Approximate Analysis of Open Exponential Queueing Networks with Blocking : General Configuration

Technical Report 87-13

HYO-SEONG LEE

STEPHEN M. POLLOCK

Department of Industrial and Operations Engineering

The University of Michigan

Ann Arbor, Michigan 48109

May, 1987

ABSTRACT

An arbitrary configuration of an open queueing network with exponential service times and finite buffers is analysed. We offer an iterative procedure for approximating the marginal occupancy probabilities for each queue of the system. The method decomposes the queueing network into individual queues and analyses each in isolation using information from only its nearest neighbors. Based upon the SIMP approximation previously used for tandem queues, it replaces each server's service time with a clearance time(which includes blocking), and each server's arrival rate by an equivalent "acceptance" rate. The procedure is easy to implement and requires modest memory and computer time. Extensive numerical experiments, performed for various topologies, yield accurate results compared with those obtained by exact or simulation methods.

1.Introduction

A queuing network, a set of arbitrarily connected queues, can represent many processes of interest in manufacturing systems, computer systems, telecommunications, etc. If the buffer space between servers is infinite and service times at each queue are exponential, these networks can be exactly analysed by Jackson's decomposition method [13]. Jackson's method, however, ignores an important feature of many real queueing systems, i.e., blocking due to the finiteness of buffer space. In this case the product form property does not hold, and very complicated conditions of dependency exist among the queues; the number of states needed for an exact analysis grows combinatorially with number of queues and buffers. For this reason, most analyses are based on approximation, numerical or simulation methods.

There are various configurations of queueing networks with blocking. The tandem (or serial) network, the most basic structural configuration, has been studied (assuming exponential service times) by Hillier and Boling[12], Caseau and Pujolle[7], Latouche and Neuts[16], Boxma and Konheim[5], Altiok[1], Perros and Altiok[18], Bocharov and Rokhas[4], Brandwajn and Jow[6], Foster and Perros[9]. For non-exponential service times, Gershwin[10] and Choong and Gershwin[8] present algorithms for special service time distributions, representing probabilistic failure and repair of the server. The SIMP approximate procedure of Pollock, Birge and Alden[20] allows for general service time distributions.

Analysis of other configurations, particularly split and merge, have been reported by Boxma and Konheim[5], Altiok and Perros[2] and Lee and Pollock[17]. With the exception of allowing some servers to have general service time distributions in [17], these all assume exponential service times.

The general system, being a combination of tandem, split and merge configurations, is

the most complicated one to analyse. Takahashi, Miyahara and Hasegawa[21] assumed that "effective" service times follow an exponential distribution, and developed a set of simultaneous non-linear equations that must be solved to get performance measures. Labetoulle and Pujolle[15] and Kerbache and Smith[14], allowing for non-exponential service times, use a diffusion approximation that may restrict its validity[19]. Perros and Altiok[3] use phase-type distributions for approximately characterizing effective service times. Their procedure appears to be restricted to small networks due to the inherent complexity of the phase-type mechanism. Recently, Perros and Snyder[19] developed a similar algorithm, using a two-phase Coxian distribution to approximate effective service times, as an improvement over [3]. However, this algorithm is not accurate in important boundary cases, such as when queues receiving exogenous inputs have very large buffers. In these previous analyses, as with the work presented in this paper, the networks are restricted to have no feedback loops.

In a tandem queueing network, representing a manufacturing system, the assumption that service times have an exponential distribution is not realistic, although it might be quite suitable for modelling computer systems. However, general queueing network models of manufacturing systems usually represent job-shop situations, where various types of product are produced on common facilities. Since each product type has a different service time at the same facility, the overall service time at each facility may be reasonably represented by an exponential distribution, even though the service time for one particular product is definitely not.

In this paper, we present an approximation method for analysing the general configuration of an open queueing network with blocking. This algorithm is based on two earlier algorithms; one proposed by Pollock, Birge and Alden[20] for tandem queues and the other by Lee and Pollock[17] for merge queues. This new algorithm solves a large network quickly, and yields robust and accurate results.

2.Description of the Network and Formulation of the Problem

The network we consider is identical to that in Altiok and Perros[3] and Perros and Snyder[19] except that we also allow external arrivals at any server. It consists of the set $\{ i: i=1,2,\cdots,M\}$ of single server queues, connected arbitrarily via arcs (i,j) with the restriction that there is no directed cycle. Since there is no directed cycle, we can number each queue in such a way that every arc (i, j) has i less than j. The service time at queue i follows an exponential distribution with rate μ_i and external arrivals to queue i are independent Poisson processes with rate λ_i. The buffer size of the i-th queue is B_i, and its capacity (including the one in service) is $N_i = 1 + B_i$. Units at each queue are served in a FIFO manner. If an external arrival encounters a queue i when it is full, the arrival is simply lost. A unit which has completed service at queue i gets its next service at queue j with "routing probability" r_{ij}. The probability that a unit leaves the queueing system after completing service at queue i is r_{i0}. Fig. 1 shows an example consisting of four queues.

Suppose a unit has just finished service at queue i and the next service required is at queue j. If queue j has no available space at that time, the unit must wait in the i-th server until it can enter queue j. During this time the i-th server cannot serve other units that might be waiting in its buffer: in this case, the i-th server is said to be <u>blocked</u> and j-th queue is <u>blocking</u>. Note that in Figure 1, queue 1 cannot be blocking and queue 4 cannot be blocked.

<Insert Fig. 1>

One difficulty in the analysis comes from the fact that a queue may be simultaneously blocking more than one "upstream" queue. We assume blocked units enter the destination queue on a "First-Blocked-First-Enter" basis [3]. Suppose k queues merge into queue j: any combination of these can be blocked simultaneously. In the worse case, all k queues can be blocked by queue j at the same time.

Since a unit blocked by queue j is ready to proceed to queue j whenever there is space in the buffer of queue j, it is effectively waiting in line to be served by server j. Therefore we can interpret the server position of a blocked unit to be part of the buffer capacity of the blocking queue. In this way, we can consider the capacity of each queue to be augmented by the number of upstream queues directly connected to it, so that the effective capacity of queue j is N_j+k. In the next section, we develop a procedure that exploits the augmented buffer size for each queue.

3. Analysis of the Model

3.1. Approach and General Relationships

Our general approach is to focus on the measures that seem to be important (for example, individual queue steady state probabilities) and to seek an approximation that will produce these measures fairly accurately. To do this, we analyse each individual queue separately, using information from only its nearest neighbors. We also use the important fact that the time for a unit to clear service, or the clearance time, has two components: the actual service time plus a term due to the occasional and probabilistic delay caused by blocking.

