DECOMPOSITION METHODS AND COMPUTATIONAL ALGORITHMS FOR MULTIPLE-CHAIN CLOSED QUEUEING NETWORKS

by

Adrian E. Conway, B.A.Sc., M.Sc., D.I.C.

A dissertation submitted to the Faculty of Science and Engineering of the University of Ottawa in partial fulfilment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical Engineering.

Ottawa, Canada
September 1985

ISBN 0-315-33353-7

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

The hierarchical analysis of product-form multiple-chain closed queueing networks, by the Simon and Ando decomposition and aggregation technique, is investigated. We introduce a new type of network decomposition, called decomposition by chain, as an alternative to the decomposition by node technique of Courtois. The decomposition by chain technique resolves the analysis of a multiple-chain closed queueing network into that of a hierarchy of single-chain closed queueing network problems. A useful consequence of adopting this approach is that a reduced system, itself having a product-form distribution, is constructed around one particular closed routing chain in the multiple-chain network. This then facilitates the parametric analysis of a particular closed chain of interest, with respect to the routing parameters, and forms a complementary result to the so-called Norton's Theorem for queues.

The decomposition by chain technique leads to a new recursive expression for the normalization constant which is associated with the probability distribution of multiple-chain closed queueing networks. The recursion relates the normalization constant of an r-chain network to those of a set of networks having (r-1) chains. The time and space requirements to compute the normalization constant, using this recursion, are polynomial in the number of routing chains. When there are many chains in the network, this method is substantially more efficient than the well-known convolution algorithm, which has exponential time and space requirements. An hybrid type algorithm is introduced that attempts to take advantage of the best features of both methods.

The new recursion, for computing the normalization constant, forms the basis of a new efficient algorithm, named RECAL, for computing the mean performance measures of multiple-chain closed queueing networks. The algorithm relies on the artifice of breaking down each chain into constituent sub-chains that each have a population of one. The complexity of this algorithm is shown to be polynomial in the number of chains. This algorithm extends the range of queueing networks which can be analyzed efficiently by exact means. A simple dynamic scaling procedure is developed to alleviate the effects of the phenomenon of numerical instability.

An efficient algorithm, for the solution of a class of so-called semi-homogeneous queueing networks, is developed from RECAL by exploiting the assumptions of semi-homogeneity, the class of semi-homogeneous networks being a generalization of the class of homogeneous networks, which has been considered by Balbo et al for the performance modeling of local area networks. The algorithm has a complexity which is closely related to the number of unordered partitions of a positive integer. It is shown that the order of the complexity is less than exponential in the square-root of the number of closed routing chains. This establishes the effectiveness of the algorithm over the direct application of a general purpose exact solution technique to the class of semi-homogeneous networks.

ACKNOWLEDGEMENTS

I am especially grateful to my research supervisor, Professor N.D. Georganas, for his advice and guidance throughout the course of my studies, without which this research would not have been accomplished.

I am also grateful to Professor C. Woodside (Department of Systems and Computer Engineering, Carleton University, Ottawa) for bringing the subject of homogeneous queueing networks to my attention.

Most of the research reported in this dissertation was done while Professor Georganas was on sabbatical leave at the Advanced Studies Department of Bull-Transac, Paris, France, and the Institut National de Recherche en Informatique et Automatique (Inria), Rocquencourt, France. I am very grateful to N. Naffah, Director of the Advanced Studies Department, for allowing me to accompany my research supervisor and utilize the facilities of Bull-Transac.

The financial assistance provided by a sequence of Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarships and University of Ottawa Graduate Research Scholarships is also gratefully acknowledged.

Finally, I am grateful to Line Robitaille for typing the manuscript.

PREFACE

This dissertation is based mostly on the following papers, all authored by A.E. Conway and N.D. Georganas:

(1) "Decomposition and aggregation by class in closed queueing networks", IEEE Transactions on Software Engineering, to be published.

(2) "RECAL: A new efficient algorithm for the exact analysis of multiple-chain closed queueing networks", Journal of the Association for Computing Machinery, to be published.
(A preliminary version has appeared as INRIA Rapports de Recherche No. 373, Paris, March 1985, and was presented at the ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, Austin, Texas, August, 1985).

(3) "A new method to compute the normalization constant of multiple-chain queueing networks", INFOR: Canadian Journal of Operations Research and Information Processing, to be published.
(A preliminary version appears in Proceedings of the 8th ACM International Computing Symposium, Florence, Italy, March 1985).

(4) "An efficient algorithm for semi-homogeneous queueing network models", Performance Evaluation Review (Special Issue on the Proceedings of the ACM SIGMETRICS/PERFORMANCE '86 Conference), to be published.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 - INTRODUCTION

THE SANDS OF HERE POURED

UPON THE SANDS OF THERE

STONE AFTER STONE TO FORM A BRIDGE

Lawrence Weiner, Nouvelle Biennale de Paris 1985.

1.1 OVERVIEW

Networks of queues arise quite naturally as mathematical models of systems in which there is a set of interconnected resources among which customers circulate. Over the years, the study of queueing networks has been increasing in intensity as a result of an expansion in the domain of applicability. The original motivation was provided by applications as diverse as job-shops [24], coal mining [27] and the evolution of populations [27]. Interest in queueing networks increased considerably with their applicability as models of computer systems [22,34,55] and packet-switching communication networks [33,65]. More recently, they have also emerged as models of distributed computer systems [18] and local area networks of workstations [3]. The use of queueing network models, as performance evaluation tools, is likely to increase in importance as larger and more complex systems are contemplated.

An important factor in the popularity of queueing network models

has been the discovery of the class of Jackson networks. Over the years, the class of networks originally considered by Jackson [23] has been generalized to include a wide variety of modeling features. Today, this generalized class of networks is known simply as the class of product-form networks since the general form of the state-distribution is a product of terms.

Although the state-distribution for product-form networks is known, it is not a simple matter to compute the mean performance measures which are the quantities that are of interest in practice. This is because there is, in general, no known simple closed-form expression for the constant of normalization that is associated with the probability distribution. For open queueing networks, that is networks in which there are exogeneous arrivals and departures from the network, the multidimensional state-space is countably infinite and the normalization constant can be determined in a closed form. However, for closed queueing networks, the customer population is fixed, the state-space is finite and the summation that defines the normalization constant cannot be reduced to a simple form that can be computed readily. For this reason, the mean performance measures cannot be computed by direct appeal to their definitions. There is a genuine need for algorithms that can obtain the mean performance measures, and other statistics that may be of interest, in an efficient manner. Such algorithms are the main subject of this dissertation.

In this dissertation, we begin by investigating the application of the Simon and Ando decomposition and aggregation technique to the

analysis of product-form queueing networks. Decomposition and aggregation is a technique which is applicable to Markov chains in general. With this technique, the problem of determining the equilibrium distribution is broken down into a set of smaller problems the solutions of which are combined to yield the distribution for the larger system. The technique is suited naturally to systems that are made up of a set of loosely coupled subsystems [15]. The application of this technique to queueing networks was introduced by Courtois [13]. He showed that the analysis of a network of queues could be broken down into the analysis of a hierarchy of subnetworks that consist of subsets of the service centers. As Courtois has pointed out [13], the analysis of product-form queueing networks by this technique does not yield any computational advantage over the convolution algorithm of Buzen [10]. Rather, its main use, for product-form networks, is "to essentially unearth the basic relations that exist among levels of aggregation and among aggregate resources in networks of stochastic service systems". These relations provide considerable insight into the hierarchic structure of queueing networks.

As mentioned above, the network decomposition of Courtois is made according to subsets of the service centers. Our first observation is that this is only one particular way to decompose product-form queueing networks. Courtois [14] has given the necessary and sufficient conditions under which the general decomposition and aggregation procedure will yield exact results. There has not, however, been an attempt to decompose product-form networks other than according to

subsets of the service centers. Using some results from the theory of reversible Markov chains, we are able to show that almost arbitrary decompositions of product-form networks will yield exact results. This observation is a starting point in the search for useful decompositions. We are interested in network decompositions that have physical meaning, ones which provide insight and which lead to useful results.

Our first contribution is to introduce a new type of network decomposition, called decomposition by chain, which is applicable to multiple-chain closed queueing networks. Rather than decomposing by service center, as Courtois and others have done [13,14,62,63], we proceed to decompose the network according to subsets of the closed routing chains. The decomposition by chain technique resolves the analysis of a multiple-chain network into a hierarchy of single-chain closed queueing network problems. A useful consequence of adopting this approach is that a reduced system, itself having a product-form distribution, is constructed around one particular closed routing chain. This reduced system facilitates the parametric analysis of a particular closed routing chain of interest with respect to the routing parameters. This result can be considered as a complementary result to 'Norton's Theorem' for queues [11] which is used for the parametric analysis of particular service centers of interest.

Apart from providing insight into the hierarchic structure of queueing networks, the fact that product-form networks can be decomposed almost arbitrarily also shows one how to design

computational algorithms for the numerical solution of product-form queueing networks. This observation provides a general framework within which we may search for efficient solution algorithms. The main contribution of this dissertation is the discovery of a solution algorithm which is substantially more efficient than hitherto adopted methods, when there are many routing chains in the network. Our new algorithm has a complexity which is polynomial in the number of chains. The established algorithms have exponential complexity.

The underlying principle of the new algorithm is the notion of decomposing by chain. The new algorithm, therefore, follows quite naturally from the decomposition by chain technique. The algorithm is a so-called normalization constant approach since, like the convolution algorithm [10,47], it is based on the computation of the normalization constant.

We present the algorithm in two parts. The first is concerned with the new method of computing the normalization constant. This quantity is of interest in itself. The second part is concerned with obtaining the mean performance measures. These are the quantities that are of most interest in practice.

Our final contribution is an efficient algorithm for the solution of a class of product-form queueing networks that arise in the performance modeling of local area networks of workstations. We introduce the class of semi-homogeneous queueing networks as a generalization of the class of homogeneous networks, which has been considered by Balbo et al [2,3] for the modeling of local area

networks. Our new algorithm, for the solution of semi-homogeneous networks, is based directly on the new algorithm mentioned in the previous paragraph. We exploit the inherent structure of semi-homogeneous queueing networks to reduce the complexity of the solution procedure. This algorithm for semi-homogeneous networks is, in general, considerably more efficient than the general purpose algorithms. It is less efficient than the algorithm of Balbo et al [3], but the class of networks it can solve is more general than the class of homogeneous networks. Our algorithm may, therefore, be situated between the algorithm of Balbo et al and the general purpose algorithms, both in terms of generality and efficiency.

In the remainder of this Chapter, we shall overview the theory of product-form queueing networks and some of their current applications. In Chapter 2, we overview the decomposition and aggregation technique and the work of Courtois. Chapter 3 presents a summary of the computational algorithms which have been developed for closed queueing networks. This will serve to situate our new algorithms among the established ones. The new results, that constitute this thesis, are presented in Chapters 4, 5, 6 and 7. We begin in Chapter 4 with the decomposition by chain technique and then proceed in Chapters 5 and 6 to describe the new methods of obtaining the normalization constant and the mean performance measures. The algorithm for the solution of semi-homogeneous networks is presented in Chapter 7. We conclude, in Chapter 8, by suggesting several further avenues of research that may lead to useful results.

1.2 INTRODUCTION TO QUEUEING NETWORKS

A network of queues consists of a set of service centers arranged at the nodes of a graph. Customers travel along the edges between the nodes and receive service at the various service centers. A network is closed if there are no exogeneous arrivals and no departures from the network. A closed network contains a fixed population of customers that circulate continuously among the service centers. A network is open if customers may enter and leave the network. The population of an open network is unconstrained. A network is mixed if for certain customers the network is closed while for others it is open.

The description of a queueing network includes a specification of the mechanism whereby customers arrive at and depart from the network, of the routing of the customers between the nodes, of the service requirements and of the queueing disciplines at the service centers. In the construction of a queueing network model for a system, it is customary to make certain assumptions so that the model will be mathematically tractable. The types of systems usually modeled using queueing networks contain features that are random in nature. The analysis of these queueing networks is a problem in the field of applied probability. The assumptions that are usually made are ones which will render the queueing network Markovian.

