1979

Solution of General Queueing Networks Using Norton's Theorem

Stephen Tolopka

Report Number:
79-314

Solution of General Queueing Networks Using Norton's Theorem

Stephen Tolopka

Technical Report TR-314
Computer Sciences Department
Purdue University

September 1979

1. Introduction

Closed queueing networks have exact, product-form solutions only if they fit into the categories defined by Baskett et. al. [BASK75]. Other types of networks which may be of interest do not quite conform to these specifications. One such class contains queues with first-come-first-served queueing disciplines and non-exponential service distributions. These networks may be solved approximately by using an iterative technique based on the so-called Norton's Theorem decomposition which reduces a network to a corresponding (solvable) two-queue network.

This report discusses a program written in Pascal at Purdue University which employs Norton's Theorem. Perhaps more importantly, it attempts to fill in some of the reasoning and implementation details omitted by the more formal papers on the subject.

Section 2 discusses Norton's Theorem and the manner in which it was applied in the program. Section 3 discusses the techniques which were used to solve the two-queue model; emphasis is placed on a recursive technique developed by Sauer [SAUE75]. Section 4 describes the extension of the basic algorithm to queues with load-dependent service rates. The input format of the program is described in Section 5, with some comments on the applicability of the program in Section 6. There are two appendices: the first contains derivations of the formulas used for modeling non-exponential distributions by the method of stages, while the second contains the equations used in the recursive solution of the two-queue model.

2. Norton's Theorem

Given a network of M queues, Norton's Theorem states that it is possible to replace (M-1) of the queues with a single queue with load-dependent service rates such that performance characteristics (throughput, utilization, mean queue length, etc.) at the M-th queue remain unchanged from what they were in the original model. The exact reduction is easy to make if the queues all have exponential service rates, but difficult if they do not.

To make the decomposition, we first assume that all the queues which are to be aggregated have exponential service distributions. (It is this assumption which makes the solution only approximate.) The aggregated exponential queues are then replaced by a single queue with load-dependent service rates. The rates are chosen so that the effect of the new queue upon the remaining original queue is exactly the same as that of the aggregated exponential queues. A composite coefficient of variation is then assigned to the new equivalent server. (Coefficient of variation is a measure of the skewness of a distribution, defined as the standard deviation divided by the mean.)

This decomposition is done in the procedure EQUIVALENTSERVER in the program. Suppose that we have M queues in the network, and that

we wish to aggregate all but the k-th queue. Let Y_i be the relative visit count to station i, obtained by solving the equations

$$Y_i = \sum_{j=1}^{M} Y_j p_{ji}$$

where p_{ji} is the transition probability from station j to station i. Then if μ_i is the service rate at station i, $X_i = Y_i/\mu_i$ is the relative throughput at station i. The X_i's could then normally be used with Buzen's algorithm [BUZE73] to find the normalizing constants G_n for various loads, n, and, finally, the throughputs for each of the queues in the network. What we do instead is to "short-circuit" server k by setting X_k to zero (i.e., $\mu_i=\infty$), and measure the throughput through station k. This measures the throughput of the rest of the network with station k removed, and can be used as the service rate for an equivalent server.

Assuming that there are N customers in the network, we compute G_n (n=0,1,...,N) using the computed X_i's with $X_k=0$. Buzen then tells us that the throughput through station i with n customers present is

$$T_i = \frac{\mu_i X_i G_{n-1}}{G_n} = \frac{Y_i G_{n-1}}{G_n}$$

The throughput at station k must be the sum of the throughputs arriving at k from the other queues in the network; i.e.,

$$T_k = \sum_{j=1}^{M} p_{jk} T_j = \sum_{j=1}^{M} p_{jk} \frac{Y_j G_{n-1}}{G_n}$$

$$= \frac{G_{n-1}}{G_n} \sum_{j=1}^{M} p_{jk} Y_j$$

$$= \frac{G_{n-1}}{G_n} Y_k \ .$$

We then take T_k as the service rate of the equivalent server when there are n customers at the equivalent server.

The composite coefficient of variation, C*, is computed as a weighted average of the coefficients of variation of the individual queues comprising the equivalent server. The weights used are the relative throughput rates; i.e.,

$$C^* = \frac{\sum_{\substack{i=1 \\ i \neq k}}^{M} Y_i C_i / \mu_i}{\sum_{\substack{i=1 \\ i \neq k}}^{M} Y_i / \mu_i}$$

The equivalent server is now completely parameterized, and we can solve this two queue model by one of the techniques described in section 3 for the throughput T_k and mean queue length $\bar{n}_k$ at queue k. (We are uninterested in the statistics for the composite queue.)

Remember that these solutions are only approximate due to the exponential assumption. We need to check the "goodness" of the solutions, and possibly refine them before accepting them as a final answer. The method used is due to Chandy, Herzog, and Woo [CHAN75].

Suppose that we have gone through the decomposition-solution procedure described above for each of the queues in the network. One obvious way to check solution validity is to compare the sum of the mean queue lengths with the number of customers in the network: they should match. We should also check to see that the throughputs are correct. To do this, we first compute a normalized throughput as $T_i' = T_i/Y_i$. The normalized throughputs should be equal, since

$T_j' = T_j/Y_j$

$= \frac{1}{Y_j} \sum_{i=1}^{M} p_{ij} T_i$ (throughput at j must be the sum of the incoming throughputs)

$= KY_j/Y_j$ (notice that the T_i's are merely another solution to the equations which produced the Y_i's, and so must be some constant multiple of the Y_i's)

$= K = KY_k/Y_k$

$= \frac{1}{Y_k} \sum_{i=1}^{M} p_{ik} T_i$

$= T_k/Y_k = T_k'$

We can then compute a mean normalized throughput as

$$MNT = \sum_{i=1}^{M} T_i'/M \quad ,$$

and compare the individual throughputs with the mean.

We will say that this iteration of the solution has excess queue length if

$$\sum_{i=1}^{M} \bar{n}_i > N(1+\varepsilon) ,$$

and insufficient queue length if

$$\sum_{i=1}^{M} \bar{n}_i < N(1-\varepsilon) ,$$

where ε is some error tolerance, typically .01 or .05. In addition, a queue will be said to have excess throughput if $T_i' > MNT(1+\varepsilon)$, and insufficient throughput if $T_i' < MNT(1-\varepsilon)$.