The first step, in approximating by a simple model what is, in fact, a very complicated and dependent state of affairs, is to make the following gross assumptions:

a) Arrivals from queue i to queue j are Poisson with effective rate λ^*_{ij}, as long as the

i-th queue is not blocked by queue j. When the i-th queue is blocked by queue j, there is of course no arrival from queue i to queue j.

b) The service clearance time (having two components) is exponentially distributed, with effective rate μ_i^*.

c) A unit at server i, at the instant service is completed, sees the destination queue in steady state.

The relation between the effective rates (λ_{ij}^* and μ_i^*) and the actual rates (λ_i and μ_i) is discussed below.

These conditions are, of course, far different from what actually happens in the system: a) is clearly not true since the outputs of each queue are anything but Poisson. We anticipate, however, that Poisson assumption may be approximately true if the service time at each queue is exponential; b) is also a heroic assumption since the clearance time, in general, being the sum of two random variables does not follow an exponential distribution. However, our previous experience with tandem queues [20,17] shows that this does not degrade results as compared to those obtained by allowing the clearance time to have a general distribution; c) allows the use of well-known steady state analyses, and does not appear to be crucial – particularly in contrast to the oversimplifications of a) and b).

3.2. Definitions and Underlying Relationships

We define (where, unless otherwise stated, the index i always runs from 1 to M)

$S_i \equiv$ service time at server i (excluding any delay due to blocking),

$T_i \equiv$ clearance time for server i, i.e., the time between when a unit enters service in queue i, and when it leaves queue i,

F_i = predecessor set of queue i $\equiv$ { k : queue k can pass units directly to queue i },

$k_i = |F_i| \equiv$ number of upstream queues directly connected to queue i,

B_i = successor set of queue i $\equiv$ { k : queue k can receive units directly from queue i },

$\bar{\lambda}_{ij} \equiv$ flow rate from queue i to queue j $\quad i=1,\cdots,M-1, \quad j\in B_i,$

$\bar{\lambda}_{0i}$ ($\bar{\lambda}_{i0}$) $\equiv$ flow rate from outside the system to queue i (from queue i to outside the system),

$\lambda^*_{ij} \equiv$ arrival rate to queue j from queue i as long as queue i is not blocked by queue j $\quad i=1,\cdots,M-1, \quad j\in B_i,$

$P_i(k) \equiv$ steady state probability that there are k units at queue i,

$b_{ij}(n) \equiv$ probability that n units are blocked by queue j including one at queue i, $\quad j=2,\cdots,M, \quad i\in F_j,$

$\alpha_{ij}(k) \equiv$ conditional probability that, upon service completion at server i, a unit which has queue j as its destination sees k units at queue j $\quad j=2,\cdots,M, \quad i\in F_j,$

$f_i \equiv$ Probability {i-th queue is full}

These definitions, assumptions a), b) and c), and the structure of the system, produce the following relationships:

1) Since the buffer space of each queue is augmented by the number of upstream queues directly connected to it, the probability that queue i is full is

$$f_i = \sum_{n=0}^{k_i} P_i(N_i+n) \tag{3.1}$$

2) Since units cannot enter from outside of the system to queue i unless it is not full, the flow rate from outside is given by,

$$\bar{\lambda}_{0i} = \lambda_i(1 - f_i) \tag{3.2}$$

3) The total flow rate into queue j, denoted by $\bar{\lambda}_j$, is

$$\bar{\lambda}_j = \sum_{i=0}^{j-1} \bar{\lambda}_{ij} \tag{3.3}$$

(For those queues i which have only inputs from outside, so that $F_i = \phi$, the flow rate is

$$\bar{\lambda}_i = \bar{\lambda}_{0i} = \lambda_i(1\text{-}f_i) \qquad \{ i : F_i = \phi \} \tag{3.4}$$

4) By the conservation of average flow, the flow rate from queue i to queue j (or to outside of the system) is given by

$$\bar{\lambda}_{ij} = \bar{\lambda}_i \, r_{ij} \tag{3.5}$$

5) The total arrival rate to queue j when queue j is not full is

$$\lambda_j^* \equiv \sum_{i=1}^{j-1} \lambda_{ij}^* + \lambda_j \tag{3.6}$$

6) The expected clearance time at queue i, given that queue j is the destination, is given by

$$E(T_i | j) = \frac{1}{\mu_i} + \sum_{n=0}^{k_j - 1} (n+1)\alpha_{ij}(N_j + n)E(T_j) \tag{3.7}$$

The first term in (3.7) represents the service time at queue i. The second term is the expected delay time due to blocking. If a unit whose destination is queue j sees n other units blocked by queue j at the instant of its service completion at queue i, it must wait n (independent) clearance times plus one residual clearance time at queue j before it feeds into queue j. (By assumption b), the residual clearance time is simply another clearance time)

Since the probability that queue j will be the destination queue is r_{ij}, the expected clearance time for queue i is

$$E(T_i) = \sum_{j \in B_i} r_{ij} E(T_i | j) \tag{3.8.a}$$

If queue i has no directly connected downstream queues, i.e., $B_i = \phi$,

$$E(T_i) = E(S_i) \tag{3.8.b}$$

7) The service completion rate at queue i is the reciprocal of the expected clearance time. Thus,

$$\mu_i^* = \frac{1}{E(T_i)} \tag{3.9}$$

3.3. Analysis of Model

Analysis given effective arrival rates and expected clearance times

We are now in a position to analyse each queue of the network. Any queue which has no directly connected upstream queue is easy to analyse, for example, consider queue 1 in Figure 1. Suppose we know the value of $E(T_1)$. Then we can analyse queue 1 by using a $M/M/1/N_1$ model since we assume that its clearance time follows an exponential distribution, and the arrivals to queue 1 are Poisson. Therefore, the occupancy probabilities of queue 1 can be obtained as:

$$P_1(j) = \frac{(1 - \rho_1)\rho_1^j}{1 - \rho_1^{N_1+1}} \qquad j=0,1,\cdots,N_1 \tag{3.10}$$

where $\rho_1 = \lambda_1 E(T_1)$. The probability that queue 1 is full is

$$f_1 = P_1(N_1) \tag{3.11}$$

It is more difficult to analyse queues which have directly connected upstream queues. In order to obtain the occupancy and blocking probabilities of such queues, we use the

procedure developed in [17] for merge queues, as outlined briefly below.

Consider queue j which has k_j directly connected upstream queues. Assume we know $E(T_j)$ and thus μ_j^* from (3.9). Define the state of queue j to be the number of units in queue j if it is not blocking. If queue j is blocking, the state is defined to be $(N_j+n,\mathbf{v})$, $n=1,\cdots,k_j$, where N_j+n represents the number of units in queue j, including blocked ones, and $\mathbf{v}$ is the n-component vector representing the units which are being blocked, in order. In order to obtain the blocking probability $b_{ij}(n)$, we must find the occupancy probabilities for each of these ordered states. Ordinarily, these occupancy probabilities can be obtained only by solving a very large set of steady state balance equations, which are impracticable to solve if k_j is fairly large. However, as shown in [17], we can obtain these occupancy probabilities very simply by considering an equivalent aggregated state space, and its associated simple birth-and-death equations. The aggregated state is the number of units in queue j, disregarding the order of units being blocked, so that blocking states which have the same number of units are aggregated into one state. Let $\hat{\lambda}_j(n)$ denote the arrival rate to state $n+1$ from state n in this aggregation. The following theorem allows us to compute appropriate values for the $\hat{\lambda}_j(n)$.

