STATE UNIVERSITY OF NEW YORK AT

STONY BROOK

CEAS TECHNICAL REPORT 607

Fast Decomposable Solution for a Class of Non-Product Form Queueing Networks

R.-X. Ni and T.G. Robertazzi

August 7, 1991

Fast Decomposable Solution for a Class of Non-Product Form Queueing Networks

RONG-XIANG NI and THOMAS G. ROBERTAZZI

Department of Electrical Engineering

SUNY at Stony Brook

Stony Brook, NY 11794

ABSTRACT

This paper considers two different solution methods for a non-product form solution decomposable queueing network. The first method is to solve for all the equilibrium state probabilities simultaneously. The second method is to partition the state transition diagram into several subsets and solve each subset of states for the equilibrium state probabilities one at a time. In this paper, we compare these two methods in terms of the actual CPU time.

1 Introduction

For an open network of N queues with Poisson arrival processes and exponential service time distributions, Jackson showed that the equilibrium state probability of the system can be written as a product of the equilibrium state probabilities of each queue:

$$P(n_1, n_2, \cdots, n_N) = P(n_1)\, P(n_2) \cdots P(n_N) \qquad (1\text{-}1)$$

Equation (1-1) is referred to as the product form solution. The introduction of the product form solution makes the queueing model analysis easier. However, not every queueing network has the product form solution.

In this paper, we will discuss such a queueing network, which has no product form solution but can be decomposed to produce an efficient mean of solution. This model is shown in figure 1. The system contains two queues, the primary queue Q_1 and the secondary queue Q_2. Whenever a customer arrives at this system, it always tries to go to the primary queue first, which can hold up to K_1 customers. Q_1 has an exponential service rate μ. In the case that the primary queue is full, the customer, then, is sent to the secondary queue ("overflow pool"), which can hold up to K_2 customers. Each customer in the secondary queue will try to reenter the primary queue after an exponential delay of γ in the secondary queue. The state transition diagram is shown in figure 2.

In section 2, we investigate two solution methods to solve the equilibrium state probabilities of the system. In section 3, we compare these two different

solutions in terms of the actual CPU time needed to solve different size problems.

2 Two solution methods

From figure 2, it is clear that this protocol has no product form solution since its state transition diagram does not possess the usual complete building block structure [LAZA & ROBE 1984]. For this protocol, we have two different ways to solve the equilibrium state probabilities of the system. The first solution method is to solve $(K_1+1)\times(K_2+1)$ linear equations simultaneously. There are $(K_1+1)\times(K_2+1)$ states in figure 2. Each state has a global balance equation but one of them is redundant. This is replaced by the equation of probability normalization to solve for $(K_1+1)\times(K_2+1)$ equilibrium state probabilities of the system. These equations are shown as follows:

For the lower left corner state, we have:

$$-\lambda P(0,0) + \mu P(1,0) = 0$$

From the left edge, we obtain:

$$-(\lambda + j\gamma)P(0,j) + \mu P(1,j) = 0, \quad 1 \le j \le K_2$$

From the bottom edge, we obtain:

$$-(\lambda+\mu)P(i,0) + \lambda P(i-1,0) + \gamma P(i-1,1) + \mu P(i+1,0) = 0$$
$$1 \le i \le K_1-1$$

For the states in the middle of the state transition diagram, we have:

$$-(\lambda+\mu+j\gamma)P(i,j) + \lambda P(i-1,j) + (j+1)\gamma P(i-1,j+1)$$
$$+ \mu P(i+1,j) = 0, \quad 1 \le i \le K_1-1, \quad 1 \le j \le K_2-1$$

From the upper boundary, we obtain:

$$-(\lambda+\mu+K_2\gamma)P(i,K_2) + \lambda P(i-1,K_2) + \mu P(i+1,K_2) = 0$$
$$1 \leq i \leq K_1-1$$

From the right boundary, we obtain:

$$-(\lambda+\mu)P(K_1,j) + \lambda P(K_1-1,j) + (j+1)\gamma P(K_1-1,j+1) = 0$$
$$1 \leq j \leq K_2-1$$

From the probability normalization, we obtain:

$$\sum_{i=0}^{K_1}\sum_{j=0}^{K_2} P(i,j) = 1$$

The second solution method is based on a decomposition of the state transition diagram which is possible for a class of non-product form queueing diagrams possessing certain common structure [ROBE 1989]. One of two possible common structures is referred to as type A structure and is illustrated in figure 3. Here each circular subset represents a state or a group of states. For the i th subset the rule is that there must be only one state external to the subset, with *unknown* probability, from which a transition(s) entering the subset originates. The subsets are solved sequentially, starting from the first subset to the second and so on. There is no restriction on the number of transitions which may leave the i th subset for destinations in the $j = i+1, i+2, \cdots$ subsets.