Although the Markovian assumptions simplify considerably the mathematical structure of a queueing network, the state description of queueing networks is multidimensional in nature and the cardinality of the state-space is usually extremely large. This, in general,

precludes the analysis of Markovian queueing networks by matrix methods [59] that involve solving the system

$$\begin{cases} \underline{v}\, Q = \underline{0} \\ \underline{v}\, \underline{1}^T = 1 \end{cases} \qquad (1.1)$$

for the equilibrium state probability distribution $\underline{v}$, where Q is the infinitesimal generator for the continuous-time Markov chain associated with the queueing network. Fortunately, a very general class of Markovian queueing networks has been discovered which has a known equilibrium distribution. This is the class of Jackson networks [23], or so-called product-form networks. For this class of networks, the problem is no longer to solve eq. 1.1 but rather to compute the performance measures of interest, knowing the state-distribution. This is the problem with which we are primarily concerned.

In the following Section we shall overview the basic model of Jackson and the main extensions that have been made to it.

1.3 THE THEORY OF PRODUCT-FORM QUEUEING NETWORKS

Product-form queueing networks have their beginnings with the network of Jackson [23]. Subsequent generalizations remain closely related to it. We shall first consider the classical work of Jackson and then overview the general class of product-form networks, as it stands today. A recent survey of the theory of product-form queueing networks in equilibrium appears in [37].

The queueing network of Jackson is a generalization of the classical M/M/K/∞ queue. There are N nodes with service centers of the $\cdot$/M/K/∞ type. The network is open and customers from outside the system arrive to the centers according to a Poisson process. The rate of external arrivals to node i, $1 \leq i \leq N$, is λ_i. The service time distribution for the customers is assumed exponential. The mean service time for a customer at node i is $1/\mu_i$. The number of parallel servers at node i is k_i. A customer who completes service at node i proceeds next to node j with probability p_{ij}. The state of the system is $\underline{n} = (n_1,\ldots,n_N)$, where n_i is the number of customers at node i. The Jackson network is a continuous-time Markov chain over the state-space $S = \{(n_1,\ldots,n_N): \; n_i \geq 0; \; 1 \leq i \leq N\}$. Jackson proved that, when the equilibrium distribution exists, it is given by

$$\Pr(\underline{n}) = \prod_{i=1}^{N} \psi_i (n_i) \tag{1.2}$$

where.

$$\psi_i(n_i) = \begin{cases} C_i \, (k_i \rho_i)^{n_i} / n_i!, & \text{if } 0 \leq n_i \leq k_i, \\ C_i \, k_i^{k_i} \, \rho_i^{n_i} / k_i!, & \text{if } n_i > k_i + 1, \end{cases}$$

C_i is a normalization constant chosen so that $\sum_{x=0}^{\infty} \psi_i(x) = 1$, $\rho_i = e_i / k_i \mu_i$ and the quantities e_i, $1 \leq i \leq N$, are given by the solution to the conservation of flow equations

$$e_i = \lambda_i + \sum_{j=1}^{N} p_{ji} e_j, \quad 1 \leq i \leq N.$$

The necessary condition for the equilibrium distribution to exist is that $\rho_i < 1$ for $1 \leq i \leq N$. As can be seen from the product-form nature of eq. 1.2, the queues behave independently as if they were isolated $M/M/k_i/\infty$ queues with input rates e_i.

Jackson later generalized his results to include an external arrival rate that depends on the population of the network and, as well, servers whose service rates are functions of the node populations [24]. These results include the closed network as a special case, which is usually (incorrectly [20]) attributed to Gordon and Newell [19]. In [43], the network considered by Gordon and Newell is generalized to include travel times between the nodes. The network of [43] is further generalized in [44] to include the case where we have different types of customers in the network which can be distinguished by their service time requirements, routing probabilities and travel times.

The class of product-form queueing networks was extended greatly, and presented in a unified fashion, by Baskett, Chandy, Muntz and Palacios [5]. This class is the one most widely referred to in practice, and is known as the class of BCMP networks. We shall overview this class of networks in some detail since it is the one with which we are concerned here, and this will serve to establish the notation which is to be adopted.

In the class of BCMP networks there are different types of customers. Customers of the same type are said to belong to the same

closed routing chain. There are, as well, different classes of customers. A customer may change its class when it completes service at a node, but it may never change its type. Let the number of types of customers be R and let the set of classes to which a type r customer may belong be C(r). The sets of classes C(1),...,C(R) are all disjoint. Let the number of nodes in the network be N and let the set of classes to which a type r customer may belong at node i be $C_i(r)$. Clearly $\bigcup_{i=1}^{N} C_i(r) = C(r)$. Also let $C_i = \bigcup_{r=1}^{R} C_i(r)$. A type r customer who completes service at node i in class c next proceeds to node j in class d with probability $p^{(r)}_{i,c;j,d}$ or leaves the network with probability

$$\left(1 - \sum_{j=1}^{N} \sum_{d \in C_j(r)} p^{(r)}_{i,c;j,d}\right).$$

It is assumed that type r customers arrive at node i in class c, $c \in C_i(r)$, according to a Poisson process at rate $\lambda^{(r)}_{ic}$. The transition probability matrix $P^{(r)} = \left[p^{(r)}_{i,c;j,d} : 1 \le i, j \le N;\ c \in C_i(r);\ d \in C_j(r)\right]$ is said to be the routing chain for type r customers. If $\lambda^{(r)}_{ic} = 0$ for all $1 \le i \le N$ and $c \in C_i(r)$, then the routing chain r is said to be closed, otherwise it is open. If routing chain r is closed, then the number of type r customers in the network is fixed. We denote this number by K_r. The mean service time requirement for a class c customer at node i is m_{ic}.

In a BCMP network, four types of queueing disciplines are allowed at the service centers, namely, first-come first-served (FCFS), processor sharing (PS), last-come first-served preemptive resume

(LCFSPR) (with service resumed at the point of interruption) and infinite server (IS). At an IS center there is no queue since there are an unlimited number of servers. If node i is a FCFS center, then it is required that the service time distribution be exponential and $m_{ic} = m_i$ for all $c \in C_i$. At LCFSPR, PS or IS service centers, the service-time distributions may take on a general form to be described below.

The required form for the service-time distribution at LCFSPR, PS or IS centers is that it have a rational Laplace transform. Such distributions can be constructed by having a network of exponential stages which a customer, who in the process of being served, passes through. Cox [16] has shown that any such distribution can be constructed with a network of exponential stages as illustrated in Figure 1.1. In the network of Figure 1.1, for example, upon completion of stage 1, the customer proceeds for further service in stage 2 with probability a_1 or completes service with probability b_1. Equivalently, we may view the customer as passing through a sequence of classes, one class corresponding to each stage of service [35]. These classes are supplementary to the one the customer was in upon arrival at the node.

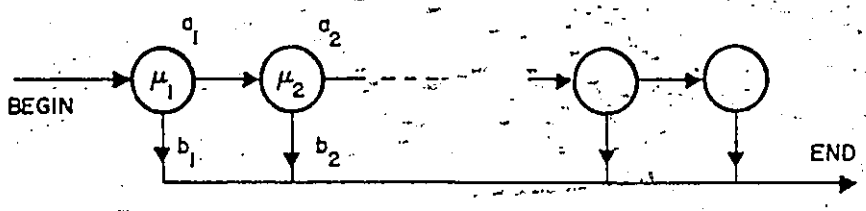

Figure 1.1 Coxian representation of a general service-time distribution by a sequence of exponential stages.

The state description of the network includes the number of each class of customers at the nodes, the stage of service they have reached, and the ordering in the queue. Baskett et al [5] give the state-distribution at this level of detail. They also give the marginal distribution corresponding to the number of each class of customers at each node. For open networks, they give the marginal distribution for the total number of customers at each node. Here we shall only consider the marginal distribution for the number of customers of each type at the nodes.

Let n_{ir} be the number of type r customers at node i, $n_i^{(R)} = \sum_{r=1}^{R} n_{ir}$, $\underline{n}_i^{(R)} = (n_{i1},\ldots,n_{iR})$ and $\underline{n}^{(R)} = (\underline{n}_1^{(R)},\ldots,\underline{n}_N^{(R)})$. The marginal state-distribution, if it exists, for the number of customers of each type at the nodes is

$$\Pr(\underline{n}^{(R)}) = G^{-1} \prod_{i=1}^{N} f_i(\underline{n}_i^{(R)}), \quad \underline{n}^{(R)} \in S^{(R)}, \tag{1.3}$$

where

$$f_i(\underline{n}_i^{(R)}) = \begin{cases} n_i^{(R)}! \prod_{r=1}^{R} w_{ir}^{n_{ir}} / n_{ir}!, & \text{if center i is FCFS,} \\ & \text{LCFSPR or PS,} \\ \prod_{r=1}^{R} w_{ir}^{n_{ir}} / n_{ir}! & \text{, if center i is IS,} \end{cases}$$

$$w_{ir} = t_{ir} e_{ir},$$

$$e_{ir} = \sum_{c \in C_i(r)} \alpha_{ic}^{(r)},$$

$$t_{ir} = \sum_{c \in C_i(r)} \alpha_{ic}^{(r)} m_{ic} / e_{ir};$$

the quantities $\alpha_{ic}^{(r)}$, $1 \le i \le N$, $c \in C(r)$, satisfy the set of equations

$$\alpha_{ic}^{(r)} = \lambda_{ic}^{(r)} + \sum_{j=1}^{N} \sum_{d \in C_j(r)} \alpha_{jd}^{(r)} p_{j,d;i,c}^{(r)}, \quad 1 \le i \le N, \ c \in C(r),$$

$$S^{(R)} = \{\underline{n}^{(R)}: \ n_{ir} > 0; \ \sum_{i=1}^{N} n_{ir} = K_r \text{ if routing chain } r \text{ is closed};$$

$$1 \le i \le N; \quad 1 \le r \le R\},$$

and where

$$G = \sum_{\underline{n}^{(R)} \in S^{(R)}} \prod_{i=1}^{N} f_i(\underline{n}_i^{(R)}).$$

If chain r is open, then the quantity $\alpha_{ic}^{(r)}$ is the throughput of class c customers, $c \in C_i(r)$, at node i. On the other hand, if chain r is closed, then $\alpha_{ic}^{(r)}$ is unique only up to a multiplicative constant. It is then interpreted as a 'relative' arrival rate, or 'visit ratio'.

The distribution given by eq. 1.3 exists if the network is closed or if it is mixed or open and $G < \infty$. We note that at FCFS centers we have $t_{ir} = t_i$ for $1 \le r \le R$. From eq. 1.3 we see that the marginal state-distribution depends on the service-time distribution, at LCFSPR, PS or IS centers, only through the mean. This is known as the

insensitivity property.

BCMP queueing networks may also accommodate various forms of state-dependency. The rate of external arrivals may be made to depend on the population of the network. The rate at which work is accomplished by a server may be a function of the state of the node, or of the number of customers at several nodes.

Let $\beta_i(a)$ be the rate at which work is accomplished at node i when the total population of the node is a. This type of state-dependency is applicable to FCFS, LCFSPR, and PS centers. With this state-dependency, the marginal distribution is obtained by replacing $f_i(\underline{n}_i^{(R)})$, in eq. 1.3, with

$$f_i(\underline{n}_i^{(R)}) \Big/ \prod_{a=1}^{n_i^{(R)}} \beta_i(a).$$

State-dependency can be used to construct useful modeling features. For example, if node i is FCFS and

$$\beta_i(a) = \begin{cases} a, & \text{if } 1 \leq a \leq k_i, \\ k_i, & \text{if } a > k_i+1, \end{cases}$$

then we have a FCFS queue with k_i parallel servers. We shall not consider explicitly the other forms of state-dependency which have been considered by Baskett et al [5].

Further generalizations to the class of BCMP networks have been made. Kelly [25,26,27] has developed a generalized class of queueing disciplines that give rise to a product-form distribution. Networks with population-dependent lost and triggered arrivals have been treated

by Lam [30]. State-dependent routing is considered by Towsley [60]. Pittel [42] has shown that networks with certain forms of blocking may support a product-form distribution under certain reversibility conditions (to be considered in Section 1.4).

1.4 REVERSIBLE QUEUEING NETWORKS

Under certain conditions, a Markov chain is said to be reversible. A reversible Markov chain has a very simple structure which can be viewed as a multidimensional extension of the classical birth-death process. Reversibility is a sufficient condition for a product-form distribution. In the following, we present certain elementary results for reversible Markov chains which will be referred to in subsequent chapters. Our only use of the notion of reversibility will be as a vehicle to simplify our arguments, specifically those of Sections 4.2 and 5.2.

