Annals of Operations Research 79(1998)181 – 206

Performance evaluation of open queueing networks with arbitrary configuration and finite buffers

H.S. Lee[a], A. Bouhchouch[b], Y. Dallery[c] and Y. Frein[b]

[a]*Department of Industrial Engineering, Kyung Hee University, Kiheung, Yongin-goon, 449-701 Kyunggi-do, Korea*

[b]*Laboratoire d'Automatique de Grenoble, UMR 5528 CNRS, Institut National Polytechnique de Grenoble, BP 46, F-38402 Saint Martin d'Hères, France*

[c]*Laboratoire MASI, URA 818 CNRS, Université Pierre et Marie Curie, 4, Place Jussieu, F-75252 Paris Cedex 05, France*

We consider an open queueing network consisting of servers linked in an arbitrary configuration with exponential service times and separated by intermediate finite buffers. The model allows any number of saturated servers (never starved) and exit servers (never blocked). The considered blocking mechanism is blocking-after-service. In the case of simultaneous blocking, blocked customers enter the destination node on a first-blocked-first-enter basis. If feedback loops exist in the network, deadlocks may arise. We assume that a deadlock is detected and resolved instantaneously by transferring all the blocked customers simultaneously. In this paper, we present an approximation method to analyze this kind of networks. The method decomposes the original network into subsystems. Each subsystem is composed of one or many upstream servers and one downstream server, separated by a finite buffer. The upstream and downstream servers are characterized by exponential service time distributions. Based on the symmetrical approach [2], we develop an algorithm to determine the unknown parameters corresponding to each subsystem. The class of models considered in this paper includes the class of open loss models for feed-forward networks considered by Lee and Pollock [11]. For this class of models, we can show that the system of equations in our algorithm is equivalent to the one used in the algorithm proposed by Lee and Pollock. As a result, our algorithm provides the same results as the algorithm of Lee and Pollock for this class of models. However, it is observed that our algorithm takes less CPU execution time than the one proposed by Lee and Pollock. For the cases of networks with feedback loops, extensive numerical experiments show that the new algorithm, in general, converges very fast and yields accurate results compared with those obtained by simulation as long as deadlocks do not occur too frequently. Moreover, for the merge configuration, we provide the proof of the convergence of the algorithm as well as the existence and uniqueness of the solution by exploiting the properties associated with a symmetric approach.

Keywords: queueing networks with finite buffers, blocking, performance analysis, decomposition techniques

1. Introduction

Queueing networks with finite buffers are useful for modeling and analyzing discrete event systems, including manufacturing systems, computer systems and communication systems. In the case of manufacturing systems, flow line systems can be modeled as tandem configuration of queueing networks with finite buffers while job shop systems can be modeled as arbitrary configuration of queueing networks with finite buffers.

In queueing networks with finite buffers, blocking may occur because of the finiteness of buffers. Different types of blocking mechanisms have been considered in the literature [13]: blocking-after-service (also referred to as type-1 blocking, transfer blocking, and manufacturing blocking), blocking-before-service (also referred to as type-2 blocking, service blocking, and communication blocking), and repetitive-service blocking (also referred to as type-3 blocking, and rejection blocking). In blocking-after-service, a server is blocked if the destination buffer of the customer is full after completion of the service of a customer. In blocking-before-service, the service of a customer is not allowed to start until there is room available in its destination buffer. In repetitive-service blocking, a customer attempts to join its destination buffer upon service completion. If this buffer is full, the customer receives another service and this is repeated until space becomes available in the destination buffer. A comparison of these types of blocking can be found in Onvural and Perros [12] and Perros [13]; see also [3].

In general, queueing networks with finite buffers are difficult to analyze due to blocking. Therefore, only under limited conditions, can exact solutions be obtained. For this reason, most analyses are based on approximation, numerical or simulation methods. Many approximation methods have been proposed for analyzing open queueing networks with finite buffers. See Perros [13] and Dallery and Gershwin [3] for a complete list of references. Most of the approximation analyses are based on decomposition. The basic idea is to decompose the queueing network into individual subsystems and analyze each subsystem in isolation.

According to the recent work by Dallery and Frein [2] for the tandem configuration under blocking-after-service, all decomposition methods involve three steps: (1) the characterization of the subsystem, (2) the derivation of a set of equations to determine the unknown parameters of each subsystem, (3) the derivation of an algorithm for solving this set of equations. They observed that although three sets of equations can be derived to determine the unknown parameters in step 2, only two sets of equations are actually used. Therefore, decomposition methods can be classified into three main approaches according to which equation sets are used in step 2. They found that these three approaches yield the same results if subsystems are characterized and solved in the same way. One of these approaches, which offers a symmetrical view of decomposition, is of special interest because the algorithm based on this symmetrical approach is, in general, faster than those based on the other approaches. Moreover, in the case of exponential characterization of subsystems, they proved the convergence of the algorithm as well as the existence and uniqueness of the solution associated with the symmetrical approach.

In this study, we extend the symmetrical approach to the arbitrary configuration of open queueing networks with exponential service times and under blocking-after-service. Some approximate algorithms have been developed for the analysis of arbitrary configurations of open queueing networks with finite buffers. Labetoulle and Pujolle [9] and Kerbache and Smith [5] developed decomposition algorithms under repetitive-service blocking with fixed destination. Kouvatsos and Xenios [7] developed an approximation algorithm under repetitive-service blocking with random destination using a maximum entropy method. In this study, service times as well as interarrival times are assumed to have a generalized exponential distribution. They extended the maximum entropy algorithm to the queueing networks with multiple server nodes and under repetitive-service blocking [8]. The maximum entropy algorithm was further generalized by Kouvatsos and Denazis [6] to the open queueing networks under repetitive-service blocking with random destination and multiple job classes.

For the analysis of open queueing networks with arbitrary configuration and under blocking-after-service, some approximate algorithms have been developed by Takahashi et al. [15], Altiok and Perros [1], Perros and Snyder [14], Jun and Perros [4] and Lee and Pollock [11]. In these algorithms, except for the algorithm by Jun and Perros, feedback loops are not allowed, i.e., only networks with feed-forward topologies are considered. In these previous algorithms, open loss models were analyzed in which external arrivals occur to one [1, 14, 15] or more than one node [4, 11] in accordance with Poisson processes. External arrival which occurs when the buffer is full is assumed to be lost. Another common feature of these algorithms is the way the network is decomposed. In these algorithms, each subsystem consists of a finite buffer and a server fed by an external arrival process. In the algorithm of Takahashi et al., the arrival process to each subsystem is characterized by a Poisson process and the service time at each subsystem is characterized by an exponential distribution. In this algorithm, a set of simultaneous nonlinear equations are developed to determine the unknown parameters. In the algorithm of Altiok and Perros, the arrival process to each subsystem is characterized by a Poisson process, while the service time at each subsystem is characterized by a phase-type distribution. Perros and Snyder developed an improved algorithm over that by Altiok and Perros. They characterized service times at each subsystem by a two-phase Coxian (C_2) distribution. Jun and Perros extended this algorithm to the network with feedback loops. In this algorithm, it is assumed that a deadlock is resolved instantaneously by exchanging all the blocked customers simultaneously. In this algorithm, the arrival process to each subsystem is characterized by a Poisson process and the service time at each subsystem is characterized by a C_2 distribution, as in the algorithm of Perros and Snyder. However, in order for this C_2 distribution to reflect all the possible blocking delays and deadlocks in the original system, a very complicated phase-type distribution should be constructed beforehand. This phase-type distribution is then simplified to the C_2 distribution using a three-moment approximation. Although this algorithm is reported to be very accurate, it appears that due to the complicated feature, the use of this algorithm is restricted to networks which consist of nodes with at most two directly linked upstream servers. In the algorithm of Lee and Pollock, service time at each subsystem is characterized by an exponential distribution. However, in this algorithm, the arrival process to each subsystem is not characterized by a single Poisson process. Instead, it is characterized by several independent Poisson processes, each of which is considered to be generated by the upstream server directly linked to the node represented by the subsystem. In addition to using a more accurate representation of the arrival process, each subsystem is analyzed exactly by using an efficient state aggregation technique. Due to this fact, Lee and Pollock's algorithm is reported to provide very accurate results despite the simplification of exponential characterizations.

Owing to this merit, we use, in our algorithm, the procedure developed in Lee and Pollock to analyze each subsystem. In fact, our algorithm characterizes and analyzes the subsystem in the same way as Lee and Pollock's algorithm. Consequently, for the class of models considered by Lee and Pollock, our algorithm yields the same results as their algorithm. However, our algorithm is more general than the algorithm of Lee and Pollock in that our algorithm can analyze a more general class of networks than the one considered by Lee and Pollock, namely networks with loops. In addition, since our algorithm is based on the symmetrical approach, it has some advantages over the previous algorithms as observed in the case of tandem configuration. That is, our algorithm is faster than the previous algorithms and, in case of merge configuration, the convergence of the algorithm as well as the existence and uniqueness of the solution can be proved.

This paper is organized as follows. In section 2, after describing the model, we present the new approximation method. Three steps which are involved in the decomposition method are described in detail. In section 3, to show the performance of the algorithm, computational results are reported for the various problem sets with different topologies. In section 4, we establish some theoretical properties for the merge configuration such as convergence, existence and uniqueness. Finally, some conclusions are given in section 5.

