AN ALGORITHM FOR CLOSED QUEUEING NETWORKS BASED ON NUMERICAL TRANSFORM INVERSION

Gagan L. Choudhury,[1] *Kin K. Leung and Ward Whitt*

AT&T Bell Laboratories

December 13, 1993

ABSTRACT

We propose a new algorithm for closed queueing networks and related product-form models based on numerical inversion of the generating function of the normalization constant (or partition function). It is known that the generating function of the normalization constant often has a remarkably simple form, but numerical inversion evidently has not been considered before. We also show that moments of steady-state distributions can be calculated directly by only performing two inversions. For closed queueing networks with p closed chains, the generating function is p dimensional. For these generating functions, the algorithm recursively performs p one-dimensional inversions. The required computation grows exponentially in the dimension, but we show that the dimension can often be reduced dramatically by exploiting special structure. Other key ingredients in the algorithm are scaling and the computation of large sums efficiently by Euler summation. Numerical examples indicate that this new algorithm can usefully complement previous algorithms.

1. Introduction

Closed queueing networks have played a major role in the performance analysis of computer systems, communication systems and other complex systems. The success of these models is largely due to the effective recursive algorithms that have been developed for computing the difficult normalization constant (or partition function), such as the convolution algorithm, MVA and RECAL, which are reviewed in Lavenberg [8] and Conway and Georganas [7]. While these recursive algorithms have been very successful, they do encounter difficulties when the model becomes large in one way or another. Thus, special approaches for analyzing large closed networks also have been developed, such as the one based on asymptotic expansions of integral representations in McKenna and Mitra [9].

In this paper we propose a radically different algorithm for calculating the performance measures of closed queueing networks and related product-form models, which we believe usefully complements existing algorithms, because it applies to both large and small models. In contrast to the recursive approach of the non-asymptotic algorithms, we directly calculate the difficult normalization constant at a desired argument (total population vector) by numerically inverting its generating function. Moreover, we directly calculate mean queue lengths by performing only two inversions.

[1] Speaker: AT&T Bell Laboratories, Room 3K-603, Holmdel, NJ 07733-3030 (gagan @buckaroo.att.com, 908-949-4028).

Our algorithm depends upon having a convenient expression for the generating function of the normalization constant and an effective numerical inversion algorithm. Generating functions of normalization constants are discussed in Bertozzi and McKenna [2], but it has long been known that these generating functions can be useful; see Reiser and Kobayashi [10]. For numerical inversion, we rely on the LATTICE-POISSON algorithm in Abate and Whitt [1] as extended in Choudhury, Lucantoni and Whitt [6].

In the present paper we give a concise account of our algorithm for closed queueing networks; an expanded discussion appears in [3]. The algorithm also applies to other product-form models. We treat the special cases of circuit-switched communication network models and resource-sharing models in [4,5]. The basic algorithm is the same for these other models; in [4,5] we show that the generating functions again have relatively simple expressions and we develop appropriate scaling algorithms.

2. Multi-Chain Networks with Only SS and IS Queues

We consider multi-chain closed queueing networks with only single-server (SS) and infinite-server (IS) queues. An SS queue is often referred to as a service center with load-independent service rate.

In describing the model, to a large extent we follow Bertozzi and McKenna [2]. Let p denote the number of *closed chains* and let $M \geq p$ be the number of *job classes.* Let N be the number of queues, with the understanding that queues $1, \ldots, q$ are SS, while queues $q+1, \ldots, N$ are IS. As usual, the SS queues must have one of the product-form disciplines such as processor sharing. In the case of first-come first-served (FCFS), the service times of all job classes at that queue are assumed to have an exponential distribution with a common mean. We call the pair (r,i) a *stage* where r is a job class and i is a queue. Let C_j be the set of stages associated with the j^{th} closed chain. Let K_j be the fixed number of jobs in the j^{th} closed chain and let $\mathbf{K} \equiv (K_1, \ldots, K_p)$ be the *total population vector.* Let n_{ri} be the number of jobs of class r at queue i and let $\mathbf{n} \equiv (n_{ri};\ 1 \leq r \leq M,\ 1 \leq i \leq N)$ be the *job vector,* which is the state variable. Let $n_i = \sum_{r=1}^{M} n_{ri}$. Let $S(\mathbf{K})$ be the *state space* of allowable job vectors, i.e.,

$$S(\mathbf{K}) = \{\mathbf{n}\colon n_{ri} \in \mathbf{Z}^+ \quad \text{and} \sum_{(r,i) \in C_j} n_{ri} = K_j\ ,\ 1 \leq j \leq p\} \tag{2.1}$$

where $\mathbf{Z}^+$ is the set of nonnegative integers. Let e_{ri} be the *visit ratio* of stage (r,i), obtained from solving the traffic rate equations using the routing matrix (which we do not introduce explicitly). let t_{ri} be the *mean service time* for class r at queue i and let $\rho'_{ri} = t_{ri} e_{ri}$ be the *relative traffic intensities.*

With this notation, the steady-state probability mass function is

$$p(\mathbf{n}) = g(\mathbf{K})^{-1} f(\mathbf{n})\ , \tag{2.2}$$

where the *normalization constant* or *partition function* is

$$g(\mathbf{K}) = \sum_{n \in S(\mathbf{K})} f(\mathbf{n}) \tag{2.3}$$

and

$$f(\mathbf{n}) = \left[\prod_{i=1}^{q} n_i! \prod_{r=1}^{M} \frac{\rho'^{\,n_{ri}}_{ri}}{n_{ri}!}\right]\left[\prod_{i=q+1}^{N} \prod_{r=1}^{M} \frac{\rho'^{\,n_{ri}}_{ri}}{n_{ri}!}\right]. \tag{2.4}$$

As shown in (2.25) of Bertozzi and McKenna [2], the generating function of $g(\mathbf{K})$ has a remarkably simple form. Allowing for multiplicities in the denominator factors, it is

$$G(\mathbf{z}) \equiv \sum_{K_1 = 0}^{\infty} \cdots \sum_{K_p = 0}^{\infty} g(\mathbf{K}) \prod_{j=1}^{p} z_j^{K_j} = \frac{\exp\left[\sum_{j=1}^{p} \rho_{j0} z_j\right]}{\prod_{i=1}^{q'} \left[1 - \sum_{j=1}^{p} \rho_{ji} z_j\right]^{m_i}} , \qquad (2.5)$$

where $m_1 + \ldots + m_{q'} = q$ and ρ_{ji} are *aggregate relative traffic intensities*, i.e.,

$$\rho_{j0} = \sum_{i=q+1}^{N} \sum_{(r,i) \in C_j} \rho'_{ji} \quad \text{and} \quad \rho_{ji} = \sum_{(r,i) \in C_j} \rho'_{ri} \ , \ 1 \le i \le q \ , \qquad (2.6)$$

We calculate the normalization constant $g(\mathbf{K})$ by numerically inverting $G(\mathbf{z})$ in (2.5).

We can also calculate moments by preforming only two inversions. We start with the known expression for the mean of the number, say q_{1i}, of chain 1 customers at queue *i*, namely,

$$E[q_{1i}] = \sum_{k=1}^{K_1} \rho_{1i}^{k} g(\mathbf{K} - k\mathbf{1}_j)/g(\mathbf{K}) \qquad (2.7)$$

where $\mathbf{1}_j$ is the vector of 0's except a 1 in the j^{th} place; e.g., see (3.258) of Lavenberg [8].

We rewrite (2.7) as

$$E[q_{1i}] = \rho_{1i}^{K_1} \frac{h(\mathbf{K})}{g(\mathbf{K})} - 1 \ , \qquad h(\mathbf{K}) = \sum_{k=0}^{K_1} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \qquad (2.8)$$

and $\mathbf{K}_2 \equiv (K_2, \ldots, K_p)$. We then see that the generating function of $h(\mathbf{K})$ is

$$H(\mathbf{z}) \equiv \sum_{K_1 = 0}^{\infty} \cdots \sum_{K_p = 0}^{\infty} h(\mathbf{K}) \prod_{j=1}^{p} z_j^{K_j} = \frac{G(z_1/\rho_{1i}, z_2, \ldots, z_p)}{1 - z_1} \ . \qquad (2.9)$$

We thus can calculate $E[q_{1i}]$ via (2.8) by two inversions, one to calculate $g(\mathbf{K})$ and the other to calculate $h(\mathbf{K})$. The same approach can also be used for higher moments [3].

3. Dimension Reduction

Our approach to inverting the *p*-dimensional generating functions is to recursively perform *p* one-dimensional inversions. In general, the computational complexity is exponential in the dimension *p*, but a dramatic reduction in computation often occurs due to special structure if we perform the one-dimensional inversions in a good order.

We look for *conditional decomposition*. We select *d* variables which we are committed to invert. We then look at the generating function with these *d* variables fixed, and we write the function of the remaining $p - d$ variables as a product of factors, where no two factors have any variables in common. The maximum dimension of the additional inversion required beyond the designated *d* variables is equal to the maximum number of the $p - d$ remaining variables appearing in one of the factors, say *m*. The overall inversion can then be regarded as being of dimension $d + m$. The idea, then, is to select an appropriate *d* variables, so that the resulting dimension $d + m$ is small.

To carry out this dimension reduction, we exploit the representation of the generating function $G(\mathbf{z})$ as a product of factors, i.e.,

$$G(\mathbf{z}) = \prod_{i=1}^{m} \hat{G}_i(\hat{\mathbf{z}}_i) \tag{3.1}$$

where $m \geq 2$ and $\hat{\mathbf{z}}_i$ is a subset of $\{z_1, z_2, \ldots, z_p\}$. (Here the same variable z_j may appear in more than one factor.) We assume that each $\hat{G}_i(\hat{\mathbf{z}}_i)$ cannot be further factorized into multiple factors, unless at least one of the latter is a function of all variables in the set $\hat{\mathbf{z}}_i$.

We now represent the conditional decomposition problem as a graph problem. We construct a graph, called an *interdependence graph*, to represent the interdependence of the variables z_k in the factors. We let each variable z_k be represented by a node in the graph. For each factor $G_i(\hat{\mathbf{z}}_i)$ in (3.1), form a fully connected subgraph Γ_i by connecting all nodes in the set $\hat{\mathbf{z}}_i$. Then let $\Gamma = \bigcup_{i=1}^{m} \Gamma_i$.

Now for any subset D of Γ, we identify the *maximal connected subsets* $S_i(D)$ of $\Gamma - D$; i.e., $S_i(D)$ is connected for each i, $S_i(D) \cap S_j(D) = \emptyset$ when $i \neq j$ and $\bigcup_i S_i(D) = \Gamma - D$. Let $|A|$ be the cardinality of the set A. Then the dimension of the inversion resulting from the selected subset D is

$$\text{inversion dimension} = |D| + \max_i\{|S_i(D)|\} \ . \tag{3.2}$$

For the small-to-moderate number of variables that we typically encounter, we can choose a subset D to minimize (3.2) by inspection or by enumeration of the subsets of Γ in increasing order of cardinality. Since our overall algorithm is likely to have difficulty if the reduced dimension is not relatively small (e.g., ≤ 10), it is not necessary to consider large sets D in (3.2).

4. The Basic Algorithm

Given the p-dimensional generating function $G(\mathbf{z})$, we first do the dimension reduction analysis to determine the order of the variables to be inverted. Given that the order has been specified, we perform (up to) p one-dimensional inversions recursively.

