{array}\right]}_{\mathbf{A}(\vec{N})}
\underbrace{\left[\begin{array}{c}
G^{+1,1} \\ G^{+1,1}_1 \\ G^{+1,2} \\ G^{+1,2}_1 \\ G^{+2,2} \\ G^{+2,2}_1 \\ G^{+1} \\ G^{+1}_1 \\ G^{+2} \\ G^{+2}_1
\end{array}\right]}_{\vec{V}(\vec{N})}$$

$$= \underbrace{\left[\begin{array}{cccccccccc}
D_{1,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & D_{1,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & D_{2,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & D_{2,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
d_{1,1,2} & \cdot & d_{0,2,2} & \cdot & \cdot & \cdot & Z_2 & \cdot & \cdot & \cdot \\
\cdot & d_{1,1,2} & \cdot & d_{0,2,2} & \cdot & \cdot & \cdot & Z_2 & \cdot & \cdot \\
\cdot & \cdot & d_{0,1,2} & \cdot & d_{1,2,2} & \cdot & \cdot & \cdot & Z_2 & \cdot \\
\cdot & \cdot & \cdot & d_{0,1,2} & \cdot & d_{1,2,2} & \cdot & \cdot & \cdot & Z_2
\end{array}\right]}_{\mathbf{B}(\vec{N})}
\underbrace{\left[\begin{array}{c}
G^{+1,1}_2 \\ G^{+1,1}_{1,2} \\ G^{+1,2}_2 \\ G^{+1,2}_{1,2} \\ G^{+2,2}_2 \\ G^{+2,2}_{1,2} \\ G^{+1}_2 \\ G^{+1}_{1,2} \\ G^{+2}_2 \\ G^{+2}_{1,2}
\end{array}\right]}_{\vec{V}(\vec{N}-\vec{1}_R)}$$

where $\cdot$ indicates a zero element, and the four blocks of the coefficient matrices represent from top: the CE (4) for $k = 1$, the CE for $k = 2$, the PC (5) for $r = 1$, and the PC for $r = 2$. The basis of normalizing constants is depicted in Figure 1. We remark that, for each element in the figure, the basis includes both normalizing constants for the populations $\vec{N}$ and $\vec{N} - \vec{1}_1$, hence the total number of elements is $card(\vec{V}(\vec{N})) = 10$.

The fundamental observations presented in this paper to improve MoM and, specifically, to considerably reduce the cost of computing $\vec{V}(\vec{N})$, are as follows:

1. we first note that it is possible to add independent equations to the above linear system by taking in consideration a generalization of the convolution expression (4) explained later in the paper; this generalization provides independent information and makes the linear system over-determined.
2. We show that, if the linear system is over-determined, then the basis $\vec{V}(\vec{N})$ can be defined smaller, while still preserving the capability of solving exactly queueing networks. The basis size reduction leads to remarkable computational savings compared to the MoM approach.
3. However, as we explain in Section 4, for models of arbitrary size the additional independent information comes at the price of additional recursions over models with different number of queues. We investigate in the rest of the paper if accepting these additional recursions is convenient with respect to the computational savings implied by the basis size reduction.

The previous observations are further illustrated in the next subsection.

3.1. Improved Computation of the Basis of Normalizing Constants

We begin by observing that (4) can be seen as a specialization of the recursive equation used by the Convolution Algorithm [5, 21], which we call the *general convolution expression* (GCE)

$$G(\vec{m}, \vec{N}) = G(\vec{m} - \vec{1}_k, \vec{N}) + \sum_{r=1}^{R} D_{k,r} G(\vec{m}, \vec{N} - \vec{1}_r), \quad (7)$$

for all $1 \leq k \leq M$. Here queues are removed through the parameter $\vec{m} - \vec{1}_k$, instead of being added as in (4). This implies that a recursion involving (7) may also evaluate models which contain fewer queues than in the original queueing network, while (4) operates on networks with multiplicity $\vec{m}' \geq \vec{m}$ only. However, by instantiating (7) on a model with multiplicity $\vec{m} + \vec{1}_k$ instead of $\vec{m}$, it is found that (7) becomes identical to (4), thus (4) specifies a subset of (7); this also explains the term "general" used in the GCE acronym. Whenever (7) is instantiated on models with fewer queues than in the original network, the information provided by (7) is independent with respect to the one provided by (4), because the two equations are defined over models with different network structure. For example, equation (7) may be added to the linear system of the simple queueing network

considered before if instantiated as

$$G(\vec{m}+2\cdot\vec{1}_1,\vec{N}) = G(\vec{m}+2\cdot\vec{1}_1-\vec{1}_2,\vec{N}) + \sum_{r=1}^{R} D_{2,r}G(\vec{m}+2\cdot\vec{1}_1,\vec{N}-\vec{1}_r). \quad (8)$$