2. Decomposition method

2.1. Description of the model

The network we consider consists of servers linked in an arbitrary configuration with exponential service times separated by intermediate finite buffers. The model allows any number of saturated servers (never starved) and exit servers (never blocked). We assume that there are M non-saturated servers and each non-saturated server has an intermediate finite buffer preceding it. The non-saturated servers will be denoted by S_i, for $i = 1,\dots,M$, and the buffer preceding server S_i is denoted by B_i, for $i = 1,\dots,M$. In comparison with the classical open model, we call this type of network the saturated model since there is no external arrival process.

Let C_i be the capacity of buffer B_i, including the server space in front of server S_i. A server and its preceding buffer will be called a node in this paper. Let a saturated server linked to buffer $B_i$be denoted by S_{oi}. The service time of server S_i is exponentially distributed with rate μ_i and the service time of server S_{oi} is exponentially distributed with rate μ_{oi}. A customer who has completed service at S_i gets its next service at S_j with probability r_{ij}. A customer who has completed service at S_{oi} gets its next service at S_i. The probability that a customer leaves the queueing system after completing service at S_i is r_{io}. (Note that $r_{io} = 1$ if S_i is an exit server.) Figure 1 shows an example of a network with four non-saturated servers and two saturated servers.

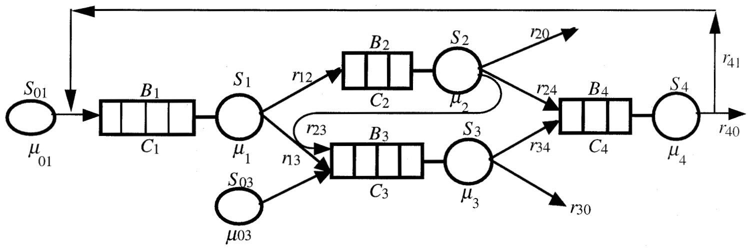


Figure 1. An example of a network with four non-saturated servers and two saturated servers.

The blocking mechanism assumed in this paper is blocking-after-service. Suppose, upon completion of its service at S_i, a customer attempts to enter node j. If buffer B_j is full at that instant, the customer is forced to stay at node i until a space becomes available at node j. During this period, server S_i is blocked and node j is blocking. Since there are more than one upstream servers directly linked to node j, more than one upstream server can be blocked simultaneously by node j. In this case, we assume that the corresponding blocked customers enter node j on a first-blocked-first-enter basis.

Since the nodes are linked in an arbitrary configuration, deadlocks may occur. Consider for example the network of figure 1. Suppose that nodes 1, 2 and 4 are full and that S_1 is blocked by node 2 and S_2 is blocked by node 4. A deadlock will occur if, when a service completion occurs at S_4, the customer chooses to go to node 1. We assume that a deadlock is detected and resolved instantaneously by exchanging all the blocked customers simultaneously as assumed in the paper by Jun and Perros [4]. This deadlock resolution mechanism may cause the violation of the first-blocked-first-enter priority rule. For instance, in the case of the deadlock we have just illustrated, suppose that both S_2 and S_3 were blocked by node 4 but S_3 had been blocked longer than S_2. In this case, when a deadlock occurs, a customer at S_2 instead of the one at S_3 enters node 4 according to the deadlock resolution mechanism.

We have assumed that the blocking mechanism at any server is blocking-after-service. Alternatively, we also allow saturated servers to operate under a repetitive-service blocking mechanism. By doing that, the class of models considered in this paper includes the class of open loss models considered by previous authors [1,4,11, 14,15], in which each node can be fed by an external Poisson arrival process. The open loss model equivalent to the saturated model in figure 1 assuming that the blocking mechanism at S_{01} and S_{03} is repetitive-service blocking is shown in figure 2.

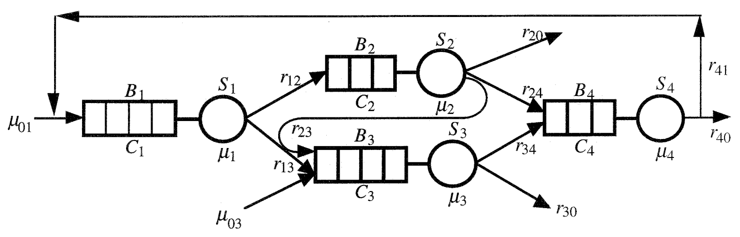


Figure 2. The equivalent open loss model.

Let U_i denote the index set of upstream servers directly linked to buffer B_i and D_i denote the index set of downstream buffers directly linked to server S_i. Let N_i denote $|U_i|$, i.e., the number of elements in set U_i. Also, let $f(i, j)$, for $i = 1,\dots,N_j$, be the index of the ith upstream server directly linked to buffer B_j. If there is a saturated server preceding B_j, this server is always considered to be the first upstream server of node j. For instance, for the network in figure 1, we have:

$$U_1 = \{01, 4\},\ \ U_3 = \{03, 1, 2\},\ D_2 = \{3, 4\},\ \ N_1 = 2,\ N_3 = 3,$$
$$f(1,3) = 03,\ f(2,3) = 1,\ f(3,3) = 2.$$

2.2. Decomposition of the original system

To analyze the network, we will use the decomposition method which decomposes the original network into M subsystems, $T(j)$, $j = 1,\dots,M$. Subsystem $T(j)$ is composed of a set of upstream servers $S_{ui}(j)$, $i = 1,\dots,N_j$ and a downstream server $S_d(j)$, separated by a finite buffer $B(j)$. The upstream servers $S_{ui}(j)$ are never starved and the downstream server $S_d(j)$ is never blocked. Subsystem $T(j)$ approximates the flow of customers in buffer B_j of the original system. In $T(j)$, the intermediate buffer $B(j)$ has the same capacity C_j as that of buffer B_j. Server $S_{ui}(j)$ represents node $f(i, j)$ and its upstream nodes in the original model, and server $S_d(j)$ represents server S_j and its downstream nodes in the original model. Figure 3 shows how the original model of figure 1 is decomposed into subsystems.

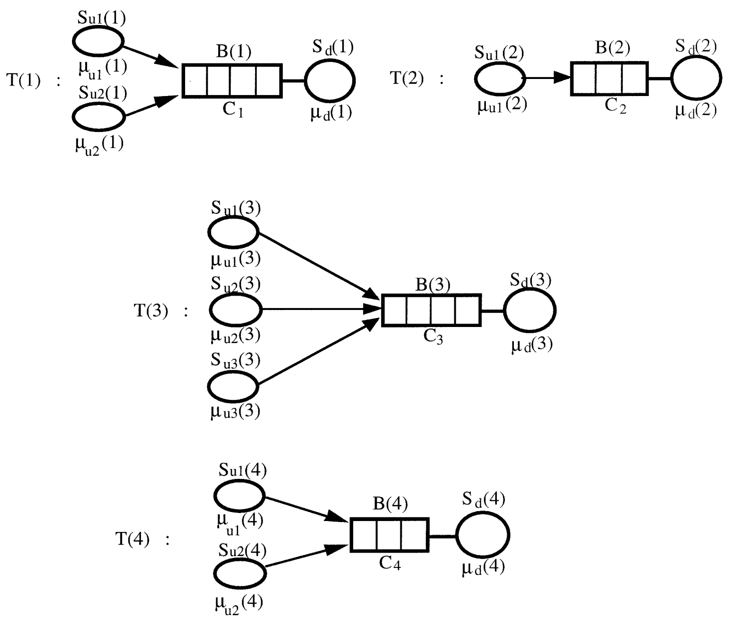


Figure 3. Decomposition of the model into subsystems.

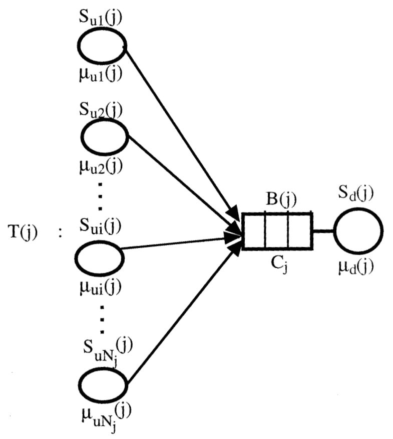


Figure 4. Subsystem $T(j)$.

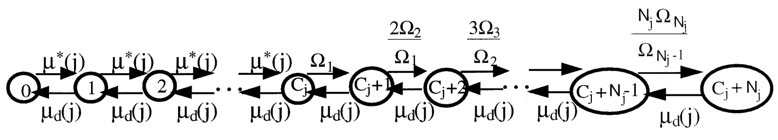
2.3. *Characterization and analysis of subsystems*

In this paper, we characterize the upstream and downstream servers at each subsystem by exponential distributions. Let the service rates of $S_{ui}(j)$ and $S_d(j)$ be $\mu_{ui}(j)$ and $\mu_d(j)$, respectively. If the values of $\mu_{ui}(j)$, $i = 1,\dots,N_j$, and $\mu_d(j)$ are given, we can analyze subsystem $T(j)$ exactly by the procedure developed in Lee and Pollock [11]. To analyze subsystem $T(j)$ (see figure 4), we define the state of $T(j)$ to be the number of customers in $B(j)$ plus the ones being blocked by $S_d(j)$ if $S_d(j)$ is blocking. Thus, $C_j + n$ represents the state that n upstream servers are being blocked by $S_d(j)$.