To represent the recursive inversion, we define *partial generating functions* by

$$g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_j=0}^{\infty} g(\mathbf{K}) \prod_{i=1}^{j} z_i^{K_i} \quad \text{for } 1 \leq j \leq p \ , \tag{4.1}$$

where $\mathbf{z}_j = (z_1, z_2, \ldots, z_j)$ and $\mathbf{K}_j = (K_j, K_{j+1}, \ldots, K_p)$ for $1 \leq j \leq p$. Let $\mathbf{z}_0$ and $\mathbf{K}_{p+1}$ be null vectors. Clearly, $\mathbf{K} = \mathbf{K}_1, \mathbf{z} = \mathbf{z}_p$, $g^{(p)}(\mathbf{z}_p, \mathbf{K}_{p+1}) = G(\mathbf{z})$ and $g^{(0)}(\mathbf{z}_0, \mathbf{K}_1) = g(\mathbf{K})$.

Let I_j represent inversion with respect to z_j. Then the step-by-step nested inversion approach is

$$g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j) = I_j[g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1})] \ , \quad 1 \leq j \leq p \ , \tag{4.2}$$

starting with $j = p$ and decreasing j by 1 each step. In the actual program implementation, we attempt the inversion shown in (4.2) for $j = 1$. In order to compute the righthand side we need another inversion with $j = 2$. This process goes on until at step p the function on the righthand side becomes the p-dimensional generating function and is explicitly computable.

In each step we use the LATTICE-POISSON inversion algorithm in [1] with modifications to improve precision and allow for complex inverse function as in [6]. We show below the inversion formula at the j^{th} step. For simplicity, we suppress those arguments which remain constant during this inversion, letting $g_j(K_j) = g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j)$ and $G_j(z_j) = g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1})$. With this notation, the inversion formula (4.2) is

$$g_j(K_j) = \frac{1}{2l_jK_jr_j^{K_j}} \sum_{k_1=0}^{l_j-1} e^{-\frac{\pi i k_1}{l_j}} \sum_{k=-K_j}^{K_j-1} (-1)^k G_j(r_j e^{\frac{\pi i(k_1+l_jk)}{l_jK_j}}) - e_j , \tag{4.3}$$

where $i = \sqrt{-1}$, l_j is a positive integer and e_j represents the *aliasing error*, which is given by

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2nl_jK_j) r_j^{2nl_jK_j} . \tag{4.4}$$

Note that, for $j = 1$, $g_1(K_1) = g(\mathbf{K})$ is real, so that $G_1(\overline{z}_1) = \overline{G_1(z_1)}$. This enables us to cut the computation in (4.3) by about one half [3].

To control the aliasing error, we choose $r_j = 10^{-a_j}$ for $a_j = \gamma_j / 2l_jK_j$. Then (4.4) becomes

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2nl_jK_j) 10^{-\gamma_j n} . \tag{4.5}$$

As is clear from (4.5), a bigger γ_j decreases the aliasing error. Also, as explained in [6], the parameter l_j controls roundoff error, with bigger values causing less roundoff error. An inner loop of the inversion requires more accuracy than an outer loop since the inverted values in an inner loop are used as transform values in an outer loop. With a goal of about eight significant digit accuracy, the following sets of l_j and γ_j typically are adequate: i) $l_1 = 1$, $\gamma_1 = 11$, ii) $l_2 = l_3 = 2$, $\gamma_2 = \gamma_3 = 13$, iii) $l_4 = l_5 = l_6 = 3$, $\gamma_4 = \gamma_5 = \gamma_6 = 15$, assuming that computations are done using double-precision arithmetic. It is usually not a good idea to use the same l_j for all *j*, because then more computation is done to achieve the same accuracy.

In [1,6] the inverse function was mainly assumed to be a probability, so that the aliasing error e_j in (4.5) could be easily bounded. In contrast, here the normalization constants may be arbitrarily large and therefore the aliasing error e_j may also be arbitrarily large. Thus, we scale the generating function in each step by defining a *scaled generating function* as

$$\overline{G}_j(z_j) = \alpha_{0j} G_j(\alpha_j z_j) , \tag{4.6}$$

where α_{0j} and α_j are positive real numbers. We invert this scaled generating function after choosing α_{0j} and α_j so that the errors are suitably controlled.

5. Scaling

The most difficult aspect of the numerical inversion is choosing the scaling parameters in (4.6). A general scaling strategy is described in Section 2.2 of [3] and a detailed scaling algorithm for the class of closed queueing networks considered here is developed in Section 5 of [3]. Here we describe the resulting scaling algorithm. From this quick description, it should be clear that the scaling is not difficult to implement. We set

$$\alpha_{0j} = e^{-\alpha_j \rho_{j0}} \quad \text{and} \quad \alpha_j = \underset{i}{Min} \left\{ \frac{K_j}{\rho_{j0}} , \frac{a_{ij}}{\overline{\rho}_{ji}} \right\} , \tag{5.1}$$

where

$$a_{ij} = \left[\sum_{l=1}^{N_{ij}} \frac{K_j + l}{K_j + 2l_jK_j + l} \right]^{1/2l_jK_j} , \quad N_{ij} = \overline{m}_i - 1 + \sum_{k=j+1}^{p} K_k \eta_{ki} , \tag{5.2}$$

$$\overline{\rho}_{ji} = \frac{1}{i} \sum_{k=1}^{i} \tilde{\rho}_{rj} , \quad \tilde{\rho}_{ji} > 0 , \qquad \overline{m}_i = \sum_{k=1}^{i} \tilde{m}_k , \tag{5.3}$$

with $\overline{\rho}_{ji} = 0$ if $\tilde{\rho}_{ji} = 0$, $\eta_{ki} = 1$ if $\rho_{ki} \neq 0$ and 0 otherwise, $\{\tilde{\rho}_{ji} : 1 \leq i \leq q'\}$ being a sorted version in decreasing order of magnitude of $\{\rho_{ji}/(1-\sum_{k=1}^{j-1}\rho_{ki}|z_k|) : 1 \le i \le q'\}$ and $\{\tilde{m}_i : 1 \le i \le q'\}$ is the rearranged version of $\{m_i : 1 \le i \le q\}$ associated with $\{\tilde{\rho}_{ji}\}$.

6. Other Algorithm Features

The inversion algorithm given by (4.2) and (4.3) with scaling (4.6) is really a family of algorithms, one for each vector $\mathbf{l} \equiv (l_1, \ldots, l_p)$. Our experience is that there exists a minimum vector $\mathbf{l}_{\min} \equiv (l_{1,\min}, \ldots, l_{p,\min})$ such that the algorithm will be sufficiently accurate whenever $\mathbf{l} \ge \mathbf{l}_{\min}$. However, the required computation increases as $\mathbf{l}$ increases, so we do not want $\mathbf{l}$ larger than necessary. Typically $l_{1,\min} = 1$, $l_{2,\min} = l_{3,\min} = 2$, $l_{4,\min} = l_{5,\min} = l_{6,\min} = 3$, etc.

In addition to being a means to achieve greater accuracy, we use the vectors $\mathbf{l}$ to verify the accuracy. If we run the algorithm with $\mathbf{l}_A$ and $\mathbf{l}_B$, where $\mathbf{l}_{A,}\ \mathbf{l}_B \ge \mathbf{l}_{\min}$, $\mathbf{l}_A \ne \mathbf{l}_B$ and the answers agree up to t significant places (with large t, say 6 or higher), then we claim that both answers are almost surely accurate up to t significant places.

In [1,6] the Euler transformation (or summation) is used to calculate infinite series arising in the inversion of Laplace transforms. We also use Euler summation here to speed up the calculation of large finite series in (4.3). Consider partial sums

$$S_j = \sum_{k=0}^{j} (-1)^k a_k \ . \tag{6.1}$$

Euler summation approximates S_∞ by

$$E(m,n) = S_n + (-1)^{n+1} \sum_{k=0}^{m-1} (-1)^k 2^{-(k+1)} \Delta^k a_{n+1} = \sum_{k=0}^{m} \begin{bmatrix} m \\ 2 \end{bmatrix} 2^{-k} S_{n+k} \ , \tag{6.2}$$

where Δ is the finite-difference operator. We use the Euler sum on the inner sums in (4.3) whenever K_j is large. We typically use $n = 11$ and $m = 20$ or 40.

7. A Numerical Example

In this section we give numerical results for our algorithm applied to one closed queueing network example. Other examples are discussed in [3]. We calculate the normalization constant $g(\mathbf{K})$ in (2.3) for specified population vectors $\mathbf{K}$ from the generating function $G(\mathbf{z})$ in (2.5). Thus the parameters are the number of chains, p, the number of distinct single-server queues, q', the multiplicities m_i, the aggregate relative traffic intensities ρ_{ji}, $1 \le j \le p$, $0 \le i \le q'$, and the desired population vector $\mathbf{K}$.

Note that the normalization constant $g(\mathbf{K})$ only depends on these parameters p, q', m_i, ρ_{ji} and $\mathbf{K}$. Hence, we do not fully specify the model. In particular, we do not give the routing probabilities or the mean service times. Thus, there are many detailed models consistent with our partial model specifications. One possible routing matrix consistent with the data that we provide is a cyclic routing matrix, all of whose entries are 0's and 1's, which yields visit ratios $e_{ri} = 1$ for all stages (r,i). If we consider this case, then $t_{ri} = \rho'_{ri}$ and the throughputs coincide with the normalization constant ratios $g(\mathbf{K}-\mathbf{1}_j)/g(\mathbf{K})$. We display some of these ratios along with the values of $g(\mathbf{K})$ in our numerical results below.

The total number of single-server queues is $q = 50$. However, we consider $q' = 10$ distinct queues, each with multiplicity $m_i = 5$. This reduces the computational complexity for our algorithm, but not for other algorithms. We also allow for an arbitrary number of IS queues, with almost no additional computational complexity. Our example has $p = 11$ chains, but the inversion dimension can be reduced from eleven to two. The numerical results are shown to seven significant figures, which is more than adequate for most applications. However the realized accuracy was found to be 8-12 significant places. This was determined by using two sets of l_j values for each case. Each number below is expressed as *aek*, which means $a \times 10^k$. Even with the scaling, some of the normalization constants assume values outside the range of computation on the computer. This difficulty is circumvented by working with logarithms [3]. We used Euler summation whenever K_j exceeds 31, which resulted in the computation of 31 terms. Our experience is that Euler summation works consistently, providing tremendous computational savings whenever a particular K_j is large.

For this example, the aggregate relative traffic intensities are:

$$\begin{aligned} \rho_{j0} &= 5j - 10 \text{ for } j = 2,3,\ldots,11 , \\ \rho_{jj} &= 0.1(j-1) \quad \text{for} \quad j = 2,\ldots,11 , \\ \rho_{1j} &= 1 + 0.1(j-1) \quad \text{for} \quad j = 2,\ldots,11 , \end{aligned} \tag{7.1}$$

with $\rho_{ji} = 0$ for all other (j,i). The generating function for this example is

$$G(\mathbf{z}) = \exp\left[\sum_{j=1}^{p} \rho_{j0} z_j\right] \Big/ \prod_{i=2}^{p} \left[1 - \rho_{ii} z_i - \rho_{1i} z_1\right]^{m_i} \tag{7.2}$$

and the interdependence graph is given in Figure 1 below.

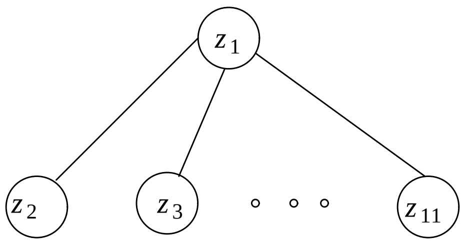

Figure 1. Interdependence graph Γ for the generating function in (7.2)

Thus, by inverting z_1 first and then $z_2, \ldots, z_{11}$ independently, we succeed in reducing the inversion dimension from 11 to 2. The numerical results for eight cases are given in Table 1 below.

chain populations K_j for $2 \le j \le 11$,	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K} - \mathbf{1}_1)/g(\mathbf{K})$
2	2	1.235628e25	1.492001e-2
2	20	7.503087e13	1.296652e-1
2	200	5.970503e129	4.477497e-1
2	2000	1.937826e683	4.982502e-1
$5(j-1)$	2	3.004462e107	8.039024e-3
$5(j-1)$	20	1.677866e133	6.803960e-2
$5(j-1)$	200	8.032122e260	2.858885e-1
$5(j-1)$	2000	1.617153e926	4.746674e-1

Table 1. Numerical results for the Example.