For the remainder of this discussion, we must keep two sets of rates in mind: a set of service rates, S_i, which may be adjusted to reduce the error in the solution, and are used only in computing the composite rates for the equivalent server; and a set of real rates, R_i, which are the original service rates in the model, and are used to parameterize the isolated queue and provide performance measures.

The service rates are adjusted as follows:

1) If there is excess queue length, then for each queue having insufficient throughput, set $S_i' = S_i T_i'/MNT$.

2) If there is insufficient queue length, then for each queue having excess throughput, set $S_i' = S_i T_i'/MNT$.

3) If there is excess queue length or insufficient queue length, and no adjustments were made in either 1) or 2), then set

$$S_i' = S_i N \Big/ \sum_{j=1}^{M} \bar{n}_j$$

for each queue in the network.

4) If there is neither excess nor insufficient queue length, then compute $S_i' = S_i T_i'/MNT$ for each queue.
If $|S_i' - S_i| < \varepsilon$ for all i, then the model is within the error tolerance, and the solution may be accepted.

If the stopping criteria in 4) were not met, then we take the new S_i' and go back to the decomposition-solution stage to try again.

The reasoning behind the adjustments performed in 1) is as follows: we must reduce the sum of the mean queue lengths. If queue i has insufficient throughput, then $S_i' < S_i$. Remember that the S_i's are used only when determining composite queue rates. Thus the rates of the equivalent servers corresponding to every queue except queue i will be reduced. This implies that the throughput of the equivalent server (and hence, of the isolated queue) will be reduced, and that the mean queue length at the equivalent server will increase, providing a corresponding decrease at the isolated queue. Thus the sum of the mean queue lengths will decrease (which makes the throughput at queue i more nearly equal to the others), and the new model should be closer to the convergence criteria. Similar reasoning may be used for the other adjustments.

In summary then, an algorithmic form for the solution technique.

1. For each queue k, $1 \leq k \leq M$, do

   a. Parameterize the server equivalent to the (M-1) queues not including queue k.

   b. Solve the resulting two queue model, obtaining the throughput and mean queue length for queue k.

2. Make adjustments to the service rates as necessary.

3. If the convergence criteria are satisfied, stop. Otherwise, go back to step 1.

3. Solving the Two Queue Model

In order to solve the two queue network, we must first develop a representation of the two queues which will reflect their non-exponential nature, and yet is not too difficult to solve. It has been shown that any distribution can be represented as a sum of exponential distributions. We use this fact to develop models which match the mean and coefficient of variation (i.e., the first two moments) of the given distribution. These models are primarily due to Sauer and Chandy [SAUE75a], and Sauer [SAUE75b].

Consider first a queue with mean service rate μ and coefficient of variation $c < 1$. (Figure 1). This queue may be represented by a series of exponential queues, called a Modified Erlang model (Figure 2) , where

$$1/(r-1)^{1/2} > c \geq 1/r^{1/2}, \quad p = \frac{2rc^2+r-2-(r^2+4-4rc^2)^{1/2}}{2(c^2+1)(r-1)},$$

and $\mu_1=\mu_2=\ldots=\mu_r=(r-p(r-1))\mu$. (See Appendix I).

The probability that a customer leaves the station after passing through the first stage is p; with probability (1−p) he passes through the remaining (r−1) stages before departing.

On the other hand, if c > 1, then we may represent the queue with a hyperexponential model (Figure 3) , where

$$p = \frac{c^2+1-(c^2-1)^{1/2}}{2(c^2+1)} , \qquad \mu_1=2p\mu \text{ , and } \mu_2=2(1-p)\mu.$$

In order to simplify the solution to the problem, we can redraw this model to resemble the Modified Erlang by making sure that $\mu_1 \geq \mu_2$, and then letting $p' = p + (1-p)^2/p$, obtaining the model shown in Figure 4.

Since we have a two queue network, we can model each queue as above, and then denote the state space as the set of (i,j,n)'s, where i is the stage in which the customer at queue 1 is currently residing, j is the stage in which the customer at queue 2 is currently residing, and n is the number of customers at queue 1.

There are several ways to solve for the probabilities of the various states. One way is to set up and solve all the balance equations directly. This technique is easy to implement, but severely limits the size of the state space. Even for models with two stages per queue, the number of customers is limited to about forty before the storage space needed for the coefficient matrix has grown too large to be tractable.

The solution technique actually used in the program is an extension of work done by Sauer [SAUE75b], and makes use of the fact that it is possible to solve for P(i,j,n) in terms of only the P(i,j,n−1)'s and P(i,j,n−2)'s. We first assign an arbitrary value to the P(1,j,0)'s, and then solve for all the other state probabilities in terms of these. We can then determine how all the other probabilities depend on the P(1,j,0)'s, and can solve for the P(1,j,0)'s. A second pass through the state space using the correct starting values produces all the desired probabilities.

We need to take a closer look at how this is actually accomplished. First, we should note that each P(i,j,n) is actually a vector with R2 elements. (R2 is the number of stages in the representation of queue 2.) The initial assignment is $P(1,j,0) = e_j$, where e_j is a column vector with a one in the j−th row, and zeroes elsewhere. The reason for doing this is so that later we know that if

$$P(i,j,n) = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \alpha_{R2} \end{bmatrix}$$

then $P(i,j,n)=\alpha_1 P(1,1,0)+\alpha_2 P(1,2,0)+\ldots+\alpha_{R2} P(1,R2,0)$.

Next, we pass through the state space starting at the P(i,j,1)'s and finishing with the P(i,j,N)'s. Notice that we only need to have the P(i,j,n-1)'s and P(i,j,n-2)'s available at any one time. Hence, the number of customers in the network has no bearing on the amount of memory needed for this technique. As we pass through the state space, we keep a sum vector S which adds up all the individual probability vectors. (See Appendix II for the equations and proof of their validity.)