This decomposition allows one to solve for the unnormalized equilibrium state probabilities of a sub-set of states one at a time. In terms of the model in question, in the context of this technique, one first solves the equilibrium state

probabilities of the same row and the right most state in the next row, at a time, from top to bottom. At the end, one sums up all the unnormalized state probabilities, and uses this sum as the normalization factor. In this method, we select state $P(K_1,K_2)$ as the reference state, that is, we initially set $P(K_1,K_2)=1$.

The first set of the linear equations is as follows:

From the left most state, we have:

$$-(\lambda+K_2\gamma)P(0,K_2)+\mu P(1,K_2)=0$$

From the middle states, we have:

$$-(\lambda+K_2\gamma+\mu)P(i,K_2)+\lambda P(i-1,K_2)+\mu P(i+1,K_2)=0$$
$$1\leq i\leq K_1-1$$

From the right most state, we have:

$$-\mu P(K_1,K_2)+\lambda P(K_1-1,K_2)+\lambda P(K_1,K_2-1)=0$$

From the reference state, we have:

$$P(K_1,K_2)=1$$

The middle sets of the linear equations are similar to the first set. They are as follows:

From the left most state, we have:

$$-(\lambda+j\gamma)P(0,j)+\mu P(1,j)=0,$$

From the middle states, we have:

$$-(\lambda+\mu+j\gamma)p(i,j)+\lambda P(i-1,j)+(j+1)\gamma P(i-1,j+1)$$

$$+ \mu P(i+1,j) = 0, \quad 1 \leq i \leq K_1-1$$

From the right most state, we have:

$$-(\lambda+\mu)P(K_1,j) + \lambda P(K_1-1,j) + (j+1)\gamma P(i-1,j+1) + \lambda P(K_1,j-1) = 0$$

The right most state probability has been solved in the *previous* equation set. Let it be t_j, then we have:

$$P(K_1,j) = t_j$$

In all of these, j is initially set to be K_2-1. Each time when a middle set is solved j decreases by 1. At the time j reaches zero, the last set of the linear equations will be solved. The last set of the linear equations is somewhat different from the previous equation sets since this is the last row and there is no next row. Therefore only K_1+1 linear equations are needed. They are as the follows:

From the left most state, we have:

$$-\lambda P(0,0) + \mu P(1,0) = 0$$

From the middle states, we have:

$$-(\lambda+\mu)P(i,0) + \lambda P(i-1,0) + \mu P(i+1,0) + \gamma P(i-1,1) = 0 \qquad 1 \leq i \leq K_1-1$$

The right most state has been solved by *previous* equation set. Let it be t_0, then we have:

$$P(K_1,0) = t_0$$

3 Performance results of these two solution methods

For the first method, we need to solve $(K_1+1)\times(K_2+1)$ linear equations. When both K_1 and K_2 are very large, the time needed is proportional to $K_1^3 K_2^3$. For the second method, we need to solve K_1+2 linear equations K_2 times and K_1+1 linear equations once. When both K_1 and K_2 are very large, the time needed is proportional to $K_1^3 K_2$. It is clear that the second method is far more efficient.

We have compared these two methods by using different sizes of K_1 and K_2. The result is shown in table 1. In this table, we call the first method "whole" and call the second method "partial". Comparing the data in the table, when $K_1=K_2=5$, the time needed is 0.9 second for the first method and 0.2 seconds for the second method. The efficiency is only 4.5 times. As K_1 and K_2 increase, say $K_1=K_2=20$, the time needed is 333.5 seconds for the first method and 3.9 seconds for the second method. The efficiency has increased to 85.5 times.

In table 1, we only finish up to 20×20 for the first method since it will be too time consuming for K_1 and K_2 beyond this value. For the second method, we finish up to 100×100. It is interesting that when K_1 and K_2 are very large, i.e. $K_1=K_2=60$, the last set of the linear equations is too singular to solve. Therefore, for the value beyond this, we solve for the equilibrium state probabilities of the last row recursively.

4 Conclusion

In this paper we have introduced two different solution methods for a class of non-product form decomposable queueing network. We have also compared their performances in the terms of the actual CPU time needed. We feel that the first method is simple in terms of procedure while the second method is efficient in practice.