Consider a continuous-time Markov chain with infinitesimal generator $Q = [q_{ij}: i,j \in S]$, state-space S and equilibrium distribution $\underline{v} = [v_i: i \in S]$.

Theorem 1.1: (Adapted from Theorem 1.3 in [27]) A continuous-time Markov chain in equilibrium is reversible if and only if

$$v_i q_{ij} = v_j q_{ji}, \quad i,j \in S.$$

Corollary 1.1: The state-distribution for a reversible Markov chain is given by the product-form

$$v_i = C \frac{q_{aj_1} q_{j_1 j_2} \cdots q_{j_n i}}{q_{j_1 a} q_{j_2 j_1} \cdots q_{i j_n}}, \quad i \in S,$$

where $a \in S$ is an arbitrary reference state, $a, j_1, j_2, \ldots, j_n, i$ is any sequence of states leading from a to i, and C is a normalization constant.

Corollary 1.2: (Adapted from Corollary 1.10 in [27]) If a reversible Markov chain is truncated to a set $A \subset S$, where A is a set of communicating states, then the resulting Markov chain is reversible in equilibrium and has equilibrium distribution $\underline{v}'$ where

$$v'_i = v_i \Big/ \sum_{j \in A} v_j, \quad i \in A.$$

Under certain special conditions, the underlying Markov chain of BCMP queueing networks will be reversible. Consider a closed BCMP queueing network in which all service-time requirements are exponential. By checking that Theorem 1.1 holds, it may be shown that it will be reversible if $R = 1$ and

$$\alpha_{ic}^{(r)} \, p_{i,c;j,d}^{(r)} = \alpha_{jd}^{(r)} \, p_{j,d;i,c}^{(r)} \tag{1.4}$$

for all $1 \le r \le R$, $1 \le i, j \le N$, $c \in C_i(r)$, and $d \in C_j(r)$. If $R > 2$, then there must be no FCFS centers in the network.

1.5 APPLICATIONS OF PRODUCT-FORM QUEUEING NETWORKS

As mentioned in Section 1.1, the applications of product-form queueing network models are numerous. We shall concern ourselves with queueing network models of computer systems, distributed computer systems, packet-switching communication networks and local area networks of workstations. General references on the modeling of computer systems include [1], [22], [34] and [55]. The modeling of a distributed computer system, using BCMP networks, is considered by Goldberg et al [18]. General surveys of the queueing network modeling of communication networks include [33], [64] and [65]. Balbo et al [3] use product-form models for local area networks of workstations. In this Section we will show, by a series of examples, how these systems map quite naturally into queueing network models.

A computer system can be modeled by a closed network of queues known as the central-server model. An example is illustrated in Figure 1.2. There is one customer associated with each 'logged-on' user at the terminals. While the user is thinking or typing, this customer is at the 'terminals' node. When a carriage return is made, a transaction is submitted to the system. The customer then leaves the 'terminals' and proceeds through a sequence of CPU-I/O device cycles. Once the transaction has been completed, the customer returns to the 'terminals'. The queueing disciplines at the CPU and I/O devices may take on any of the forms that support a product-form distribution. The PS discipline is used to approximate round-robin service at the CPU. The 'terminals' node is an IS center (the users can work simultaneously

at their terminals). Using the class-switching feature of BCMP networks, the routing of a customer, among the CPU and I/O devices, may be made to be any finite deterministic sequence of nodes. We need not have random routing as in the original Jackson network. By having different types of customers, we may allow the routing and service requirements to depend on the routing chain to which a customer belongs. We may model 'batch' users by introducing an open routing chain.

A model of a distributed computer system can be created by interconnecting a group of central-server models of the type in Figure 1.2. This is illustrated in Figure 1.3. In this model, we have a number of customers associated with the terminals of each central-server network. There is a closed routing chain associated with each such group of customers. The customers place demands on the resources local to their sites as well as on those at remote sites. The communication channel that links the sites can be modeled by a FCFS queue through which the customers, proceeding from one site to another, must pass.

In a model of a packet-switching communication network, we associate a customer with each packet in the network. Each transmission channel is modeled by a FCFS queue. For window flow-controlled networks, we associate a closed routing chain with each virtual-channel that is set up between the source-destination pairs in the network. The chain population is made equal to the window size. Acknowledgements are taken into account by introducing a delay before

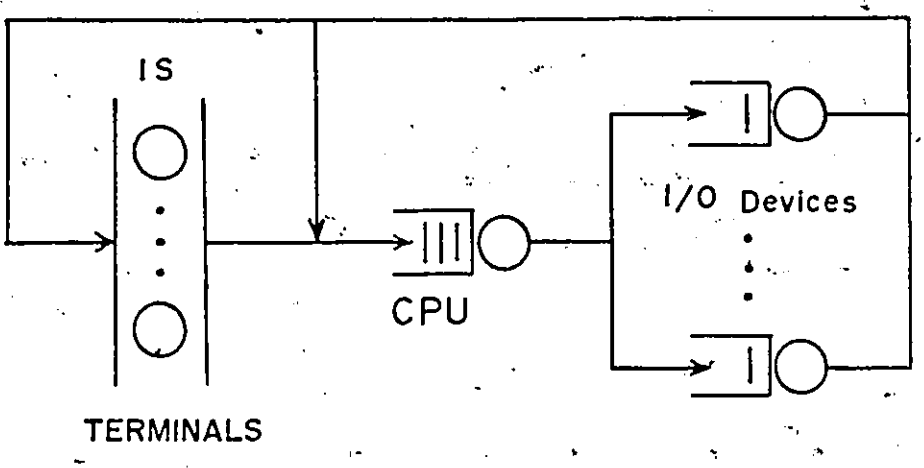

Figure 1.2 Central-server model of a computer system

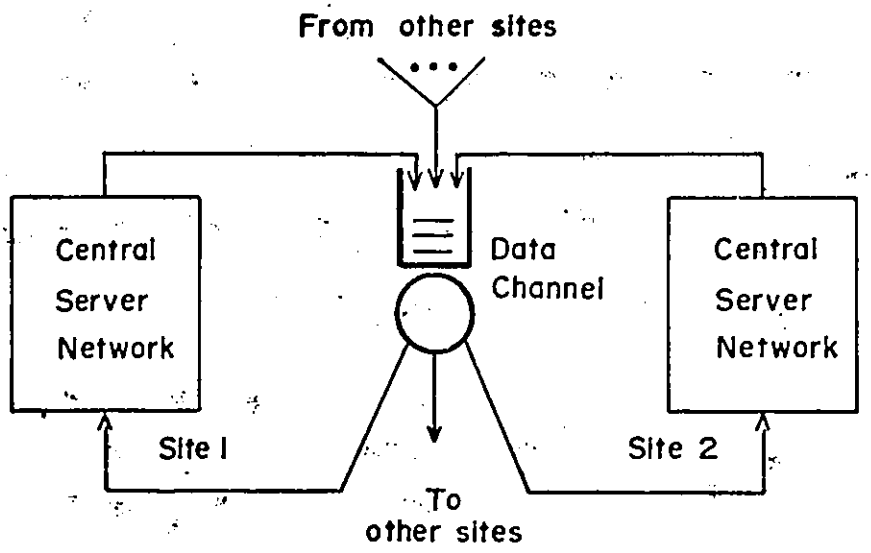

Figure 1.3 Queueing network model of a distributed computer system

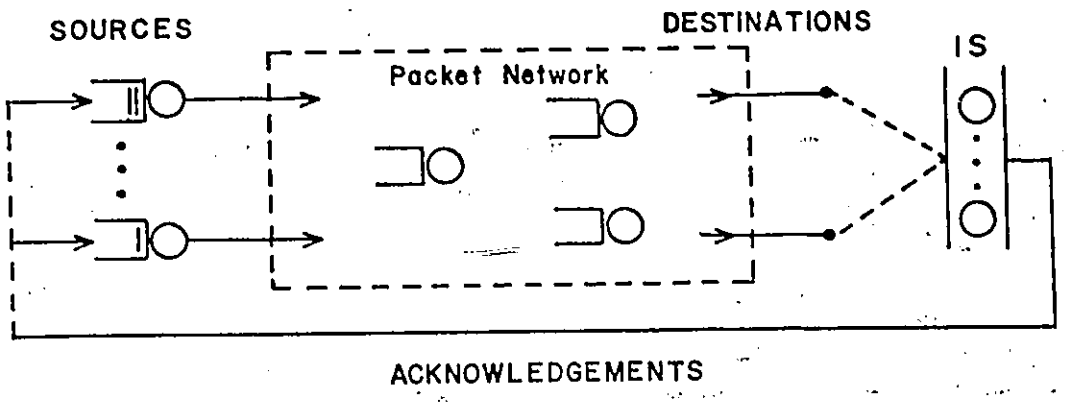

Figure 1.4 Queueing network model of a packet-switching communication network

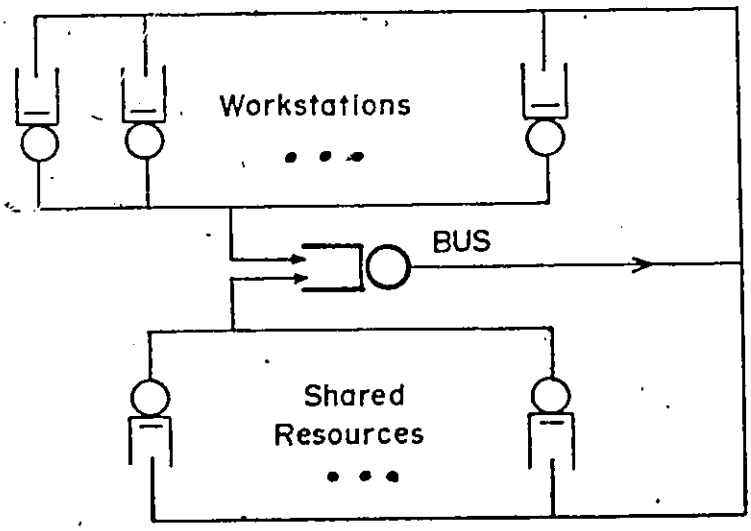

Figure 1.5 Queueing network model of a local area network

which a new packet may be injected into the network by a source. The delaying action may be achieved by having an IS center through which the customers, who have reached their destinations, must pass before they return to the source. We illustrate this communication network model in Figure 1.4.

We finally consider queueing network models of local area networks of workstations. In Chapter 7 we shall develop a new efficient algorithm for the solution of the queueing networks that arise in this particular application. The model is illustrated in Figure 1.5. In this model we associate a queue with each workstation and shared resource (e.g. bus, fileserver) in the network. Users at the workstations can perform a task at their own station or issue one to the shared resources, or even other stations, via the communication channel. We model the demands placed by the users on the resources of the network by associating a closed routing chain, having unit population, with each workstation in the network. The customer contained within each chain is allowed to visit all service centers in the network. As mentioned earlier, by the use of class-switching, we can make the routing of a customer depend on the past route so that, for example, a customer may take the following route - workstation, bus, fileserver, bus, workstation.

CHAPTER 2 - DECOMPOSITION AND AGGREGATION IN QUEUEING NETWORKS

2.1 INTRODUCTION

This Chapter provides an overview of the decomposition and aggregation technique of Simon and Ando. We consider, in some detail, how Courtois applied this technique to queueing network analysis. We begin by describing the class of nearly-completely decomposable Markov chains, since it was this class of systems that provided the initial motivation for Simon and Ando. We conclude by showing how the technique gives rise to reduced systems that are useful for the purposes of parametric analysis. The Chapter also serves to establish the notation to be adopted in Chapter 4.

2.2 NEARLY-COMPLETELY DECOMPOSABLE MARKOV CHAINS

Consider an irreducible infinitesimal generator Q, associated with a continuous-time Markov chain, that can be partitioned into square matrices as illustrated in Figure 2.1. Let $n(\alpha)$ be the order of the square matrix $Q(\alpha,\alpha)$. The notation $q_{ij}(\alpha,\beta)$ is used to denote the (i,j) element of $Q(\alpha,\beta)$ and the off-diagonal block row sum is written as $s_{i\alpha} = \sum_{\substack{\beta=1 \\ \beta\neq\alpha}}^{M_1} \sum_{j=1}^{n(\beta)} q_{ij}(\alpha,\beta)$. A left eigenvector, associated with eigenvalue one, of the matrix $(I+Q)$, where I is the identity matrix, is denoted by v and partitioned as $v = [v_1 \ldots v_{M_1}]$, where $v_\alpha = (v_{1\alpha},\ldots,v_{n(\alpha)\alpha})$ so as to conform with the partitioning of Q. Similarly, let $s = [s_1 \ldots s_{M_1}]$ where $s_\alpha = (s_{1\alpha},\ldots,s_{n(\alpha)\alpha})$.