Theorem 3.1. The aggregated states of queue j are equivalent (in terms of producing identical $P_j(i)$) to the original states of queue j if the arrival rates to the aggregated states are:

$$\hat{\lambda}_j(i) = \lambda_j^* \qquad \text{for } i=0,\cdots,N_j$$

$$\hat{\lambda}_j(N_j+n) = \frac{(n+1)\Omega_{n+1}}{\Omega_n} \qquad \text{for } n=1,\cdots,k_j-1 \tag{3.12}$$

where $\Omega_k \equiv \sum_{i_1<\cdots<i_k} \prod_{j=1}^{k} \lambda^*_{i_j}, \quad i_j \in F_j$

Once we obtain the occupancy probabilities of the aggregated states, we can find the occupancy probabilities of the original states by the following theorem:

Theorem 3.2. The relationship between the occupancy probabilities of the original states and those of aggregated states is:

$$P_j(N+n,i_1\cdots i_n) = \frac{\prod_{j=1}^{n} \lambda^*_{i_j}}{n!\ \Omega_n} P_j(N+n) \qquad \text{for } n=1,\cdots,k_j \tag{3.13}$$

These theorems and their proofs are exactly the same as in [17], with the modest extension that the network here has external arrivals to queue j and so equation (3.6) is used. For more detail on the analysis of merged queues and proof of the theorems, refer to [17].

The aggregated occupancy probability $P_j(i)$ needed to obtain $P_j(N_j+n,i_1\cdots i_n)$ in (3.13) can be found trivially from the simple birth and death formulae:

$$P_j(i) = P(0) \prod_{j=1}^{i} \frac{\hat{\lambda}_j(j)}{\mu} \qquad i=1,2,\cdots,N_j+k_j \tag{3.14.a}$$

$$\sum_{i=0}^{N_j+k_j} P_j(i) = 1 \tag{3.14.b}$$

Finding effective arrival rates

In order to use these results, however, we need to know the values of λ^*_{ij} (so that the λ^*_i can be gotten from 3.6). If we have available the values of $\bar{\lambda}_{ij}$ and $b_{ij}(n)$, however, then the λ^*_{ij} can be gotten from

$$\lambda_{ij}^{*}(1 - \sum_{n=1}^{k_j} b_{ij}(n)) = \bar{\lambda}_{ij}, \tag{3.15}$$

a conservation equation which yields the arrival rate λ_{ij}^{*} needed in order to produce the given value $\bar{\lambda}_{ij}$.

Finding effective expected clearance times

Once we have values of $P_j(N_j+n,i_1\cdots i_n)$ from (3.13), we can obtain $b_{ij}(n)$, the probability{n units are blocked by queue j including one at queue i} from:

$$b_{ij}(n) = \frac{\lambda_{ij}^{*}\Omega_{n-1\backslash i}}{\Omega_n} P_j(N+n) \qquad n=1,\cdots,k_j, \qquad i\in F_j \tag{3.16}$$

where $\Omega_{n-1\backslash i} \equiv \sum_{i_1<\cdots<i_{n-1}} \prod_{j=1}^{n-1} \lambda_{i_j}^{*}$, $i_j \in F_j\backslash\{i\}$

From $b_{ij}(n)$, we can now compute the conditional probability $\alpha_{ij}(n)$ that, upon service completion at server i, a unit which has queue j as its destination queue sees k units at queue j. From assumption c) in section 3.1 and the fact that a unit cannot be served (and therefore cannot have completed service) at queue i if queue i is blocked by queue j, we see that $\alpha_{ij}(N_j+n)$ is the conditional probability that there are N_j+n units at queue j given that queue i is not blocked by queue j. Since the probability that queue i is blocked by queue j is $\sum_{n=1}^{k_j} b_{ij}(n)$,

$$\alpha_{ij}(N_j+n) = \frac{P_j(N_j+n)-b_{ij}(n)}{1-\sum_{n=1}^{k_j} b_{ij}(n)} \qquad n=0,\cdots,k_j-1, \qquad i\in F_j \tag{3.17}$$

where $b_{ij}(0)$ is defined to be 0. We now can use this value of $\alpha_{ij}(N_j+n)$ in equation (3.7) to obtain $E(T_i|\, j\,)$.

The occupancy probabilities of each queue can now be found by an iterative procedure. Each iteration consists of two sets of calculations: the effective arrival rates λ^*_{ij} are calculated in forward order and occupancy probabilities and $E(T_i|\, j\,)$ are calculated in backward order.

If external arrivals occur at only the first queue, only a single set of calculations needed. However, if more than one queue has external arrivals, a two way analysis is unavoidable for the following reason. Suppose more than one queue has external arrivals. The analysis of queue i ($B_i \neq \phi$) yields an updated value of f_i, as well as other values of performance measures, and this updated value of f_i, in turn, gives the updated $\bar{\lambda}_{0i}$ and $\bar{\lambda}_i$. But the updated value of $\bar{\lambda}_i$ affects λ^*_{ij} for all the upstream queues, which can change the occupancy probabilities of upstream queues already obtained. In order to avoid this dilemma, we use a two way analysis:

Forward calculation

Each iteration begins with queue 1 and proceeds to queue 2,3,···,M. f_1, the probability that queue 1 is full, is gotten using (3.10), (3.11); the values of $\bar{\lambda}_{1j}$ come from (3.5) for all $j \in B_1$. From $\bar{\lambda}_{1j}$ and $b_{1j}(n)$ (obtained in a previous backward analysis), we can find λ^*_{1j} for all $j \in B_1$ using equation (3.15). Then, we consider queue 2. At queue 2, since we have available λ^*_{i2} for all $i \in F_2$, we can find f_2, the probability that queue 2 is full, which makes it possible to obtain $\bar{\lambda}_{02}$. Now we can find $\bar{\lambda}_2$ using (3.3) and $\bar{\lambda}_{2j}$ for all $j \in B_2$ using (3.5). From $\bar{\lambda}_{2j}$ and $b_{2j}(n)$ (obtained in a previous backward analysis), we find λ^*_{2j} for all $j \in B_2$. Then queue 3 is considered, etc., until the values of λ^*_{iM} are obtained for all $i \in F_M$.