In each case, the inversions for variables $z_2, z_3, \ldots, z_{11}$ are done explicitly, using

$$g^{(1)}(\mathbf{z}_1, \mathbf{K}_2) = \frac{\exp\left[\rho_{10} z_1\right]}{\prod_{i=2}^{p}\left[1-\rho_{1i} z_1\right]^{m_i}} \prod_{i=2}^{p} \sum_{k=0}^{K_i} \frac{\left[\rho_{i0}\right]^k}{k!} \begin{bmatrix} m_i + K_i - k - 1 \\ K_i - k \end{bmatrix} \frac{\rho_{ii}^{K_i - k}}{\left[1 - \rho_{1i} z_1\right]^{K_i - k}} .$$

Hence no l_j is involved for $2 \leq j \leq 11$ and only a one-dimensional algorithm is required for this example. The accuracy is checked by doing the calculation with $l_1 = 1$ and $l_1 = 2$.

Acknowledgment. We thank Yaakov Kogan for discussions that helped motivate this work.

REFERENCES

[1] ABATE, J. and WHITT, W. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems* 10 (1992) 5-88.

[2] BERTOZZI, A. and MCKENNA, J. Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review* 35 (1993) 239-268.

[3] CHOUDHURY, G. L., LEUNG, K. K. and WHITT, W. Calculating normalization constants of closed queueing networks by numerically inverting their generating functions. submitted to *J. ACM*, 1993.

[4] CHOUDHURY, G. L., LEUNG, K. K. and WHITT, W. Calculating normalizations constants of circuit-switched communications network models by numerically inverting their generating functions. in preparation.

[5] CHOUDHURY, G. L., LEUNG, K. K. and WHITT, W. Calculating normalization constants in resource-sharing models by numerically inverting their generating functions. in preparation.

[6] CHOUDHURY, G. L., LUCANTONI, D. M. and WHITT, W. Multidimensional transform inversion with applications to the transient M/G/1 queue. *Ann. Appl. Prob.*, to appear.

[7] CONWAY, A. E. and GEORGANAS, N. D. *Queueing Networks — Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation.* MIT Press, Cambridge, MA, 1989.

[8] LAVENBERG, S. S. (ed.) *Computer Performance Modeling Handbook,* Academic Press, Orlando, FL, 1983.

[9] MCKENNA, J. and MITRA, D. Integral representations and asymptotic expansions for closed Markovian queueing networks: normal usage. *Bell System Tech. J.* 61 (1982) 661-683.

[10] REISER, M. and KOBAYASHI, H. Queueing networks with multiple closed chains: theory and computational algorithms. *IBM J. Res. Dev.* 19 (1975) 283-294.

To represent the recursive inversion, we define *partial generating functions* by

$$g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_j=0}^{\infty} g(\mathbf{K}) \prod_{i=1}^{j} z_i^{K_i} \quad \text{for } 1 \leq j \leq p , \tag{2.1}$$

where $\mathbf{z}_j = (z_1, z_2, \ldots, z_j)$ and $\mathbf{K}_j = (K_j, K_{j+1}, \ldots, K_p)$ for $1 \leq j \leq p$. Let $\mathbf{z}_0$ and $\mathbf{K}_{p+1}$ be null vectors. Clearly, $\mathbf{K} = \mathbf{K}_1, \mathbf{z} = \mathbf{z}_p$, $g^{(p)}(\mathbf{z}_p, \mathbf{K}_{p+1}) = G(\mathbf{z})$ and $g^{(0)}(\mathbf{z}_0, \mathbf{K}_1) = g(\mathbf{K})$.

Let I_j represent inversion with respect to z_j. Then the step-by-step nested inversion approach is

$$g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j) = I_j[g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1})] , \quad 1 \le j \le p , \tag{2.2}$$

starting with $j = p$ and decreasing j by 1 each step. In the actual program implementation, we attempt the inversion shown in (2.2) for $j = 1$. In order to compute the righthand side we need another inversion with $j = 2$. This process goes on until at step p the function on the righthand side becomes the p-dimensional generating function and is explicitly computable. By simply relabeling the p transform variables, we see that the scheme above can be applied to the p variables in any order.

In each step we use the lattice-Poisson inversion algorithm in Abate and Whitt [1,2] with modifications to improve precision and allow for complex inverse function as in Choudhury, Lucantoni and Whitt [9]. We show below the inversion formula at the j^{th} step. For simplicity, we suppress those arguments which remain constant during this inversion, letting $g_j(K_j) = g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j)$ and $G_j(z_j) = g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1})$. With this notation, the inversion formula is

$$g_j(K_j) = \frac{1}{2 l_j K_j r_j^{K_j}} \sum_{k_1=0}^{l_j-1} e^{-\frac{\pi i k_1}{l_j}} \sum_{k=-K_j}^{K_j-1} (-1)^k G_j\left(r_j e^{\frac{\pi i (k_1 + l_j k)}{l_j K_j}}\right) - e_j , \tag{2.3}$$

where $i = \sqrt{-1}$, l_j is a positive integer and e_j represents the *aliasing error*, which is given by

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2 n l_j K_j) r_j^{2 n l_j K_j} . \tag{2.4}$$

Note that, for $j = 1$, $g_1(K_1) = g(\mathbf{K})$ is real, so that $G_1(\overline{z}_1) = \overline{G_1(z_1)}$. This enables us to cut the computation in (2.3) by about one half. For $j = 1$, we replace (2.3) by

$$g_1(K_1) = \frac{1}{2 l_1 K_1 r_1^{K_1}} \left[G_1(r_1) - (-1)^{K_1} G_1(-r_1) + 2 \sum_{k_1=1}^{l_1} e^{\frac{-\pi i k_1}{l_1}} \right.$$
$$\left. \sum_{k=0}^{K_1-1} G_1\left[r_1 e^{\pi i (k_1 + l_1 k)/l_1 k_1} \right] \right] - e_1 . \tag{2.5}$$

To control the aliasing error, we choose

$$r_j = 10^{-\frac{\gamma_j}{2 l_j K_j}} . \tag{2.6}$$

Inserting (2.6) into (2.4), we get

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2 n l_j K_j) 10^{-\gamma_j n} . \tag{2.7}$$

As is clear from (2.7), a bigger γ_j decreases the aliasing error. Also, as explained in [9,10], the parameter l_j controls roundoff error, with bigger values causing less roundoff error. An inner loop of the inversion requires more accuracy than an outer loop since the inverted values in an inner loop are used as transform values in an outer loop. With a goal of about eight significant digit accuracy, the following sets of l_j and γ_j typically are adequate: i) $l_1 = 1, \gamma_1 = 11$, ii) $l_2 = l_3 = 2$, $\gamma_2 = \gamma_3 = 13$, iii) $l_4 = l_5 = l_6 = 3$, $\gamma_4 = \gamma_5 = \gamma_6 = 15$, assuming that computations are done using double-precision arithmetic. It is usually not a good idea to use the same l_j for all j, because then more computation is done to achieve the same accuracy.

In [1,2,9] the inverse function was mainly assumed to be a probability, so that the aliasing error e_j in (2.7) could be easily bounded. In contrast, here the normalization constants may be arbitrarily large and therefore the aliasing error e_j in (2.7) may also be arbitrarily large. Thus, in order to control errors, we scale the generating function in each step by defining a *scaled generating function* as

$$\overline{G}_j(z_j) = \alpha_{0j} G_j(\alpha_j z_j) \ , \tag{2.8}$$

where α_{0j} and α_j are positive real numbers. We invert this scaled generating function after choosing α_{0j} and α_j so that the errors are suitably controlled.

Let $\overline{g}_j(K_j)$ represent the inverse function of $\overline{G}_j(z_j)$. The desired inverse function $g_j(K_j)$ may then be recovered from $\overline{g}_j(K_j)$ by

$$g_j(K_j) = \alpha_{0j}^{-1} \alpha_j^{-K_j} \overline{g}_j(K_j) \ . \tag{2.9}$$

To compute steady-state performance measures, we are typically interested in ratios of normalization constants. Let $\mathbf{1}_j$ be the vector with a 1 in the j^{th} place and 0's elsewhere. Note that the ratio $g(\mathbf{K}-\mathbf{1}_j)/g(\mathbf{K})$ can be computed in terms of the scaled inverse function $\overline{g}_1(K_1)$ by

$$\frac{g(\mathbf{K}-\mathbf{1}_j)}{g(\mathbf{K})} = \frac{g_1^{-j}(K_1)}{g_1(K_1)} = \alpha_1 \frac{\overline{g}_1^{-j}(K_1)}{\overline{g}_1(K_1)} \ , \tag{2.10}$$

where $g_1^{-j}(K_1) = g^{(0)}(\mathbf{z}_0, \mathbf{K}_1 - \mathbf{1}_j) = g(\mathbf{K}-\mathbf{1}_j)$ and similarly for $\overline{g}_1^{-j}(K_1)$.

$$\alpha_{0j} = e^{-\alpha_j \rho_{j0}} \quad \text{and} \quad \alpha_j = \underset{i}{Min} \left\{ \frac{K_j}{\rho_{j0}} \ , \ \frac{a_{ij}}{\overline{\rho}_{ji}} \right\} , \tag{5.41}$$

where

$$a_{ij} = \left[\sum_{l=1}^{N_{ij}} \frac{K_j + l}{K_j + 2 l_j K_j + l} \right]^{1/2 l_j K_j} , \tag{5.42}$$

$$N_{ij} = \overline{m}_i - 1 + \sum_{k=j+1}^{p} K_k \eta_{ki} \tag{5.43}$$

$$\overline{\rho}_{ji} = \begin{cases} 0 & \tilde{\rho}_{ji} = 0 \\ \frac{1}{i} \sum_{k=1}^{i} \tilde{\rho}_{jk} \ , & \tilde{\rho}_{ji} > 0 \ , \end{cases} \tag{5.44}$$

$$\overline{m}_i = \sum_{k=1}^{i} \tilde{m}_k \ , \tag{5.45}$$

$$\eta_{ki} = \begin{cases} 1 & \text{if} \quad \rho_{ki} \neq 0 \\ 0 & \text{if} \quad \rho_{ki} = 0 \end{cases} \tag{5.45}$$

with $\{\tilde{\rho}_{ji} : 1 \leq i \leq q'\}$ being the sorted version (in decreasing order of magnitude) of $\{\rho_{ji}/(1 - \sum_{k=1}^{j-1} \rho_{ki} |z_k|) : 1 \leq i \leq q'\}$ and $\{\tilde{m}_i : 1 \leq i \leq q'\}$ is the rearranged version of $\{m_i : 1 \leq i \leq q\}$ associated with $\{\tilde{\rho}_{ji}\}$.

8. Numerical Examples

In this section we give numerical results for our algorithm applied to four closed queueing network examples. For each example, we calculate the normalization constant $g(\mathbf{K})$ in (1.2) and (4.3) for specified population vectors $\mathbf{K}$ from the generating function $G(\mathbf{z})$ in (4.5). Thus the parameters are the number of chains, p, the number of distinct single-server queues, q', the multiplicities m_i, the aggregate relative traffic intensities ρ_{ji}, $1 \leq j \leq p$, $0 \leq i \leq q'$, and the desired population vector $\mathbf{K}$.