After passing completely through the state space, we observe that we could directly compute the P(1,j,N-1)'s from their own balance equations now that we have the P(i,j,N)'s. Call these recomputed probabilities the P'(1,j,N-1)'s. If we have a solution, it should be true that $P(1,j,N-1)-P'(1,j,N-1)=0$, or $D_j=0$. These difference equations, coupled with the fact that all of the probabilities must sum to 1, produce the following set of equations:

$$\begin{bmatrix} D_2 \\ D_3 \\ \cdot \\ \cdot \\ \cdot \\ D_{R2} \\ S \end{bmatrix} \quad X = \quad \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}$$

which, when solved, give us the correct values for the P(1,j,0)'s! We can now plug these in as the starting values, and repeat this procedure to get all the probabilities. Performance measures of interest can be computed along the way.

A third solution technique is to solve the balance equations by iteration. Here, we pick some arbitrary values for all the probabilities, and then recompute them using the balance equations directly. The resulting probabilities are then normalized so that they sum to one, and the process is repeated until the normalization constant is sufficiently close to one. Note that here, we will probably need more than two iterations to obtain solutions. However, we hope to improve the accuracy of the solution by avoiding the difference equations.

4. Extensions

The basic techniques described above assume that all the queues in the network have load-independent service rates. However, we can produce equivalent servers for queues with load-dependent service rates by using the corresponding algorithms of Buzen [BUZE73] or Bruell [BRUE78]. Solving the two-queue model calls for a straightforward extension of the recursive technique; now both servers

(instead of only the equivalent server) may be load-dependent.

Although general load-dependent service rates have not been implemented in the program, the special case of multiple identical servers at a service center has been handled. This effectively includes the infinite server queueing discipline, which is implemented as a service center with as many servers as there are customers in the network. All the necessary equations for solving the two-queue network my be found in Appendix II.

An option permitting the user to generate a file of performance criteria of interest for later input to a plotting routine was never completed. However, the vestigial plot flag remains as part of the input specification. This flag should always be turned off.

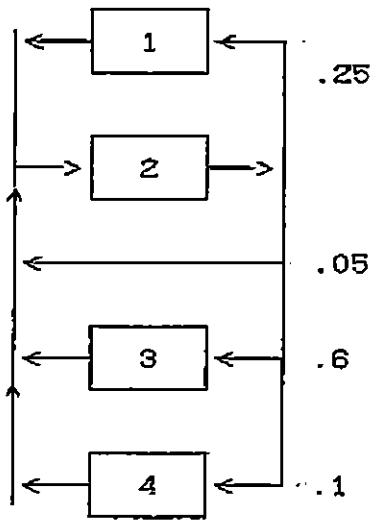
5. Input Format

Input to the program should be in the following format.

```
                M
                NMIN NMAX {INCR}      (Model is solved for NMIN to NMAX
                                       customers in increments of INCR;
                                       INCR may be omitted if NMIN=NMAX)
          ⎧     NUMBER OF SERVERS AT DEVICE i        (-1 means infinite)
          ⎪     FLAG Si Ci           (where FLAG=0 => Si is a service time;
one set   ⎪⎧                  ⎫                 FLAG=1 => Si is a service rate)
of these  ⎪⎪ I J1 P(I,J1)      ⎪
for each  ⎨⎪ I J2 P(I,J2)      ⎬    Branching probabilities from device i
device    ⎪⎪    .              ⎪    to device Jk.  The list is ended with
          ⎪⎪    .              ⎪    a -1.
          ⎪⎪ I Jl P(I,Jl)      ⎪
          ⎪⎪ -1                ⎪
          ⎩⎩                   ⎭
                PLOTFLAG             (0=> OFF; 1 => ON)
```

Any number of models may be described in one input file.

Example:



Device	Service Time (secs.)	C	Nr. Servers
1	11.012	1.12	∞
2	0.043	2.36	2
3	0.181	0.77	3
4	7.189	0.95	∞

For N=50 to 100 by 5's
Plot flag off

```
4
50 100 5
-1
0 11.012 1.12
1 2 1.0
-1
2
0 0.043 2.36
2 1 0.25
2 2 0.05
2 3 0.6
2 4 0.1
-1
3
0 0.181 0.77
3 2 1.0
-1
-1
0 7.189 0.95
4 2 1.0
-1
0
```

6. Applicability

When the program derives solutions for a particular model, they tend to be quite good. Errors, even for mean queue lengths, are only a few percent. However, solutions cannot always be obtained. The number of customers for which a particular model can be successfully solved is dependent on the model. The more balanced the model, the more difficult it is to obtain a solution. Some models will admit solutions for only thirty customers, while others (similar to that described in section 5) have been successfully solved for as many as eighty-five customers. The difficulties seem to lie in the fact that as the state space grows, we have to subtract very small probabilities from other very small probabilities, while the answer loses significant digits.

The third solution technique described in section 3 was an attempt to avoid this problem by avoiding subtractions. Unfortunately, the technique uses many iterations, and in fact does not seem to converge as well as the implemented technique. The reasons for this are currently unknown.

Throughout this paper it has been implicitly assumed that the queueing networks under discussion contain only a single customer class. Some thought has been given to implementing a multiple-class version of the program. Although this might be successfully accomplished for small numbers of classes (i.e., two or three), it does not seem feasible in the general case since one must produce a service rate for every possible combination of customers which could be at the equivalent server. For the single-class case, this involves only N solutions (where N is the number of customers in the network); for multi-class networks, the number of required solutions grows combinatorially. Thus it would seem that this problem requires a different approach.

Appendix I

This appendix contains the derivations of the formulas for the branching probabilities and service rates of the stages of the Modified Erlang and hyperexponential models, as well as proof that the hyperexponential can also be modeled as a Modified Erlang.