Backward calculation

We start from queue M, and obtain the occupancy probabilities and $E(T_i|\, j\,)$ for each queue. Since queue M is never blocked $E(T_M)$ is always equal to $E(S_M)$. Using μ^*_M and λ^*_{iM} (obtained in a previous forward analysis), we can find $E(T_i|M)$ for all $i \in F_M$ as well as

the occupancy probabilities of queue M. Then consider queue M-1. At queue M-1, we first obtain $E(T_{M-1})$ (and therefore, μ^*_{M-1}) using equation (3.8.a). From this updated μ^*_{M-1} and λ^*_{iM-1} (obtained from a previous forward analysis), we can obtain occupancy probabilities of queue M-1, and $E(T_i|M\text{-}1)$ for $i \in F_{M-1}$. This procedure continues until $E(T_1)$ is obtained.

At the end of the backward analysis, we check whether the convergence condition is satisfied, by using a suitable comparison, such as the values of the $E(T_i)$, for successive iterations. If convergence does not occur, another iteration is performed.

Note that, in the forward analysis, occupancy probabilities of disaggregated states are not obtained, since the only occupancy probability computed is f_i from (3.12) and (3.14). Thus the computational effort of the two way analysis is not critically increased over that needed for the one way analysis.

3.3.1. Approximate Algorithm 1

The analysis above is incorporated into the following iterative algorithm to obtain an approximate solution to the system's steady state probabilities.

0. (Set-up) Set the values of λ_i, μ_i, N_i for $i=1,\cdots,M$

1. (Initialization – The conditions here are as if all queues are unblocked.)
   Set $E(T_i) = \frac{1}{\mu_i}$ for $i=1,\cdots,M$ $\rho_1 = \lambda_1 E(T_1)$,
   Find f_1 using (3.10) and (3.11)
   Set $\bar{\lambda}_1 = \lambda_1(1 - f_1)$
   Find $\bar{\lambda}_{1j}$ using (3.5) for all $j \in B_1$
   Set $\lambda^*_{1j} = \bar{\lambda}_{1j}$ for all $j \in B_1$
   For i=2, M-1 do
   begin
   find $\hat{\lambda}_i(n)$ using (3.12)

find $P_i(n)$ using (3.14)
find f_i using (3.1)
find $\bar{\lambda}_{0i}$ using (3.2)
find $\bar{\lambda}_i$ using (3.3)
find $\bar{\lambda}_{ij}$ using (3.5) for all $j \in B_i$
set $\lambda^*_{ij} = \bar{\lambda}_{ij}$ for all $j \in B_i$
end

2. (Backward Analysis – Find $E(T_i|j)$ and occupancy probabilities for each queue)
For $j = M, 2, -1$ do
begin
find $E(T_j)$ using (3.8)
find $\hat{\lambda}_j(n)$ using (3.12) for $n = 0, \cdots, N_j + k_j - 1$
find $P_j(n)$ using (3.14) for $n = 0, \cdots, N_j + k_j$
find $b_{ij}(n)$ using (3.16) for all $i \in F_j$
find $\alpha_{ij}(N_j + n)$ using (3.17) for all $i \in F_j$
find $E(T_i|j)$ using (3.7) for $i \in F_j$
end
Find $E(T_1)$ using (3.8)

3. (Convergence check)
If updated values of $E(T_i)$ show little change from the previous ones for all i (i.e., convergence) go to step 5. Else, go to step 4.

4. (Forward analysis – find λ^*_{ij} for each queue)
Set $\rho_1 = \lambda_1 E(T_1)$,
Find f_1 using (3.10) and (3.11)
Set $\bar{\lambda}_1 = \lambda_1(1 - f_1)$
Find $\bar{\lambda}_{1j}$ using (3.5) for all $j \in B_1$
Find λ^*_{1j} using (3.15) for all $j \in B_1$
For $i = 2, M - 1$ do
begin
find $\hat{\lambda}_i(n)$ using (3.12)
find $P_i(n)$ using (3.14)
find f_i using (3.1)
find $\bar{\lambda}_{0i}$ using (3.2)
find $\bar{\lambda}_i$ using (3.3)
find $\bar{\lambda}_{ij}$ using (3.5) for all $j \in B_i$
find λ^*_{ij} using (3.15) for all $j \in B_i$

end
go to step 2

5. (Calculate occupancy probabilities)
For queue 2 through M, these have already been obtained in step 2. For queue 1, find $P_1(n)$ using (3.10) for $n=0,\cdots,N_1$

3.4 Approximate Algorithm 2

If external arrivals occur only at the first queue, only the backward analysis is needed in every iteration because all $\bar{\lambda}_{ij}$ are completely determined by f_1 which is obtained at the end of the backward analysis. The occupancy probabilities are then obtained from the following simplified algorithm.

0. (Set-up) Same as in algorithm 1

1. (Initialization – The conditions here are as if all queues are unblocked.)
Set $E(T_i) = \frac{1}{\mu_i}$ for $i=1,\cdots,M$ $\rho_1 = \lambda_1 E(T_1)$,
Find f_1 using (3.10) and (3.11)
Set $\bar{\lambda}_1 = \lambda_1(1 - f_1)$
Find $\bar{\lambda}_{1j}$ using (3.5) for all $j\in B_1$
Set $\lambda^*_{1j} = \bar{\lambda}_{1j}$ for all $j\in B_1$
For i=2,M-1 do
begin
find $\bar{\lambda}_i$ using (3.3)
find $\bar{\lambda}_{ij}$ using (3.5) for all $j\in B_i$
Set $\lambda^*_{ij} = \bar{\lambda}_{ij}$ for all $j\in B_i$
end

2. (Backward Analysis) Same as step 2 in algorithm 1

3. (Convergence check) Same as step 3 in algorithm 1

4. (Find λ^*_{ij} for each queue)
Set $\rho_1 = \lambda_1 E(T_1)$,

Find f_1 using (3.10) and (3.11)

Set $\bar{\lambda}_1 = \lambda_1(1 - f_1)$

Find $\bar{\lambda}_{1j}$ using (3.5) for all $j \in B_1$

Find $\overset{*}{\lambda}_{1j}$ using (3.15) for all $j \in B_1$

For i=2, M − 1 do

begin

find $\bar{\lambda}_i$ using (3.3)

find $\bar{\lambda}_{ij}$ using (3.5) for all $j \in B_i$

find $\overset{*}{\lambda}_{ij}$ using (3.15) for all $j \in B_i$

end

go to step 2

5. (Calculate occupancy probabilities) Same as step 5 in algorithm 1

4. Computational Results

In order to test the accuracy of our approximation method, the algorithm was implemented on IBM 3090-400 and tested on a variety of problems. Tables 1-8 give comparisons with three- to eight-node network problems in the literature, all of which have only one queue with external arrivals. In those cases where exact solutions have not been obtained, we use simulation results reported by previous authors.