Note that the normalization constant $g(\mathbf{K})$ only depends on these parameters p, q', m_i, ρ_{ji} and $\mathbf{K}$. Hence, we do not fully specify the models below. In particular, we do not give the routing probabilities $R_{ri,sj}$ or the mean service times t_{ri}. Thus, there are many detailed models consistent with our partial model specifications. One possible routing matrix consistent with the data that we provide is a cyclic routing matrix, all of whose entries are 0's and 1's, which yields visit ratios $e_{ri} = 1$ for all stages (r,i) from (4.2). If we consider this case, then $t_{ri} = \rho'_{ri}$ and the throughputs θ_{ri} in (4.7) coincide with the normalization constant ratios $g(\mathbf{K} - \mathbf{1}_j)/g(\mathbf{K})$. We display some of these ratios along with the values of $g(\mathbf{K})$ in our numerical results below. We note that the throughputs for any more detailed model can be found by solving (4.2) for the visit ratios e_{ri} and then applying (4.7).

For the first three examples, the total number of single-server queues is the same, namely, $q = 50$. However, in each example we consider ten distinct queues, each with multiplicity five. Thus, $q' = 10$ in each example. This reduces the computational complexity for our algorithm, but not for the others. We also allow for an arbitrary number of IS queues, with almost no additional computational complexity. Multiplicities and the presence of IS queues evidently complicate the theory of residues [3].

What is different in our examples is the number of closed chains. The first example has only one chain, and is thus the easiest example. The second example has four chains, while the third and fourth examples have eleven chains. However, the dimension can be reduced from eleven to two in the last two examples, whereas the dimension cannot be reduced below four in the second example, so that for our algorithm the second example is most difficult. The last case of the second example took about an hour on a SUN SPARC-2 workstation.

The numerical results below are shown to seven significant figures, which is more than adequate for most applications. However the realized accuracy was found to be 8-12 significant places. This was determined by using two sets of l_j values for each case. (The parameter l_j appears in (2.3); see Section 2.3.) Each number below is expressed as *aek*, which means $a \times 10^k$. Even with the scaling, some of the normalization constants assume values outside the range of computation on the computer. This difficulty is circumvented by working with logarithms, as discussed in Section 2.2.

For all the examples, we used Euler summation whenever K_j exceeds 31, which resulted in the computation of 31 terms. Euler summation works in all these examples, providing tremendous computational savings whenever a particular K_j is large.

8.1 Choosing Scaling Parameters

Note that the inversion procedure in (2.2) is a nested procedure, so that scaling done at one step will also modify the functions in subsequent steps. By (2.7), the aliasing error term for computing $\bar{g}_j(K_j)$ from the scaled generating function $\bar{G}_j(z_j)$ in (2.8) is

$$\bar{e}_j = \sum_{n=1}^{\infty} \bar{g}_j(K_j + 2nl_jK_j)10^{-\gamma_j n} . \tag{2.11}$$

Since the quantities needed in product-form models typically involve ratios of normalization constants, we really care about the relative error and not the absolute error. The relative error is given by

$$e'_j \equiv \frac{\bar{e}_j}{\bar{g}_j(K_j)} = \sum_{n=1}^{\infty} \frac{\bar{g}_j(K_j + 2nl_jK_j)}{\bar{g}_j(K_j)} 10^{-\gamma_j n} , \tag{2.12}$$

so that it can be bounded via

$$|e'_j| \leq \sum_{n=1}^{\infty} \left| \frac{\bar{g}_j(K_j + 2nl_jK_j)}{\bar{g}_j(K_j)} \right| 10^{-\gamma_j n} . \tag{2.13}$$

Let

$$C_j = \underset{n}{Max} \left\{ \left| \frac{\bar{g}_j(K_j + 2nl_jK_j)}{\bar{g}_j(K_j)} \right| \right\}^{1/n} . \tag{2.14}$$

Then

$$|e'_j| \leq \sum_{n=1}^{\infty} C_j^n 10^{-\gamma_j n} \leq \frac{C_j 10^{-\gamma_j}}{1 - C_j 10^{-\gamma_j}} \approx C_j 10^{-\gamma_j} . \tag{2.15}$$

Hence, to limit the aliasing error, we want C_j in (2.14) to be not too large, i.e.,

$$C_j << 10^{\gamma_j} . \tag{2.16}$$

Our main purpose in scaling is to satisfy (2.16), which in turn controls the relative aliasing error. However, note that only the scale parameter α_j is useful for this purpose since C_j is independent of the other scale parameter α_{0j}. We use the other scale parameter α_{0j} to make it more likely that $\bar{g}_j(K_j)$ does not exceed the range of floating point computation that is, we want

$$10^{-\delta} < \bar{g}_j(K_j) < 10^{\delta} , \tag{2.17}$$

where $(10^{-\delta}, 10^{\delta})$ is the range of floating point computation. However, scaling to satisfy (2.17) is not always possible. Therefore, we use a second measure to avoid floating point range exception. We compute and store only the logarithms of large quantities or the ratios of large quantities.

Here is what we do: All computations involve either products or sums. In the case of products, we express the logarithm of the product as the sum of the logarithms. In the case of a sum $s = \sum_{i=1}^{n} a_i$, we write

$$s = a_1(1 + \sum_{i=2}^{n} (a_i/a_1)) , \tag{2.18}$$

where $|a_1| = \max_i \{|a_i|\}$, and then

$$\log s = \log a_1 + \log(1 + \sum_{i=2}^{n} (a_i / a_1))$$
$$= \log a_1 + \log(1 + \sum_{i=2}^{n} \exp(\log a_i - \log a_1)) \ . \qquad (2.19)$$

To implement (2.19) we need to identify the largest term in the sum (2.l8). For us the relevant sum is (2.3). The largest term there is easy to identify, it corresponds to $k_1 = k = 0$.

An obvious problem with condition (2.16) is that we do not know $g_j(K_j)$ explicitly. However, since 10^{γ_j} is very large, it should be possible to satisfy (2.16) by roughly controlling the growth rate of $g_j(K_j)$ based on the structure of the generating function. Specifically, in many cases it is possible to express $g_j(K_j)$ as

$$g_j(K_j) = \sum_{i=1}^{m} A_i B_i(K_j) \ , \qquad (2.20)$$

where the A_i's are usually unknown constants, but the B_i's are known functions of K_j. Indeed, we show that this structure holds for the closed queueing networks in Section 5. Then our strategy is to identify the fastest growing function $B_i(K_j)$, and introduce a scaled version

$$\bar{B}_i(K_j) = \alpha_{0j} \alpha_j^{K_j} B_i(K_j) \qquad (2.21)$$

as in (2.8), so that

$$\underset{n}{Max} \left\{ \left| \frac{\bar{B}_i(K_j + 2nl_j K_j)}{\bar{B}_i(K_j)} \right| \right\}^{1/n} \leq \beta \ , \qquad (2.22)$$

where β is of the order of 1. Indeed, we identify the fastest growing function by requiring that (2.22) hold for *all i*. For the scaling of closed queueing networks in Section 5, we find that it suffices to consider only the case $n = 1$ in (2.22). Given (2.22), we use the scaling in (2.21) in (2.8).

We have just described a general scaling strategy. We present a specific scaling algorithm for a class of closed queueing networks that follows this strategy in Section 5. With the aid of (2.22), we are able to treat multiplicities in the factors of the generating function. However, to treat near multiplicities, we must go beyond (2.22), as we do in Section 5.5. Hence, (2.22) should be regarded only as a reasonable starting point. Difficult cases will require refinement.

8.2 Verification of the Accuracy of Computation

9. Dimension Reduction by Decomposition

In general, the inversion of a p-dimensional generating function $G(\mathbf{z})$ represents a p-dimensional inversion whether it is done directly or by our proposed recursive technique. Fortunately, however, it is often possible to reduce the dimension significantly by exploiting special structure. To see the key idea, note that if $G(\mathbf{z})$ can be written as a product of factors, where no two factors have common variables, then the inversion of $G(\mathbf{z})$ can be carried out by inverting the factors separately and the dimension of the inversion is thus reduced. The factors can be treated separately because factors not involving the variable of integration pass through the sum in (2.3). We call this an *ideal decomposition.* It obviously provides reduction of computational complexity, but we do not really expect to be able to exploit it, because it essentially amounts to having two or more completely separate models, which we would not have with proper model construction. (We would treat them separately to begin with.)

Even though ideal decomposition will virtually *never* occur, key model elements (e.g., closed chains) are often only *weakly coupled*, so that we can still exploit a certain degree of decomposition to reduce the inversion dimensionality, often dramatically. The idea is to look for *conditional decomposition.* This dimension reduction is illustrated in Section 5.4 and Examples 3 and 4 in Section 7.

10. Closed Queueing Networks

In this section we consider multi-chain closed queueing networks with only single-server queues (service centers with load-independent service rates) and infinite-server queues. This section closely follows Sections 2.1 and 2.2 of Bertozzi and McKenna [3], which in turn closely follow Bruel and Balbo [5]. However, we do not consider the most general models in [3,5]. We use the following notation:

- p = number of closed chains
- M = number of job classes ($M \ge p$).
- N = number of queues (service centers). Queues $1,\ldots,q$ are assumed to be of the single-server type and queues $q+1,\ldots,N$ are assumed to be of the infinite-sever (IS) type. As usual, for the single-server queues, the service discipline may be first-come first-served (FCFS), last-come first-served preemptive-resume (LCFSPR) or processor sharing (PS). In the case of FCFS, the service times of all job classes at a queue are assumed to be exponential with the same mean.
- $R_{ri,sj}$ = routing matrix entry, probability that a class r job completing service at queue i will next proceed to queue j as a class s job for $1 \le i,j \le N$, $1 \le r,s \le M$ (i.e., class hopping is allowed). The pair (r,i) is referred to as a stage in the network.
- class vs. chain: Two classes r and s communicate with each other if for some i and j, stage (s,j) can be reached from stage (r,i) in a finite number of steps and vice versa. With respect to the relation of communication, all the classes can be divided into mutually disjoint equivalence classes called (closed) chains (ergodic sets in Markov chain theory). All classes within a chain communicate. No two classes belonging to different chains communicate. Since we are considering the steady-state distribution of a model with only closed chains, we do not need to consider any transient stages, i.e., stages (r,i) that will not be reached infinitely often.
- K_j = number of jobs in the j^{th} closed chain, $1 \le j \le p$, which is fixed.
- $\mathbf{K} = (K_1, \ldots, K_p)$, the population vector, specified as part of the model data.
- n_{ri} = number of jobs of class r in queue i, $1 \le r \le M$, $1 \le i \le N$.
- n_i = number of jobs in queue i, $n_i = \sum_{r=1}^{M} n_{ri}$. $1 \le i \le N$.
- $\mathbf{n} = (n_{ri})$, $1 \le r \le M$, $1 \le i \le N$, the job vector, the queue lengths, the state variable.
- C_j = set of stages in the j^{th} closed chain. Clearly, $\sum_{(r,i) \in C_j} n_{ri} = K_j$, $1 \le j \le p$.
- $q_{ji} = \sum_{r\,:\,(r,i) \in C_j} n_{ri}$, number of jobs from chain j at queue i.
- $s(\mathbf{K})$ = state space of allowable job vectors or queue lengths (including those in service), i.e.,

$$S(\mathbf{K}) = \left\{ \mathbf{n} : n_{ri} \in \mathbf{Z}^{+} \quad \text{and} \quad \sum_{(r,i)\in C_j} n_{ri} = K_j , 1 \le j \le p \right\} , \tag{4.1}$$

where $\mathbf{Z}^{+}$ is the set of nonnegative integers.