Let

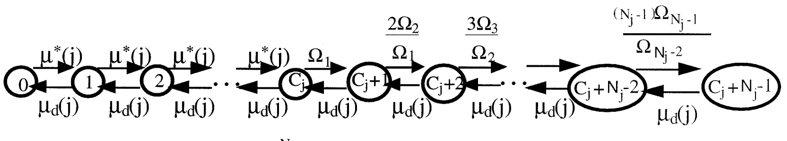
$P(n:j)$	=	steady state probability that $T(j)$ is in state n, $n = 0,\dots,C_j + N_j$,
$b_{ui}(n:j)$	=	probability that n servers are blocked by $S_d(j)$ including $S_{ui}(j)$, $n = 1,\dots,N_j$,
$b_{ui}(j)$	=	probability that $S_{ui}(j)$ is blocked by $S_d(j)$,
$P_s(j)$	=	probability that $S_d(j)$ is starved at the service completion instant,
$P_{bi}(n:j)$	=	probability that at the service completion instant, the server $S_{ui}(j)$ sees n other servers being blocked by $S_d(j)$, $n = 0,\dots,N_j - 1$.

Then, from the results in Lee and Pollock [11], the steady-state probability $P(n:j)$ can be obtained by solving the birth and death process shown in figure 5 or figure 6. Figure 5 represents the state-transition-rate diagram when the service mechanism at $S_{u1}(j)$ is blocking-after-service, while figure 6 represents the state-transition-rate diagram when the service mechanism at $S_{u1}(j)$ is repetitive-service blocking.


$\mu^*(j)$
Ω_1
$\frac{2\Omega_2}{\Omega_1}$
$\frac{3\Omega_3}{\Omega_2}$
$\frac{N_j\Omega_{N_j}}{\Omega_{N_j-1}}$
0
1
2
C_j
C_j+1
C_j+2
C_j+N_j-1
C_j+N_j
$\mu_d(j)$

where $\mu^*(j) = \sum_{i=1}^{N_j} \mu_{ui}(j)$, $\Omega_n = \sum_{i_1<\dots<i_n} \prod_{k=1}^{n} \mu_{ui_k}(j)$, $i_k \in \{1,2,\dots,N_j\}$

Figure 5. State-transition-rate diagram when the service mechanism at $S_{u1}(j)$ is blocking-after-service.


$\mu^*(j)$
Ω_1
$\frac{2\Omega_2}{\Omega_1}$
$\frac{3\Omega_3}{\Omega_2}$
$\frac{(N_j-1)\Omega_{N_j-1}}{\Omega_{N_j-2}}$
0
1
2
C_j
C_j+1
C_j+2
C_j+N_j-2
C_j+N_j-1
$\mu_d(j)$

where $\mu^*(j) = \sum_{i=1}^{N_j} \mu_{ui}(j)$, $\Omega_n = \sum_{i_1<\dots<i_n} \prod_{k=1}^{n} \mu_{ui_k}(j)$, $i_k \in \{2,3,\dots,N_j\}$

Figure 6. State-transition-rate diagram when the service mechanism at $S_{u1}(j)$ is repetitive blocking.

Once the steady-state probabilities are obtained, the blocking probabilities can be computed by again using the results in Lee and Pollock as follows:

$$b_{ui}(n:j) = \frac{\mu_{ui}(j)\Omega_{n-1}^{i}}{\Omega_n} P(C_j + n:j), \quad n = 1,2,\dots N_j, \tag{1}$$

where $\Omega^i_{n-1} \equiv \sum_{i_1<\cdots<i_{n-1}} \prod_{k=1}^{n-1} \mu_{ui_k}(j)$, $i_k \in \{1, 2,\dots, i-1, i+1,\dots, N_j\}$ in the case of blocking-after-service at $S_{u1}(j)$ and $i_k \in \{2, 3,\dots, i-1, i+1,\dots,N_j\}$ in the case of repetitive-service blocking at $S_{u1}(j)$.

The steady-state probability that server $S_{ui}(j)$ is blocked by server $S_d(j)$ can now be computed as

$$b_{ui}(j) = \sum_{n=1}^{N_j} b_{ui}(n:j). \tag{2}$$

Finally, the starvation probabilities and the blocking probabilities at the service completion epochs are given by (for details, refer to Lee et al. [10])

$$P_s(j) = \frac{P(1:j)}{1-P(0;j)}, \tag{3}$$

$$P_{bi}(n:j) = \frac{P(C_j+n:j) - b_{ui}(n:j)}{1-b_{ui}(j)}, \quad n = 0, 1, 2, \dots, N_j - 1. \tag{4}$$

2.4. Decomposition equations

In section 2.3, we analyzed the subsystem assuming that the parameter values are known. These parameter values, however, are not known but should be determined. Notice that the unknown parameters are the set of service rates $\mu_{ui}(j)$, $i = 1, 2,\dots,N_j$, $j = 1, 2,\dots,M$ and $\mu_d(j)$, $j = 1, 2,\dots,M$. As a result, we have a total of $M + \sum_{j=1}^{M} N_j$ unknowns. To determine these unknowns, we need a set of $M + \sum_{j=1}^{M} N_j$ independent equations. First consider the saturated server S_{0j} and exit server S_k in the original model. Since S_{0j} has no upstream nodes in the original model, server $S_{u1}(j)$ in subsystem $T(j)$ represents S_{0j} in the original model. Similarly, since S_k has no downstream nodes in the original model, server $S_d(k)$ in subsystem $T(k)$ represents S_k in the original model. Consequently, we have $\mu_{u1}(j) = \mu_{0j}$, $\mu_d(k) = \mu_k$ for these servers. Therefore, we have as many boundary conditions as the number of saturated servers and exit servers in the original model.

We now derive three different sets of equations that can be used to determine the remaining set of unknowns. In deriving the sets of equations to determine the unknown parameters, we do not consider the effect of a deadlock because consideration of it will make the model too complicated to solve the networks in a general case. Therefore, sets of equations are derived under the conditions as if there is no deadlock. One set of equations is related to the service processes of the downstream servers $S_d(j)$. Since the server $S_d(j)$ represents the part of the original model downstream of buffer $B(j)$, the service time of $S_d(j)$ corresponds to the time between the beginning of a service at S_j and the transfer of the customer into one of the downstream buffers. This time is composed of a service time of server S_j and, possibly, a blocking time. Blocking occurs if the destination buffer is full at the service completion instant at S_j. Note that a blocking of server S_j by S_m is represented by a blocking of server $S_{uk}(m)$ by $S_d(m)$ in subsystem $T(m)$, where $f(k, m) = j$. Hence, the probability that at the service completion instant at S_j a customer whose destination is node m sees n other customers being blocked by S_m, is approximated by $P_{bk}(n:m)$, where $f(k, m) = j$. Therefore, the mean service time of $S_d(j)$ is given by

$$\frac{1}{\mu_d(j)} = \frac{1}{\mu_j} + \sum_{m \in D_j} r_{jm} \sum_{n=0}^{N_j - 1} P_{bk}(n:m)(n+1) \frac{1}{\mu_d(m)}, \tag{5}$$

where $f(k, m) = j$ for each non-exit server S_j.

A similar set of equations is related to the service processes of the upstream servers $S_{ui}(j)$. Since server $S_{ui}(j)$ represents the server $S_{f(i,j)}$ and its upstream part in the original model, the service time of $S_{ui}(j)$ corresponds to the time between the transfer of a customer from server $S_{f(i,j)}$ into buffer B_j and the service completion of the next customer with destination S_j at $S_{f(i,j)}$. Let $f(i, j) = k$. Then, we can approximate the mean time between the transfer of a customer from server S_k into buffer B_j and the next service completion at S_k by

$$P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_k},$$

where $\mu^*(k) = \sum_{i=1}^{N_k} \mu_{ui}(k)$. Therefore, if the destination of the next customer is node j, this becomes the mean service time of $S_{ui}(j)$. On the other hand, if the destination of the next customer is not node j, say node m, then an additional time of $\sum_{n=0}^{N_m - 1} P_{bl}(n:m)(n+1)/\mu_d(m)$ on average is required to clear this customer from server S_k after completion of its service, where $f(l, m) = k$. Due to the Markov routing, the mean time from the clearance of this customer until the service completion of the next customer with destination S_j is again $1/\mu_{ui}(j)$. Thus, we can express $1/\mu_{ui}(j)$ as

$$\begin{aligned} \frac{1}{\mu_{ui}(j)} &= r_{kj} \left\{ P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_k} \right\} \\ &+ \sum_{m \in D_k} r_{km} \left\{ P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_k} + \sum_{n=0}^{N_m - 1} P_{bl}(n:m)(n+1) \frac{1}{\mu_d(m)} + \frac{1}{\mu_{ui}(j)} \right\}, \end{aligned} \tag{6}$$

where $f(i, j) = k$, $f(l, m) = k$.