In this case, the normalizing constant $G(\vec{m}+2\cdot\vec{1}_1-\vec{1}_2,\vec{N})$ lies outside the basis $\vec{V}(\vec{N})$, thus equation (8) does not reduce to a CE and provides independent information. Note also that $G(\vec{m}+2\cdot\vec{1}_1-\vec{1}_2,\vec{N})$ is the normalizing constant of a model where queue 2 has been completely removed since we have assumed $\vec{m}=(1,1)$, thus it can be computed easily with closed-form formulas for the balanced network case [20] and therefore the addition of (8) does not increase the number of unknowns in the linear system. *The main idea investigated in this paper is that this independent information can be exploited effectively to reduce the size of the basis* $\vec{V}(\vec{N})$. In fact, consider a new basis $\vec{V}_{new}(\vec{N})$ composed by normalizing constants with $R-2 \leq \sum_k m_k \leq R-1$ additional queues instead of the $R-1 \leq \sum_k m_k \leq R$ as in the original definition of $\vec{V}(\vec{N})$. Then, using (4), (5), and (8), we can define a linear system with square matrix of coefficients

$$\underbrace{\left[\begin{array}{cccccc} 1 & -D_{1,1} & \cdot & \cdot & -1 & \cdot \\ \hline \cdot & \cdot & 1 & -D_{2,1} & -1 & \cdot \\ \hline 1 & -D_{2,1} & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & -d_{0,1,1} & \cdot & -d_{0,2,1} & N_1 & -Z_1 \\ \hline \cdot & \cdot & \cdot & \cdot & N_2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & N_2 \end{array}\right]}_{\mathbf{A}_{new}(\vec{N})} \underbrace{\left[\begin{array}{c} G^{+1} \\ G_1^{+1} \\ G^{+2} \\ G_1^{+2} \\ G \\ G_1 \end{array}\right]}_{\vec{V}_{new}(\vec{N})}$$

$$= \underbrace{\left[\begin{array}{c} \cdot \\ \cdot \\ \hline G^{+1,-2} \\ \hline \cdot \\ \hline \cdot \\ \cdot \end{array}\right]}_{\vec{V}^{-k}_{new}(\vec{N})} + \underbrace{\left[\begin{array}{cccccc} D_{1,2} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & D_{2,2} & \cdot & \cdot & \cdot \\ \hline D_{2,2} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline d_{0,1,2} & \cdot & d_{0,2,2} & \cdot & Z_2 & \cdot \\ \cdot & d_{0,1,2} & \cdot & d_{0,2,2} & \cdot & Z_2 \end{array}\right]}_{\mathbf{B}_{new}(\vec{N})} \underbrace{\left[\begin{array}{c} G_2^{+1} \\ G_{1,2}^{+1} \\ G_2^{+2} \\ G_{1,2}^{+2} \\ G_2 \\ G_{1,2} \end{array}\right]}_{\vec{V}_{new}(\vec{N}-\vec{1}_R)}$$

where the new vector $\vec{V}^{-k}_{new}(\vec{N})$ includes the normalizing constant $G^{+1,-2} \equiv G(\vec{m}+\vec{1}_1-\vec{1}_2,\vec{N})$, and the blocks of the coefficient matrix are from the top: the CE for $k=1$, the CE for $k=2$, the GCE (7), the PC for $r=1$, and the PC for $r=2$. The new linear system may be written compactly as

$$\mathbf{A}_{new}(\vec{N})\vec{V}_{new}(\vec{N}) = V^{-k}_{new}(\vec{N}) + \mathbf{B}_{new}(\vec{N})\vec{V}_{new}(\vec{N}-\vec{1}_R) \quad (9)$$

with square coefficient matrix, thus if $\mathbf{A}^{-1}_{new}(\vec{N})$ exists *the solution of the linear system (9) provides an alternative way to recursively compute normalizing constants that is cheaper than the original linear system (6), since (9) halves the order of the coefficient matrix with respect to (6)*. We stress that without (8) the new system (9) would be underdetermined, thus resorting to the GCE equations is critical for this new approach.

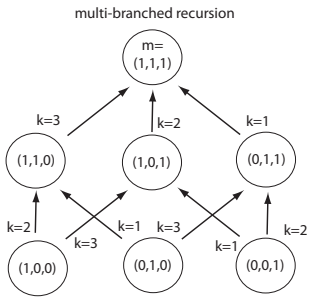
It is also important to remark that, for queueing networks larger than the one considered in this experiment, the normalizing constants in $V^{-k}_{new}(\vec{N})$ may not be available from closed-form expressions. In this case, the computation of $V^{-k}_{new}(\vec{N})$ requires additional recursions over models with different number of queues; we show in the next section that, if multiple equations (7) are used simultaneously, this yields a multi-branched recursive structure for the MoM algorithm, where one needs to evaluate recursively also models with fewer queues that are not considered in the original MoM recursion.