Now let Q be written in the form $Q=Q^{*}+\varepsilon C$ where $Q^{*}=\mathrm{diag}(Q_1^{*}\ldots Q_{M_1}^{*})$, Q_α^{*} is constructed from $Q(\alpha,\alpha)$ by distributing the quantity $s_{i\alpha}$ over the elements of the ith row of $Q(\alpha,\alpha)$, for $1\leq i\leq n(\alpha)$, and $\varepsilon = \underset{i,\alpha}{\mathrm{Max}}\{s_{i\alpha}: 1\leq\alpha\leq M_1;\ 1\leq i\leq n(\alpha)\}$. The quantity ε is referred to as the maximum degree of coupling between the subsystems $Q(\alpha,\alpha)$, $1\leq\alpha\leq M_1$. If ε is sufficiently small [58], then Q is said to be nearly-completely decomposable.

Simon and Ando [58] proved that if a system is nearly-completely decomposable, then its dynamic behaviour v(t), with time t, evolves through four distinct stages.

(1) Short-term dynamics: $v(t)$ and $y^{*}(t)$ evolve similarly.

(2) Short-term equilibrium: $v_\alpha(t)$ and $v_\alpha^{*}(t)$ are reaching a similar equilibrium for $1\leq\alpha\leq M_1$.

(3) Long-term dynamics: $v_\alpha(t)$, for $1\leq\alpha\leq M_1$, continues to move towards equilibrium. The relative values among $(v_{1\alpha},\ldots,v_{n(\alpha)\alpha})$ are maintained approximately. The marginal distribution $\left[\sum_{i=1}^{n(\alpha)} v_{i\alpha}(t): 1\leq\alpha\leq M_1\right]$ evolves.

(4) Long-term equilibrium: The marginal distribution moves towards equilibrium. v(t) is reaching an overall equilibrium.

2.3 THE SIMON AND ANDO TECHNIQUE

The time decomposition of the previous Section suggests the following space decomposition technique for obtaining an approximation z to the equilibrium distribution v. The technique consists of two

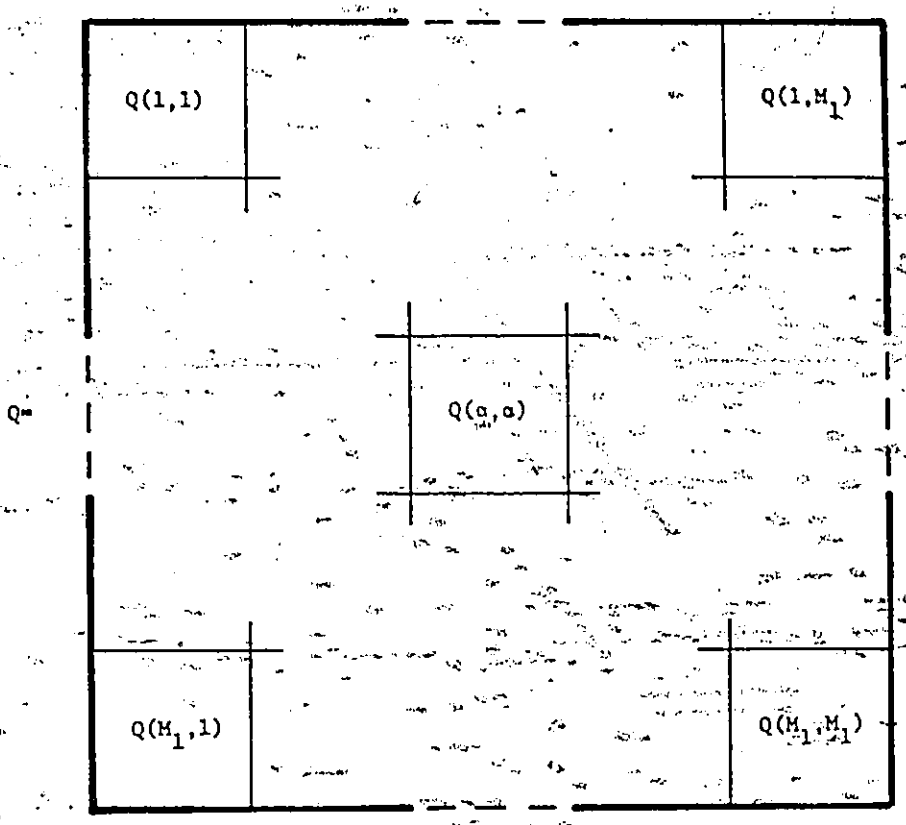

Figure 2.1 Single-level decomposition of Q

parts, decomposition and aggregation, and proceeds as follows.

Step 1: (DECOMPOSITION) Construct Q^* and find v^* where $v^*(I+Q^*)=v^*$, $v^*_\alpha \underline{1}^T=1$ for $1\leq\alpha\leq M_1$ and $\underline{1}$ is a row vector all of whose elements are one.

Step 2: (AGGREGATION) Construct the $(M_1 x M_1)$ infinitesimal generator A, of the reduced system, as follows. With a_{ij} as the (i,j) element of A, let

$$a_{ij} = \begin{cases} v_i^* Q(i,j)\underline{1}^T, & \text{for } i\neq j, \\ -\sum\limits_{\substack{k=1 \\ k\neq i}}^{M_1} a_{ik}, & \text{for } i=j. \end{cases}$$

Step 3: Find u where $u(I+A)=u$ and $u\underline{1}^T=1$.

Step 4: Compute the vector $z=[[u_1 v_1^*]\ldots[u_{M_1} v_{M_1}^*]]$.

As can be seen in Step 1, the construction of Q^* from Q is unspecified and is left open to us. How we construct Q^* will affect the error (z-v) which will be obtained. If certain special conditions are satisfied, however, it is well-known [14, Section 4] that the above procedure will yield exact results regardless of the actual value of ϵ.

Theorem 2.1: The decomposition and aggregation procedure, as described above, yields the exact equilibrium distribution v for the continuous-time Markov chain with infinitesimal generator Q if and only if Q^* is constructed in such a way that, for $1\leq\alpha\leq M_1$, the vector v^*_α is

equal to the true conditional equilibrium distribution $C_\alpha^{-1}(v_{1\alpha},\ldots,v_{n(\alpha)\alpha})$ where $C_\alpha=v_\alpha \underline{1}^T$.

The decomposition and aggregation procedure, as described above, can be extended into a hierarchical or multiple-level analysis of Q if the matrices Q_α^*, $1\leq\alpha\leq M_1$, can themselves be decomposed into square matrices in the same manner as Q. In such a situation, the decomposition and aggregation procedure can be used in Step 1 to obtain approximations $z(\alpha)$ to the vectors v_α^* rather than directly solving the eigenvector problems $v_\alpha^*(I+Q_\alpha^*)=v_\alpha^*$. Figure 2.2 illustrates a hierarchical decomposition of Q into L levels ℓ, $1\leq\ell\leq L$, and provides a notation which will be used to identify the submatrices of Q.

To describe an L-level decomposition and aggregation procedure the following notation is adopted. Let the order of the square matrix $Q(\prod_{i=\ell}^{1}(\alpha_i,\alpha_i))$ be[1] denoted by $n(\prod_{i=\ell}^{1}\alpha_i)$ and let $q_{ij}(\prod_{i=\ell}^{1}(\alpha_i,\beta_i))$ denote the (i,j) element of $Q(\prod_{i=\ell}^{1}(\alpha_i,\beta_i))$. Denote a left eigenvector of $Q^*(\prod_{i=\ell}^{1}(\alpha_i,\alpha_i))$ by $v^*(\prod_{i=\ell}^{1}\alpha_i)$. The reduced systems are denoted by $A(\prod_{i=\ell}^{1}\alpha_i)$ where the (i,j) element of $A(\prod_{i=\ell}^{1}\alpha_i)$ is $a_{ij}(\prod_{i=\ell}^{1}\alpha_i)$ and determined as

$$a_{ij}\left(\prod_{i=\ell}^{1}\alpha_i\right)=\begin{cases} v_i^*\left(\prod_{i=\ell}^{1}\alpha_i\right)Q\left((i,j)\prod_{i=\ell}^{1}(\alpha_i,\alpha_i)\right)\underline{1}^T, & \text{for } i\neq j,\\ -\sum_{\substack{k=1\\k\neq i}}^{M_{\ell+1}} a_{ik}\left(\prod_{i=\ell}^{1}\alpha_i\right), & \text{for } i=j.\end{cases}$$

---

[1] The expression $\prod_{i=\ell}^{1}(\alpha_i,\alpha_i)$ is to be interpreted, for the purposes of notation, as the ordered product $(\alpha_\ell,\alpha_\ell)\ldots(\alpha_1,\alpha_1)$.