Table 1 gives comparisons for the triangular network of fig. 2, as reported in Takahashi et. al [21] and Altiok and Perros[3]. Arrivals are at queue 1 with rate 1, and every queue has a buffer of size one. The routing probabilities are $r_{10}=0$, $r_{12}=r_{13}=0.5$, $r_{23}=1$. Comparisons are based on $P_1(N_1)$, which determines the throughput of the system since other queues do not have external arrivals. As can be seen in the table, our method performs better than Takahashi's method and is comparable with Altiok and Perros' method. We also note that Altiok and Perros' method underestimates $P_1(N_1)$, the probability queue 1 is full, if the service rates are low (e.g., μ=1,1.1,1.2) and

overestimates it if the service rates are high (e.g., $\mu = 1,2,3$). This pattern suggests that it might have a larger error for very high or low service rates, even though in the intermediate range shown their method is accurate. On the other hand, our method appears to be more robust in that it shows a consistent pattern of overestimating throughput for all explored service rates, by fairly small deviations, i.e., 0.004 − 0.005 in absolute error.

<Insert Fig. 2,3>

Table 1
Approximations to $P_1(N_1)$ from Takahashi et al., Altiok and Perros, and our algorithm

μ_1	μ_2	μ_3	Exact	Altiok & Perros	Takahashi	Algo 2.
1	1.1	1.2	0.55963	0.54698	0.58669	0.56301
1	1.2	1.4	0.54634	0.53736	0.57344	0.55020
1	1.3	1.6	0.53681	0.53049	0.56324	0.54094
1	1.4	1.8	0.52980	0.52541	0.55538	0.53404
1	1.5	2.0	0.52451	0.52153	0.54904	0.52876
1	1.6	2.2	0.52043	0.51850	0.54398	0.52462
1	1.7	2.4	0.51724	0.51608	0.53975	0.52133
1	1.8	2.6	0.51469	0.51411	0.53619	0.51866
1	1.9	2.8	0.51264	0.51250	0.53318	0.51646
1	2.0	3.0	0.51096	0.51115	0.53058	0.51464

Our method was also tested for nine other three-node network problems, as well as the four four-node network problems and ten eight-node network problems analysed in [3] and [19]. Figs. 1, 2 and 3 show the topologies of these networks. Tables 3–8 present some numerical results selected from among these problems, showing the average (and maximum) absolute deviations and average (and maximum) relative errors with respect to exact or simulated values. Since Altiok and Perros' algorithm cannot solve the eight node-

network due to memory limitations, we compare our results in this case with only those of Perros and Snyder's.

Table 2 is a summary of the comparisons with Altiok and Perros' and Perros and Snyder's for the averages of nine three-node network problems, four four-node network problems and ten eight-node network problems, for the various performance measures in table 3–8. As can be seen, our algorithm give better results in both average(maximum) absolute deviation and average(maximum) relative error. In addition, while Perros and Snyder's algorithm is inaccurate if the first queue has infinite buffers (see for example table 6), ours appears to work quite well for all cases shown.

Table 2
Summary of Comparisons with the approximations of
Altiok and Perros, and Perros and Snyder

Network Configuration	Measures	Altiok and Perros	Perros and Snyder	New
3 node network (avg. of 9 prob.)	avg. abs. dev.	0.0095	0.0082	0.0066
	max. abs. dev.	0.0322	0.0201	0.0152
	avg. rel. err.	0.0545	0.0547	0.0311
	max. rel. err.	0.1991	0.2132	0.1019
4 node network (avg. of 4 prob.)	avg. abs. dev.	0.0213	0.0198	0.0135
	max. abs. dev.	0.0442	0.0442	0.0257
	avg. rel. err.	0.0679	0.0638	0.0433
	max. rel. err.	0.1691	0.1401	0.1045
8 node network (avg. of 10 prob.)	avg. abs. dev.	–	0.014	0.007
	max. abs. dev.	–	0.043	0.020
	avg. rel. err.	–	0.079	0.045
	max. rel. err.	–	0.338	0.229

It is important to state that we have not proven the convergence of our algorithm. However, in all of the problems we have tested to date, we have not found any which did not converge. The maximum number of iterations needed for convergence was

11; in most cases convergence to one part in 10^5 occurred within 4–7 iterations. From a practical point of view, we note that the maximum CPU time required for an eight node network problems was 0.008 seconds. We also do not have a-priori bounds on the accuracy of the method; these are currently being explored.

Conclusions

We have presented a new approximate algorithm for analysing a general configuration of an open queueing network with blocking. Besides being accurate and fast, our algorithm has the following advantages over those previously reported:

Generality : It can solve not only networks with a large number of servers, but also general topologies including external arrivals at more than one queue.

Robustness: It yields accurate results regardless of whether the queues with external arrivals have infinite buffers or not, or whether the service rates are high or low.

Simplicity: There are no numerical procedures involving the solution of simultaneous nonlinear equations or fixed point problems.

Considering the generality, robustness, simplicity and accuracy of the algorithm, and its significant improvement over previous methods, it holds promise to be a useful tool in the study of networks of queues.

Acknowledgement

This study was supported in part by a contract from the General Motors Corporation to the University of Michigan, for the development of analytical tools for production

systems. We wish, in particular, to thank Jeff Alden and Larry Burns, of G.M., for their continuing interest and support.