4. The Multi-Branched Method of Moments

The integration of the GCE (7) into the MoM linear system can be done in different ways depending on the number of equations (7) simultaneously instantiated into the matrix difference equation. As observed earlier, integrating GCEs into MoM allows to reduce the basis size; this reduction is given by a decrease in the number of queues added to the multiplicity vectors in the basis, which is equivalent to considering queue length moments of smaller order. Specifically, if the new basis $V_{new}(\vec{N})$ includes models with only $l-1$ and l added queues, one can integrate a single or multiple GCEs *for each model with* l *additional queues only*[5]. A comparison of the recursion trees arising from the two alternatives (single or multiple GCEs) is given in Figure 2.

Using a single GCE implies an additional M recursions which first remove from the model queue M, followed by queue $M-1$, and so forth up to a trivial model with a single queue. Instead, using all possible GCEs implies that the additional recursions first consider *all* possible $\binom{M}{M-1}$ models with $M-1$ queues, followed by *all* possible $\binom{M}{M-2}$ models with $M-2$ queues, and so on up to models with a single queue. The latter approach appears to be the most expensive, at least if one ignores the basis size reduction, because it has a number of new recursions that grows combinatorially instead of linearly. Yet, while limiting to a single GCE seems a natural choice to control the number of new recursions, we have noted that in practice the additional information of the multiple GCEs implies a much larger reduction of the basis than in the case of a single GCE. This in turn provides computational savings often greater than the additional overheads imposed by the extra recursions. Thus, in this section we investigate the trade-off imposed by different types of integrations of GCEs and consider the general case of simultaneously considering up to B, $1 \leq B \leq M$, GCEs in the linear system. We also provide a complexity

5 The GCE is not needed for models with $l-1$ additional queues, since their normalizing constants are all immediately computed from the basis for $\vec{V}(\vec{N}-1_R)$ using the PC of class R.



(a) Single GCE (b) Multiple GCEs

Figure 2. ***Structure of the MoM recursion after addition of a single or multiple GCEs (7) on a model with*** $M = 3$ ***queues. The label*** $k = 1$***, e.g., indicates a GCE instantiated for*** $k = 1$ ***on all models with*** l ***added queues in the redefined basis. Labels within a circle indicate the multiplicity vector*** $\vec{m}$ ***on which the basis is defined, e.g.,*** $(1, 0, 1)$ ***is the model obtained from the original queueing network by removing queue*** 2***.***

analysis to evaluate the best choice of this *branching factor* B as a function of the other model parameters.

4.1. Basis Reduction

We now investigate the reduction of the basis cardinality as a function of the number of GCE equations added to the MoM matrix difference equation. Indeed, the most interesting cases are 1) when (7) is added for a single value of k or 2) when all possible equations in (7) are added; in fact, intermediate cases imply a combinatorial branching of the recursion growing in computational complexity similar to the second case. Let us define a *basis of level* l, $l \geq 1$, as the set

$$\vec{V}_l(\vec{N}) = \{G(\vec{m}', \vec{N}), G(\vec{m}', \vec{N}-\vec{1}_1), \ldots, G(\vec{m}', \vec{N}-\vec{1}_{R-1}) \\ | \, \vec{m}' = \vec{m} + (\delta_1, \ldots, \delta_M), \ \textstyle\sum_{k=1}^{M} \delta_k = l, \delta_k \in \{0, 1\}\},$$

which is the set of normalizing constants with l additional replicated queues. Note that a basis of level l has cardinality $card(\vec{V}_l(\vec{N})) = \binom{M+l-1}{l} R$, thus *a decrease, thanks to the GCEs, of* l *even by a few units implies a quick combinatorial reduction of the number of elements in the basis*. We define $\vec{V}_l^{-k}(\vec{N})$ as a vector similar to $\vec{V}_l(\vec{N})$, but in which all normalizing constants refer to models where $m'_k = m_k - 1$ and $m'_j = m_j$, for $j \neq k$. According to this definition, in MoM the basis is composed by the union of $\vec{V}_R(\vec{N})$ and $\vec{V}_{R-1}(\vec{N})$. In particular, it can be easily shown that the MoM recursion can be rewritten using the new notation as

$$\mathbf{A}_l(\vec{N})\vec{V}_l(\vec{N}) = \mathbf{B}'_{l-1}(\vec{N})\vec{V}_{l-1}(\vec{N}) + \mathbf{B}'_l(\vec{N})\vec{V}_l(\vec{N} - \vec{1}_R),$$

$$N_R\vec{V}_{l-1}(\vec{N}) = \mathbf{C}'_{l-1}(\vec{N})\vec{V}_l(\vec{N} - 1_R),$$