Solving equation (6) in terms of $1/\mu_{ui}(j)$, we have

$$\frac{1}{\mu_{ui}(j)} = \left\{ P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_k} - r_{kj} \sum_{n=0}^{N_j - 1} P_{bi}(n:j)(n+1) \frac{1}{\mu_d(j)} \right\} \frac{1}{r_{kj}}, \tag{7}$$

where $f(i, j) = k$.

Finally, a third set of equations is related to the conservation of flow. Let $X_d(i)$ be the throughput of $S_d(i)$ (or $T(i)$) and $X_{ui}(j)$ be the throughput of $S_{ui}(j)$. Then, the following relationships should hold between the throughput of two subsystems to satisfy the conservation of flow:

$$X_{ui}(j) = X_d(k) r_{kj}, \tag{8}$$

where $f(i, j) = k$, for $i = 1, 2, \ldots, N_j$, $j = 1, 2, \ldots, M$.

We have obtained three sets of equations. However, it can be easily checked that the number of equations in any two sets of equations is the same as the number of unknown parameters. Therefore, to determine the unknown parameters, we only need two sets of equations. Let $SE1$ be a system of equations consisting of equations (5) and (8). Let $SE2$ be a system of equations consisting of equations (7) and (8). Similarly, let $SE3$ be a system of equations consisting of equations (5) and (7). Since we can choose any system of equations among $SE1$, $SE2$, and $SE3$ to determine the unknown parameters, three different types of algorithms can be devised. As Dallery and Frein proved for the tandem configuration [2], we can prove that the following lemma holds for the arbitrary configuration.

Lemma 1. Any solution of $SE3$ satisfies the conservation of flow equation (8).

Proof. Note that throughputs $X_d(k)$ and $X_{ui}(j)$ can be expressed, respectively, as:

$$X_d(k) = \left[\frac{1}{\mu_d(k)} + P_s(k) \frac{1}{\mu^*(k)} \right]^{-1}, \tag{9}$$

$$X_{ui}(j) = \left[\frac{1}{\mu_{ui}(j)} + \sum_{n=0}^{N_j - 1} P_{bi}(n:j)(n+1) \frac{1}{\mu_d(j)} \right]^{-1}. \tag{10}$$

From equation (10) together with equations (7) and (9), we have

$$\begin{aligned} X_{ui}(j) = &\left[\left\{ P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_d(j)} - r_{kj} \sum_{n=0}^{N_j - 1} P_{bi}(n:j)(n+1) \frac{1}{\mu_d(j)} \right\} \frac{1}{r_{kj}} \right. \\ &\left. + \sum_{n=0}^{N_j - 1} P_{bi}(n:j)(n+1) \frac{1}{\mu_d(j)} \right]^{-1} = r_{kj} \left[P_s(k) \frac{1}{\mu^*(k)} + \frac{1}{\mu_d(k)} \right]^{-1} = X_d(k) r_{kj}, \end{aligned}$$

where $f(i, j) = k$. □

Lemma 1 implies that any solution of $SE3$ will satisfy (8). The proof of lemma 1 can be reversed to show that (5) and (8) imply (7), and similarly (7) and (8) imply (5). As a result, we have the following corollary.

Corollary 1. The three systems of equations $SE1$, $SE2$, $SE3$ are equivalent.

Corollary 1 indicates that different algorithms based on different systems of equations yield the same results if subsystems are characterized and solved in the same way. The previous algorithms such as those of Altiok and Perros [1], Perros and Snyder [14], Jun and Perros [4], and Lee and Pollock [11] are all developed based on $SE1$. The algorithm we develop in this paper is based on $SE3$. As will be shown later, the algorithm based on $SE3$ has several advantages over the algorithms based on $SE1$ or $SE2$, in addition to offering a symmetrical view of decomposition.

2.5. The computational algorithm

This section describes an algorithm to solve the system of equations, $SE3$. Our algorithm is an iterative algorithm with a single loop. Since the network is linked in an arbitrary configuration, each subsystem can be analyzed in any order within a loop. For convenience, we analyze subsystems $T(j)$ in the order $j = 1,\dots,M$. Within each loop, to analyze subsystem $T(j)$, we first calculate the service rate of the downstream server, $\mu_d(j)$, using equation (5). To do this, we need the values of $P_{bk}(n:m)$ for all $m \in D_j$, where $f(k, m) = j$. We use the values of $P_{bk}(n:m)$ which have been obtained most recently. That is, if subsystem $T(m)$ has already been analyzed in this loop, we use the values updated in this loop. On the other hand, if subsystem $T(m)$ has not been analyzed in this loop, we use the values updated in the previous loop. Once $\mu_d(j)$ is updated, using the values of $\mu_{ui}(j)$ which have been updated most recently, we analyze subsystem $T(j)$ using the procedure presented in section 2.3. Then we calculate $P_s(j)$ and $P_{bi}(n:j)$ using equations (3) and (4), respectively. After these probabilities are obtained, we calculate new values of $\mu_{ui}(k)$ for all $k \in D_j$, where $f(i, k) = j$, using equation (7). In each iteration step of the algorithm, some computational effort can be saved because some parameter values do not have to be calculated for the following cases: (i) $P_s(j)$ if S_j is an exit server, (ii) $\mu_{u1}(j)$ if the first upstream server linked to B_j is a saturated server, (iii) $\mu_d(j)$ if S_j is an exit server, (iv) $P_{bi}(n:j)$ if a saturated server is the only upstream server linked to B_j. The algorithm continues until the upstream and downstream service rates become close enough between two successive iterations. The initial values of the parameters $\mu_d(j)$, $\mu_{ui}(j)$ and $P_{bi}(n:j)$ are obtained by assuming that neither blocking nor starvation exist. This algorithm, which we will call algorithm 1 in this paper, is summarized as follows:

Algorithm 1

Initialization:

```
Set  $\mu_d(j) = \mu_j,\ j = 1,\dots,M$
     $\mu_{ui}(j) = \mu_k * r_{kj}$,  where $f(i,j) = k$,  for $i = 1,\dots,N_j,\ j = 1,\dots,M$
     $P_{bi}(n:j) = 0$,  for $i = 1,\dots,N_j,\ j = 1,\dots,M$
```

Iteration step:

```
For $j$ =1, 2,…,$M$ do:
1. Calculate $\mu_d(j)$ using (5).
2. Calculate $P_s(j)$ using (3).
3. Calculate $P_{bi}(n:j)$ using (4), $n = 0, 1,\ldots,N_j - 1$, $i = 1, 2,\ldots,N_j$.
4. Calculate $\mu_{ui}(k)$ using (7), for all $k \in D_j$, where $f(i,k) = j$.
```

Convergence step:

```
If convergence condition is satisfied, stop.
Otherwise, go to the iteration step.
```

This algorithm, in general, solves *SE*3 very quickly. In addition, algorithm 1 always converged in all the problems we have tested to date. However, for the networks with no feedback loop, we can devise a more efficient algorithm, which we will call algorithm 2 in this paper. In algorithm 2, each loop consists of a forward pass (step 1) and a backward pass (step 2). The forward pass calculates new values of the service rates of the upstream servers $\mu_{ui}(j)$ as well as the starvation probabilities $P_s(j)$. The backward pass calculates new values of the service rates of the downstream servers $\mu_d(j)$ as well as the blocking probabilities $P_{bi}(n:j)$. The initialization is done in the same way as in algorithm 1.

Algorithm 2

Initialization:

```
Set $\mu_d(j) = \mu_j$, $j = 1,\ldots,M$
    $\mu_{ui}(j) = \mu_k * r_{kj}$, where $f(i,j) = k$, for $i = 1,\ldots,N_j$, $j = 1,\ldots,M$
    $P_{bi}(n:j) = 0$, for $i = 1,\ldots,N_j$, $j = 1,\ldots,M$
```

Iteration step:

```
Step 1: For $j = 1, 2,\ldots,M$ do:
        1.1. Calculate $P_s(j)$ using (3).
        1.2. Calculate $\mu_{ui}(k)$ using (7) for all $k \in D_j$, where $f(i,k) = j$.
Step 2: $j = M,\ldots,2,1$ do:
        2.1. Calculate $\mu_d(j)$ using (5).
        2.2. Calculate $P_{bi}(n:j)$ using (4), $n = 0,1,\ldots,N_j - 1$, $i = 1, 2,\ldots,N_j$.
Step 3: If convergence condition is satisfied, stop.
        Otherwise, go to step 1.
```

Since both algorithm 1 and algorithm 2 solve the same system of equations, they always yield the same results. However, the speeds of these two algorithms are different. In most of the test problems of the acyclic networks, algorithm 2 was observed to converge faster than algorithm 1. For the test problems of the networks with very few feedback loops, the speeds of these two algorithms were observed to be almost comparable. However, for the networks with many feedback loops, it was observed that algorithm 1 usually converged faster than algorithm 2. Through numerical experiments, it was also found that algorithm 2 did not converge in some large-size network problems with many feedback loops, although it converged in all the acyclic network problems tested to date. For this reason, we recommend that algorithm 1 be used for the networks with feedback loops, while algorithm 2 be used for the networks without feedback loops. We have not been able to prove the convergence of the algorithm in the general case. In the case of a merge configuration, however, as will be shown in section 4, we did prove the convergence of algorithm 2 as well as the existence and uniqueness of the solution by exploiting the properties pertaining to the system of equations, *SE*3.