Modified Erlang

The easiest way to derive formulas for p and μ is to use Laplace transforms. In [KLEI75], we see that the Laplace transform for the Erlang model as we have defined it is

$$B^*(s) = p(\mu_1/(s+\mu_1)) + (1-p)\prod_{i=1}^{r} \mu_i/(s+\mu_i),$$

where μ_i is the service rate of stage i. Since in our case, $\mu_1=\mu_2=\ldots=\mu_r=\mu$, we can simplify this to

$$B^*(s) = p\mu/(s+\mu) + (1-p)\mu^r/(s+\mu)^r$$

$$= p\mu(s+\mu)^{-1} + (1-p)\mu^r(s+\mu)^{-r}$$

Differentiating, we find

$$B^{*(1)}(s) = -p\mu(s+\mu)^{-2} - r(1-p)\mu^r(s+\mu)^{-(r+1)}.$$

From [KLEI75], we learn that the first moment, X_1, of any distribution may be found from its Laplace transform thusly:

$$X_1 = -B^{*(1)}(0) = p/\mu + r(1-p)/\mu \ .$$

Differentiating again, we obtain

$$B^{*(2)}(s) = 2p\mu(s+\mu)^{-3} + r(r+1)(1-p)\mu^r(s+\mu)^{-(r+2)},$$

and since the second moment of the distribution, X_2, can be found as $X_2 = B^{*(2)}(0)$, we obtain

$$X_2 = 2p/\mu^2 + r(r+1)(1-p)/\mu^2 \ .$$

Let d be the mean service rate of the device being modeled, and c be its coefficient of variation. Then since $c = (X_2-X_1)^{1/2}/ X_1$, we see that

$$c^2X_1^2 + X_1^2 = X_2$$

$$X_2 = (c^2+1)X_1^2 = (c^2+1)/d^2 \ ,$$

and so we find that

$$(1) \qquad p/\mu + r(1-p)/\mu = 1/d, \text{ and}$$

$$(2) \qquad 2p/\mu^2 + r(r+1)(1-p)/\mu^2 = (c^2+1)/d^2 \ .$$

Multiplying (1) by μd, we obtain

$$(3) \qquad \mu = d(p+r(1-p)).$$

Substituting (3) into (2) and cancelling the common d^2 term, we find

$$2p + r(r+1)(1-p) = (c^2+1)(p+r-rp)^2$$

$$2p + (r^2+r) - p(r^2+r) = (c^2+1)(p^2+r^2+r^2p^2+2pr-2p^2r-2pr^2)$$

$$p(2-r^2-r) + (r^2+r) = (c^2+1)(p^2(1+r^2-2r)+p(2r-r^2)+r^2)$$

$$(c^2+1)p^2(1+r^2-2r) + p(2rc^2-2r^2c^2+2r-2r^2-2+r^2+r) = -r^2c^2+r$$

$$p^2(c^2+1)(r-1) + p(r(-2c^2-1)+2) = (-r^2c^2+r)/(r-1)$$

$$p = \frac{\{2rc^2+r-2-[4r^2c^4+r^2+4+4r^2c^2-4r-8rc^2+4(k^2+1)(-r^2c^2+r)]^{1/2}\}}{2(c^2+1)(r-1)}$$

$$= \frac{\{2rc^2+r-2-(4r^2c^4+r^2+4+4r^2c^2-4r-8rc^2-4r^2c^4+4r+4rc^2-4r^2c^2)^{1/2}\}}{2(c^2+1)(r-1)}$$

$$= \{2rc^2+r-2-(-4rc^2+r^2+4)^{1/2}\}/2(c^2+1)(r-1) \ ,$$

which is the desired result. From (3), we can rearrange terms, and find that $\mu = d(r-p(r-1))$.

Hyperexponential

For the hyperexponential, we find that [KLEI75]

$$B^*(s) = p\mu_1/(s+\mu_1) + (1-p)\mu_2/(s+\mu_2) \ .$$

As before, we differentiate and find that

$$B^{*(1)}(s) = -p\mu_1(s+\mu_1)^{-2} - (1-p)\mu_2(s+\mu_2)^{-2}$$

$$-B^{*(1)}(0) = p/\mu_1 + (1-p)/\mu_2 = 1/d$$

$$B^{*(2)}(s) = 2p\mu_1(s+\mu_1)^{-3} + 2(1-p)\mu_2(s+\mu_2)^{-3}$$

$$B^{*(2)}(0) = 2p/\mu_1^2 + 2(1-p)/\mu_2^2 = (c^2+1)/d^2 \qquad (4)$$

Now if we choose $\mu_1 = 2dp$, we obtain

$$p/2dp + (1-p)/\mu_2 = 1/d$$

$$1/2d + (1-p)/\mu_2 = 1/d$$

$$\mu_2/2 + d(1-p) = \mu_2$$

$$\mu_2 = 2d(1-p),$$

verifying the formulas for the service rates of the two stages. We also find by substituting into (4) that

$$2p/(2dp)^2 + 2(1-p)/(2d(1-p))^2 = (c^2+1)/d^2$$

$$2p/4p^2 + 2(1-p)/4(1-p)^2 = c^2+1$$

$$2(1-p) + 2p = 4(c^2+1)p(1-p)$$

$$1 = 2(c^2+1)p(1-p)$$

$$2(c^2+1)p^2 - 2(c^2+1)p + 1 = 0$$

$$p = \{2(c^2+1) - (4c^4+8c^2+4-8c^2-8)^{1/2}\} / 4(c^2+1)$$

$$= \{2(c^2+1) - (4c^4-4)^{1/2}\} / 4(c^2+1)$$

$$= \{c^2 + 1 - (c^4-1)^{1/2}\} / 2(c^2+1) .$$

Finally, we need to show that the hyperexponential distribution can be successfully modeled by a modified Erlang as described in section 3.