where we have exploited the structure of PCs noting that $\vec{V}_{l-1}(\vec{N})$ is computed immediately from $\vec{V}_{l-1}(\vec{N}-1)$ using the PCs of class R only. Here, $\mathbf{A}_l(\vec{N})$ and $\mathbf{B}'_l(\vec{N})$ are submatrices of $\mathbf{A}(\vec{N})$ and $\mathbf{B}(\vec{N})$ including coefficients of all CEs and of all PCs of classes $r = 1, \ldots, R-1$; conversely $\mathbf{C}'_l(\vec{N})$ includes coefficients of all PCs of class $r = R$. After obtaining $V_{l-1}(\vec{N})$ from the second matrix equation, the above expressions lead to the compact recursion for MoM

$$\mathbf{A}_l(\vec{N})\vec{V}_l(\vec{N}) = \mathbf{B}_l(\vec{N})\vec{V}_l(\vec{N} - \vec{1}_R). \tag{10}$$

The last expression is useful for derivation of the general structure of recursion after addition of GCEs as described in the next fundamental theorems.

Theorem 1. *The inclusion in the MoM matrix difference equation of the GCEs (7) for* $k = 1, \ldots, M$ *on all models having* l *additional queues in the basis* $\vec{V}_l(\vec{N})$ *allows to define a linear system of the type*

$$\mathbf{A}(\vec{N})\vec{V}_l(\vec{N}) = \mathbf{C}(\vec{N})\vec{V}_l^{-k}(\vec{N}) + \mathbf{B}(\vec{N})\vec{V}_l(\vec{N} - \vec{1}_R), \tag{11}$$

which, assuming $\vec{V}_l^{-k}(\vec{N})$ *known, has at least as many equations as unknowns if* $l \geq \max\{1, R - M\}$*. Therefore, the basis has minimum cardinality for* $l = \max\{1, R - M\}$*.*

Proof. A basis of level l has $\binom{M+l-1}{l} R$ normalizing constants, while the total number of CEs and PCs is $\binom{M+l-2}{l-1}(M + R - 1)$ since there exist M CEs and $R - 1$ PCs relating constants in $V_l(\vec{N})$, see [6, 7]. From the above, we have that, in absence of GCEs, the matrix equation is not under-determined if $\binom{M+l-2}{l-1}(M + R - 1) \geq \binom{M+l-1}{l} R$, which is true for all $l \geq R$.

We now add to the previous condition the number of additional GCEs which do not specialize into CEs and that we can formulate for models with l additional queues, which is $\sum_{h=1}^{\min\{M,l\}} \binom{M}{h}\binom{l-1}{l-h}(M-h)$. This can be explained as follows. Consider a model with normalizing constant in $\vec{V}_l(\vec{N})$ and where we have added l queues. Denote by h the number of distinct queues among the l queues we have added. It is possible to see that removing any of these h queues using a GCE involves only normalizing constants with $l-1$ added queues that are already known from the bases $\vec{V}_{l-1}(\vec{N})$ and $\vec{V}_{l-1}(\vec{N} - \vec{1}_R)$ that are computed easily from $\vec{V}_l(\vec{N})$ and $\vec{V}_l(\vec{N} - \vec{1}_R)$, thus these specific GCE equations are identical to CEs already considered in MoM and do not provide independent information. Therefore, for a model with h distinct additional queues, only $M-h$ GCEs are different from the existing CEs. Note that there are $\binom{M}{h}$ ways of choosing the h distinct queues and $\binom{h+(l-h)-1}{l-h} = \binom{l-1}{l-h}$ ways of adding l queues to the model chosen among these h distinct ones under the constraint that each of the h queues is chosen at least once. Combining these expressions gives

the number of GCEs that are not CEs, which simplifies to $\sum_{h=1}^{\min\{M,l\}} \binom{M}{h}\binom{l-1}{l-h}(M-h) = \binom{M+l-2}{l-1}(M+R-1) = \binom{M+l-2}{l}M$, where the first passage follows by Vandermonde convolution [11].

Adding the number of GCEs that are not CEs, we evaluate the following condition for (11) to have more equations than unknowns $\binom{M+l-2}{l}M + \binom{M+l-2}{l-1}(M+R-1)$ $\geq \binom{M+l-1}{l}R$. Suppose first $R > M+1$ and thus $l = \max\{1, R-M\} = R-M$, then we consider the condition $\binom{R-2}{R-M}M + \binom{R-2}{R-M-1}(M+R-1) \geq \binom{R-1}{R-M}R$ which using the property of binomial coefficients $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ on the right hand side gives $\binom{R-2}{R-M}M + \binom{R-2}{R-M-1}(M+R-1) \geq \binom{R-2}{R-M}R + \binom{R-2}{R-M-1}R$ that simplifies to $\binom{R-2}{R-M-1}(M-1) \geq \binom{R-2}{R-M}(R-M)$ which is actually an equality because, after expanding the binomial coefficients, both sides are found identical. Hence, since $l = R-M$ always returns an equality between number of equations and number of unknowns, it is easy to verify that $l < R-M$ would always give an under-determined system and thus $l = R-M$ gives the minimum allowable basis size for the case $R > M+1$.