3. Computational results

In order to test the accuracy and the speed of the approximation method, both algorithm 1 and algorithm 2 were implemented on an IBM PC 586 and tested on a variety of problems. The results of the approximation method were compared with those of simulation. Each simulation was run until at least 300,000 customers departed from the system. If the network has no feedback loop and the blocking mechanism at the saturated servers is repetitive-service blocking, our algorithm yields the same results as the algorithm of Lee and Pollock [11] because decomposition into subsystems, characterization of the subsystem and analysis of the subsystem are done in the same way in these algorithms. In order to compare the speed of algorithm 2 to that of Lee and Pollock's algorithm, we have tested both algorithms on the set of problems presented in Lee and Pollock. From the results of the experiments, we have observed that the number of iterations needed for convergence by algorithm 2 is always less than or equal to that needed for convergence by Lee and Pollock's algorithm. As a result, algorithm 2 takes less CPU time than the previous algorithm using the same convergence criterion. The CPU execution time as well as the number of iterations was observed to be reduced by about 15% on average in algorithm 2 compared with that in Lee and Pollock's algorithm. The maximum number of iterations needed for convergence was eight in algorithm 2, while it was ten in Lee and Pollock's algorithm on the set of problems presented in Lee and Pollock [11].

We have also tested our algorithm for the networks with feedback loops. The results are summarized in tables 1 to 4 for the various performance measures. In each table, the probability that there are n customers at node i is denoted by $P_i(n)$ and the mean queue length at node i is denoted by L_i. Table 1 gives results for the three-node network in figure 7 and table 2 gives results for the four-node network in figure 8. Table 3 gives results for the eight-node network in figure 9. The blocking mechanism at the saturated servers is assumed to be repetitive-service blocking for the networks

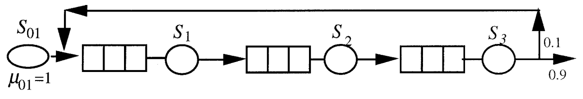


Figure 7. The three-node network.

Table 1

Comparisons with simulation for the three-node networks with feedback loops.

Case	Measures	Simulation	Approx.	Rel. error (%)
C = (2, 2, 2), μ = (1, 1, 1)	$P_1(0)$	0.206	0.210	1.94
	$P_1(2)$	0.458	0.483	5.46
	L_1	1.252	1.273	1.68
	$P_2(0)$	0.265	0.287	8.30
	$P_2(2)$	0.450	0.452	0.44
	L_2	1.185	1.166	– 1.60
	$P_3(0)$	0.376	0.389	3.46
	$P_3(2)$	0.334	0.335	0.30
	L_3	0.959	0.946	– 1.36
C = (1, 1, 1), μ = (1.5, 1.5, 1.5)	$P_1(0)$	0.510	0.478	– 6.27
	$P_1(1)$	0.490	0.522	6.53
	L_1	0.490	0.522	6.53
	$P_2(0)$	0.526	0.532	1.14
	$P_2(1)$	0.474	0.468	– 1.27
	L_2	0.474	0.468	– 1.27
	$P_3(0)$	0.610	0.620	1.64
	$P_3(1)$	0.390	0.380	– 2.56
	L_3	0.390	0.380	– 2.56
C = (3, 3, 3), μ = (1.5, 1.5, 1.5)	$P_1(0)$	0.312	0.311	– 0.32
	$P_1(3)$	0.191	0.202	5.76
	L_1	1.300	1.317	1.31
	$P_2(0)$	0.332	0.338	1.81
	$P_2(3)$	0.222	0.232	4.50
	L_2	1.301	1.307	0.46
	$P_3(0)$	0.391	0.396	1.28
	$P_3(3)$	0.175	0.179	2.29
	L_3	1.128	1.130	0.18
C = (2, 2, 2), μ = (2, 2, 2)	$P_1(0)$	0.489	0.481	– 1.64
	$P_1(2)$	0.195	0.210	7.69
	L_1	0.706	0.729	3.26
	$P_2(0)$	0.498	0.501	0.60
	$P_2(2)$	0.213	0.228	7.04
	L_2	0.715	0.727	1.68
	$P_3(0)$	0.547	0.550	0.55
	$P_3(2)$	0.180	0.186	3.33
	L_3	0.634	0.636	0.32

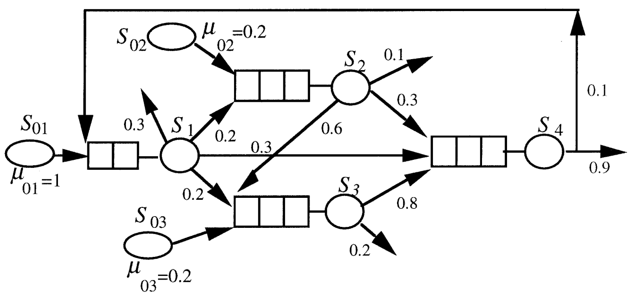


Figure 8. The four-node network.

Table 2

Comparisons with simulation for the four-node networks with feedback loops.

Case	Measures	Simulation	Approx.	Rel. error (%)
$C = (1, 1, 1, 1)$, $\mu = (1.0, 0.5, 0.5, 1.0)$	$P_1(0)$	0.364	0.338	– 7.14
	$P_1(1)$	0.636	0.662	4.09
	L_1	0.636	0.662	4.09
	$P_2(0)$	0.490	0.477	– 2.65
	$P_2(1)$	0.510	0.523	2.55
	L_2	0.510	0.523	2.55
	$P_3(0)$	0.389	0.394	1.29
	$P_3(1)$	0.611	0.606	– 0.82
	L_3	0.611	0.606	– 0.82
	$P_4(0)$	0.589	0.589	0.00
	$P_4(1)$	0.411	0.411	0.00
	L_4	0.411	0.411	0.00
$C = (2, 2, 2, 2)$, $\mu = (1.5, 0.8, 0.8, 1.5)$	$P_1(0)$	0.367	0.351	– 4.36
	$P_1(2)$	0.311	0.323	3.86
	L_1	0.945	0.972	2.86
	$P_2(0)$	0.516	0.509	– 1.36
	$P_2(2)$	0.186	0.203	9.14
	L_2	0.670	0.693	3.43
	$P_3(0)$	0.363	0.366	0.83
	$P_3(2)$	0.342	0.352	2.92
	L_3	0.980	0.986	0.61
	$P_4(0)$	0.523	0.523	0.00
	$P_4(2)$	0.210	0.216	2.86
	L_4	0.688	0.693	0.73
$C = (2, 2, 2, 2)$, $\mu = (2.0, 1.0, 1.0, 2.0)$	$P_1(0)$	0.482	0.471	– 2.28
	$P_1(2)$	0.213	0.217	1.88
	L_1	0.731	0.746	2.05
	$P_2(0)$	0.595	0.589	– 1.01
	$P_2(2)$	0.134	0.145	8.21
	L_2	0.538	0.557	3.53
	$P_3(0)$	0.445	0.445	0.00
	$P_3(2)$	0.263	0.273	3.80
	L_3	0.818	0.828	1.22
	$P_4(0)$	0.599	0.600	0.17
	$P_4(2)$	0.149	0.154	3.36
	L_4	0.550	0.554	0.73

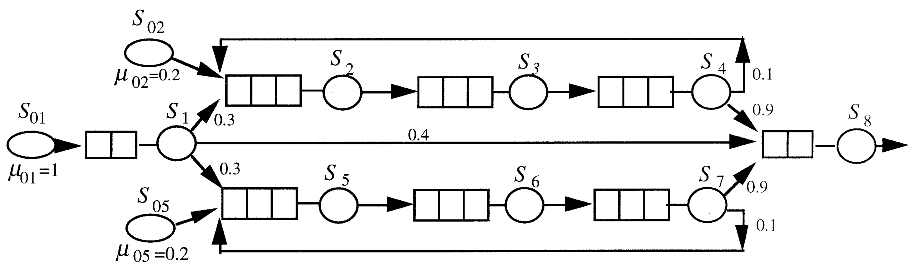


Figure 9. The eight-node network.

Table 3

Comparisons with simulation for the eight-node networks with feedback loops.