Let p be the probability of branching to stage 1 in the hyperexponential, and q=1-p be the probability of branching to stage 2. We know that $\mu_1=2p\mu$ and $\mu_2=2q\mu$, where μ is the service rate of the device being modeled. In the modified Erlang, the branching probability p' is

$$p' = p + q^2/p = (p^2+q^2)/p ,$$

and so the Laplace transform is

$$B^*(s) = p'\mu_1/(\mu_1+s) + (1-p')\mu_1\mu_2/(\mu_1+s)(\mu_2+s)$$

$$= \frac{(p^2+q^2)(2p\mu)}{p(2p\mu+s)} + \frac{(p-q-p^2)(2p\mu)(2q\mu)}{p(2p\mu+s)(2q\mu+s)}$$

$$= \frac{-4p^3q\mu^2+4pq^3\mu^2+4p^2q\mu^2-4pq^3\mu^2-4p^3q\mu^2+(p^2+q^2)(2p\mu)s}{p(2p\mu+s)(2q\mu+s)}$$

$$= \frac{2\mu s(p^2+q^2)+4pq\mu^2}{(2p\mu+s)(2q\mu+s)} = \frac{2p^2\mu}{2p\mu+s} + \frac{2q^2\mu}{2q\mu+s}$$

$$= p\mu_1/(\mu_1+s) + q\mu_2/(\mu_2+s) ,$$

which is the Laplace transform of the hyperexponential distribution.

Appendix II

This appendix contains the equations used to recursively solve the two queue model, and proofs of their correctness. Important results are marked with a (*) as they appear.

The following terminology will be used:

- $\mu_{i,n}$: service rate of the i-th stage of queue 1 with n customers present
- $\lambda_{j,n}$: service rate of the j-th stage of queue 2 with n customers present at queue 1
- R1 : number of stages at queue 1
- R2 : number of stages at queue 2
- p : probability of departing queue 1 after receiving service at the first stage
- q : probability of departing queue 2 after receiving service at the first stage
- δp : (1-p) if we are talking about a transition from stage 1 to stage 2 of queue 1; 1 otherwise
- δq : (1-q) if we are talking about a transition from stage 1 to stage 2 of queue 2; 1 otherwise
- N : number of customers in the network

The state space transition diagrams in Figures 5 and 6 may aid in understanding the following material.

The system is solved in the following manner: for each $1 \leq n \leq N$, we first compute P(1,1,n), then all the P(i,1,n)'s, then the P(1,j,n)'s, and, finally, the remaining P(i,j,n)'s.

Conjecture 1:

For $1 \leq n \leq N-1$, $2 \leq l \leq R1$,

$$P(l,1,n) = \sum_{i=2}^{l} \left(\frac{\lambda_{R2,n-1}}{\mu_{l,n}} \prod_{k=i}^{l} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,R2,n-1) + \frac{q\lambda_{1,n-1}}{\mu_{l,n}} \prod_{k=i}^{l} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,1,n-1) \right) + \prod_{k=2}^{l} \frac{\delta p \mu_{k-1,n}}{\mu_{k,n}+\lambda_{1,n}} P(1,1,n)$$

Pf.: By induction on l; n is fixed, but arbitrary.

For l=2, we obtain

$$P(2,1,n) = \left(\frac{\lambda_{R2,n-1}}{\mu_{2,n}}\right)\left(\frac{\mu_{2,n}}{\mu_{2,n}+\lambda_{1,n}}\right) P(2,R2,n-1)$$

$$+ \left(\frac{q\lambda_{1,n-1}}{\mu_{2,n}}\right)\left(\frac{\mu_{2,n}}{\mu_{2,n}+\lambda_{1,n}}\right) P(2,1,n-1)$$

$$+ \left(\frac{(1-p)\mu_{1,n}}{\mu_{2,n}+\lambda_{1,n}}\right)\left(1\right) P(1,1,n)$$

$$= \{1/(\mu_{2,n}+\lambda_{1,n})\}\ \{\lambda_{R2,n-1}P(2,R2,n-1)$$

$$+ q\lambda_{1,n-1}P(2,1,n-1) + (1-p)\mu_{1,n}P(1,1,n)\},$$

which is correct from the flow diagram.

Assume true for l. Show true for l+1.

$$P(l+1,1,n) = \{1/(\mu_{l+1,n}+\lambda_{1,n})\}\ \{\lambda_{R2,n-1}P(l+1,R2,n-1)$$

$$+ q\lambda_{1,n-1}P(l+1,1,n-1) + \mu_{1,n}P(1,1,n)\}$$

(from the flow diagram)

$$= \frac{\lambda_{R2,n-1}}{\mu_{l+1,n}+\lambda_{1,n}} P(l+1,R2,n-1) + \frac{q\lambda_{1,n-1}}{\mu_{l+1,n}+\lambda_{1,n}} P(l+1,1,n-1)$$

$$+ \frac{\mu_{1,n}}{\mu_{l+1,n}+\lambda_{1,n}} \left\{ \sum_{i=2}^{l} \left\{ \frac{\lambda_{R2,n-1}}{\mu_{1,n}} \prod_{k=i}^{l} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,R2,n-1) \right.\right.$$

$$\left. + \frac{q\lambda_{1,n-1}}{\mu_{1,n}} \prod_{k=i}^{l} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,1,n-1) \right\}$$

$$\left. + \prod_{k=2}^{l} \frac{\delta p \mu_{k-1,n}}{\mu_{k,n}+\lambda_{1,n}} P(1,1,n) \right\}$$

$$= \sum_{i=2}^{l+1} \left\{ \frac{\lambda_{R2,n-1}}{\mu_{l+1,n}} \prod_{k=i}^{l+1} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,R2,n-1) \right.$$

$$\left. + \frac{q\lambda_{1,n-1}}{\mu_{l+1,n}} \prod_{k=i}^{l+1} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{1,n}} P(i,1,n-1) \right\}$$

$$+ \prod_{k=2}^{l+1} \frac{\delta p \mu_{k-1,n}}{\mu_{k,n}+\lambda_{1,n}} P(1,1,n)$$

Q.E.D.

If we let β_1 equal the summation, and α_1 be the coefficient of $P(1,1,n)$, we have $P(1,1,n)=\beta_1+\alpha_1 P(1,1,n)$.