Consider now the other case $R \leq M+1$ where l takes the minimum possible value $l = \max\{1, R-M\} = 1$, we have then $\binom{M}{1}(M-1) + \binom{M-1}{0}(M+R-1) \geq \binom{M}{1}R$ which is equivalent to $M(M-1)+(M+R-1) \geq MR$ and assuming the worst case $R = M+1$ we get $M(M-1)+2M \geq M(M+1)$ which is always true because the two sides simplify to the same identical value. This means that the linear system is always square if we use the minimum value $l = 1$ when $R \leq M+1$.

We can summarize the above findings saying that $l = \max\{1, R-M\}$ always implies a $\mathbf{A}(\vec{N})$ matrix that is square or over-determined and that smaller values of l instead result in under-determined systems for certain values of M and R. This concludes the proof of the theorem. □

Theorem 2. *The inclusion of a single GCE (7) for given k in the MoM matrix difference equation allows to define a linear system similar to (11), but which has more equations than variables if $l \geq \max\{1, R-1\}$. In particular, the basis has minimum cardinality for $l = \max\{1, R-1\}$.*

Proof. The proof differs from that of Theorem 1 for the number of GCE equations that are not CEs. Suppose that GCEs for given k are used, and assume without loss of generality that the GCE of station $k = M$ is the one included in the matrix difference equation. Then the number of GCEs that are not CEs is

$$\sum_{h=1}^{\min\{M,l\}} \binom{M}{h}\binom{l-1}{l-h} - \sum_{h=1}^{\min\{M-1,l-1\}} \binom{M}{h}\binom{l-1}{l-h},$$

where the left term follows similarly to the number of GCEs in Theorem 1, but for the case where one GCE is added, instead of M, to the models in $\vec{V}_l(\vec{N})$ with l additional queues. The right term counts instead the number of times this GCE is identical to an existing CE. Using Vandermonde convolution on the last expression we have at least as many equations as unknowns if $\binom{M+l-2}{l} + \binom{M+l-2}{l-1}(M+R-1) \geq \binom{M+l-1}{l}R$. Now using $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ on the right hand side we get $\binom{M+l-2}{l} + \binom{M+l-2}{l-1}(M+R-1) \geq \binom{M+l-2}{l}R + \binom{M+l-2}{l-1}R$, which is equivalent to $\binom{M+l-2}{l-1}(M-1) \geq \binom{M+l-2}{l}(R-1)$. Expanding the binomial coefficients it is found that the two sides are identical if $l = \max\{1, R-1\}$ which completes the proof. □

The results in Theorem 1 and Theorem 2 show that: (1) if all GCEs are added to the MoM matrix difference equation, then the basis level can be decreased by up to M units; (2) if a single GCE is used, the basis level can instead be decreased by a single unit. Following the same line of the proofs of Theorem 1 and Theorem 2 it is then straightforward to show the following corollary.

Corollary 1. *If B GCEs, $1 \leq B \leq M$, are added to the matrix difference equation, then the basis can be decreased by up to B levels and the minimal basis size is obtained with the basis level $l = \max\{1, R-B\}$.*

The next section investigates the computational implications of the last result.

5. Computational Complexity

Corollary 1 enables the evaluation of the optimal choice of the *branching factor* B as a function of the model size. In practice, we are interested in understanding when the additional recursions implied by a branching factor B give an overhead that is less that the savings implied by the reduction of the basis level from $l = R$ of the original MoM to $l = \min\{1, R-B\}$ of MoM with GCEs.

We first observe that if B GCEs are used in the MoM linear system, then the basis $\vec{V}_l^{-k}(\vec{N})$ in (11) is computed recursively from B bases of models with a queue less. These models have $M-1$ queues, thus the branching factor in this case is upper bounded by $B \leq M-1$. That is, the maximum number of GCEs added to the linear system changes according to the distance d, $d = 0, \ldots, M-1$, in the recursion tree from the root (i.e., the original model). For $d = 0, \ldots, B-1$, only up to d GCEs can be added to the linear system, while for distances $d = B, \ldots, M$ we can always add B GCEs. This can be seen immediately from Figure 2, where the number of GCE equations instantiated for a model with three queues are three ($k = 1$, $k = 2$, and $k = 3$), two for a model with two queues, and they decrease progressively during the recursion.

Starting from the previous consideration, we analyze below the computational complexity of MoM with GCEs for

the two limit cases $B = 1$ and $B = M$, and provide discussion about the intermediate cases $1 < B < M$ at the end of this subsection.