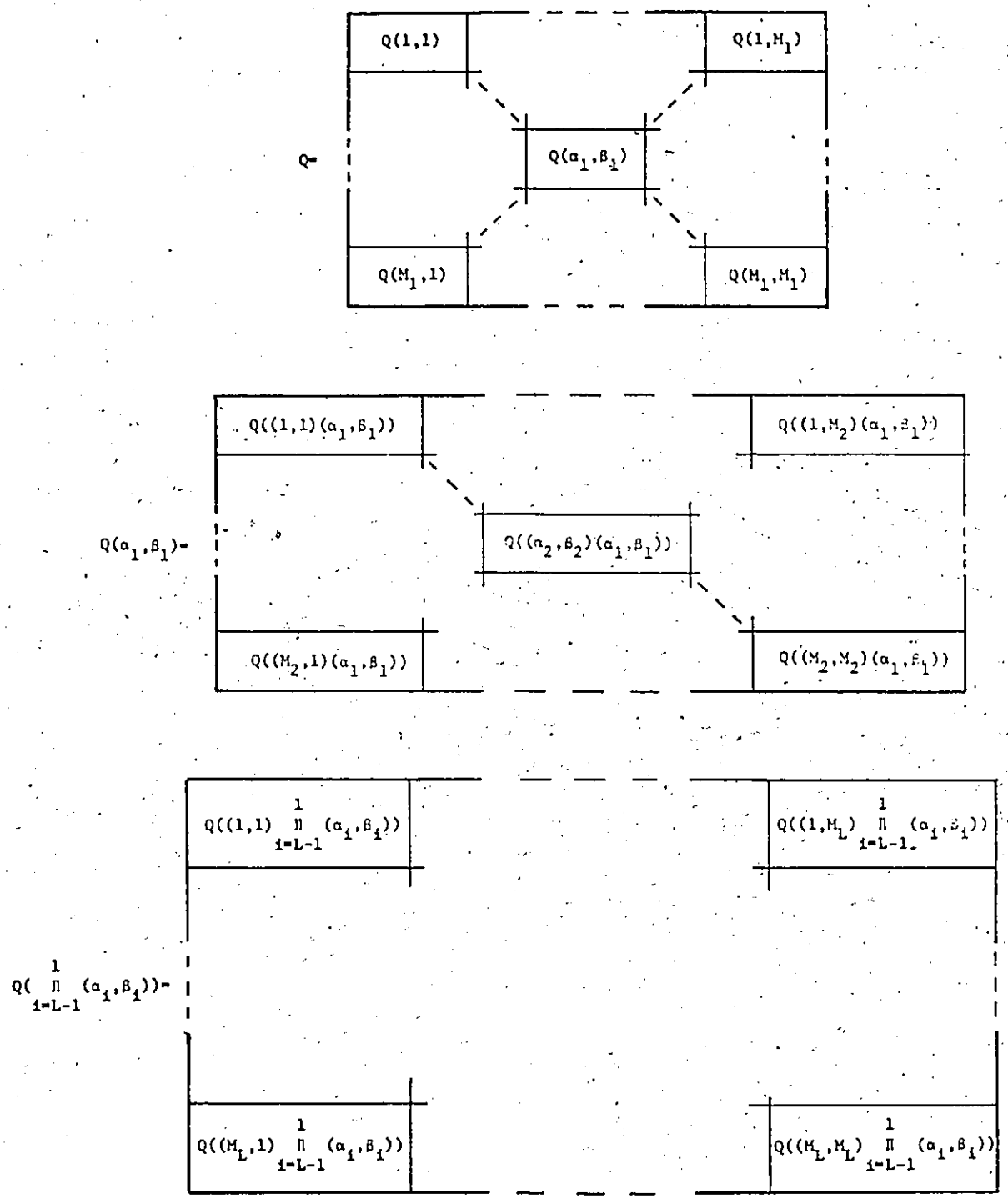

Figure 2.2 L-level decomposition of Q

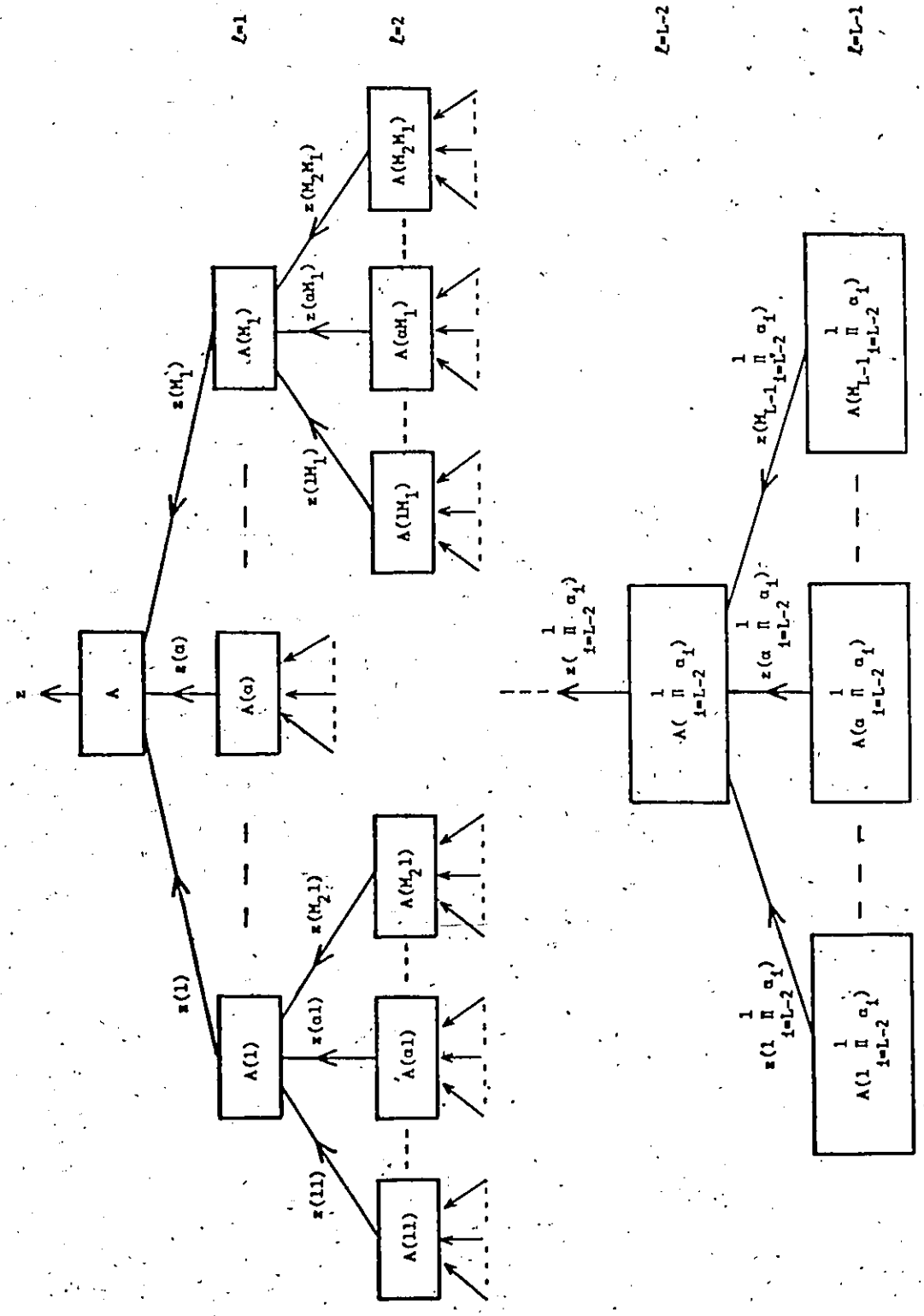

Figure 2.3 Solution tree for the L-level decomposition and aggregation procedure

Using this notation the multiple-level procedure is conveniently presented in the form of a solution tree, as depicted in Figure 2.3, where each box represents an application of Steps 2, 3 and 4 of the single-level-analysis.

2.4 THE LEVEL ANALYSIS OF COURTOIS

Courtois [13], [14] applied the multiple-level decomposition and aggregation procedure to the analysis of product-form queueing networks of the type considered by Gordon and Newell [19]. This is the class of closed BCMP networks with a single closed routing chain (R=1) and no class-switching. We shall assume, for the sake of simplicity, that there is a single fixed-rate server at each node ($\beta_i(a) = 1$). The state of the network under consideration is $\underline{n}^{(1)} = (\underline{n}_1^{(1)}, \ldots, \underline{n}_N^{(1)})$. We let $B(N,K_1)$ denote a closed single-chain BCMP network having N nodes and population K_1. The state-space of $B(N,K_1)$ is denoted by $S^{(1)}$.

Consider Fig. 2.1 and suppose Q is the generator corresponding to $B(N,K_1)$. Courtois decomposes Q according to the number of customers at some particular node, say the Nth. The α partition consists of the set of states $S(\alpha) = \{\underline{n}^{(1)}: \ \underline{n}^{(1)} \in S^{(1)}; \ n_{N1} = \alpha\}$, where $0 \leq \alpha \leq K_1$. The number of partitions is $M_1 = K_1 + 1$. An example of decomposition of the system B(4,2) is illustrated by the solid lines in Figure 2.4. If we now construct $Q^*(\alpha,\alpha)$ by simply truncating Q to the state-space $S(\alpha)$ and distribute the quantity $s_{i\alpha}$ over the nonzero elements in the ith row of $Q(\alpha,\alpha)$, then $Q^*(\alpha,\alpha)$ is the generator for the system $B(N-1,K_1-\alpha)$. Courtois, therefore, decomposes $B(N,K_1)$ into the set of systems

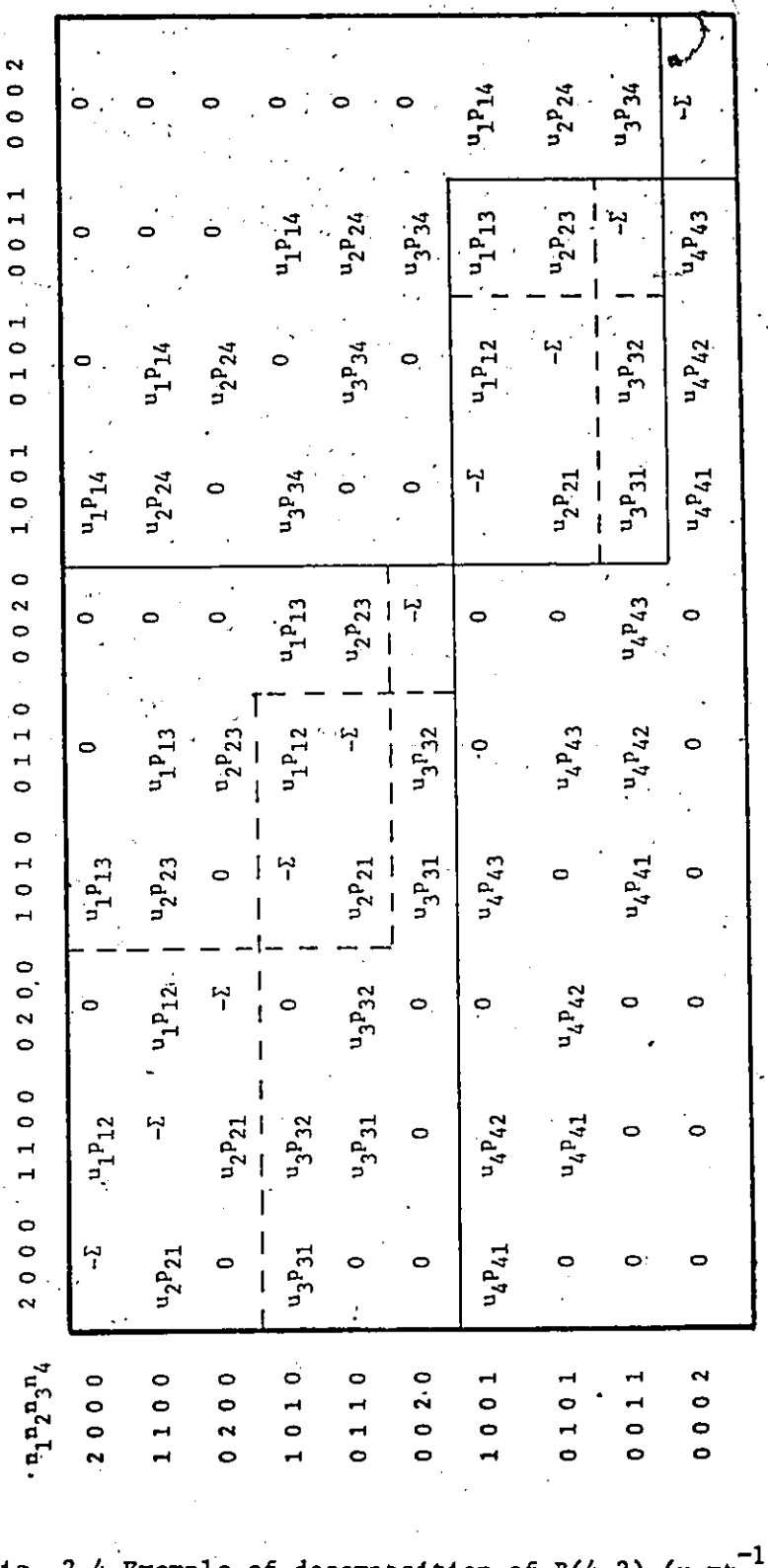

Fig. 2.4 Example of decomposition of B(4,2) ($u_i=t_i^{-1}$).

$\{B(N-1,\alpha): \ 0<\alpha<K_1\}$. Each of these subsystems can, in turn, be decomposed in a similar fashion. Such further decomposition is illustrated by the dashed lines in Figure 2.4. In general, the system $B(n,k)$ can be decomposed into the set of systems $\{B(n-1,x): \ 0<x<k\}$.

We now turn our attention to the reduced system A that arises in the aggregation procedure. The state-space of A is $\{\alpha: \ 0<\alpha<K_1\}$. By inspection of Figure 2.4, it is easy to convince oneself that

$$a_{ij} = \begin{cases} u_N(1-p_{NN}^{(1)}), & \text{if } j=i-1, \\ \sum_{n=1}^{N-1} u_n U_n(i)\, p_{nN}^{(1)}, & \text{if } j=i+1, \\ -\sum_{\substack{K=1 \\ K\neq i}}^{K_1} a_{ik}, & \text{if } j=i, \end{cases} \tag{2.1}$$

where $u_i=t_i^{-1}$ and $U_n(i)$ is the utilization of server n in the subsystem $B(N-1,K_1-i)$. The reduced system A is, therefore, simply a FCFS single-server queue with service rate $u_N(1-p_{NN}^{(1)})$ and arrival rate $\lambda(n_N) = \sum_{n=1}^{N-1} u_n U_n(n_N) p_{nN}^{(1)}$, when the number of customers at the queue is n_N.

The single-level decomposition and aggregation procedure of Section 2.3, as applied to $B(N,K_1)$, may now be summarized as follows.

Step 1: For $0<\alpha<K_1$: construct $B(N-1,K_1-\alpha)$ from $Q(\alpha,\alpha)$ by distributing $s_{i\alpha}$ over the nonzero elements of the ith row of $Q(\alpha,\alpha)$. Obtain v_α^*

corresponding to $B(N-1,K_1-\alpha)$ and compute $U_n(\alpha)$ for $1<n<N-1$.