- e_{ri} = visit ratio, i.e., solution of the traffic rate equation

$$\sum_{(r,i)\in C_k} e_{ri} R_{ri,sj} = e_{sj} \quad \text{for all} \quad (s,j) \in C_k \quad \text{and} \quad 1 \le k \le p \; . \tag{4.2}$$

For each chain there is one degree of freedom in (4.2). Hence, for each chain *j*, the visit ratios $\{ e_{ri} : (r,i) \in C_j \}$ are specified up to a constant multiplier.

- t_{ri} = the mean service time for class *r* at queue *i*.
- $\rho'_{ri} = t_{ri} e_{ri}$, $1 \le r \le M$, $1 \le i \le N$, the relative traffic intensities.
- The steady-state distribution is given by (1.1) and the partition function by (1.2), where

$$f(\mathbf{n}) = \left[\prod_{i=1}^{q} n_i ! \prod_{r=1}^{M} \frac{\rho'^{\,n_{ri}}_{ri}}{n_{ri} !} \right] \left[\prod_{i=q+1}^{N} \prod_{r=1}^{M} \frac{\rho'^{\,n_{ri}}_{ri}}{n_{ri} !} \right] . \tag{4.3}$$

- $\rho_{j0} = \sum_{i=q+1}^{N} \sum_{(r,i)\in C_j} \rho'_{ri}$ and $\rho_{ji} = \sum_{(r,i)\in C_j} \rho'_{ri}$ for $i = 1,2,...,q$, the aggregate relative traffic intensities.
- The generating function $G(\mathbf{z})$ is given by (1.4), using (1.2) and (4.3). As shown in (2.25) of Bertozzi and McKenna [3], $G(\mathbf{z})$ can be expressed simply as

$$G(\mathbf{z}) = \frac{\exp \left[\sum_{j=1}^{p} \rho_{j0} z_j \right]}{\prod_{i=1}^{q} \left[1 - \sum_{j=1}^{p} \rho_{ji} z_j \right]} \; . \tag{4.4}$$

In general, there may be multiplicity in the denominator factors of (4.4) if two or more queues are identical with respect to visits by customers of all classes. In such a situation (4.4) becomes

$$G(\mathbf{z}) = \frac{\exp \left[\sum_{j=1}^{p} \rho_{j0} z_j \right]}{\prod_{i=1}^{q'} \left[1 - \sum_{j=1}^{p} \rho_{ji} z_j \right]^{m_i}} \; , \tag{4.5}$$

where

$$\sum_{i=1}^{q'} m_i = q. \tag{4.6}$$

For us, (4.5) is the relevant form, not (4.4); i.e., the key parameters are p and q'. Our algorithm simplifies by having different queues with identical single-server parameters. (Evidently the reverse is true for the theory of residues [3].)

Given the normalization constant $g(\mathbf{K})$ in (1.2) and (4.3), we can directly compute the steady-state probability mass function $p(\mathbf{n})$ in (1.1). Moreover, several important performance measures can be computed directly from ratios of normalization constants. For example, the throughput of class r jobs at queue i is

$$\theta_{ri} = e_{ri} \frac{g(\mathbf{K}-\mathbf{1}_j)}{g(\mathbf{K})} \quad \text{for } (r,i) \in C_j \ , \tag{4.7}$$

where $\mathbf{1}_j$ is the vector with a 1 in the j^{th} place and 0's elsewhere; e.g., see p. 147 of [26]. The means $E[n_{ri}]$ and $E[q_{ji}]$ and higher moments $E[n_{ri}^k]$ and $E[q_{ji}^k]$ can also be computed directly from the normalization constants, but the standard formulas involve more than two normalization constant values. We develop an improved algorithm for means and higher moments via generating functions in Section 6 below.

11. Scaling for Closed Queueing Networks with Single-Server and Infinite-Server Queues

In this section we indicate how to choose the scale parameters α_{0j} and α_j in (2.8) in order to invert the generating function $G(\mathbf{z})$ in (4.5) for the class of closed queueing networks considered here. Our final scaling algorithm is given in Section 5.5 beginning with (5.41). We develop it by starting with more elementary cases and work our way up to the full generality of (4.5). The general strategy has already been described in Section 2.

11.1 Scaling for a Single Chain with Only Single-Server Queues

In this case we have (4.5) with $p = 1$ without the term in the numerator corresponding to infinite-server queues. Using (2.8), we see that the scaled generating function is given by

$$\overline{G}(z_1) = \frac{\alpha_{01}}{\prod_{i=1}^{q'} \left[1 - \rho_{1i}\alpha_1 z_1\right]^{m_i}} \ . \tag{5.1}$$

When there are no multiplicities (i.e., $m_i = 1$ for all i) and one aggregate relative traffic intensity ρ_{1i} is dominant (substantially larger than all others), it is easy to see that we should have $\alpha_1 = 1/\max\{\rho_{1i}\}$. We now give a more careful analysis to account for the multiplicities. See Section 5.5 below for a discussion of near multiplicities.

Carrying out a partial fraction expansion with (5.1), we get

$$\overline{G}(z_1) = \alpha_{01} \sum_{i=1}^{q'} \sum_{j=1}^{m_i} \frac{A_{ij}}{\left[1 - \rho_{1i}\alpha_1 z_1\right]^j} \ , \tag{5.2}$$

where

$$A_{ij} = \frac{(-1)^{m_i - j}}{\alpha_{01}(m_i - j)!(\rho_{1i}\alpha_1)^{m_i - j}} \left[\frac{d^{m_i - j}}{dz_1^{m_i - j}} \overline{G}(z_1) \right] \Bigg|_{z_1 = \frac{1}{\rho_{1i}\alpha_1}} \ . \tag{5.3}$$

In general, the constants A_{ij} in (5.3) are difficult to compute (except when $m_i = 1$ for all i), so we will treat them as unknown constants. Through a term-by-term binomial expansion and by collecting the coefficient of $z_1^{K_1}$, we get

$$\bar{g}(K_1) = \sum_{i=1}^{q'} \sum_{j=1}^{m_i} A_{ij} \bar{B}_{ij}(K_1) \ , \tag{5.4}$$

where

$$\bar{B}_{ij}(K_1) = \alpha_{01} \begin{bmatrix} j+K_1-1 \\ K_1 \end{bmatrix} \rho_{1i}^{K_1} \alpha_1^{K_1} \ . \tag{5.5}$$

Note that (5.4) is of the same form as (2.20). Hence, we can get the scale parameters by focusing on the fastest growing function $B_{ij}(K_1)$, and imposing condition (2.22). Note that, for any given i, the term corresponding to $j = m_i$ is the fastest growing; therefore we concentrate only on that term. It can be shown that the most restrictive condition in (2.22) comes from $n = 1$ and that (2.22) will be satisfied with $\beta = 1$ by setting

$$\alpha_{01} = 1 \quad \text{and} \quad \alpha_1 = \underset{i}{Min}\{a_i / \rho_{1i}\} \ , \tag{5.6}$$

where

$$a_i = \begin{cases} 1 \quad \text{for} \quad m_i = 1 \\ \left[\prod_{j=1}^{m_i - 1} \frac{K_1 + j}{K_1 + 2l_1 K_1 + j} \right]^{1/2l_1 K_1} \qquad \text{for} \quad m_i > 1 \ . \end{cases} \tag{5.7}$$

From (5.6) and (5.7), we see that we do indeed have the obvious scaling with $a_i = 1$ when $m_i = 1$ for all *i*. Moreover, we have this same scaling when K_1 becomes very large, because $a_i \rightarrow 1$ for all *i* as $K_1 \rightarrow \infty$ when $m_i > 1$.

Note that in this case we do not use α_{01} to satisfy (2.17), but as explained in Section 2 we do not have any problem if (2.17) is not satisfied, because we work with logarithms.

11.2 Scaling for a Single Chain with Single-Server and Infinite-Server Queues

In this case the term in the numerator of (4.5) corresponding to the infinite-server queues has to be included. Instead of (5.2), from the partial-fraction expansion we get

$$\overline{G}(z_1) = \alpha_{01} \sum_{i=1}^{q'} \sum_{j=1}^{m_i} \frac{A_{ij} e^{\rho_{10}\alpha_1 z_1}}{\left[1 - \rho_{1i}\alpha_1 z_1\right]^j} \ . \tag{5.8}$$

At first assume $m_i = 1$ for all *i*. For ease of notation, also let $A_{i1} \equiv A_i$. Then (5.8) becomes

$$\overline{G}(z_1) = \alpha_{01} \sum_{i=1}^{q'} \frac{A_i e^{\rho_{10}\alpha_1 z_1}}{\left[1 - \rho_{1i}\alpha_1 z_1\right]} \ . \tag{5.9}$$

Collecting the coefficient of $z_1^{K_1}$ on the right side of (5.9), and after some manipulation, we get

$$\overline{g}(K_1) = \sum_{i=1}^{q'} A_i \overline{B}_i(K_1) \ , \tag{5.10}$$

where

$$\overline{B}_i(K_1) = \alpha_{01}(\rho_{1i}\alpha_1)^{K_1} \sum_{j=0}^{K_1} \frac{(\rho_{10}/\rho_{1i})^j}{j!} \ . \tag{5.11}$$

Again (5.10) is of the same form as (2.20). The fastest growing term in this case is the one corresponding to the largest ρ_{1i}. Let

$$\rho_{1,max} = \underset{i}{Max}\{\rho_{1i}\} \ . \tag{5.12}$$

To proceed further with (5.11), we consider two cases: In Case 1, $\rho_{10}/\rho_{1,max} \leq K_1$, while in Case 2, $\rho_{10}/\rho_{1,max} > K_1$. In Case 1, by noticing the connection to the Poisson distribution, it can be shown that

$$\frac{1}{2} e^{(\rho_{10}/\rho_{1,max})} \leq \sum_{j=0}^{l} \frac{(\rho_{10}/\rho_{1,max})^j}{j!} \leq e^{(\rho_{10}/\rho_{1,max})} \qquad \text{for } l \geq K_1 \ . \tag{5.13}$$

Using (5.13), we see that condition (2.22) is satisfied (with $\beta = 2$) by making the following choices of scale parameters

$$\alpha_{01} = e^{-(\rho_{10}/\rho_{1,max})} \quad \text{and} \quad \alpha_1 = 1/\rho_{1,max} \ . \tag{5.14}$$

In Case 2, relation (5.13) does not hold. In this case we control the growth rate of the fastest growing term in (5.11). Let $B_i'(K_1)$ represent this term, which is given by

$$B_i'(K_1) \;=\; \alpha_{01} \frac{(\alpha_1 \rho_{10})^{K_1}}{K_1!} \approx \frac{\alpha_{01} (\alpha_1 \rho_{10})^{K_1}}{\sqrt{2\pi K_1 K_1^{K_1} e^{-K_1}}} \;. \tag{5.15}$$

The approximate expression in (5.15) is obtained by Stirling's approximation to the factorial. Note that $B_i'(K_1)$ is independent of i (since Case 2 ensures that $\rho_{10}/\rho_{1i} > K_1$ for all i). Therefore, we can control the growth rate of $B_i'(K_1)$ for all i by choosing the scale parameters to cancel out the two dominant terms in the expression for the factorial. This is done with the choice

$$\alpha_{01} \;=\; e^{-K_1} \quad \text{and} \quad \alpha_1 \;=\; \frac{K_1}{\rho_{10}} \;. \tag{5.16}$$