Acknowledgements

We will like to thank Miss S. Pareek for lending us figure 1 & figure 2. The research in this paper was supported in part by the National Science Foundation under Grant no. NCR-8703689 and by the SDIO/IST and managed by the U.S. Office of Naval Research under Grant no. N00014-85-K0610.

References

1. Lazar, A. A. and Robertazzi, T. G., "The Geometry of Lattices for Multi-class Markovian Queueing Network", Proceedings of the 1984 Conference on Information Sciences and Systems, Princeton, March 1984, pp. 164-168.

2. Jackson, J. R., "Networks of Waiting Lines", Operations Research, 5, 1957, pp. 518-521.

3. Robertazzi, T. G., "Recursive Solution of a Class of Non-Product Form Protocol Models", Proceedings of IEEE INFOCOM'89, Ottawa, Canada, April 1989, pp. 38-46.

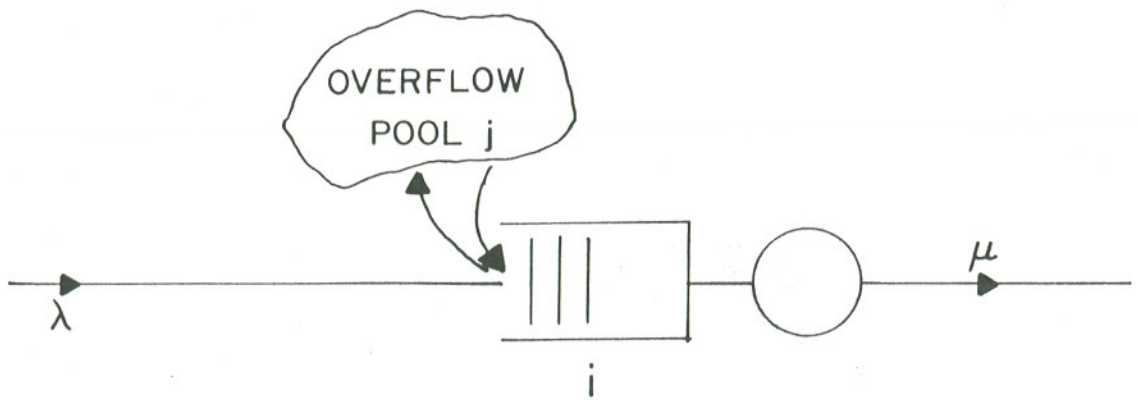

FIGURE 1

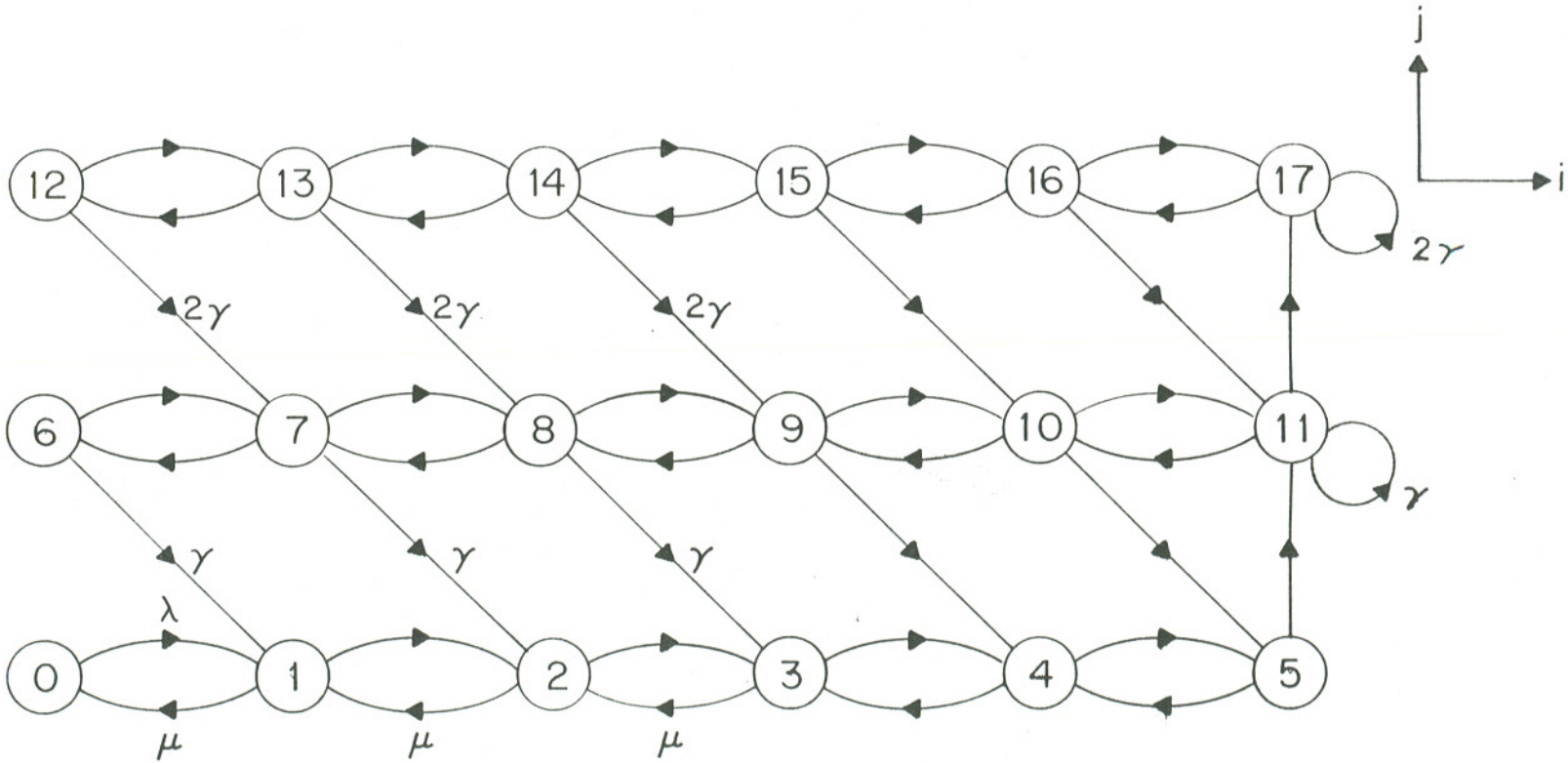

FIGURE 2

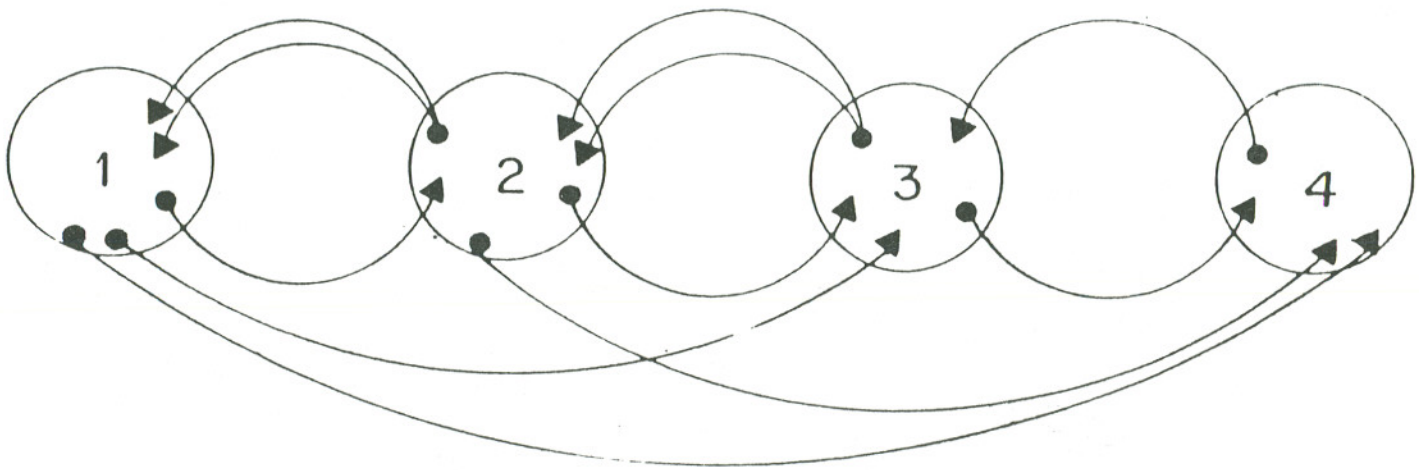

FIGURE 3

$K_1 \times K_2$	Actual CPU time (second)	
	Whole	Partial
5×5	0.9	0.2
10×10	14.1	0.9
15×15	86.7	2.0
20×20	333.5	3.9
30×30		11.0
40×40		23.6
50×50		44.6
60×60		72.6
70×70		111.7
80×80		162.6
90×90		226.9
100×100		292.5

TABLE 1