Now from the balance equation for $P(1,1,n-1)$, we obtain

$$(\mu_{1,n-1}+\lambda_{1,n-1})P(1,1,n-1) = p\mu_{1,n}P(1,1,n) + \mu_{R1,n}P(R1,1,n)$$

$$+ \lambda_{R2,n-2}P(1,R2,n-2) + q\lambda_{1,n-2}P(1,1,n-2)$$

$$= p\mu_{1,n}P(1,1,n) + \mu_{R1,n}(\beta_{R1}+\alpha_{R1}P(1,1,n))$$

$$+ \lambda_{R2,n-2}P(1,R2,n-2) + q\lambda_{1,n-2}P(1,1,n-2)$$

$$= (p\mu_{1,n}+\mu_{R1,n}\alpha_{R1})P(1,1,n) + \mu_{R1,n}\beta_{R1}$$

$$+ \lambda_{R2,n-2}P(1,R2,n-2) + q\lambda_{1,n-2}P(1,1,n-2)$$

or

$$(*) \quad P(1,1,n) = \{1/(p\mu_{1,n}+\mu_{R1,n}\alpha_{R1})\} \{(\mu_{1,n-1}+\lambda_{1,n-1})P(1,1,n-1)$$

$$- \mu_{R1,n}\beta_{R1} - \lambda_{R2,n-2}P(1,R2,n-2) - q\lambda_{1,n-2}P(1,1,n-2)\}$$

Notice that all the terms on the right hand side contain probabilities of degree n-1 or n-2. Hence, we can solve for $P(1,1,n)$ knowing only these earlier probabilities.

To solve for the $P(i,1,n)$'s, we can use the balance equations directly from the figure.

$$(*) \quad P(i,1,n) = \{1/(\mu_{i,n}+\lambda_{1,n})\} \{\delta p \mu_{i-1,n}P(i-1,1,n)$$

$$+ \lambda_{R2,n-1}P(i,R2,n-1) + q\lambda_{1,n-1}P(i,1,n-1)\} ,$$

for $2 \leq i \leq R1$, $1 \leq n \leq N$.

Conjecture 2:

$$P(1,j,n) = \sum_{i=2}^{1} \frac{\delta q \lambda_{j-1,n}}{\mu_{1,n}} \prod_{k=i}^{1} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{j,n}} P(i,j-1,n)$$

$$+ \prod_{k=2}^{1} \frac{\delta p \mu_{k-1,n}}{\mu_{k,n}+\lambda_{j,n}} P(1,j,n)$$

for $2 \leq 1 \leq R1$, $2 \leq j \leq R2$, $1 \leq n \leq N-1$.

Pf: By induction on l; j and n are fixed, but arbitrary.

For l=2, we obtain

$$P(2,j,n) = \frac{\delta q \lambda_{j-1,n}}{\mu_{2,n}} \cdot \frac{\mu_{2,n}}{\mu_{2,n}+\lambda_{j,n}} P(2,j-1,n)$$

$$+ \frac{(1-p)\mu_{1,n}}{\mu_{2,n}+\lambda_{j,n}} P(1,j,n)$$

$$= \{1/(\mu_{2,n}+\lambda_{j,n})\}\{\delta q \lambda_{j-1,n} P(2,j-1,n) + (1-p)\mu_{1,n} P(1,j,n)\}$$

which is true by inspection of the figure and the balance equations.

Now suppose the conjecture is true for l. Show it is true for l+1:

$$P(l+1,j,n) = \{1/(\mu_{l+1,n}+\lambda_{j,n})\}\{\delta q \lambda_{j-1,n} P(l+1,j-1,n)$$

$$+ \mu_{1,n} P(l,j,n)\}$$

$$= \frac{\delta q \lambda_{j-1,n}}{\mu_{l+1,n}+\lambda_{j,n}} P(l+1,j-1,n) +$$

$$\frac{\mu_{1,n}}{\mu_{l+1,n}+\lambda_{j,n}} \left\{ \sum_{i=2}^{l} \frac{\delta q \lambda_{j-1,n}}{\mu_{1,n}} \prod_{k=i}^{l} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{j,n}} P(i,j-1,n) \right.$$

$$\left. + \prod_{k=2}^{l} \frac{\delta p \mu_{k-1,n}}{\mu_{k,n}+\lambda_{j,n}} P(1,j,n) \right\}$$

$$= \frac{\delta q\lambda_{j-1,n}}{\mu_{l+1,n}+\lambda_{j,n}} P(l+1,j-1,n) + \sum_{i=2}^{l} \frac{\delta q\lambda_{j-1,n}}{\mu_{l+1,n}} \prod_{k=i}^{l+1} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{j,n}} P(i,j-1,n)$$

$$+ \prod_{k=2}^{l+1} \frac{\delta p\mu_{k-1,n}}{\mu_{k,n}+\lambda_{j,n}} P(1,j,n)$$

$$= \sum_{i=2}^{l+1} \frac{\delta q\lambda_{j-1,n}}{\mu_{l+1,n}} \prod_{k=i}^{l+1} \frac{\mu_{k,n}}{\mu_{k,n}+\lambda_{j,n}} P(i,j-1,n) + \prod_{k=2}^{l+1} \frac{\delta p\mu_{k-1,n}}{\mu_{k,n}+\lambda_{j,n}} P(1,j,n)$$

Q.E.D.

If we let β_1 be the above summation, and α_1 be the coefficient of $P(1,j,n)$, we can write $P(l,j,n) = \beta_1 + \alpha_1 P(1,j,n)$, and then

$$(\mu_{1,n-1}+\lambda_{j,n-1})\ P(1,j,n-1) = p\mu_{1,n}P(1,j,n) + \delta q\lambda_{j-1,n-1}P(1,j-1,n-1)$$

$$+ \mu_{R1,n}P(R1,j,n)$$

$$(\mu_{1,n-1}+\lambda_{j,n-1})\ P(1,j,n-1) = p\mu_{1,n}P(1,j,n) + \delta q\lambda_{j-1,n-1}P(1,j-1,n-1)$$

$$+ \mu_{R1,n}(\beta_{R1} + \alpha_{R1}P(1,j,n))$$

$$(*)\quad P(1,j,n) = \{1/(p\mu_{1,n}+\mu_{R1,n}\alpha_{R1})\}\{(\mu_{1,n-1}+\lambda_{j,n-1})\ P(1,j,n-1)$$

$$- \delta q\lambda_{j-1,n-1}P(1,j-1,n-1) - \mu_{R1,n}\beta_{R1}\ \}$$

Notice again that we can now solve for the $P(1,j,n)$'s using only those probabilities previously determined.