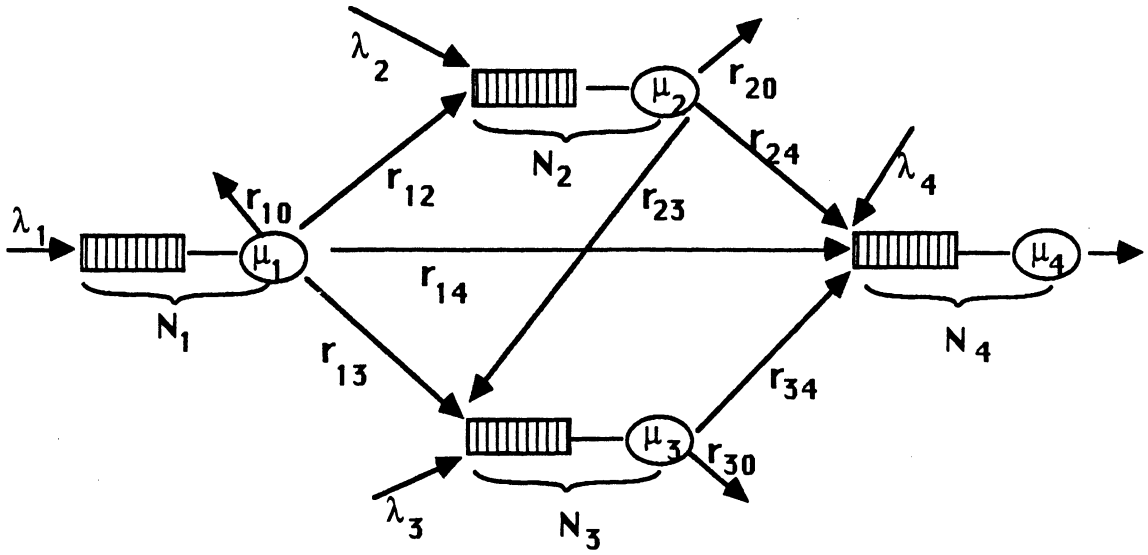

Figure 1. the four node network

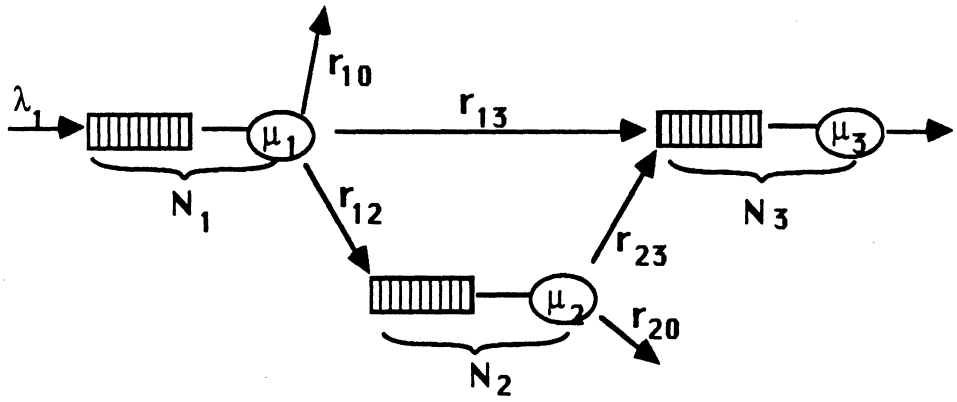

Figure 2. the three node network

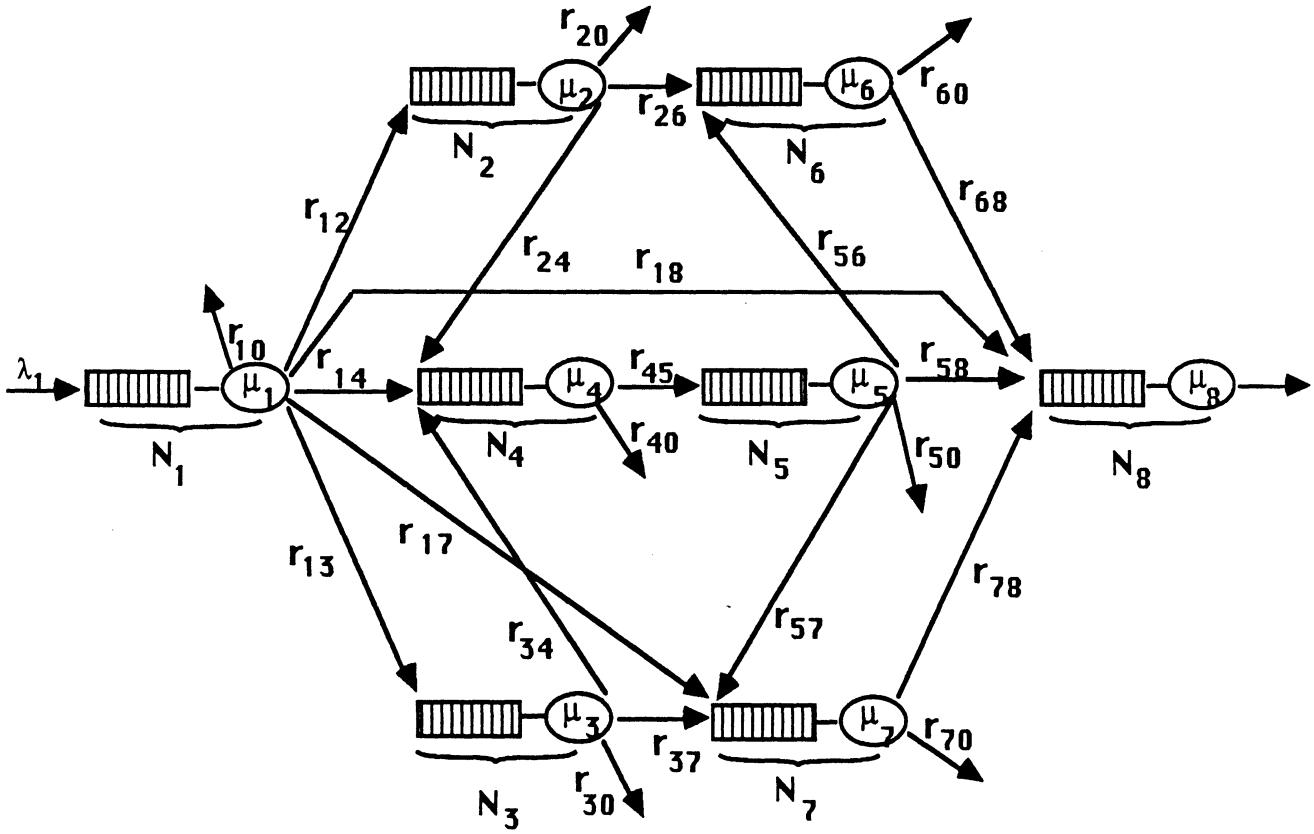

Figure 3. the eight node network

Table 3
Comparison with the approximations of Altiok and Perros, and Perros and Snyder for a three node network

Measures	Exact	Altiok and Perros	Perros and Snyder	New
$P_1(0)$	0.1078	0.1261	0.1248	0.1048
$P_1(1)$	0.0946	0.1096	0.1111	0.0938
$P_1(2)$	0.0836	0.0956	0.0961	0.0840
$P_1(3)$	0.0743	0.0835	0.0833	0.0752
$P_1(4)$	0.0662	0.0731	0.0724	0.0673
$P_1(5)$	0.0592	0.0639	0.0632	0.0603
$P_2(0)$	0.6231	0.6489	0.6490	0.6161
$P_2(1)$	0.2401	0.2294	0.2311	0.2399
$P_2(2)$	0.0919	0.0818	0.0805	0.0934
$P_2(3)$	0.0449	0.0399	0.0395	0.0506
$P_3(0)$	0.4563	0.4560	0.4561	0.4560
$P_3(1)$	0.2684	0.2608	0.2608	0.2679
$P_3(2)$	0.2753	0.2832	0.2831	0.2761
avg. abs. deviation	0.0000	0.0102	0.0102	0.0018
max. abs. deviation	0.0000	0.0258	0.0259	0.0070
avg. rel. error	0.0000	0.0876	0.0880	0.0192
max. rel. error	0.0000	0.1698	0.1744	0.1269