Time Requirements. If $B = 1$, then $V_l^{-k}(\vec{N})$ is computed by M recursions. During the dth recursive step $d = 0, \ldots, M-1$, the model has $M - d$ queues; the basis is always of level $l = R - 1$ for all steps. Assuming quadratic costs in the solution of the linear system, e.g., using a method like the Wiedemann algorithm, we have that the time for computing $\vec{V}(\vec{N})$ from $\vec{V}(\vec{N} - \vec{1}_R)$ grows as

$$\sum_{d=0}^{M-1} \left(\binom{M-d+R-2}{R-1} R \right)^2 S_{exact}^d, \tag{12}$$

where the term between parentheses is the coefficient matrix order in (11) and

$$S_{exact}^d \approx (N \log(M-d+N)) \binom{M-d+R-2}{R-1} R$$

is the overhead of exact algebra for a model with $M - d$ queues and assuming that the linear system solver uses multiprecision arithmetic [6]. In the expression (12) we have ignored the exact number of iterations of the solution algorithm and thus the expression may be regarded as a cost per iteration of the linear system solver. However, this captures correctly the scalability with respect to the total population N that is the critical source of inefficiencies for existing solution methods.

In the case where we use all possible GCEs, it is $B = M$ at the first recursive step, then $B = M - 1$ at the second step, and $B = M - d$ at the dth recursive step[6]. In addition, the level used at the dth step of the recursion is $l \equiv l(d) = \max\{1, R - M + d\}$, which is thus a function of the distance d from the root of the recursion tree. Following these observations, the time requirements grow as

$$\sum_{d=0}^{M-1} \binom{M}{M-d} \left(\binom{M-d+l(d)-1}{l(d)} R \right)^2 S_{exact}^d$$

where $l(d) = \max\{1, R - M + d\}$ and the term $\binom{M}{M-d}$ accounts for the combinatorial branching of the recursion and is the number of all possible queueing network models with $M - d$ distinct queues chosen among the initial M. For example, when $M < R$

$$\sum_{d=0}^{M-1} \binom{M}{M-d} \left(\binom{R-1}{R-M+d} R \right)^2 S_{exact}^d$$

which is significantly smaller than (12) since the binomial coefficient does not longer depend on the sum of M and R.

Similarly to the case $B = 1$, the time requirements expression is a cost per solver iteration and the term raised to square is the linear system order. Compared to the above expressions, the original MoM algorithm has a time requirement per iteration of

$$\left(\binom{M+R-1}{R} R \right)^2 S_{exact}^d. \tag{13}$$

Figure 3 quantifies the savings per solver iteration of the new algorithm for $B = 1$ and $B = M$ compared to the costs of the original MoM. Since the costs are dependent on M, R, and the population size N, we simplify the evaluation and consider the variation of M and R under a quite large $N = 100$. The cost surfaces indicate that the algorithm with $B = M$ is typically the most efficient except for very low values of M where it is much more expensive than $B = 1$ and the original MoM, although the cost per iteration remains quite small. Overall, the savings of the $B = 1$ case are quite limited compared to the original MoM, while massive cost reduction is achieved with the multi-branched case $B = M$. This is quite counter-intuitive, since one would at first expect that the wide recursion tree in Figure 2(b) is a major source of computational cost compared to the linear recursive structure in Figure 2(a). Yet, Figure 3 indicates that, for multiclass models that can be solved in acceptable times with commonly-available hardware, the cost of the combinatorial branching in Figure 2(b) is not yet a performance bottleneck and it is justified by the massive computational saving of the basis size reduction.

Space Requirements. The space requirement of the case $B = 1$ is upper bounded by the cost of storing the linear system (11) in memory when it is largest, i.e., for the original model with M queues. This is approximately given by

$$2 \left(\binom{M+R-2}{R-1} R \right)^2 + 3 \left(\binom{M+R-2}{R-1} R \right) S_{exact}^{R-1}. \tag{14}$$

The evaluation of memory requirements in the case $B = M$ is similar, but requires to take into account the width of the multi-branched recursion tree, since all basis vectors for models with $k - 1$ queues should be available before the evaluation of models with k queues. Thus, the memory occupation is

$$\max_{d=1,\ldots,M} \binom{M}{M-d} \left(2 \left(\binom{M-d+R-2}{R-1} R \right)^2 + 2 \left(\binom{M-d+R-2}{R-1} R \right) S_{exact} \right), \tag{15}$$

where the first term is the cost of storing $\mathbf{A}(\vec{N})$ and $\mathbf{B}(\vec{N})$ for the currently evaluated linear system, while the second

6 This observation holds true under the assumption that the model is composed initially by queues that have all multiplicity $m_k = 1$, i.e., which are all distinct. The case of models with replicated queues has more favorable computational costs if the total number of queues (including the non-replicated ones) is the same, thus our analysis is a worst-case scenario when M is interpreted as the total number of queues instead of the number of distinct ones.