Once we have all the above probabilities, it is easy to solve for the remaining ones directly from the balance equations.

$$(*)\qquad P(i,j,n) = \{1/(\mu_{i,n}+\lambda_{j,n})\}\ \{\delta p\mu_{i-1,n}P(i-1,j,n)$$

$$+ \delta q\lambda_{j-1,n}P(i,j-1,n)\ \}$$

$$\text{for } 2 \leq i \leq R1,\ 2 \leq j \leq R2,\ 1 \leq n \leq N-1$$

We need one more equation to cover the special case where $n=N$. It is derived directly from the figure, and is given by

$$(*)\quad P(1,1,N) = \frac{q\lambda_{1,n-1}\ P(1,1,N-1) + \lambda_{R2,n-1}\ P(1,R2,N-1)}{\mu_{1,n}}$$

References

[BASK75] Baskett, F., Chandy, K.M., Muntz, R.R, and Palacios, F. "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers." J. ACM 22, 2 (April 1975), 248-260.

[BRUE78] Bruell, S.C. "On Single and Multiple Job Class Queueing Network Models of Computer Systems." Ph.D Thesis, Computer Sciences Department, Purdue University, 1978.

[BUZE73] Buzen, J.P. Computational Algorithms for Closed Queueing Networks with Exponential Servers. Comm. ACM 16, 9 (Sept. 1973), 527-531.

[CHAN75] Chandy, K.M., Herzog, U., and Woo, L. Approximate Analysis of General Queuing Networks. IBM J. R&D 19, 1 (Jan. 1975), 43-49.

[KLEI75] Kleinrock, L. Queueing Systems, Vol. I. Wiley, New York, 1975.

[SAUE75a] Sauer, C.H., and Chandy, K.M. Approximate Analysis of Central Server Models. IBM J. R&D 19, 3 (May 1975), 301-313.

[SAUE75b] Sauer, C.H. Configurations of Computing Systems: An Approach Using Queuing Network Models. Ph.D. Diss., U. of Texas at Austin, Austin, Texas, 1975.

Figure 1. Non-exponential queue.

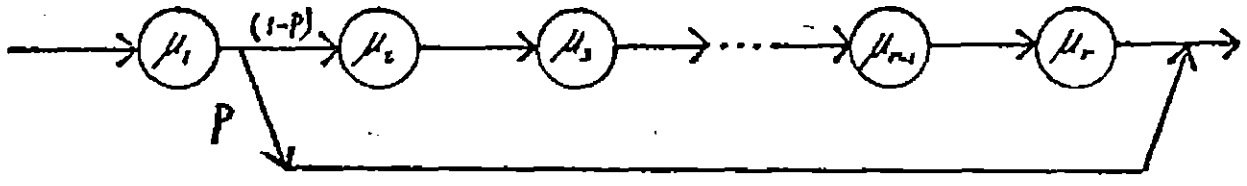


Figure 2. Modified Erlang Model.

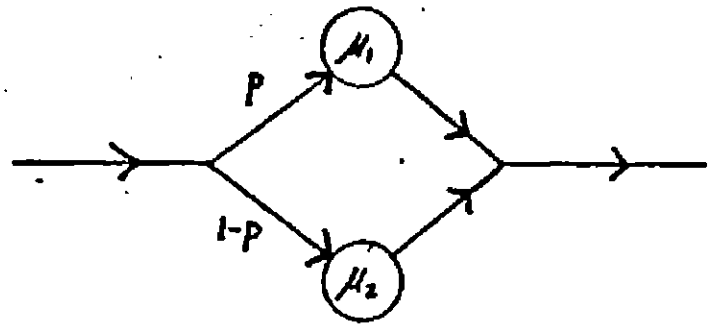


Figure 3. Hyperexponential Model.

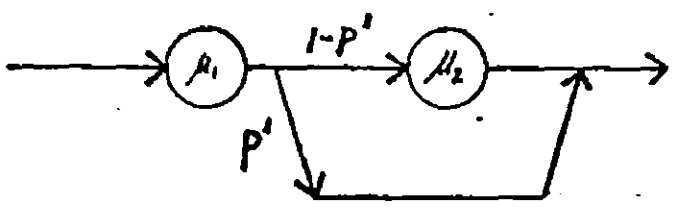


Figure 4. Modified Erlang form of Hyperexponential Model.