$\lambda = (0.8,0,0)$ $\mu = (1,1,1)$ $N = (\infty,3,2)$
$r_{10}=0.2$, $r_{12}=0.4$, $r_{13}=0.4$, $r_{20}=0.3$, $r_{23}=0.7$
CPU time = 0.001 second

Table 4
Comparison with the approximations of Altiok and Perros,
and Perros and Snyder for a three node network

Measures	Exact	Altiok and Perros	Perros and Snyder	New
$P_1(0)$	0.2154	0.2142	0.2135	0.2092
$P_1(1)$	0.7846	0.7858	0.7865	0.7909
$P_2(0)$	0.7051	0.7231	0.7241	0.6968
$P_2(1)$	0.2949	0.2770	0.2759	0.3032
$P_3(0)$	0.6123	0.6144	0.6158	0.6235
$P_3(1)$	0.3877	0.3856	0.3842	0.3765
avg. abs. deviation	0.0000	0.0071	0.0081	0.0086
max. abs. deviation	0.0000	0.0180	0.0190	0.0112
avg. rel. error	0.0000	0.0170	0.0196	0.0207
max. rel. error	0.0000	0.0607	0.0644	0.0289

λ = (3.0,0,0) μ = (1,1,1) N = (1,1,1)
r_{10}=0.2, r_{12}=0.4, r_{13}=0.4, r_{20}=0.5, r_{23}=0.5
CPU time = 0.002 seconds

Table 5
Comparison with the approximations of Altiok and Perros,
and Perros and Snyder for a four node network

Measures	Exact	Altiok and Perros	Perros and Snyder	New
$P_1(0)$	0.0208	0.0246	0.0174	0.0207
$P_1(1)$	0.1370	0.1405	0.1448	0.1325
$P_1(2)$	0.8422	0.8349	0.8379	0.8467
$P_2(0)$	0.6001	0.6266	0.6371	0.5931
$P_2(1)$	0.2636	0.2391	0.2405	0.2530
$P_2(2)$	0.1363	0.1363	0.1225	0.1539
$P_3(0)$	0.6311	0.6605	0.6618	0.6253
$P_3(1)$	0.2533	0.2228	0.2267	0.2426
$P_3(2)$	0.1156	0.1167	0.1115	0.1321
$P_4(0)$	0.2764	0.2429	0.2570	0.2976
$P_4(1)$	0.2436	0.2056	0.2114	0.2410
$P_4(2)$	0.4800	0.5515	0.5316	0.4615
avg. abs. deviation	0.0000	0.0225	0.0212	0.0100
max. abs. deviation	0.0000	0.0715	0.0516	0.0212
avg. rel. error	0.0000	0.0797	0.0813	0.0453
max. rel. error	0.0000	0.1827	0.1635	0.1427

$\lambda = (5,0,0,0)$ $\mu = (1,1,1,1)$ $N = (2,2,2,2)$
$r_{10}=0.05$, $r_{12}=0.35$, $r_{13}=0.30$, $r_{14}=0.30$, $r_{20}=0.05$,
$r_{23}=0.05$, $r_{24}=0.90$, $r_{30}=0.05$, $r_{34}=0.95$
CPU time = 0.003 seconds

Table 6
Comparison with the approximations of Perros and Snyder for an eight node network

Measures	Simulation	Perros and Snyder	New
$P_1(0)$	0.099	0.262	0.092
$P_1(1)$	0.071	0.191	0.083
$P_1(2)$	0.056	0.129	0.076
$P_1(3)$	0.048	0.090	0.069
$P_1(4)$	0.041	0.065	0.062
$P_1(5)$	0.037	0.048	0.057
$P_2(0)$	0.614	0.656	0.597
$P_2(1)$	0.235	0.233	0.252
$P_2(2)$	0.151	0.110	0.151
$P_3(0)$	0.607	0.656	0.599
$P_3(1)$	0.244	0.233	0.251
$P_3(2)$	0.149	0.111	0.150
$P_4(0)$	0.566	0.618	0.560
$P_4(1)$	0.240	0.239	0.255
$P_4(2)$	0.195	0.143	0.186
$P_5(0)$	0.461	0.600	0.420
$P_5(1)$	0.262	0.250	0.277
$P_5(2)$	0.277	0.150	0.303
$P_6(0)$	0.493	0.595	0.484
$P_6(1)$	0.282	0.249	0.266

Measures	Simulation	Perros and Snyder	New
$P_6(2)$	0.224	0.156	0.249
$P_7(0)$	0.405	0.471	0.414
$P_7(1)$	0.273	0.273	0.263
$P_7(2)$	0.322	0.256	0.323
$P_8(0)$	0.202	0.200	0.200
$P_8(1)$	0.189	0.177	0.199
$P_8(2)$	0.609	0.623	0.601
avg. abs. deviation	0.000	0.050	0.013
max. abs. deviation	0.000	0.163	0.041
avg. rel. error	0.000	0.348	0.108
max. rel. error	0.000	1.690	0.541

λ = (5.0.0,0,0,0,0,0) μ = (4,1,1,2,2,2,2,3.5)
N = (2,2,2,2,2,2,2,2)
$r_{10}=0.0$ $r_{12}=0.2$ $r_{13}=0.2$ $r_{14}=0.2$ $r_{17}=0.2$ $r_{18}=0.2$
$r_{20}=0.0$ $r_{24}=0.5$ $r_{26}=0.5$ $r_{30}=0.0$ $r_{34}=0.5$ $r_{37}=0.5$
$r_{40}=0.0$ $r_{45}=1.0$ $r_{50}=0.0$ $r_{56}=0.3$ $r_{57}=0.3$ $r_{58}=0.4$
$r_{60}=0.0$ $r_{68}=1.0$ $r_{70}=0.0$ $r_{78}=1.0$ $r_{80}=1.0$
CPU time = 0.005 seconds

Table 7
Comparisons with the approximations of Perros and Snyder for an eight node network

Measures	Simulation	Perros and Snyder	New
$P_1(0)$	0.321	0.323	0.299
$P_1(1)$	0.313	0.335	0.332
$P_1(2)$	0.366	0.342	0.369
$P_2(0)$	0.601	0.593	0.590
$P_2(1)$	0.254	0.254	0.254
$P_2(2)$	0.146	0.153	0.156
$P_3(0)$	0.584	0.587	0.570
$P_3(1)$	0.261	0.256	0.259
$P_3(2)$	0.155	0.157	0.171
$P_4(0)$	0.564	0.551	0.560
$P_4(1)$	0.258	0.253	0.255
$P_4(2)$	0.178	0.196	0.185
$P_5(0)$	0.557	0.579	0.548
$P_5(1)$	0.266	0.258	0.264
$P_5(2)$	0.177	0.163	0.188
$P_6(0)$	0.750	0.768	0.749
$P_6(1)$	0.195	0.180	0.190
$P_6(2)$	0.056	0.053	0.062
$P_7(0)$	0.539	0.529	0.533