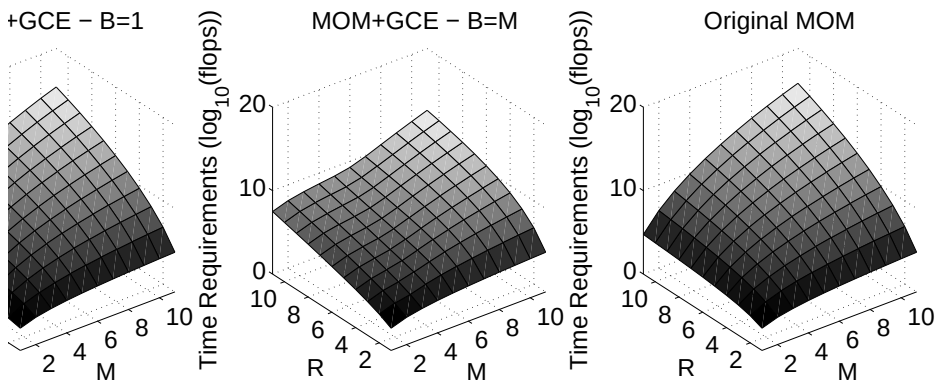


Figure 3. **Time requirements of MoM and the divide-and-conquer MoM for different number of queues M and number of service classes R. All queues are assumed distinct. The results indicate that assuming a branching level $B = M$ is far superior to $B = 1$, unless a small number of queues is considered in the model ($M \leq 4$). The total population in the network is set to $N = 100$.**

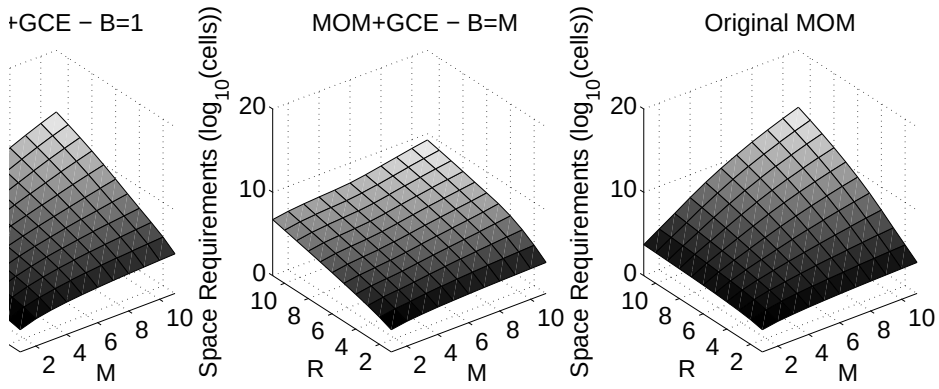


Figure 4. **Space requirements of MoM and the divide-and-conquer MoM for different number of queues M and number of service classes R. The interpretation of the results is qualitatively similar to the one for the time requirements in Figure 3, with the best branching level being $B = M$ unless the number of queues M is small. The total population in the network is set to $N = 100$.**

term accounts for the basis for populations $\vec{N}$ and $\vec{N} - \vec{1}_R$ of all models at distance d from the root of the recursion.

Finally, the computational costs of the original MoM are given by [6]

$$2\left(\binom{M+R-1}{R}R\right)^2 + 3\left(\binom{M+R-2}{R-1}R\right)S^R_{exact}, \tag{16}$$

which is quite similar to the cost of the case $B = 1$.

The comparison of the space requirements of the three different methods is shown in Figure 4 for different values of M and R; we set again the total population to $N = 100$. Results are qualitatively similar to the time requirement case: the GCE equations provide the largest savings in space requirements compared to the original MoM only if $B = M$. The case $B = 1$ is 1 to 2 orders of magnitude faster then the original MoM for models with few queues ($M \leq 4$), while as M increases the algorithm with $B = M$ scales much better. In particular, for the most challenging model with $M = 11$ and $R = 11$, the computational saving of the modified algorithm with $B = M$ is about four orders of magnitude over the original MoM, thus making the case that the inclusion of the GCE equations is highly-valuable also for the space requirements.

Intermediate cases $1 < B < M$. Following the result in Corollary 1 it is immediately found that the size of the basis for intermediate choices of the branching level B is always bounded by the choices $B = 1$ and $B = M$ and computa-

tional requirements are typically within those of these limit cases. For example, assume that B queues are chosen for removal and the multi-branched recursion is operated only on these queues such that the recursion is terminated by solving with the original MoM models with $M - B$ queues. In this case, we have found that the computational costs of the choices $B = 1$ and $B = M$ are always better than these intermediate cases, unless $M - B = 1$. Therefore the savings of these intermediate cases seem marginal and do not motivate a specialized implementation of the algorithm. As a result, we believe that the multi-branched recursion approach is best implemented with a choice $B = M$ which provides the biggest savings with respect to the original MoM on the largest number of choices of M and R.