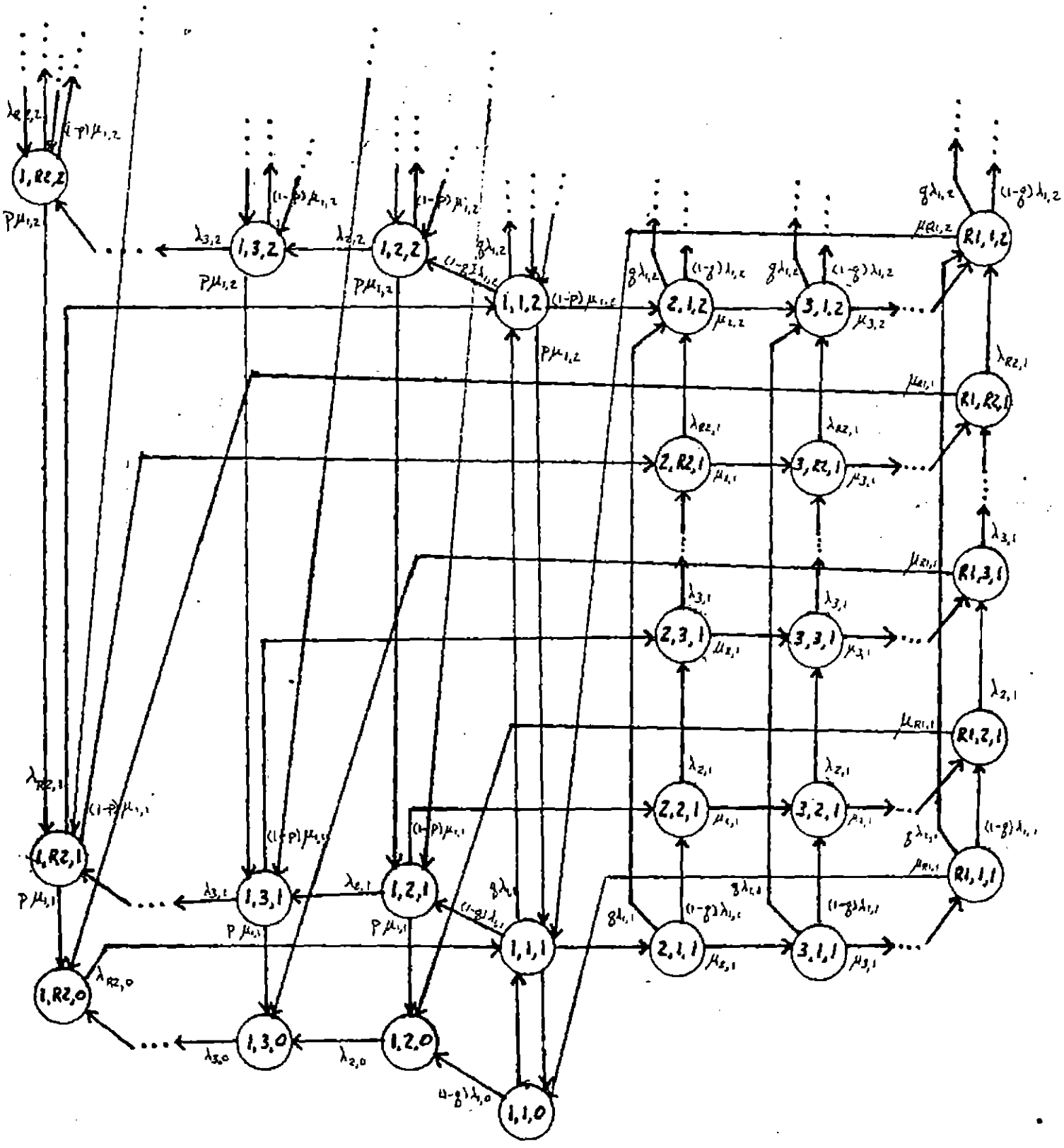


Figure 5. Lower portion of statespace for two queues with exponential stages.

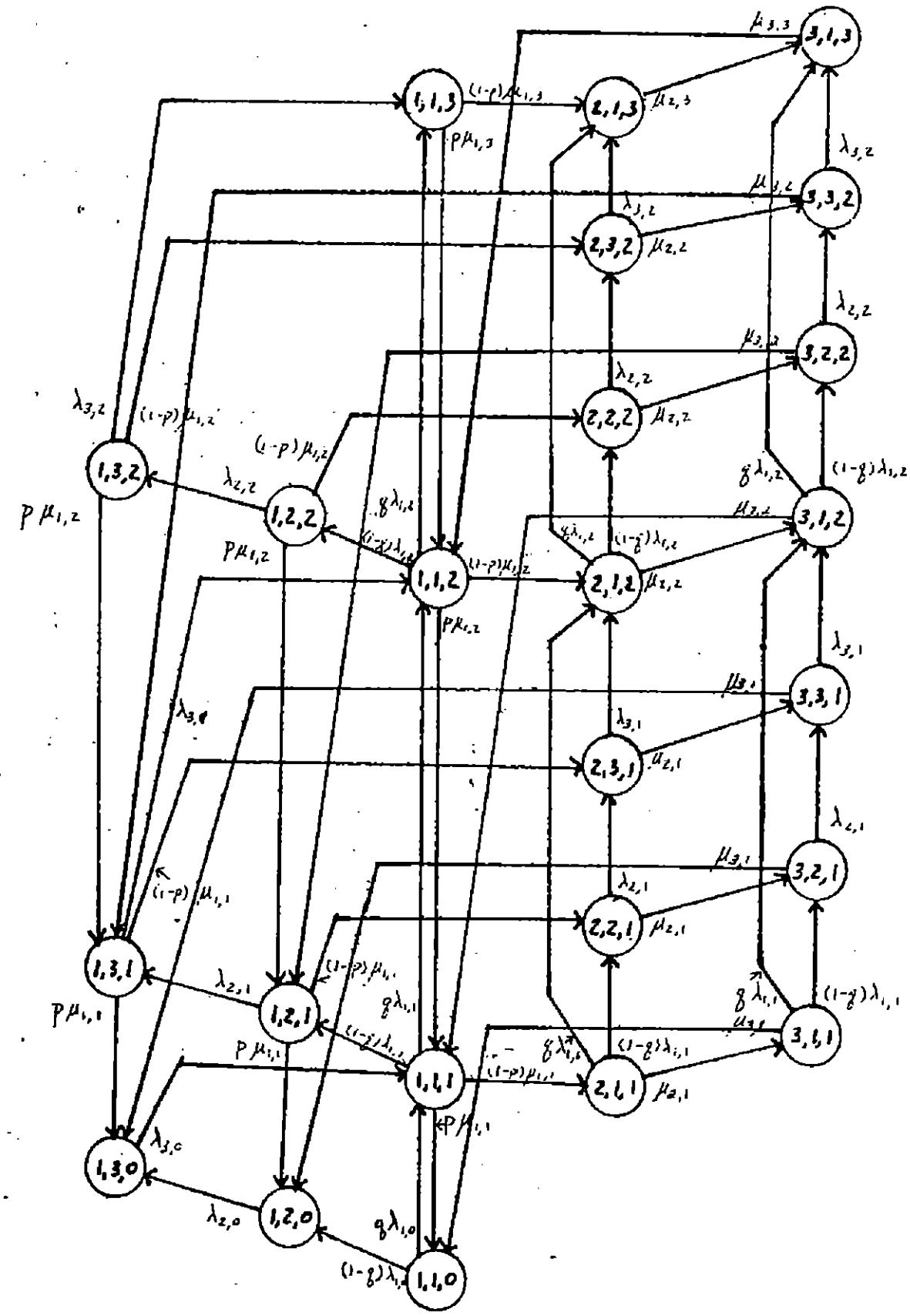


Figure 6. Complete statespace for two queue model with R1=3, R2=3, N=3.