Measures	Simulation	Perros and Snyder	New
$P_7(1)$	0.270	0.254	0.259
$P_7(2)$	0.191	0.216	0.208
$P_8(0)$	0.457	0.436	0.459
$P_8(1)$	0.262	0.250	0.262
$P_8(2)$	0.281	0.314	0.279
avg. abs. deviation	0.000	0.013	0.008
max. abs. deviation	0.000	0.033	0.022
avg. rel. error	0.000	0.046	0.033
max. rel. error	0.000	0.131	0.107

λ = (3,0,0,0,0,0,0,0) μ = (4,1,1,2,2,2,2,3.5)
N = (2,2,2,2,2,2,2,2)
routing is same as in table 5
CPU time = 0.006 seconds

Table 8
Comparisons with the approximations of Perros and Snyder for an eight node network

Measures	Simulation	Perros and Snyder	New
$P_1(0)$	0.281	0.283	0.258
$P_1(1)$	0.247	0.263	0.253
$P_1(2)$	0.226	0.229	0.247
$P_1(3)$	0.246	0.224	0.242
$P_2(0)$	0.535	0.527	0.526
$P_2(1)$	0.261	0.258	0.257
$P_2(2)$	0.122	0.125	0.126
$P_2(3)$	0.082	0.090	0.092
$P_3(0)$	0.522	0.520	0.505
$P_3(1)$	0.261	0.259	0.259
$P_3(2)$	0.128	0.127	0.133
$P_3(3)$	0.089	0.093	0.103
$P_4(0)$	0.506	0.499	0.503
$P_4(1)$	0.255	0.254	0.256
$P_4(2)$	0.131	0.127	0.130
$P_4(3)$	0.108	0.121	0.112
$P_5(0)$	0.491	0.511	0.478
$P_5(1)$	0.262	0.260	0.260
$P_5(2)$	0.136	0.130	0.142
$P_5(3)$	0.111	0.110	0.119

Measures	Simulation	Perros and Snyder	New
$P_6(0)$	0.707	0.727	0.704
$P_6(1)$	0.213	0.200	0.209
$P_6(2)$	0.061	0.054	0.062
$P_6(3)$	0.020	0.020	0.025
$P_7(0)$	0.454	0.448	0.451
$P_7(1)$	0.263	0.252	0.256
$P_7(2)$	0.148	0.140	0.145
$P_7(3)$	0.135	0.160	0.149
$P_8(0)$	0.355	0.335	0.350
$P_8(1)$	0.240	0.228	0.241
$P_8(2)$	0.164	0.155	0.166
$P_8(3)$	0.240	0.282	0.244
avg. abs. deviation	0.000	0.009	0.006
max. abs. deviation	0.000	0.042	0.023
avg. rel. error	0.000	0.047	0.042
max. rel. error	0.000	0.185	0.250

λ = (3,0,0,0,0,0,0,0) μ = (4,1,1,2.2.2.2,3.5)
N = (3,3,3,3,3,3,3,3)
routing is same as in table 5
CPU time = 0.006 seconds

REFERENCES

[1] Altiok,T.,"Approximate Analysis of Exponential Tandem Queues with Blocking",Europ. J. Oper. Res. 11, 390-398,1982

[2] Altiok,T., and H.G. Perros, "Open Networks of Queues with Blocking: Split and Merge Configurations",AIIE Trans. 18,251-261,1986

[3] Altiok,T., and H.G. Perros, "Approximate Analysis of Arbitrary Configurations of Open Queueing Networks with Blocking",Annals of OR 9,481-509,1987

[4] Bocharov,P.P. and G.P. Rokhas, "On an Exponential Queueing System in Series with Blocking", Problems of Control and Information Theory 9,441-455,1980

[5] Boxma,O.J. and A.G.Konheim, "Approximate Analysis of Exponential Queueing Systems with Blocking",Acta Informatica 15,19-66,1981

[6] Brandwajn,A. and Y.L.Jow, "An Approximate Method for Tandem Queues with Blocking",Manuscript,Amdahl Corp..1985

[7] Caseau,P. and G.Pujolle, "Throughput Capacity of a Sequence of Queues with Blocking due to Finite Waiting Room,"IEEE Trans. Soft. Eng. SE-5,631-642,1979

[8] Choong,Y.F. and S.B. Gershwin, "A Decomposition Method for the Approximate Evaluation of Capacitated Transfer Lines with Unreliable Machines and Random Processing Times", IEE Trans. 9,150-159,1987

[9] Foster,F.G. and H.G. Perros, "On the Blocking Process in Queue Networks", Europ. J. Oper. Res. 5,276-283,1980

[10] Gershwin,S.B., "An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage and Blocking", Manuscript,Lab. for Information and Decision Sciences,MIT,1983

[11] Gross,D., and C.M.Harris, Fundamentals of Queueing Theory, New York, John Wiley and Sons, 1985

[12] Hillier,F.S. and R. Boling, "Finite Queues in Series with Exponential or Erlang Service Times – Numerical Approach", Oper. Res. 15,286-303,1967

[13] Jackson,J.R., "Jobshop-like queueing systems", Management Sci. 10, 131-142. 1963

[14] Kerbache,L.,and J.M. Smith, "The generalized Expansion Method for Open Finite Queueing Networks', Technical Report, Dept. of Industrial Engr. and Operations Research, Univ. of Massachusetts, 1986

[15] Labetoulle,J.and G.Pujolle, "Isolation Method in a Network of Queues", IEEE Trans. Soft. Eng. SE-6,1980

[16] Latouche,G., and M.F. Neuts, "Efficient Algorithmic Solutions to Exponential Tandem Queues with Blocking", SIAM J. of Algebraic Discrete Methods 1, 93-106, 1980

[17] Lee,H., and S.M. Pollock, "Approximate Analysis for the Merge Configuration of an Open Queueing Network with Blocking", Technical Report, IOE Dept., 87-11, Univ. of Michigan, 1987

[18] Perros,H.G. and T. Altiok, "Approximate Analysis of Open Networks of Queues with Blocking: Tandem Configurations",IEEE Trans on Soft. Eng., SE-12,1986

[19] Perros,H.G. and P.M. Snyder, "A Computationally Efficient Approximation Algorithm for Analyzing Open Queueing Networks with Blocking",Manuscript, CS Dept., N.C. State Univ.,1986

[20] Pollock,S.M.,J.R.Birge and J.M.Alden, "Approximation Analysis for Open Tandem Queues with Blocking: Exponential and General Service Distributions", Technical Report,IOE Dept.,85-30, Univ. of Michigan,1985

[21] Takahashi,Y,H.Miyahara and T.Hasegawa, "An Approximation Method for Open Restricted Queueing Networks",Oper. Res. 28,594-602,1980