$C = (2, 2, 2, 2, 2, 2, 2, 2)$, $\mu = (1.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.5)$				$C = (2, 2, 2, 2, 2, 2, 2, 2)$, $\mu = (2.0, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 2.0)$			
Measures	Simulation	Approx.	Rel. error (%)	Measures	Simulation	Approx.	Rel. error (%)
$P_1(0)$	0.220	0.221	0.45	$P_1(0)$	0.417	0.421	0.96
$P_1(2)$	0.547	0.539	– 1.46	$P_1(2)$	0.311	0.302	– 2.89
L_1	1.317	1.318	0.08	L_1	0.895	0.881	– 1.56
$P_2(0)$	0.434	0.426	– 1.84	$P_2(0)$	0.614	0.616	0.33
$P_2(2)$	0.306	0.309	0.98	$P_2(2)$	0.145	0.142	– 2.07
L_2	0.872	0.882	1.15	L_2	0.531	0.527	– 0.75
$P_3(0)$	0.415	0.406	– 2.17	$P_3(0)$	0.599	0.600	0.17
$P_3(2)$	0.315	0.318	0.95	$P_3(2)$	0.153	0.149	– 2.61
L_3	0.900	0.912	1.33	L_3	0.554	0.550	– 0.72
$P_4(0)$	0.377	0.373	– 1.06	$P_4(0)$	0.549	0.552	0.55
$P_4(2)$	0.348	0.352	1.15	$P_4(2)$	0.187	0.185	– 1.07
L_4	0.971	0.979	0.82	L_4	0.639	0.634	– 0.78
$P_5(0)$	0.434	0.426	– 1.84	$P_5(0)$	0.613	0.616	0.49
$P_5(2)$	0.305	0.309	1.31	$P_5(2)$	0.145	0.142	– 2.07
L_5	0.871	0.882	1.26	L_5	0.532	0.527	– 0.94
$P_6(0)$	0.416	0.406	– 2.40	$P_6(0)$	0.598	0.600	– 0.33
$P_6(2)$	0.315	0.318	0.95	$P_6(2)$	0.154	0.149	– 3.25
L_6	0.898	0.912	1.56	L_6	0.556	0.550	– 1.08
$P_7(0)$	0.376	0.373	– 0.80	$P_7(0)$	0.547	0.552	0.91
$P_7(2)$	0.363	0.352	– 3.03	$P_7(2)$	0.189	0.185	– 2.12
L_7	0.972	0.979	0.72	L_7	0.643	0.634	– 1.40
$P_8(0)$	0.280	0.274	– 2.14	$P_8(0)$	0.368	0.364	– 1.09
$P_8(2)$	0.492	0.492	0.00	$P_8(2)$	0.380	0.380	0.00
L_8	1.212	1.217	0.41	L_8	1.012	1.015	0.30

in figures 7 and 8, while it is assumed to be blocking-after-service for the network in figure 9. Since all these problems have feedback loops, deadlocks occur in these problems. The frequency of deadlocks, however, is not very high, although significant.

The number of deadlocks which have been observed during simulation until 300,000 customers departed from the system ranges between 530 and 8,300.

As shown in tables 1 through 3, the new algorithm yields very good results. Most of the estimated values for the occupancy probabilities have a relative error less than 5%. However, for systems with frequent deadlocks, the results obtained by our algorithm are not as good because the algorithm does not take the effect of a deadlock into account in determining the unknown parameters. The three-node network in figure 10, which was taken from the paper of Jun and Perros [4], represents the case

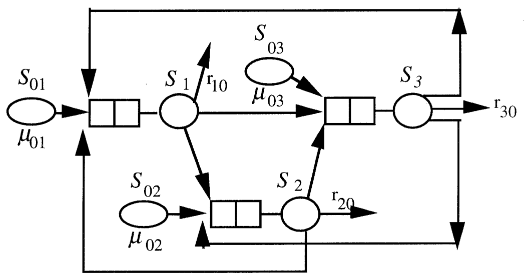


Figure 10. The three-node network in which deadlocks occur frequently.

in which deadlocks occur very frequently. In these problems, the number of deadlocks which have been observed during simulation until 300,000 customers departed from the system ranges between 11,700 and 40,000. Table 4 gives the results for these test problems. Table 4 shows that the new algorithm always overestimates the congestion measures of the system such as the mean queue length or the probability that the queue length is full. This is due to the fact that the new algorithm overestimates the delay time caused by blocking because it does not consider the effect of the deadlock resolution mechanism in which blocked customers get unblocked as soon as a deadlock occurs. As shown in table 4, some of the estimated values for the occupancy probabilities have a relative error larger than 20%. These results appear to be not as good as those of Jun and Perros' algorithm, in which most of the estimated values have a relative error less than 5%. However, the algorithm of Jun and Perros is very complicated. In addition, it has the restriction that it cannot solve the network if there is any node in the network that has more than two non-saturated upstream servers directly linked to it. For instance, the algorithm of Jun and Perros cannot solve the networks in figures 8 and 9 because node 4 (node 8) in figure 8 (figure 9) has three non-saturated upstream servers directly linked to it. Therefore, we believe that, from a practical point of view, the new algorithm is useful even for the networks in which deadlocks occur frequently although the results are not as good. The maximum number of iterations needed for convergence for the problems presented in this paper was 14 when we use algorithm 1. But in most cases, convergence occurred within 10 iterations (for a precision of 10^{-6}). The maximum CPU time was 0.06 seconds on an IBM PC-586.

Table 4

Comparisons with Jun and Perros for the three-node networks in which deadlocks occur frequently.

Case	Measures	Simulation	Jun and Perros	Approx.	Rel. error (%)
$C = (3, 3, 3)$	$P_1(0)$	0.194	0.202	0.184	– 5.15
$\mu_0 = (1.5, 2.0, 1.8)$	$P_1(3)$	0.367	0.366	0.393	7.08
$\mu = (3.0, 4.0, 3.5)$	L_1	1.768	1.750	1.823	3.11
$r_{10} = 0.5$, $r_{12} = 0.3$, $r_{13} = 0.2$,	$P_2(0)$	0.242	0.247	0.224	– 7.44
$r_{20} = 0.5$, $r_{21} = 0.2$, $r_{23} = 0.3$,	$P_2(3)$	0.296	0.303	0.326	10.14
$r_{30} = 0.5$, $r_{31} = 0.3$, $r_{32} = 0.2$	L_2	1.578	1.577	1.654	4.82
	$P_3(0)$	0.204	0.217	0.196	– 3.92
	$P_3(3)$	0.344	0.343	0.368	6.98
	L_3	1.717	1.691	1.766	2.85
$C = (2, 2, 2)$	$P_1(0)$	0.317	0.324	0.260	– 17.98
$\mu_0 = (2.0, 1.0, 1.5)$	$P_1(2)$	0.378	0.391	0.452	19.58
$\mu = (5.0, 4.0, 3.0)$	L_1	1.062	1.066	1.192	12.24
$r_{10} = 0.4$, $r_{12} = 0.2$, $r_{13} = 0.4$,	$P_2(0)$	0.326	0.336	0.277	15.03
$r_{20} = 0.2$, $r_{21} = 0.3$, $r_{23} = 0.5$,	$P_2(2)$	0.362	0.381	0.450	24.31
$r_{30} = 0.5$, $r_{31} = 0.2$, $r_{32} = 0.3$	L_2	1.035	1.046	1.173	13.30
	$P_3(0)$	0.153	0.166	0.124	– 18.95
	$P_3(2)$	0.622	0.612	0.685	10.13
	L_3	1.470	1.447	1.561	6.19
$C = (2, 2, 2)$	$P_1(0)$	0.205	0.213	0.177	– 13.70
$\mu_0 = (3.0, 1.0, 2.0)$	$P_1(2)$	0.514	0.517	0.561	9.14
$\mu = (4.0, 3.0, 2.0)$	L_1	1.309	1.304	1.384	5.73
$r_{10} = 0.5$, $r_{12} = 0.4$, $r_{13} = 0.1$,	$P_2(0)$	0.254	0.262	0.232	– 8.66
$r_{20} = 0.6$, $r_{21} = 0.3$, $r_{23} = 0.1$,	$P_2(2)$	0.470	0.474	0.513	9.15
$r_{30} = 0.4$, $r_{31} = 0.3$, $r_{32} = 0.3$	L_2	1.216	1.212	1.281	5.67
	$P_3(0)$	0.178	0.189	0.171	– 3.93
	$P_3(2)$	0.532	0.526	0.559	5.08
	L_3	1.354	1.337	1.388	2.51
$C = (2, 3, 3)$	$P_1(0)$	0.161	0.172	0.132	– 18.01
$\mu_0 = (2.0, 1.0, 1.0)$	$P_1(2)$	0.589	0.584	0.643	9.17
$\mu = (2.5, 2.0, 1.5)$	L_1	1.429	1.412	1.511	5.74
$r_{10} = 0.5$, $r_{12} = 0.3$, $r_{13} = 0.2$,	$P_2(0)$	0.161	0.175	0.140	– 13.04
$r_{20} = 0.5$, $r_{21} = 0.3$, $r_{23} = 0.2$,	$P_2(3)$	0.400	0.407	0.462	15.50
$r_{30} = 0.4$, $r_{31} = 0.4$, $r_{32} = 0.2$	L_2	1.876	1.861	2.006	6.93
	$P_3(0)$	0.113	0.125	0.101	– 10.62
	$P_3(3)$	0.481	0.480	0.537	11.64
	L_3	2.089	2.061	2.188	4.74

4. Properties of algorithm 2 for the merge configuration

In the case of a merge configuration, we can develop some theoretical results such as the convergence of the algorithm as well as the existence and uniqueness of the solution. In this section, we present these results. The merge configuration we consider consists of K saturated servers and $K + 1$ non-saturated servers, as shown in figure 11.