Step 2: Construct A according to eq. 2.1.

Step 3: Find u corresponding to A (the system A is simply a birth-death process having a product-form distribution).

Step 4: Compute $z = [[u_1 v_1^*]\ldots[u_{M_1} v_{M_1}^*]]$.

From the above, we see that the decomposition and aggregation procedure reduces the analysis of $B(N,K_1)$ into the analysis of $B(N-1,\alpha)$, for $0<\alpha<K_1$, and the reduced system A. All of these systems have a smaller state-space than that of $B(N,K_1)$.

The above procedure does not, in general, yield exact results. Vantilborgh [62] has proved that $z=v$ if and only if the system $B(N-1,K_1-\alpha)$ is constructed from $Q(\alpha,\alpha)$ in such a way that the visit ratio vector $(e_{11},\ldots,e_{(N-1)1})$, associated with $B(N-1,K_1-\alpha)$, is subparallel[1] to the vector $(e_{11},\ldots,e_{N1})$ which is associated with $B(N,K_1)$.

The single-level procedure, as described above, can be extended into a multiple-level procedure, like the one depicted in Figure 2.3, since, as mentioned previously, B(n,k) can be decomposed into the set of systems $\{B(n-1,x): 0<x<k\}$. The number of levels L is (N-1). Each reduced system in Fig. 2.3 is a birth-death process.

Up until now, we have considered networks with a single closed

---

[1] A vector $(u_1,\ldots,u_{n-1})$ is subparallel to $(v_1,\ldots,v_n)$ if $(u_1,\ldots u_{n-1}) = c(v_1,\ldots,v_{n-1})$, where c is some constant [62].

routing chain (R=1). Courtois [14] has extended his procedure to the case of multiple-chain closed queueing networks. Let $B^{(R)}(N,\underline{K})$ denote a closed BCMP queueing network with R chains, N nodes, and population vector $\underline{K}=(K_1,\ldots,K_R)$. Courtois has shown that $B^{(R)}(n,\underline{k})$ can be decomposed into the set of subsystems $\{B^{(R)}(n-1,\underline{x}): \underline{0}\leq\underline{x}\leq\underline{k}\}$. The condition for exact results to be obtained is that, for $1\leq r\leq R$, the vector $(e_{1r},\ldots,e_{(n-1)r})$ associated with $B^{(R)}(n-1,\underline{x})$, is subparallel to the vector $(e_{1r},\ldots,e_{nr})$, which is associated with $B^{(R)}(n,\underline{k})$ [14].

Although the application of the decomposition and aggregation technique to closed product-form queueing networks yields considerable insight into their hierarchic structure, it is not an efficient method to obtain the mean performance measures which are the quantities that are of interest in practice. Efficient computational algorithms are surveyed in Chapter 3. Rather, the level analysis of Courtois can be useful for analyzing networks that are more complex than $B^{(R)}(N,\underline{K})$ and whose state-distributions are unknown. It is suited to the analysis of systems that are nearly-completely decomposable. Models of multiprogrammed computer systems with a limited number of memory partitions fall naturally into this class of systems and have been analyzed by decomposition and aggregation [6]. Methods for their efficient solution have been developed in [7] and [36]. The analysis of queueing networks with clusters of strongly interacting servers is considered in [63]. Networks containing customers which have a high variability of service times can also be analyzed by the decomposition and aggregation technique [66]. In addition, we may attempt to analyze systems that are neither product-form nor nearly-completely decomposable, but this involves heuristics [29,34,39,40].

2.5 PARAMETRIC ANALYSIS

A useful consequence of the decomposition and aggregation technique is that it shows one how to construct a reduced system around a particular subsystem that may be of interest. We may then vary local parameters of interest, and observe the effect, without having to repetitively analyze the entire system.

Consider Figure 2.1 and suppose Q has been decomposed in such a way that the parameters, with respect to which we wish to carry out a parametric analysis, are contained in the off-diagonal blocks $\{Q(\alpha,\beta): 1\leq\alpha,\beta\leq M_1;\ \alpha\neq\beta\}$. Then, as can be seen from the decomposition and aggregation procedure of Section 2.3, having obtained v^* in Step 1, we need only analyze the reduced system A each time the parameters are changed.

Suppose we wish to carry out a parametric analysis with respect to the parameters which are local to node N in $B(N,K_1)$. If we adopt the decomposition of Courtois, as has been illustrated in Figure 2.4, then we see that u_4 and p_{i4}, $1\leq i\leq 4$, only appear in the off-diagonal blocks. Hence, each time we vary the parameters associated with node N we need only analyze the system A arising in Step 2 of Section 2.4. This system is simply a single-server queue subjected to a state-dependent Poisson source. Exact results will be obtained if the condition of Vantilborgh [62] is satisfied in the construction of the systems $B(N-1,\alpha)$, $0\leq\alpha\leq K_1$. The quantity $\sum_{n=1}^{N-1} u_n U_n(i)\, p_{nN}^{(1)}$, in equation 2.1, is commonly interpreted as the throughput through node N in the system $B(N,K_1-i)$, when t_N is set to zero (shorted) [62]. This parametric

analysis technique for a node in a network is widely known as flow-equivalent aggregation, or 'Norton's theorem' for queues [11]. It can be derived [11] by direct algebraic means but the above presentation, in the context of decomposition and aggregation, situates the technique within a more general framework.

The above parametric analysis procedure is readily generalized to the system $B^{(R)}(N,\underline{K})$. In this situation, the reduced system A is now a single-server queue subjected to a source of R types of customers. The rate of arrival of each type of customer will depend on the state $\underline{n}_N^{(R)}$ of the queue. This generalized procedure has been derived by direct algebraic means in [28]. We mention that one may also construct a reduced system around a set of k nodes that arbitrarily interfaces the remaining network. For example, in the case of $B(N,K_1)$, we should decompose the state-space $S^{(1)}$ so that the α partition consists of the set of states $S(\alpha) = \{\underline{n}^{(1)}: \underline{n}^{(1)} \in S^{(1)}; \sum_{a=N-k+1}^{N} n_{a1}=\alpha\}$. This generalization has been proved by direct algebraic means in [4]. It applies, as well, to the system $B^{(R)}(N,\underline{K})$.

CHAPTER 3 - COMPUTATIONAL ALGORITHMS FOR QUEUEING NETWORKS

3.1 INTRODUCTION

As mentioned in Section 1.1, the difficulty posed by closed queueing networks is that the normalization constant G cannot be written in a closed form as for open networks. A direct computation would involve a sum of [22]

$$\prod_{r=1}^{R} \binom{K_r+N-1}{N-1}$$

terms, this being the cardinality of $S^{(R)}$. Such a summation is, in general, computationally intractable. Hence, the network performance measures cannot be computed by simply summing probabilities over appropriate sets of states. As a result, much effort has been directed to the development of efficient methods for analyzing closed queueing networks.

The two principal algorithms for the analysis of closed queueing networks are the convolution algorithm and mean value analysis (MVA). These algorithms yield exact results for product-form networks. We shall consider the details of these algorithms since the new algorithms that will be developed in the following Chapters all yield exact results and we will need to compare them with the established algorithms. There are, as well, certain other closely related algorithms which we will make mention of.

3.2 THE CONVOLUTION ALGORITHM

The convolution algorithm was first proposed by Buzen [9] for

single-chain networks and subsequently extended by Reiser and Kobayashi [47] for multiple-chains. The convolution algorithm is an efficient means of obtaining the normalization constant. Most performance measures of interest, such as mean queue lengths, server utilizations, mean waiting times and nodal throughputs are readily computed from the normalization constant and certain intermediate results obtained in the convolution algorithm. It is termed a normalization constant approach since the measures of interest are obtained via normalization constants.

Consider the closed BCMP queueing network $B^{(R)}(N,\underline{K})$ with R routing chains, N nodes and population vector $\underline{K}$. Let the associated normalization constant be denoted by $C_N(\underline{K})$. It may be shown [8] that

$$C_N(\underline{K}) = (f_1 * \ldots * f_N)(\underline{K})$$

where * is an R-dimensional convolution operation. The convolution algorithm consist of computing $C_N(\underline{K})$ using the recursive expression

$$C_m(\underline{k}) = (C_{m-1} * f_m)(\underline{k}) \qquad (3.1)$$

with initial conditions $C_1(\underline{k}) = f_1(\underline{k})$ and $C_m(\underline{k}-\underline{1}_r) = 0$ if $k_r < 0$, where $\underline{1}_r$ is a unit vector pointing in the rth direction. If node m is a state-dependent server, then eq. 3.1 is explicitly

$$C_m(\underline{k}) = \sum_{n_R=0}^{k_R} \cdots \sum_{n_2=0}^{k_2} \sum_{n_1=0}^{k_1} f_m(\underline{n}) C_{m-1}(\underline{k}-\underline{n}). \qquad (3.2)$$

If node m is a constant-speed server, then eq. 3.2 reduces to [8]

$$C_m(\underline{k}) = C_{m-1}(\underline{k}) + \sum_{r=1}^{R} w_{mr} C_m(\underline{k}-\underline{1}_r). \qquad (3.3)$$

Ignoring the computation of the initial conditions and assuming all nodes have constant-speed servers, the total number of operations (multiplications, additions) to arrive at $C_N(\underline{K})$, using eq. 3.3, is

$$2R(N-1) \prod_{r=1}^{R} (K_r+1). \qquad (3.4)$$

The required storage space, in number of array locations, to implement the algorithm is [8]

$$2 \prod_{r=1}^{R} (K_r+1). \qquad (3.5)$$

If node m is a state-dependent server, then the computation of $C_m(\underline{k})$ for $\underline{0} \leq \underline{k} \leq \underline{K}$, using the previously computed values of $C_{m-1}(\underline{k})$, $\underline{0} \leq \underline{k} \leq \underline{K}$, requires of the order of (see for example [32])

$$\prod_{r=1}^{R} (K_r+1)(K_r+2)/2 \qquad (3.6)$$

operations. The space requirement is unchanged from eq. 3.5.

We now consider how the mean performance measures are obtained from the normalization constants. We shall first consider the case

where we have constant-speed servers. We denote the throughput, utilization, mean queue length (node population) and mean waiting time (queueing + service) of type r customers at node i by T_{ir}, U_{ir}, Q_{ir} and W_{ir}, respectively. Also let $T_{ir}^{(c)}$, $U_{ir}^{(c)}$, $Q_{ir}^{(c)}$, and $W_{ir}^{(c)}$ denote the same respective quantities but on a per class basis, where $c \in C_i(r)$.

The mean performance measures, for a network of constant-speed servers, can be obtained as follows [34]:

$$
\begin{aligned}
T_{ir} &= e_{ir}\, C_N(\underline{K} - \underline{1}_r) \,/\, C_N(\underline{K}), \\
U_{ir} &= t_{ir}\, T_{ir}, \\
Q_{ir} &= \sum_{a=1}^{K_r} w_{ir}^{a}\, C_N(\underline{K} - a\underline{1}_r) \,/\, C_N(\underline{K}), \\
W_{ir} &= Q_{ir} \,/\, T_{ir},
\end{aligned}
$$

$$
\begin{aligned}
T_{ir}^{(c)} &= \alpha_{ic}^{(r)}\, T_{ir} \,/\, e_{ir}, \\
U_{ir}^{(c)} &= m_{ic}\, T_{ir}^{(c)}, \\
Q_{ir}^{(c)} &= \alpha_{ir}^{(c)}\, m_{ic}\, Q_{ir} \,/\, w_{ir}, \\
\text{and} \quad W_{ir}^{(c)} &= Q_{ir}^{(c)} \,/\, T_{ir}^{(c)}
\end{aligned}
\qquad (3.7)
$$

In the case of a state-dependent server, the equations for T_{ir}, W_{ir}, $T_{ir}^{(c)}$, $Q_{ir}^{(c)}$, and $W_{ir}^{(c)}$ remain the same as in eqs. 3.7 but Q_{ir} must be computed using [34]

$$
Q_{ir} = \sum_{x=1}^{K_r} x \Pr(n_{ir} = x) \qquad (3.8)
$$

where

$$
\Pr(n_{ir} = x) = \sum_{\underline{k}_i \in \Omega(x)} \Pr(\underline{n}_i^{(R)} = \underline{k}_i),
$$

$$\Omega(x) = \{\underline{k}_i = (k_{i1},\ldots,k_{iR}): 0 \leq k_{is} \leq K_s \text{ for } 1 \leq s \leq R,\ s \neq r;\ k_{ir} = x\},$$

$$\Pr(\underline{n}_i^{(R)} = \underline{k}_i) = f_i(\underline{k}_i)\, C_{N-\{i\}}(\underline{K} - \underline{k}_i) \,/\, C_N(\underline{K}),$$

and $C_{N-\{i\}}(\underline{k})$ is the normalization constant of a network identical to $B^{(R)}(N,\underline{k})$ but with the ith node removed, that is, $C_{N-\{i\}}(\underline{k}) = (f_1 * \ldots * f_{i-1} * f_{i+1} * \ldots * f_N)(\underline{k})$.