The scaling for Cases 1 and 2 in (5.14) and (5.16) can be combined as follows:

$$\alpha_{01} \;=\; e^{-\alpha_1 \rho_{10}} \quad \text{and} \quad \alpha_1 \;=\; \underset{i}{Min} \left\{ \frac{K_1}{\rho_{10}} \;,\; \frac{1}{\rho_{1i}} \right\} \;. \tag{5.17}$$

Finally, based on (5.6) and (5.17), we extend the scaling in the case $m_i > 1$ as follows:

$$\alpha_{01} \;=\; e^{-\alpha_1 \rho_{10}} \quad \text{and} \quad \alpha_1 \;=\; \underset{i}{Min} \left\{ \frac{K_1}{\rho_{10}},\; \frac{a_i}{\rho_{1i}} \right\} \;, \tag{5.18}$$

where a_i is as in (5.7)

11.3 Scaling for Multiple Chains with Single-Server and Infinite-Server Queues

For the general case, the generating function is as in (4.5). In order to carry out the innermost level of inversion (i.e., the p^{th} or last step of inversion) with respect to z_p we may assume that z_i for $i \;=\; 1,2,...,p-1$ is constant. We rewrite (4.5) as

$$G(\mathbf{z}) \;=\; \left[\frac{\exp\left[\sum_{j=1}^{p-1} \rho_{j0} z_j \right]}{\prod_{i=1}^{q'} \left[1 \;-\; \sum_{j=1}^{p-1} \rho_{ji} z_j \right]^{m_i}} \right] \left[\frac{\exp(\rho_{p0} z_p)}{\prod_{i=1}^{q'} \left[1 \;-\; \dfrac{\rho_{pi} z_p}{1 \;-\; \sum_{j=1}^{p-1} \rho_{ji} z_j} \right]^{m_i}} \right] \;. \tag{5.19}$$

The first factor on the right side of (5.19) is a constant, while the second factor is the same as in the one-dimensional inversion considered in Section 5.2 with ρ_{10} replaced by ρ_{p0} and ρ_{1i} replaced by $\rho_{pi} / \left[1 - \sum_{j=1}^{p-1} \rho_{ji} z_j \right]$. The second parameter is complex, so we replace it by its modulus. Using (5.18), we get the scaling as

$$\alpha_{0p} \;=\; e^{-\alpha_p \rho_{p0}} \quad \text{and} \quad \alpha_p \;=\; \underset{i}{Min} \left\{ \frac{K_p}{\rho_{p0}} \;,\; (a_{ip}/\rho_{pi}) \left| 1 - \sum_{j=1}^{p-1} \rho_{ji} z_j \right| \right\} \;, \tag{5.20}$$

where

$$a_{ip} \;=\; \begin{cases} 1 \quad \text{for} \quad m_i \;=\; 1 \\ \left[\prod_{j=1}^{m_i - 1} \dfrac{K_p + j}{K_p + 2 l_p K_p + j} \right]^{1/2 l_p K_p} \quad \text{for} \quad m_i \;>\; 1 \;. \end{cases} \tag{5.21}$$

Note that it is possible to have $\rho_{pi} \;=\; 0$ for some i, but not all i. If $\rho_{pi} \;=\; 0$, then $a_{ip}/\rho_{pi} \;=\; \infty$, so that the i^{th} term does not yield the minimum in (5.20).

Next we consider the inversion with respect to z_{p-1}. The generating function in this step is the inverse function in the p^{th} step. Using (5.19), we can write it as

$$g^{(p-1)}(\mathbf{z}_{p-1}, \mathbf{K}_p) = I_p \left[G(\mathbf{z}) \right]$$

$$= \frac{\exp[\sum_{j=1}^{p-1} \rho_{j0} z_j]}{\prod_{i=1}^{q'} [1 - \sum_{j=1}^{p-1} \rho_{ji} z_j]^{m_i}} \sum_{i=1}^{q'} \sum_{j=1}^{m_i} \sum_{k=0}^{K_p} A_{ij} \frac{(\rho_{p0})^k}{k!} \begin{bmatrix} j + K_p - k - 1 \\ K_p - k \end{bmatrix} \frac{\rho_{pi}^{K_p - k}}{[1 - \sum_{j=1}^{p-1} \rho_{ji} z_j]^{K_p - k}} \quad (5.22)$$

where A_{ij} has an expression similar to (5.3). The dominant term in (5.22) is of the form

$$\frac{\exp\left[\sum_{j=1}^{p-1} \rho_{j0} z_j\right]}{\prod_{i=1}^{q'} \left[1 - \sum_{j=1}^{p-1} \rho_{ji} z_j\right]^{K_p + m_i}} . \quad (5.23)$$

Note that in (5.22) we have implicitly assumed that $\rho_{pi} \neq 0$. If instead $\rho_{pi} = 0$ for some i, then corresponding to that i the term $\rho_{pi}^{K_p - k} \Big/ \left[1 - \sum_{j=1}^{p-1} \rho_{ji} z_j\right]^{K_p - k}$ would be missing. Hence, instead of (5.23), in general the dominant term in (5.22) is

$$\frac{\exp\left[\sum_{j=1}^{p-1} \rho_{j0} z_j\right]}{\prod_{i=1}^{q'} \left[1 - \sum_{j=1}^{p-1} \rho_{ji} z_j\right]^{K_p \eta_{pi} + m_i}} , \quad (5.24)$$

where $\eta_{pi} = 1$ if $\rho_{pi} \neq 0$ and $\eta_{pi} = 0$ otherwise.

Note that (5.24) is similar to $G(\mathbf{z})$ with p replaced by $p-1$ and m_i replaced by $K_p \eta_{pi} + m_i$. Therefore, from (5.20), we get the scaling in the $(p-1)^{st}$ step as $\alpha_{0,p-1} = e^{-\alpha_{p-1} \rho_{p-1,0}}$ and

$$\alpha_{p-1} = \underset{i}{Min} \left\{ \frac{K_{p-1}}{\rho_{p-1,0}} , \ (a_{i,p-1} / \rho_{p-1,i}) \left| 1 - \sum_{j=1}^{p-2} \rho_{ji} z_j \right| \right\} , \quad (5.25)$$

where

$$a_{i,p-1} = \left[\sum_{j=1}^{N_{i,p-1}} \frac{K_{p-1} + j}{K_{p-1} + 2 l_p K_{p-1} + j} \right]^{1/2 l_{p-1} K_{p-1}} , \quad (5.26)$$

where $N_{i,p-1} = K_p \eta_{pi} + m_i - 1$ with η_{pi} defined as above.

Proceeding as above, we get the scaling in step j, $1 \le j \le p$, as

$$\alpha_{0j} = e^{-\alpha_j \rho_{j0}} \quad \text{and} \quad \alpha_j = \underset{i}{Min} \left\{ \frac{K_j}{\rho_{j0}} \ , \ (a_{ij}/\rho_{ji}) \left| 1 - \sum_{k=1}^{j-1} \rho_{ki} z_k \right| \right\} , \tag{5.27}$$

where

$$a_{ij} = \left[\sum_{l=1}^{N_{ij}} \frac{K_j + l}{K_j + 2l_j K_j + l} \right]^{1/2 l_j K_j} \tag{5.28}$$

and

$$N_{ij} = m_i - 1 + \sum_{k=j+1}^{p} K_k \eta_{ki} \ , \tag{5.29}$$

with $\eta_{ki} = 1$ if $\rho_{ki} \neq 0$ and $\eta_{ki} = 0$ otherwise. In (5.29) if an upper sum index is smaller than the lower sum index, then that should be interpreted as a vacuous sum.

Note that the scale parameters above are not constant. The scaling at the j^{th} step depends on z_k for $1 \leq k \leq j-1$. Since the z_k's change during the course of inversion, so do the scale parameters. From (2.3), it is clear that the z_k values are always on a circle and therefore the modulus $|z_k|$ is constant. Furthermore, since the parameters ρ_{ki} are nonnegative the most restrictive scaling (smallest values of α_j) comes when $z_k = |z_k|$ (i.e., the point z_k is on the positive real line) for $1 \leq k \leq j-1$. From (2.3) it is clear that this *restrictive scaling* is indeed done once during the course of the inversion algorithm. If we use the restrictive scaling for all cases, then the scale parameter at the j^{th} step becomes constant. The advantage of this scaling is that we then need to compute each scale parameter only once. Secondly, we need not recover the original inverse function from the inverse of the scaled generating function in each step using (2.9). Instead, the recovery may be done only once at the end of all inversions and all intermediate computation may be done using only scaled functions. The use of this restrictive scaling makes all the computations related to scaling insignificant compared to the overall computations. Through numerical investigations we observed that the restrictive scaling produces about the same accuracy as the scaling in (5.27), so that we recommend using it. It is as given below:

$$\alpha_{0j} = e^{-\alpha_j \rho_{j0}} \quad \text{and} \quad \alpha_j = \underset{i}{Min} \left\{ \frac{K_j}{\rho_{j0}} \ , \ (a_{ij}/\rho_{ji}) (1 - \sum_{k=1}^{j-1} \rho_{ki} |z_k|) \right\} , \tag{5.30}$$

where a_{ij} is as given in (5.28).

11.4 Scaling with Dimension Reduction

In many situations there will be dimension reduction. As indicated in Section 3, the possibility of dimension reduction can be checked using the interdependence graph approach. The scaling can be done independently of the dimension reduction, just as described in Section 5.3, except that the dimension reduction determines the order of the variables to be inverted. The d variables requiring full inversion become variables $z_1, \ldots, z_d$ and the remaining $p-d$ variables become $z_{d+1}, \ldots, z_p$; i.e., the remaining variables appear in the inner loops.

It is also possible to directly determine scaling after the dimension reduction is done. We illustrate that in this subsection with an example. We also show how it is sometimes possible to replace a numerical inversion by an explicit formula.

Consider a closed queueing network with one infinite-server queue and $p-1$ groups of single-server queues where each group i has m_i identical queues. There are p closed chains in the model. The i^{th} chain $(2 \leq i \leq p)$ goes through each single-server in group i at least once and the infinite-server queue any number of times. As for the first chain, it goes through the infinite-server queue and *all* single-server queues in the model at least once. Note that we have not fully specified the routing matrix here, which is not necessary for computing $g(\mathbf{K})$.

For this model, $q' = p-1$ and $\rho_{ji} = 0$ unless $j = 1$ or $j = i$ or $i = 0$, i.e., the generating function becomes

To construct the interdependence graph for $G(\mathbf{z})$, we form a subgraph with two nodes for each factor in the denominator, z_1 and z_i for $2 \le i \le p$. Since the numerator can be viewed as a product of p factors, each with one z variable, each factor in the numerator is represented by a one-node subgraph. Taking the union of all these subgraphs, we obtain the interdependence graph Γ for (5.31) depicted below.

If we delete the subset $D = \{z_1\}$, then Γ becomes disconnected into subsets $S_i(D) = \{z_i\}$ for $2 \le i \le p$. According to (3.2), the dimension of the inversion resulting from the subset D is 2 in this case. That is, the inversion dimension can be reduced from p to 2.

To show the inversion order more precisely, equation (5.31) can be rewritten as

$$G(\mathbf{z}) = \frac{\exp\left[\rho_{10} z_1\right]}{\prod_{i=2}^{p}\left[1-\rho_{1i} z_1\right]^{m_i}} \prod_{i=2}^{p} \frac{\exp(\rho_{i0} z_i)}{\left[1-\dfrac{\rho_{ii} z_i}{1-\rho_{1i} z_1}\right]^{m_i}} \ . \tag{5.32}$$

If we keep z_1 constant, then the first factor on the right side of (5.32) becomes constant and the second factor becomes a product of $p-1$ terms, each of which is a function of a single z variable. Each such factor may be inverted independently with respect to the corresponding z variable, and once we take a product of each inverse function, we get a function of z_1 and $K_2, K_3, \ldots, K_p$. A final inversion with respect to z_1 yields the desired normalization constant $g(\mathbf{K})$. So the p-dimensional problem is reduced to a two-dimensional one. Next comes the question of scaling.