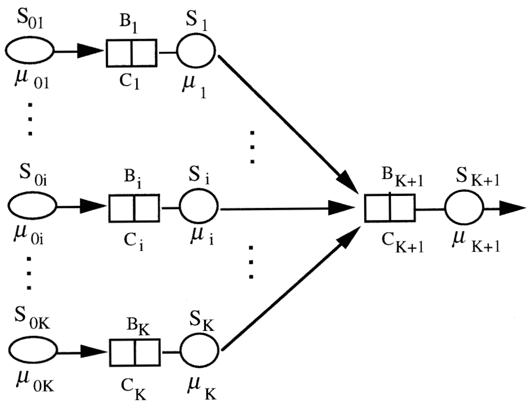


Figure 11. Merge configuration.

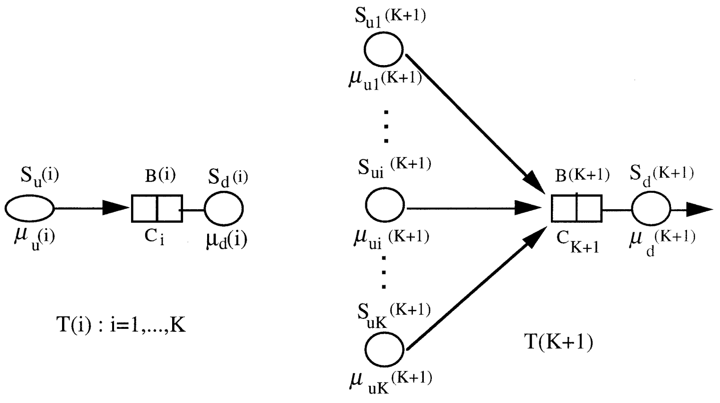


Figure 12. Decomposition of the network into subsystems.

To analyze this network, we decompose the network into $K + 1$ subsystems, $T(i)$, $i = 1,\dots,K + 1$, as shown in figure 12.

In the merge configuration, the upstream server, $S_u(i)$, in subsystem $T(i)$ represents S_{0i}, $i = 1,\dots,K$, and the downstream server, $S_d(K + 1)$, in subsystem $T(K + 1)$ represents S_{K+1} in the original model. Therefore, we have $\mu_u(i) = \mu_{0i}$, $i = 1,\dots,K$, and $\mu_d(K + 1) = \mu_{K+1}$. Using these facts, the sets of equations (5) and (7) which determine the service rates of the upstream and downstream servers in each subsystem can be simplified to:

$$\frac{1}{\mu_d(i)} = \frac{1}{\mu_i} + \sum_{n=0}^{K-1} P_{bi}(n:K+1)(n+1)\frac{1}{\mu_{K+1}}, \qquad i = 1,\dots,K, \tag{11}$$

$$\frac{1}{\mu_{ui}(K-1)} = \frac{1}{\mu_i} + P_S(i)\frac{1}{\mu_{0i}}, \qquad i = 1,\dots,K. \tag{12}$$

As a result, we have the following simple version of algorithm 2, which we will call algorithm 2(M).

Algorithm 2(M)

```
Initialization:
    Set  $\mu_u(i) = \mu_{0i}$, $i = 1,\dots,K$,  $\mu_d(i) = \mu_i$,  $i = 1,\dots,K+1$.
Iteration step:
    Step 1: For $i = 1, 2,\dots,K$ do:
        1.1. Calculate $P_S(i)$ using (3).
        1.2. Calculate $\mu_{ui}(K+1)$ using (12).
    Step 2:
        2.1. Calculate $P_{bi}(n:K+1)$ using (4), $n = 0, 1,\dots,K-1$, $i = 1, 2,\dots,K$.
        2.2. Calculate $\mu_d(i)$ using (11), $i = 1, 2,\dots,K$.
    Step 3: If convergence condition is satisfied, stop.
        Otherwise, go to step 1.
```

To prove the theoretical results pertaining to this algorithm, we need to develop some properties associated with each subsystem. We give three of these properties below. Since the proofs of these properties are very long and somewhat involved, we do not provide them in this paper for the sake of conciseness. The proof of property 1 is given in Dallery and Frein [2] and the proofs of properties 2 and 3 can be found in Lee et al. [10].

Property 1. Let $\mu_u^1(i), \mu_d^1(i)$ and $\mu_u^2(i), \mu_d^2(i)$ be two sets of parameters to subsystem $T(i)$, $1 \le i \le K$. Suppose $\mu_u^1(i) = \mu_u^2(i)$. Then the following relationship holds:

$$\text{If } \mu_d^2(i) \le \mu_d^1(i) \quad \text{then} \quad P_s^2(i) \le P_s^1(i) \quad \text{and} \quad X_d^2(i) \le X_d^1(i).$$

Property 2.

$$\sum_{n=0}^{K-1} P_{bj}(n:K-1)(n+1) \text{ is increasing in } \mu_{ui}(K+1), \qquad j = 1,\dots,K,\ i = 1,\dots,K.$$

Property 3. Consider the following two sets of parameters pertaining to subsystem $T(K+1)$:

$$(\mu^1_{u1}(K+1),\dots,\mu^1_{uK}(K+1)),\mu^1_d(K+1),$$
$$(\mu^2_{u1}(K+1),\dots,\mu^2_{uK}(K+1)),\mu^2_d(K+1).$$

Suppose $\mu^1_d(K+1)=\mu^2_d(K+1)$. Define sets A, B and C such that

$$A=\{i\in S/\mu^1_{ui}(K+1)<\mu^2_{ui}(K+1)\},$$
$$B=\{i\in S/\mu^1_{ui}(K+1)>\mu^2_{ui}(K+1)\},$$
$$C=\{i\in S/\mu^1_{ui}(K+1)=\mu^2_{ui}(K+1)\},$$

where $S=\{1,2,\dots,K\}$. Then the following relationships hold:

$$\sum_{i\in A}X^1_{ui}(K+1)<\sum_{i\in A}X^2_{ui}(K+1)\quad\text{and}\quad\sum_{i\in B}X^1_{ui}(K+1)>\sum_{i\in B}X^2_{ui}(K+1).$$

We are now ready to provide the proofs for the existence and uniqueness of the solution as well as the convergence of the algorithm. The general procedure of the proofs is the same as the one in Dallery and Frein [2] for the tandem configuration. The following property, stated as theorem 1, answers the question of uniqueness of the solution provided that the existence is guaranteed.

Theorem 1. The system of equations *SE*3 has at most one solution.

Proof. We assume that *SE*3 has two different solutions and show that this leads to a contradiction. All the quantities that pertain to the first solution (respectively, the second solution) will be denoted by a superscript 1 (respectively, 2). Let $\mu^j_d(i)$ and $\mu^j_{ui}(K+1)$, for $i=1,\dots,K$, $j=1,2$, be the two solutions. Define sets A, B and C such that

$$A=\{i\in S/\mu^1_d(i)>\mu^2_d(i)\},$$
$$B=\{i\in S/\mu^1_d(i)<\mu^2_d(i)\},$$
$$C=\{i\in S/\mu^1_d(i)=\mu^2_d(i)\},$$

where $S=\{1,2,\dots,K\}$. Then from property 1, it follows for each subsystem $T(i)$ for $i\in A$, that

$$P^2_s(i)<P^1_s(i)\quad\text{and}\quad X^2_d(i)<X^1_d(i)\quad\text{for } i\in A.$$

Applying this result to (12), we obtain

$$\mu^1_{ui}(K+1)<\mu^2_{ui}(K+1)\quad\text{for } i\in A.$$

Since *SE*3 satisfies conservation of flow, the fact that $X^2_d(i)<X^1_d(i)$ implies that

$$X^2_{ui}(K+1)<X^1_{ui}(K+1)\quad\text{for } i\in A. \tag{13}$$

Similarly, for each subsystem, $T(i)$ for $i \in B$, we have

$$\begin{aligned} &P_s^1(i) < P_s^2(i), \quad X_d^1(i) < X_d^2(i), \quad \mu_{ui}^2(K+1) < \mu_{ui}^1(K+1), \\ &X_{ui}^1(K+1) < X_{ui}^2(K+1). \end{aligned} \tag{14}$$

Inequalities (13) and (14) imply that

$$\sum_{i \in A} X_{ui}^1(K+1) > \sum_{i \in A} X_{ui}^2(K+1) \quad \text{and} \quad \sum_{i \in B} X_{ui}^1(K+1) < \sum_{i \in B} X_{ui}^2(K+1).$$

These inequalities, however, are a contradiction to property 3. □

We now provide the proof of the convergence of algorithm 2(M).

Theorem 2. Algorithm 2(M) always converges to a solution of system *SE*3.