A practical problem with the convolution algorithm is that the floating-point range of a machine may be exceeded in the course of computing $C_N(\underline{K})$. A partial solution is to scale the quantities $(e_{1r},\ldots,e_{Nr})$ by a constant factor, for each chain r. This is termed static-scaling. A scaling procedure is given in [34, Section 3.5.4.b]. Another approach is to vary the scaling factors as we compute $C_N(\underline{K})$. This is termed dynamic scaling and has been developed by Lam [31]. The algorithm to be described in the following Section avoids these potential underflow/overflow problems.

3.3 MEAN VALUE ANALYSIS

Reiser and Lavenberg [49] have developed an elegant algorithm for the solution of closed queueing networks that works directly in terms of the mean performance measures (hence mean value analysis (MVA)). It is based on the Arrival Theorem [35], [57] for product-form queueing networks and Little's result [38]. In words, the Arrival Theorem states that the stationary distribution for $B^{(R)}(N,\underline{K})$ at arrival epochs of type r customers is equal to the stationary distribution for

$B^{(R)}(N,\underline{K}-\underline{1}_r)$ at arbitrary times.

Let $P_i(j,\underline{k})$ be the queue length distribution at node i in $B^{(R)}(N,\underline{k})$. Also let $T_{ir}(\underline{k})$, $U_{ir}(\underline{k})$, $Q_{ir}(\underline{k})$ and $W_{ir}(\underline{k})$ be the mean performance measures associated with $B^{(R)}(N,\underline{k})$. The following recursive equations constitute the MVA algorithm [8,34,49]:

$$W_{ir}(\underline{k}) = \begin{cases} t_{ir}, \text{ if i is an IS center,} \\ t_{ir}(1+\sum_{s=1}^{R} Q_{is}(\underline{k}-\underline{1}_r)), \text{ if i is a single server fixed-rate center,} \\ t_{ir}\sum_{j=1}^{K} jP_i(j-1,\underline{k}-\underline{1}_r)/\beta_i(j), \text{ if i is a state-dependent center,} \end{cases}$$

$$T_{ir}(\underline{k}) = e_{ir} K_r / \sum_{j=1}^{N} e_{jr} W_{jr}(\underline{k}),$$

$$Q_{ir}(\underline{k}) = T_{ir}(\underline{k}) W_{ir}(\underline{k}),$$

$$P_i(j,\underline{k}) = \frac{1}{\beta_i(j)} \sum_{s=1}^{R} t_{is} T_{is}(\underline{k}) P_i(j-1,\underline{k}-\underline{1}_s),$$

$$P_i(0,\underline{k}) = 1-\sum_{j=1}^{K} P_i(j,\underline{k}), \qquad (3.9)$$

where $K = K_1 + \ldots + K_R$.

The MVA algorithm consists of computing the mean performance measures for $B^{(R)}(N,\underline{K})$, using eqs. 3.9, by recursion over $\underline{k}$, where $\underline{0}\leq\underline{k}\leq\underline{K}$. The performance measures on a per class basis can be obtained using eqs. 3.7.

In the case where there are no state-dependent servers in the network the time requirement, in number of operations, to obtain the

mean performance measures, is approximately the same as eq. 3.4. The space requirement, in number of array locations, is [8]

$$N \prod_{r=1}^{R} (K_r + 1). \qquad (3.10)$$

This is more than for the convolution algorithm when $N > 3$.

When there are state-dependent servers, the MVA algorithm is complicated by the need to compute and store the queue-length distribution $P_i(j,\underline{k})$. The algorithm may also fail numerically as $P_i(0,\underline{k})$ tends to zero. Reiser [50] has devised a means to circumvent this problem but it involves an increase in computational costs.

3.4 OTHER EXACT ALGORITHMS

Several other exact algorithms have been developed in an attempt to improve upon difficulties encountered in the original convolution algorithm. The Local Balance Algorithm for Normalizing Constants (LBANC) [12] is closely related to MVA and avoids truncation errors that occur in the convolution algorithm for normalization constants at the lower limit of the floating point range. LBANC requires, however, the computation and storage of unnormalized queue-length distributions for state-dependent centers. In hybrid Convolution-LBANC [34], convolution is used for the state-dependent centers and LBANC is used for the others. The Normalized Convolution Algorithm (NCA) of Reiser [50] eliminates the overflow problems of convolution. In hybrid

MVA-NCA [50], NCA is used for the state-dependent centers and MVA for the remaining centers.

A number of algorithms have as well been developed that exploit certain properties of the models being analyzed. In models of communication networks (Section 1.5), each closed routing chain may only visit a small subset of the centers in the network. This sparsity or locality property has been exploited by Lam and Lien [32] who developed a so-called Tree Convolution algorithm that, in certain situations, has dramatically lower space and time requirements than the conventional convolution algorithm. A Tree MVA algorithm has also been proposed [61]. Balbo et al [3] have developed an efficient procedure for analyzing the class of homogeneous queueing networks that arise in the modeling of local area networks. This procedure exploits the inherent symmetry of homogeneous networks (to be defined in Section 7.2).

3.5 THE SOLUTION OF MIXED NETWORKS

In this Section we consider how one may obtain the mean performance measures for a queueing network that contains both open and closed routing chains. In the following, for the sake of simplicity, we shall limit our attention to the case of a mixed network containing a single open chain and having only constant-speed or limited queue-dependent servers. A limited queue-dependent server is a state-dependent server for which

$$\beta_i(n) = \beta_i(g_i)$$

for $n > g_i$, where $g_i > 2$ [34]. Mixed networks with limited queue-dependent centers are stable if and only if

$$w_{ir_0} / \beta_i(g_i) < 1,$$

for $1 \leq i \leq N$, where r_0 is the index of the open chain [34].

The approach to analyzing a mixed network is to transform it to an equivalent closed network which has the same performance measures for the closed chains as in the original mixed network. The equivalent closed network is constructed from the original mixed network by removing the open chain and introducing modified service rate functions of the form $\beta_i^*(n)$ and given by [34,52]

$$\beta_i^*(n) = \beta_i(n)\alpha_i(n-1)/\alpha_i(n)$$

where, for limited queue-dependent servers [53],

$$\alpha_i(n) = \beta_i(g_i)^{g_i-n-1} / \left(\left\{\prod_{h=n+1}^{g_i-1} \beta_i(h)\right\} \{(1-w_{ir_0}/\beta_i(g_i))^{n+1}\}\right)$$

$$+ \sum_{a=1}^{g_i-2} \binom{n+a}{a} w_{ir_0} \left\{1 / \prod_{h=n+1}^{n+a} \beta_i(h) - 1 / \prod_{h=n+1}^{g_i-1} \beta_i(h)(\beta_i(g_j))^{n-g_j+i+1}\right\}$$

for $n < (g_i - 2)$ and

$$\alpha_i(n) = 1 / (1 - w_{ir_0} / \beta_i(g_i))^{n+1}$$

for $n \geq (g_i - 1)$. For constant-speed servers, we therefore have

$$\beta_i^*(n) = (1 - w_{ir_0}).$$

According to eq. 1.3 this implies that, for the case of constant-speed servers, in the transformation from the mixed network to the equivalent closed network, we need merely replace w_{ir} by $w_{ir}/(1-w_{ir_0})$.

Having arrived at an equivalent closed network, the performance measures for the closed chains can be obtained using the conventional algorithms that apply to closed networks. The performance measures for the open chain are obtained as follows [34]:

$$Q_{ir_0} = w_{ir_0} \sum_{k=1}^{K} (k+1)\, P_i(k,\underline{K}) \,/\, \beta_i^*(k+1)$$

if center i is limited queue-dependent, where $K = \sum_{\substack{r=1 \\ r \neq r_0}}^{R} K_r$ and $P_i(k,\underline{K})$ is the queue length distribution associated with the equivalent closed network.

$$Q_{ir_0} = (w_{ir_0} \,/\, 1 - w_{ir_0})\Big(1 + \sum_{\substack{r=1 \\ r \neq r_0}}^{R} Q_{ir}\Big) \qquad (3.11)$$

if center i is a constant-speed server.

$$Q_{ir_0} = w_{ir_0}$$

if center i is IS.

$$T_{ir_0} = e_{ir_0} \text{ and } W_{ir_0} = Q_{ir_0} \,/\, T_{ir_0}.$$

CHAPTER 4 - DECOMPOSITION AND AGGREGATION BY CHAIN

4.1 INTRODUCTION

In Section 2.4 we have shown how Courtois decomposes $R^{(R)}(N,\underline{K})$ into subsystems that consist of subsets of the nodes. In this Chapter we introduce a new technique of decomposition in which the subsystems consist of subsets of the routing chains. We call this technique decomposition by chain. With this technique the analysis of a multiple-chain closed queueing network is resolved into a hierarchy of single-chain queueing network problems. A useful consequence of adopting this new approach is that a reduced network is formed which contains only one particular closed routing chain. A parametric analysis can then be made with respect to the routing matrix of this chain without having to repetitively analyze the entire multiple-chain network. It will be proved that this can be done without introducing any error. The method proposed in this Chapter is therefore an exact decomposition and aggregation technique.

The motivation for decomposing by chain stems from the problem in the modeling of communication networks where we may wish to analyze the performance of a single virtual channel of interest which is embedded in a network. One may wish to observe the effect of various routing strategies without having to analyze the entire network. For such a parametric analysis it is clearly advantageous to be able to construct a simplified reduced network, around the particular routing chain of interest, especially if the original network consists of a large number

of chains. This is what we have accomplished by appealing to the theory of decomposition and aggregation. It is to be mentioned that there exist approximation techniques for constructing a reduced network around a particular routing chain of interest. One approach is to replace all the chains, which are not of particular interest, by a single open chain [41]. Another is to 'coalesce' them into a single closed chain [54].

The chapter is organized as follows. In the following Section we observe that if a queueing network is reversible, then the decomposition and aggregation procedure will yield exact results for almost arbitrary decompositions. This observation will serve to simplify considerably our arguments. In Section 4.3 we transform $B^{(R)}(N,\underline{K})$ into an equivalent network of exponential queues having the PS discipline so that this reversibility condition will hold. We present, in Section 4.4, for the sake of clarity, the simplest case of a single-level decomposition by chain of a multiple-chain PS network. It is shown that an R-chain PS network can be decomposed into a set of (R-1)-chain PS networks. It is proved that, with such a decomposition, exact results can be obtained by the process of aggregation. In Section 4.5, the reduced system which results from the aggregation procedure is interpreted as being a single-chain queueing network whose servers have service rates that depend on the state $\underline{n}^{(1)}$ of the network. Section 4.6 is concerned with the analysis of this queueing network. In general, such a network does not possess a product-form distribution but it is demonstrated that the network, arising here,

does have a distribution which can be expressed in the form of a product. Section 4.7 summarizes the single-level procedure and describes how a parametric analysis of a particular routing chain of interest may be made with respect to the routing matrix. In Section 4.8, the time and space requirements of this parametric analysis procedure are derived and compared with the requirements of a straightforward repetitive analysis of the network using the convolution algorithm. In Section 4.9 and 4.10, the multiple-level decomposition and aggregation by chain procedure is presented. It is shown that it too is exact. Section 4.11 illustrates the parametric analysis of a routing chain in a store-and-forward network model. Finally, in Section 4.12, we present an example of a generator matrix that is decomposed by chain.

4.2 ON DECOMPOSITION AND AGGREGATION IN REVERSIBLE QUEUEING NETWORKS

In the following Theorem we observe that if a Markov chain is reversible, then its analysis by decomposition and aggregation will be exact for almost arbitrary choices of decomposition.