From the scaling in Section 5.2, we see that the scaling required to invert the i^{th} factor ($2 \le i \le p$) in (5.32) is given by

$$\alpha_{0i} = e^{-\alpha_i \rho_{i0}} \quad \text{and} \quad \alpha_i = Min\left\{ \frac{K_i}{\rho_{i0}},\ (a_i/\rho_{ii})(1-\rho_{1i}|z_1|) \right\} , \tag{5.33}$$

where

$$a_i = \begin{cases} 1 \quad \text{for} \quad m_i = 1 \\ \left[\prod_{j=1}^{m_i-1} \dfrac{K_i + j}{K_i + 2l_i K_i + j} \right]^{1/2 l_i K_i} \quad \text{for} \quad m_i > 1 \ . \end{cases} \tag{5.34}$$

It is also possible to explicitly invert the i^{th} product factor ($2 \le i \le p$) in (5.32) and, when we do, we get The dominant term in (5.35) is of the form

$$\frac{\exp\left[\rho_{10} z_1\right]}{\prod_{i=2}^{p}\left[1 - \rho_{1i} z_1\right]^{K_i + m_i}} .$$

Therefore, from (5.18), we get the scaling in order to invert with respect to z_1 as

$$\alpha_{0,1} = e^{-\alpha_1 \rho_{10}} \quad \text{and} \quad \alpha_1 = \underset{i}{Min} \left\{ \frac{K_1}{\rho_{10}} \; , \; \frac{a_{i1}}{\rho_{1i}} \right\} , \tag{5.36}$$

where

$$a_{i1} = \left[\sum_{j=1}^{N_{i1}} \frac{K_1 + j}{K_1 + 2 l_1 K_1 + j} \right]^{1/2 l_1 K_1} \tag{5.37}$$

and

$$N_{i1} = m_i - 1 + K_i \ , \tag{5.38}$$

It is to be noted that all scaling done in this subsection could also be derived from the final scaling equations in Section 5.3 ((5.28)–(5.30)); in this section $\eta_{ji} = 1$ for $j = 1$ and $j = i$ and $\eta_{ji} = 0$ otherwise.

11.5 Near Multiplicities

We have indicated that much of the difficulty in the scaling is due to the possibility of multiple factors in the denominator of (4.5). It should be intuitively clear that these same difficulties can arise with near multiplicities, and that so far our scaling algorithm does not account for near multiplicities. Moreover, in an application we might not recognize exact multiplicities.

In this subsection we introduce a heuristic scheme to account for near multiplicities. Consider the single-chain setting with only single-server queues in Section 5.1, and assume, without loss of generality, that $\rho_{1i} \geq \rho_{1,i+1}$ for all i. (This involves a sorting, which is of low complexity.) Without multiplicities or near multiplicities, the scaling should be $a_1 = 1/\rho_{11}$, but with multiplicities or near multiplicities perhaps other terms should play a role in the minimum in (5.6). To capture the effect of near multiplicities, for the scaling only, we replace ρ_{1i} in (5.1) and (5.6) by the average of the i largest relative traffic intensities, i.e.,

$$\hat{\rho}_{1i} = \frac{1}{i} \sum_{k=1}^{i} \rho_{1k} \ . \tag{5.39}$$

Moreover, in (5.7) we act as if the multiplicity associated with the i^{th} group of queues is

$$\hat{m}_i = \sum_{k=1}^{i} m_k \ ; \tag{5.40}$$

i.e., we replace a_i in (5.7) with $\hat{a}_i$ based on $\hat{m}_i$ in (5.40). Note that $\hat{a}_1 = a_1$ and $\hat{\rho}_{11} = \rho_{11}$, but that $\hat{a}_i \leq a_i$ and $\hat{\rho}_{1i} \geq \rho_{1i}$, so that the first ratio a_1/ρ_{11} is unchanged, but the other ratios in (5.6) are decreased, which may reduce α_1 in (5.6). The extra reduction will tend to occur when there are near multiplicities.

Based on this idea, we propose replacing the restrictive scaling in (5.30) by an even more restrictive scaling. We let

Finally, as a further turning heuristic, we have found that reducing a scaling parameter successively from α_j to about 0.95 α_j can be helpful if difficulties are encountered with the standard scaling in (5.41)–(5.45).

12. Moments via Generating Functions

Given the steady-state probability mass function, we can calculate moments. Without loss of generality, let $(r,i) \in C_1$. We start with a standard expression for the probability mass function of q_{1i}, the number of chain 1 customers at queue i, namely,

$$P(q_{1i} = k) = \frac{\rho_{1i}^k (g(\mathbf{K} - k\mathbf{1}_1) - \rho_{1i} g(\mathbf{K} - (k+1)\mathbf{1}_1))}{g(\mathbf{K})} , \tag{6.1}$$

see (3.257) on p. 147 of [28]. (A similar expression holds for the mass function of n_{ri} [28]. It involves ρ'_{ri} instead of ρ_{1i}.)

From the telescoping property of (6.1), we can write the tail probabilities as

$$P(q_{1i} = k) \geq \frac{\rho_{1i}^k g(\mathbf{K} - k\mathbf{1}_1)}{g(\mathbf{K})} . \tag{6.2}$$

From (6.2) we obtain the standard formula for the mean,

$$E[q_{1i}] = \sum_{k=1}^{\infty} P(q_{1i} \geq k) = \sum_{k=1}^{K_1} \rho_{1i}^k \frac{g(\mathbf{K} - k\mathbf{1}_1)}{g(\mathbf{K})} ; \tag{6.3}$$

e.g., see (3.258) on p. 147 of [28]. Unfortunately, formula (6.3) is not too convenient for us, because it requires $K_1 + 1$ normalization function calculations and thus $K_1 + 1$ numerical inversions. We now show how this mean can be calculated by *two* inversions.

For this purpose, we rewrite (6.3) as

$$E[q_{1i}] = \frac{\rho_{1i}^{K_1} h(\mathbf{K})}{g(\mathbf{K})} - 1 , \tag{6.4}$$

where

$$h(\mathbf{K}) = \sum_{k=0}^{K_1} \rho_{1i}^{-k} g(k, \mathbf{K}_2) . \tag{6.5}$$

Let $H(z_1)$ be the generating function of $h(\mathbf{K})$ with respect to K_1. Then

$$\begin{aligned} H(z_1) &= \sum_{m=0}^{\infty} z_1^m h(m, \mathbf{K}_2) = \sum_{m=0}^{\infty} z_1^m \sum_{k=0}^{m} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \\ &= \sum_{k=0}^{\infty} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \sum_{m=k}^{\infty} z_1^m = \frac{g^{(1)}(z_1/\rho_{1i}, \mathbf{K}_2)}{1 - z_1} , \end{aligned} \tag{6.6}$$

where $g^{(1)}(\mathbf{z}_1, \mathbf{K}_2)$ is the partial generating function in (2.1). Now, if $H(\mathbf{z})$ represents the full generating function of $h(\mathbf{K})$, then from (6.6) it is clear that

$$H(\mathbf{z}) = \frac{G(z_1/\rho_{1i} , z_2, \ldots, z_p)}{1 - z_1} . \tag{6.7}$$

Since $H(\mathbf{z})$ is of the same form as $G(\mathbf{z})$ it may be inverted by the established inversion procedure. Hence, we can obtain the mean $E[q_{1i}]$ using two inversions from (6.4). We invert $G(\mathbf{z})$ and $H(\mathbf{z})$, respectively, to obtain $g(\mathbf{K})$ and $h(\mathbf{K})$.

By the same approach, we can also calculate higher moments. For example,

$$E[q_{1i}(q_{1i} - 1)] = 2 \sum_{k=0}^{\infty} k P(q_{1i} \geq k) = \frac{2\rho_{1i}^{K_1} h(\mathbf{K})}{g(\mathbf{K})} , \tag{6.8}$$

where

$$h_1(\mathbf{K}) \;=\; \sum_{k=0}^{K_1} k\rho_{1i}^{-k} g(k,\mathbf{K}_2) \;. \tag{6.9}$$

Let $H_1(z_1)$ be the generating function of $h_1(\mathbf{K})$ with respect to K_1. Then

$$H_1(z_1) \;=\; \sum_{m=0}^{\infty} z_1^m h_1(m,\mathbf{K}_2) \;=\; \sum_{m=0}^{\infty} z_1^m \sum_{k=0}^{K_1} k\rho_{1i}^{-k} g(k,\mathbf{K}_2)$$

$$= \sum_{k=0}^{\infty} k\rho_{1i}^{-k} g(k,\mathbf{K}_2) \sum_{m=k}^{\infty} z_1^m \;=\; \left[\frac{z_1}{1-z_1}\right] \frac{\partial}{\partial z_1} g^{(1)}(z_1/\rho_{1i}\;,\;\mathbf{K}_2) \;. \tag{6.10}$$

Moreover, the full generating function of $h_1(K)$ is

$$H_1(\mathbf{z}) \;=\; \frac{z_1}{1-z_1} \frac{\partial}{\partial z_1} G(z_1/\rho_{1i}\;,\; z_2, \ldots, z_p) \;. \tag{6.11}$$

Finally, note from (4.5) that

$$\frac{\partial G(\mathbf{z})}{\partial z_1} \;=\; \left[\rho_{10} + \sum_{i=1}^{q'} \frac{m_i \rho_{1i}}{(1-\sum_{j=1}^{p} \rho_{ji} z_j)}\right] G(\mathbf{z}) \;, \tag{6.12}$$

so that the new generating function is a sum of generating functions of the same form as $G(\mathbf{z})$.

For higher factorial moments, we proceed in the same way using (6.2) and

$$E[q_{1i}(q_{1i}-1)\ldots(q_{1i}-l+1)] \;=\; l \sum_{k=0}^{K_1} k(k-1)\ldots(k-l+2) P(q_{1i} \geq k) \;. \tag{6.13}$$

For more on moments, see McKenna [29] and references there.

Example 1. In the first example, there are an arbitrary number of infinite-server queues and a total of 50 single-server queues, which consist of 10 distinct queues, each with a multiplicity of 5. The model has one closed chain going through all infinite-server queues and each single-server queue at least once.

The classical closed-form expression for the normalization constant $g(K)$ due to Koenigsberg [17,22,23] holds in a single-chain model when there are no multiplicities and no load-dependent servers; see (3.9) of [3]. Bertozzi and McKenna [3] derived corresponding closed-form expressions for $g(K)$ when there are multiplicities but no load-dependent servers ((3.12) of [3]) and when there is an IS queue but no multiplicities ((3.22) of [3]), but evidently no closed-form expression has yet been derived for the case considered here in which there are both multiplicities and an IS queue. Moreover, with either multiplicities or an IS queue, our algorithm may be competitive with computation from the closed-form formulas. However, without multiplicities or load-dependent servers, the simple classical formula (3.9) of [3] seems clearly best.

Here $p = 1$, $q' = 10$ and $m_i = 5$, $1 \leq i \leq 10$. The relative traffic intensities are $\rho_{10} = 5$ and

$$\rho_{1i} \;=\; 0.1i \quad \text{for} \quad i = 1,2, \ldots, 10 \;,$$

so that ρ_{1i} ranges from 0.1 to 1.0. We consider five cases for the total population: $K_1 = 2\times 10^k$ for $k = 0$, 1,..,4. We give the numerical results below in Table 1.

population K_1	normalization constant $g(K_1)$	ratio $g(K_1 - 1)/g(K_1)$
2	5.377500e2	6.043701e-2
20	1.906584e13	4.461782e-1
200	1.381312e26	9.659115e-1
2000	1.284918e31	9.978987e-1
20,000	1.541538e35	9.997990e-1

Table 1. Numerical results for Example 1.