Proof. It is easy to check that algorithm 2 (therefore, algorithm 2(M)) is nothing but the method of Gauss–Seidel applied to the system of fixed point type equations. The aim of the algorithm is to solve system *SE*3 and therefore to obtain values for the unknown parameters $\mu_{ui}(K+1)$ and $\mu_d(i)$, for $i = 1,\dots,K$. It consists of an initialization step followed by an iteration step. During each iteration, new values of $P_s(i)$, $\mu_{ui}(K+1)$ $P_{bi}(n:K+1)$ and $\mu_d(i)$ are obtained. Let n denote the index of the iteration. Let $P_s^n(i)$, $\mu_{ui}^n(K+1)$, $P_{bi}^n(n:k+1)$ and $\mu_d^n(i)$ denote the parameters obtained during the nth iteration. Let $\mu_d^0(i)$, $i = 1,\dots,K$ denote the initial value of the service rate of $S_d(i)$ given by $\mu_d^0(i) = \mu_i$, $i = 1,\dots,K$.
During the nth iteration, steps 1 and 2 are successively executed. In step 1, subsystems $T(i)$, $i = 1,\dots,K$, are successively analyzed. The parameters of subsystem $T(i)$ are then $\mu_{ui}^n(K+1)$ and $\mu_d^{n-1}(i)$. Similarly, in step 2, subsystems $T(i)$, for $i = K,\dots,1$, are successively analyzed. The parameters of subsystem $T(i)$ are then $\mu_{ui}^n(K+1)$ and $\mu_d^n(i)$. Note that we have $\mu_u^n(i) = \mu_i$, $i = 1,\dots,K$, and $\mu_d^n(K+1) = \mu_{K+1}$ for any n.

Part 1. In the first part of the proof, we show that for any two successive iterations n and $n+1$, we have

$$\mu_{ui}^{n+1}(K+1) \geq \mu_{ui}^n(K+1), \quad \text{for } i = 1,\dots,K \tag{15}$$

and

$$\mu_d^{n+1}(i) \leq \mu_d^n(i) \quad \text{for } i = 1,\dots,K. \tag{16}$$

Part a. Consider step 1 of the algorithm. Let us show that

$$\begin{aligned} &\text{If } \mu_d^n(i) \leq \mu_d^{n-1}(i) && \text{for } i = 1,\dots,K, \ \text{then} \\ &\mu_{ui}^{n+1}(K+1) \geq \mu_{ui}^n(K+1) && \text{for } i = 1,\dots,K. \end{aligned} \tag{17}$$

Consider subsystem $T(i)$. From property 1, we obtain $P_s^{n+1}(i) \leq P_s^n(i)$ for $i = 1,...,K$. Now from (12), we get $\mu_{ui}^{n+1}(K+1) \geq \mu_{ui}^n(K+1)$. Then by induction on i, we obtain (17).

Part b. Consider step 2 of the algorithm. Let us show that

$$\begin{aligned} &\text{If } \mu_{ui}^{n+1}(K+1) \geq \mu_{ui}^n(K+1) && \text{for } i = 1,\dots,K, \quad \text{then} \\ &\mu_d^{n+1}(i) \leq \mu_d^n(i) && \text{for } i = 1,\dots,K. \end{aligned} \tag{18}$$

From property 2, we obtain

$$\sum_{j=0}^{K-1} P_{bk}^{n+1}(j:K+1)(j+1)\frac{1}{\mu_{K+1}} \geq \sum_{j=0}^{K-1} P_{bk}^{n}(j:K+1)(j+1)\frac{1}{\mu_{K+1}}.$$

Now from (11), we get $\mu_d^{n+1}(i) \leq \mu_d^n(i)$ for $i = 1,\dots,K$. Then by induction on i, we obtain (18).

Part c. From (17) and (18), we get the implication

$$\begin{aligned} &\text{If } \mu_d^n(i) \leq \mu_d^{n-1}(i), \quad i = 1,\dots,K, \quad \text{then} \\ &\mu_{ui}^{n+1}(K+1) \geq \mu_{ui}^n(K+1) \quad \text{and} \quad \mu_d^{n+1}(i) \leq \mu_d^n(i), \quad i = 1,\dots,K. \end{aligned} \tag{19}$$

From (11), it follows that

$$\mu_d^1(i) \leq \mu_d^0(i) \quad \text{for } i = 1,\dots,K. \tag{20}$$

Then (15) and (16) follow from (19) and (20).

Part 2. Equation (15) states that each series $\{\mu_{ui}^n(K+1),\ n = 1,2,\dots\}$ for $i = 1,\dots,K$ is non-decreasing. Moreover, from (12), it follows that $\mu_{ui}^n(K+1) \leq \mu_i$. Therefore, as each series is non-decreasing and upper bounded, it converges. Similarly, equation (16) states that each series $\{\mu_d^n(i),\ n = 1,2,\dots\}$ for $i = 1,\dots,K$ is non-increasing. Moreover, from (11) it follows that

$$\mu_d^n(i) \geq \frac{\mu_i \mu_{K+1}}{\mu_{K+1} + K\mu_i}.$$

This bound is obtained by considering the case where $P_{bi}^n(K-1:K+1) = 1$. Therefore, as each series is non-increasing and lower bounded, it converges. Now, since the algorithm is the application of the method of Gauss–Seidel to solve the system of equations $SE3$, the asymptotic values of $\mu_d^n(i)$, $\mu_{ui}^n(K+1)$, $i = 1, 2,\dots,K$, are indeed a solution of $SE3$. □

We know from theorem 2 that $SE3$ has at least one solution. Furthermore, from theorem 1, we know that $SE3$ cannot have two or more solutions. Therefore, we have the following property, stated as corollary 2.

Corollary 2. The solution of the system of equations $SE3$ exists and is unique.

From corollary 1, the three systems of equations are equivalent. Thus, the solution of the system of equations $SE1$ (and $SE2$) also exists and is unique.

5. Conclusions

We have presented a new approximate algorithm for analyzing an arbitrary configuration of open queueing networks with finite buffers. The approximate technique decomposes the network into a set of subsystems. Each subsystem is composed of one or many upstream servers and one downstream server separated by a finite buffer. The new algorithm is based on the symmetrical approach and offers a symmetrical representation of the decomposition method. The new algorithm is quite simple because service times at each subsystem are characterized by exponential distributions. The new algorithm is also very general in that it can analyze all the classes of models considered by the previous studies under blocking-after-service mechanism. For the class of models considered by Lee and Pollock [11], the new algorithm yields the same results as that of Lee and Pollock. The numerical results, however, have shown that the new algorithm takes less CPU execution time than the one proposed by Lee and Pollock. As reported in Lee and Pollock [11], the new algorithm yields very accurate results for the cases of networks without feedback loops. For the cases of networks with feedback loops, the numerical results have shown that the new algorithm usually converges fast and yields good results unless deadlocks occur too frequently. In particular, for the merge configuration, we could prove the convergence of the algorithm as well as the existence and uniqueness of the solution, which to the best of our knowledge has never been done previously. This new algorithm holds promise as a useful tool in the analysis of arbitrary configuration of open queueing networks with finite buffers.

Acknowledgement

The research of the first author was supported through KOSEF Grant 951-1010-012-2.

References

[1] T. Altiok and H.G. Perros, Approximate analysis of arbitrary configurations of open queueing networks with blocking, Annals of Oper. Res. 9(1987)481–509.

[2] Y. Dallery and Y. Frein, On decomposition methods for tandem queueing networks with blocking, Oper. Res. 41(1993)386–399.

[3] Y. Dallery and S.B. Gershwin, Manufacturing flow line aystems: A review of models and analytical results, Queueing Sys. 12(1992)3–94.

[4] K.P. Jun and H.G. Perros, Approximate analysis of arbitrary configuration of queueing networks with blocking, in: *Proc. 1st International Workshop on Queueing Networks with Blocking*, North-Holland, Amsterdam, 1989, pp. 259–280.

[5] L. Kerbache and J. MacGregor Smith, The generalized expansion method for open finite queueing networks, Eur. J. Oper. Res. 32(1987)448 – 461.

[6] D.D. Kouvatsos and S.G. Denazis, Entropy maximized queueing networks with blocking and multiple job classes, Perform. Eval. 17(1993)189 – 205.

[7] D.D. Kouvatsos and N.P. Xenios, Maximum entropy analysis of general queueing networks with blocking, in: *Proc. 1st International Workshop on Queueing Networks with Blocking*, North-Holland, Amsterdam, 1989, pp 281 – 309.

[8] D.D. Kouvatsos and N.P. Xenios, MEM for arbitrary queueing networks with multiple general servers and repetitive-service-blocking, Perform. Eval. 10(1989)169 – 195.

[9] J. Labetoulle and G. Pujolle, Isolation method in a network of queues, IEEE Trans. Soft. Eng. 6(1980)373 – 381.

[10] H.S. Lee, A. Bouhchouch, Y. Dallery and Y. Frein, Performance evaluation of open queueing networks with arbitrary configuration and finite buffers, Technical Report, Dept. of Industrial Eng., Kyung Hee Univ., Korea, 1996.

[11] H.S. Lee and S.M. Pollock, Approximate analysis of open acyclic exponential queueing networks with blocking, Oper. Res. 38(1990)1123 – 1134.

[12] R.O. Onvural and H.G. Perros, On equivalencies of blocking mechanisms in queueing networks with blocking, Oper. Res. Lett. 5(1986)293 – 297.

[13] H.G. Perros, *Queueing Networks with Blocking,* Oxford University Press, New York, 1994.

[14] H.G. Perros and P.M. Snyder, A computationally efficient approximation algorithm for analyzing open queueing networks with blocking, Perform. Eval. 9(1989)217 – 224.

[15] Y. Takahashi, H. Miyahara and J. Hasegawa, An approximation method for open restricted queueing networks, Oper. Res. 28(1980)594 – 602.