6. Conclusions

In this paper, we have presented a generalization of the Method of Moments (MoM), a recently proposed algorithm for the exact analysis of multiclass queueing network models which are widely used in capacity planning of computer systems and networks [6, 7]. We have integrated in the MoM equations also the recursive formula used in the Convolution Algorithm [5, 21], here called the general convolution equation (GCE). We have shown that using the GCE in MoM significantly changes the structure of its recursion leading to the evaluation of models with different number of queues, which can be solved much more efficiently than the larger models considered by MoM. As a result, the computational costs in time and space of the generalized algorithm are several orders of magnitude smaller than the original MoM recursion. Future work will focus on a similar extension for the CoMoM algorithm presented in [7][7].

References

[1] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *JACM*, 22(2):248–260, 1975.

[2] A. Bertozzi and J. McKenna. Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review*, 35(2):239–268, 1993.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. Wiley and Sons, 1998.

[4] S. C. Bruell and G. Balbo. *Computational Algorithms for Closed Queueing Networks*. North-Holland, 1980.

[5] J. P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Comm. of the ACM*, 16(9):527–531, 1973.

[6] G. Casale. An efficient algorithm for the exact analysis of multiclass queueing networks with large population sizes. In *Proc. of joint ACM SIGMETRICS/IFIP Performance*, pages 169–180. ACM Press, 2006.

[7] G. Casale. CoMoM: Efficient class-oriented evaluation of multiclass performance models. *IEEE Trans. on Software Engineering*, to appear in 2009.

[8] K. M. Chandy and D. Neuse. Linearizer: A heuristic algorithm for queuing network models of computing systems. *Comm. of the ACM*, 25(2):126–134, 1982.

[9] K. M. Chandy and C. H. Sauer. Computational algorithms for product-form queueing networks models of computing systems. *Comm. of the ACM*, 23(10):573–583, 1980.

[10] G. L. Choudhury, K. K. Leung, and W. Whitt. Calculating normalization constants of closed queuing networks by numerically inverting their generating functions. *JACM*, 42(5):935–970, 1995.

[11] D. J. A. Cohen. *Basic Techniques of Combinatorial Theory*. John Wiley and Sons, 1978.

[12] A. E. Conway and N. D. Georganas. RECAL - A new efficient algorithm for the exact analysis of multiple-chain closed queueing networks. *JACM*, 33(4):768–791, 1986.

[13] P. Cremonesi, P. J. Schweitzer, and G. Serazzi. A unifying framework for the approximate solution of closed multiclass queuing networks. *IEEE Trans. on Computers*, 51:1423–1434, 2002.

[14] P. J. Denning and J. P. Buzen. The operational analysis of queueing network models. *ACM Computing Surveys*, 10(3):225–261, 1978.

[15] W. J. Gordon and G. F. Newell. Closed queueing systems with exponential servers. *Oper. Res.*, 15(2):254–265, 1967.

[16] P. G. Harrison and S. Coury. On the asymptotic behaviour of closed multiclass queueing networks. *Performance Evaluation*, 47(2):131–138, 2002.

[17] S. Kounev and A. Buchmann. Performance modeling and evaluation of large-scale j2ee applications. In *Proc. of CMG Conference*, pages 273–283, 2003.

[18] S. Lam. Dynamic scaling and growth behavior of queueing network normalization constants. *JACM*, 29(2):492–513, 1982.

[19] S. Lam. A simple derivation of the MVA and LBANC algorithms from the convolution algorithm. *IEEE Trans. on Computers*, 32:1062–1064, 1983.

[20] D. Mitra and J. McKenna. Asymptotic expansions for closed markovian networks with state-dependent service rates. *JACM*, 33(3):568–592, July 1985.

[21] M. Reiser and H. Kobayashi. Queueing networks with multiple closed chains: Theory and computational algorithms. *IBM J. Res. Dev.*, 19(3):283–294, 1975.

[22] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *JACM*, 27(2):312–322, 1980.

[23] K. W. Ross, J. Wang. Implementation of Monte Carlo Integration for the Analysis of Product-Form Queueing Networks. *Performance Evaluation*, 273–292, May 1997.

[24] P. J. Schweitzer. Approximate analysis of multiclass closed networks of queues. In *Proc. of the Int'l Conf. on Stoch. Control and Optim.*, pages 25–29, Amsterdam, 1979.

7 The author wishes to thank the anonymous QEST reviewers for detailed comments on an earlier version of this paper that greatly helped in improving presentation quality.