To check our accuracy (see Section 2.3), in each case we preformed each calculation of $g(K_1)$ twice, once with $l_1 = 1$ and once with $l_1 = 2$. Both calculations agreed to at least the seven digits displayed in Table 1.

Being one-dimensional, this example is relatively elementary. However, this example has both multiplicities ($m_i = 5$ for all i) and near multiplicities ($\rho_{1,10} = 1$ and $\rho_{1,9} = 0.9$). Hence, neither the simple scaling $\alpha_1 = 1/\max_i \{\rho_{1i}\}$, nor the scaling in Section 5.1 is effective. For this example, we need the refined more restrictive scaling in Section 5.5.

Example 2. This example is identical to Example 1 except that now $p = 4$ and each of the four chains goes through each of the 50 single-server queues at least once and the first chain does not go through any infinite-server queue. No dimension reduction is possible in this case, because ρ_{ji} is positive for all i and j. Since we have 10 distinct queues, each with multiplicity 5, this model does not satisfy the assumptions for the residue results in [3]. However, as we have noted before, the multiplicities reduce the computational complexity here.

Example 4. This is our challenge problem for other algorithms. It is the same as Example 3 except that we change the multiplicities m_i and the chain populations K_j. We increase m_i from 5 to 100, $1 \leq i \leq 10$. Hence, now there are $q = 1000$ single-server queues, but still only $q' = 10$ distinct queues. We consider three cases. First, we increase the chain populations to $K_j = 200$ for $2 \leq j \leq 11$. We obtain three subcases by considering three different values for K_1. Our numerical results are given below in Table 4.

chain populations K_j for $2 \leq j \leq 11$	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K} - \mathbf{1}_1)/g(\mathbf{K})$
200	20	1.232036e278	4.582983e-3
200	200	2.941740e579	4.281094e-2
200	2000	3.399948e2037	2.585489e-1

Table 4. Numerical results for Case 1 of Example 4.

As in Example 3, the accuracy was checked by performing the calculations twice, once with $l_1 = 1$ and once with $l_1 = 2$. Again, the inversions for variables $z_2, z_3, \ldots, z_{11}$ are done explicitly by (5.35), so that the numerical inversion was essentially one-dimensional.

The results in Table 4 were obtained in less than one minute. This example would seem to be out of the range of existing algorithms, with the exception of the ones based on asymptotics [21,30]. (We anticipate that it should be possible to exploit the special structure with other algorithms, but that evidently is not part of the basic algorithms. It seems to be an interesting direction of research.)

From [30-32], we see that the PANACEA algorithm based on asymptotics requires that there be an IS queue and that each chain visit this queue. Unlike [30-32], the asymptotic approximation in [21] does not require any IS queue, but it seems to require that there be *no* IS queues and that *all* chain populations be large. To show that our algorithm does not have such limitations, we consider two modifications of Case 1. Case 2 has classes 1 and 2 with small populations, while the other class populations remain large. In particular, we let $K_2 = 2$ and $K_3 = 5$. Numerical results for Case 2 appear below in Table 5.

chain populations K_j for $2 \le j \le 11$	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K} - \mathbf{1}_1)/g(\mathbf{K})$
in all cases:	2	3.842031e407	5.128582e-4
$K_2 = 2,$	20	1.484823e454	5.087018e-3
$K_3 = 5$ and	200	6.003231e747	4.706391e-1
$K_j = 200,\ 4 \le j \le 11$	2000	5.442693e2154	2.705391e-1

Table 5. Numerical results for Case 2 of Example 4.

Case 3 is a modification of Case 2 in which we remove all the IS nodes, i.e., we set $\rho_{j0} = 0$ for all *j*. Numerical results for Case 3 appear below in Table 6.

chain populations K_j for $2 \le j \le 11$	K_1	normalizations constant $g(\mathbf{K})$	ratio $g(\mathbf{K} - \mathbf{1}_1)/g(\mathbf{K})$
in all cases:	2	9.959619e313	4.762073e-4
$K_2 = 2,$	20	1.447107e361	4.728591e-3
$K_3 = 5$ and	200	1.222889e660	4.417444e-2
$K_j = 200,\ 4 \le j \le 11$	2000	2.948943e2096	2.645993e-1

Table 6. Numerical results for Case 3 of Example 4.

As for Case 1, Cases 2 and 3 required about a minute on the SUN SPARC-2 workstation.

13. Conclusions

We have presented a general algorithm for calculating normalization constants in product-form models by numerically inverting their generating functions (Section 2). We have shown how the dimension can often be dramatically reduced by exploiting conditional decomposition (Section 3). We have considered in detail the special case of closed queueing networks with only single-server and IS queues (Section 4) and developed an explicit scaling algorithm for this class of models (Section 5). We have shown how to calculate mean queue lengths and higher-order moments directly by performing only two inversions of the same form as for the normalization constant (Section 6). Finally, we have presented a few numerical examples illustrating the algorithms and showing that it can solve some challenging problems (Section 7).

It remains to develop detailed scaling algorithms for other subclasses of product-form models, extending Section 5. Some of this is done in [7]; other cases are in progress. A further goal is to develop an effective automatic, dynamic scaling algorithm that requires only limited knowledge of special generating function structure, in the spirit of Lam [26].

In conclusion, we think that the numerical inversion algorithm here usefully complements existing algorithms for closed queueing networks and related models, and that there is significant potential for further progress with this approach.

Acknowledgment. We thank Yaakov Kogan for discussions that helped motivate this work.

REFERENCES

[1] ABATE, J. and WHITT, W. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems* 10 (1992) 5-88.

[2] ABATE, J. and WHITT, W. Numerical inversion of probability generating functions. *Opns. Res. Letters* 12 (1992) 245-251.

[3] BERTOZZI, A. and MCKENNA, J. Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review* 35 (1993) 239-268.

[4] BIRMAN, A., FERGUSON, D. and KOGAN, Y. Asymptotic solutions for a model of large multiprogramming systems with multiple workloads. IBM T. J. Watson Research Center, Yorktown Heights, NY, 1992.

[5] BRUEL, S. C. and BALBO, G. *Computational Algorithms for Closed Queueing Networks*, Elsevier, New York, 1980.

[6] BUZEN, J. P. Computational algorithms for the closed queueing networks with exponential servers. *Commun. ACM* 16 (1973), 527-531.

[7] CHOUDHURY, G. L., LEUNG, K. K. and WHITT, W. Calculating normalization constants in resource-sharing models by numerically inverting their generating functions. in preparation.

[8] CHOUDHURY, G. L. and LUCANTONI, D. M. Numerical computation of the moments of a probability distribution from its transforms. *Opns. Res.* (1994), to appear.

[9] CHOUDHURY, G. L., LUCANTONI, D. M. and WHITT, W. Multidimensional transform inversion with applications to the transient M/G/1 queue. 1993, submitted.

[10] CHOUDHURY, G. L., LUCANTONI, D. M. and WHITT, W. Numerical solution of

$M_t/G_t/1$ queues. 1993, submitted.

[11] CHOUDHURY, G. L., LUCANTONI, D. M. and WHITT, W. Numerical transform inversion to analyze teletraffic models. *Proceedings of International Teletraffic Congress 14*, France, 1994, to appear.

[12] CONWAY, A. E. and GEORGANAS, N. D. RECAL: a new efficient algorithm for the exact analysis of multiple-chain closed queueing networks. *J. ACM* 33 (1986) 768-791.

[13] CONWAY, A. E. and GEORGANAS, N. D. Decomposition and aggregation by class in closed queueing networks. *IEEE Trans. Software Eng.* 12 (1986) 1025-1040.

[14] CONWAY, A. E. and GEORGANAS, N. D. *Queueing Networks — Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation.* MIT Press, Cambridge, MA, 1989.

[15] CONWAY, A. E., de SOUZA e SILVA, E. and LAVENBERG, S. S. Mean value analysis by chain of product form queueing networks. *IEEE Trans. Computers* 38 (1989) 432-442.

[16] GORDON, J. J. The evaluation of normalizing constants in closed queueing networks. *Opns. Res.* 38 (1990) 863-869.

[17] HARRISON, P. G. On normalizing constants in queueing networks. *Opns. Res.* 33 (1985) 464-468.

[18] KAUFMAN, J. S. Blocking in a shared resource environment. *IEEE Trans. Commun.* COM-29 (1981) 1474-1481.

[19] KELLY, F. P. *Reversibility and Stochastic Networks*, Wiley, New York, 1979.

[20] KELLY, F. B. Blocking probabilities in large circuit-switched networks. *Adv. Appl. Prob.* 18 (1986) 473-505.

[21] KNESSL, C. and TIER, C. Asymptotic expansions for large closed queueing networks with multiple job classes. *IEEE Trans. Computers* 41 (1992) 480-488.

[22] KOENIGSBERG, E. Cyclic queues. *Opns. Res. Quart.* 9 (1958) 22-35.

[23] KOENIGSBERG, E. Comments on ‘‘On normalization constants in queueing networks,’’ by P. G. Harrison. *Opns. Res.* 34 (1986) 330.

[24] KOGAN, Y. Another approach to asymptotic expansions for large closed queueing networks. *Opns. Res. Letters* 11 (1992) 317-321.

[25] KOGAN, Y. and BIRMAN, A. Asymptotic analysis of closed queueing networks with bottlenecks. preprint.

[26] LAM, S. S. Dynamic scaling and growth behavior of queueing network normalization constants. *J. ACM* 29 (1982) 492-513.

[27] LAM, S. S. and LIEN, Y. L. A tree convolution algorithm for the solution of queueing networks. *Commun. ACM* 26 (1983) 203-215.

[28] LAVENBERG, S. S. (ed.) *Computer Performance Modeling Handbook,* Academic Press, Orlando, FL, 1983.

[29] MCKENNA, J. A generalization of Little’s law to moments of queue lengths and waiting times in closed product-form queueing networks. *J. Appl. Prob.* 26 (1989) 121-133.

[30] MCKENNA, J. and MITRA, D. Integral representations and asymptotic expansions for closed Markovian queueing networks: normal usage. *Bell System Tech. J.* 61 (1982) 661-683.

[31] MCKENNA, J., MITRA, D. and RAMAKRISHNAN, K. G. A class of closed Markovian queueing networks: integral representations, asymptotic expansions, generalizations. *Bell System Tech. J.* 60 (1981) 599-641.

[32] RAMAKRISHNAN, K. G. and MITRA, D. An overview of PANACEA, a software package for analyzing Markovian queueing networks. *Bell System Tech. J.* 61 (1982) 2849-2872.

[33] REIF, F. *Fundamentals of Statistical and Thermal Physics.* McGraw-Hill, New York, 1965.

[34] REISER, M. and KOBAYASHI, H. Queueing networks with multiple closed chains: theory and computational algorithms. *IBM J. Res. Dev.* 19 (1975) 283-294.

[35] REISER, M. and LAVENBERG, S. S. Mean value analysis of closed multichain queueing networks. *J. ACM* 27 (1980) 313-322.

[36] de SOUZA e SILVA, E. and LAVENBERG, S. S. Calculating joint queue-length distributions in product-form queueing networks. *J. ACM* 36 (1989) 194-207.

[37] WHITTLE, P. *Systems in Stochastic Equilibrium,* Wiley, New York, 1986.

[38] WILLIAMS, A. C. and BHANDIWAD, R. A. A generating function approach to queueing network analysis of multiprogrammed computers. *Networks* 6 (1976) 1-22.