Theorem 4.1: If the Markov chain with generator Q is reversible, $Q^*_\alpha = Q(\alpha,\alpha) + \text{diag}\,(s_\alpha)$ for $1 \leq \alpha \leq M_1$, and the set of states associated with Q^*_α form a closed communicating class, then the decomposition and aggregation procedure of Section 2.3 yields exact results.

Proof: Let $S(\alpha)$ be the set of states associated with Q^*_α. Now Q^*_α is

constructed by simply truncating the Markov chain to the set $S(\alpha) \subset S$ so that, by Corollary 1.2, $\bar{v}^{*}_{\alpha} = v_{\alpha} / v_{\alpha} \underline{1}^{T}$. Therefore, by Theorem 2.1, we obtain exact results.

A direct consequence of Theorem 4.1 is that if a queueing network is reversible, then we may also obtain exact results for almost arbitrary decompositions. If we are to analyze a queueing network it is, therefore, worthwhile seeing if the conditions for reversibility can be satisfied. From Section 1.4 we see that $B^{(R)}(N,\underline{K})$ is reversible if there are no FCFS centers, the service-time requirements are all exponentially distributed, and eq. 1.4 is satisfied. These conditions may appear to be very restrictive. It is possible, however, to construct a reversible queueing network that has the same marginal state distribution as one which is not reversible. We may replace FCFS centers by ones with the PS discipline. That this has no effect can be seen from eq. 1.3. By the insensitivity property, we may replace general service-time distributions by exponentional ones having the same mean. Finally, it is always possible to construct routing matrices $P^{(r)}$, $1 \leq r \leq R$, that satisfy eq. 1.4 for any given set of visit ratios $\{\alpha_{ic}^{(r)}: \ 1 \leq i \leq N; \ c \in C_i(r)\}$.

In conclusion, if we are to analyze a system $B^{(R)}(N,\underline{K})$ by decomposition and aggregation, then it is convenient to make the necessary transformations so that reversibility will hold. The transformations do not perturb the marginal distribution and will ensure that exact results will be obtained for almost any network

decomposition we may care to make. The decomposition of Courtois is only one particular network decomposition that yields exact results. In retrospect, the Vantilborgh condition [62], for exact results to be obtained in the level analysis of Courtois, is trivial.

4.3 THE EQUIVALENT PS NETWORK

In this Chapter we are concerned with applying the decomposition and aggregation technique to the system $B^{(R)}(N,\underline{K})$. We assume, for the sake of simplicity, that $B^{(R)}(N,\underline{K})$ consists of only FCFS service centers. The results of this Chapter can be extended easily to networks containing PS, LCFSPR or IS centers.

Our preliminary step is to transform $B^{(R)}(N,\underline{K})$ into a network of exponential PS service centers with routing matrices $P^{(r)}$ that satisfy eq. 1.4 so that the reversibility condition will hold. Another reason for proceeding in this way is that in the PS network one can identify readily a hierarchy of subsystems associated with subsets of the routing chains. In the following, we shall refer to this transformed network as the PS network. We stress that the PS network is indistinguishable from $B^{(R)}(N,\underline{K})$ as far as the marginal distribution (eq. 1.3) is concerned.

4.4 SINGLE-LEVEL DECOMPOSITION BY CHAIN

Consider an R-chain PS network with state-space $S^{(R)}$ and state $\underline{n}^{(R)}$. Then $S^{(R)} = \{\underline{n}^{(R)}: n_{ir} > 0 \text{ for } 1 \leq i \leq N,\ 1 \leq r \leq R: \sum_{i=1}^{N} n_{ir} = K_r \text{ for } 1 \leq r \leq R\}$.

Let $S(k_{1R},\ldots,k_{NR}) = \{\underline{n}^{(R)}: \ \underline{n}^{(R)} \in S^{(R)};\ n_{1R}=k_{1R},\ldots,n_{NR}=k_{NR}\}$ and let $L_R = \{(k_{1R},\ldots,k_{NR}): \ k_{jR} > 0 \text{ for } 1 \leq j \leq N;\ \sum_{j=1}^{N} k_{jR} = K_R\}$. Clearly then $\bigcup_{(k_{1R},\ldots,k_{NR})\in L_R} S(k_{1R},\ldots,k_{NR}) = S^{(R)}$. The infinitesimal generator Q, corresponding to this queueing network, can be decomposed according to the disposition of the chain R customers in the network, as depicted in Figure 4.1, where the partitioning is made according to the sets of states $S(k_{1R},\ldots,k_{NR})$. The number of subsystems in the decomposition is $M_1 = \binom{K_R+N-1}{N-1}$, this being the number of states in a single-chain N-node PS network with chain population K_R [8]. An example decomposition is given in Section 4.12.

Now construct Q^* by letting $Q^* = \text{diag}(Q(1,1)\ldots Q(M_1,M_1)) + \text{diag}(s)$ where s was defined in Section 2.2. The matrix Q^*_α, $1 \leq \alpha \leq M_1$, having row sums equal to zero, is then an infinitesimal generator matrix and defines a process with state space $S^{(R-1)}$. The off-diagonal elements of Q^*_α are identical to those of $Q(\alpha,\alpha)$ and hence the process defined by Q^*_α can be interpreted as an (R-1)-chain PS network in which k_{iR} customers of chain R are 'fixed' at node i and cannot depart. Such a system is equivalent to an (R-1)-chain PS network whose servers have state-dependent service rates $\beta_i(a_i^{(R-1)}) = a_i^{(R-1)}/(a_i^{(R-1)}+k_{iR})$. That is, when there are $a_i^{(R-1)} = \sum_{r=1}^{R-1} n_{ir}$ customers at service center i, each is receiving service at the rate $1/(a_i^{(R-1)}+k_{iR})$. Let this system with state-dependent servers be denoted by $M(R-1,(k_{1R},\ldots,k_{NR}))$.

It is now proved that with such a single-level decomposition, exact results are obtained by the decomposition and aggregation procedure.

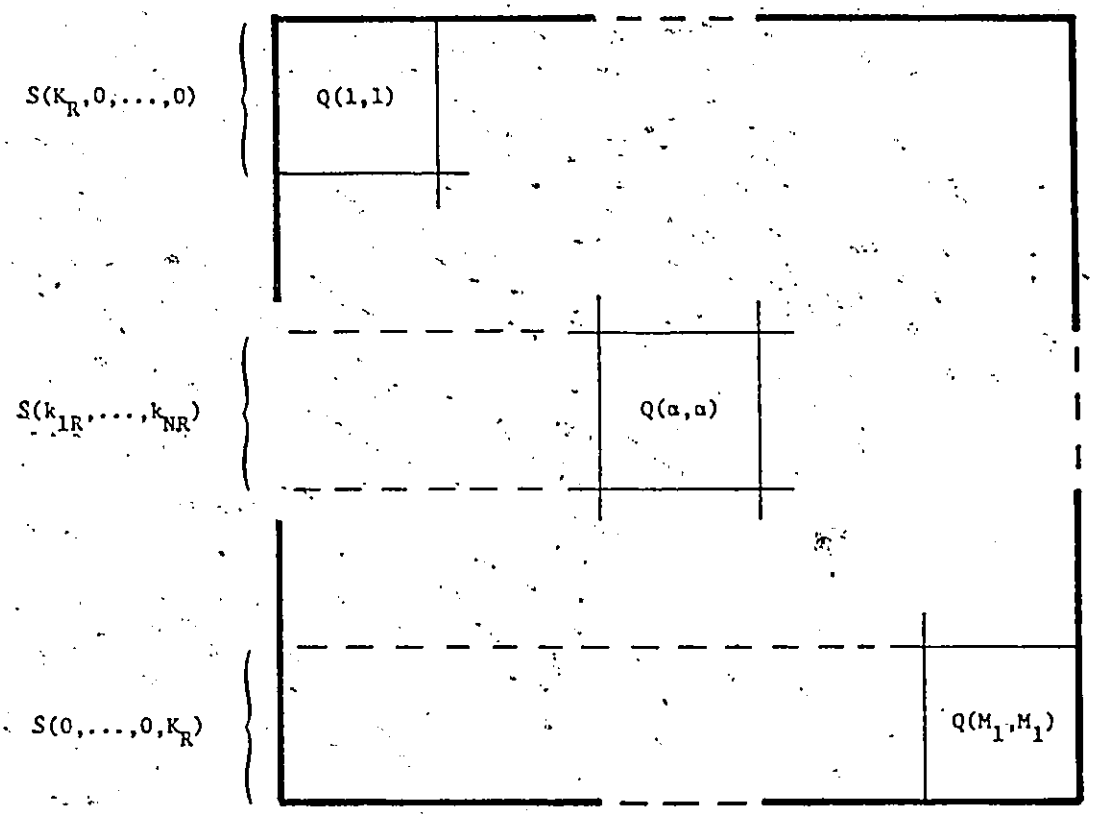

Figure 4.1 Single-level decomposition of the R-chain PS network

Theorem 4.2: If the infinitesimal generator, corresponding to the R-chain PS network, is decomposed as in Figure 4.1 and $Q^*_\alpha = Q(\alpha,\alpha) + \text{diag}(s_\alpha)$, then the stationary probabilities $Pr(\underline{n}^{(R)})$ are obtained exactly by the single-level decomposition and aggregation procedure.

Proof: By assumption, $Q^*_\alpha = Q(\alpha,\alpha) + \text{diag}(s_\alpha)$, $S(k_{1R},\ldots,k_{NR})$ is a set of communicating states and the PS network is reversible so it immediately follows from Theorem 4.1 that exact results will be obtained.

4.5 SINGLE-LEVEL AGGREGATION BY CHAIN

In the previous Section it has been shown that the analysis of an R-chain PS network can be decomposed into the analysis of M_1 (R-1)-chain PS networks and that no error would be introduced by the single-level decomposition and aggregation procedure. This Section is concerned with interpreting the result of the second step of the procedure, the aggregation step, in which the reduced system A is constructed. With the decomposition of Q, as given in Figure 4.1, it will be seen that the system A can be interpreted as the infinitesimal generator of an N-node single-chain FCFS queueing network with customer population K_R and servers whose service rates depend on the state $\underline{n}^{(1)}$ of the network. Such a queueing network, however, is not among those considered by Baskett et al [5] and does not, in general, exhibit a product-form distribution. It will be seen in the following Section, however, that the reduced system A, arising here, does have a

distribution expressible in the form of a product.

Consider Figure 4.1 where Q is partitioned according to the sets of states $S(k_{1R},\ldots,k_{NR})$, $(k_{1R},\ldots,k_{NR}) \in L_R$. Let $Q(\alpha,\alpha_{ij}^{+})$ be the submatrix of Q that contains the transitions between the set of states $S(k_{1R},\ldots,k_{NR})$ and the other set $S(k_{1R},\ldots,k_{NR})^{+ij}$, where $(k_{1R},\ldots,k_{NR})^{+ij}=(k_{1R},\ldots,k_{NR})-\delta_i+\delta_j$, and $\delta_j=(0,\ldots,0,1,0,\ldots,0)$ where the one is entered in the i'th position. The N-tuple $(k_{1R},\ldots,k_{NR})^{+ij}$ is said to be adjacent to $(k_{1R},\ldots,k_{NR})$. Since the state transition $(\underline{n}^{(R-1)}k_R) \rightarrow (\underline{n}^{(R-1)}k_R^{+ij})$, where $k_R=(k_{1R},\ldots,k_{NR})$ and $k_R^{+ij} = k_R-\delta_i+\delta_j$, can be made with a single chain R customer service completion, $q_{IJ}(\alpha,\alpha_{ij}^{+})$ is nonzero for some $1\leq I,J\leq n(\alpha)$. Now suppose $Q(\alpha,\beta)$ is the submatrix that contains the transitions between $S(k_{1R},\ldots,k_{NR})$ and some other set $S(k'_{1R},\ldots,k'_{NR})$, $(k'_{1R},\ldots,k'_{NR})\in L_R$. Then $q_{IJ}(\alpha,\beta)$ is zero for all $1\leq I,J\leq n(\alpha)$ if $(k_{1R},\ldots,k_{NR})$ is not adjacent to $(k'_{1R},\ldots,k'_{NR})$. Therefore, the element $a_{\alpha\beta}$ of A is nonzero only if $(k_{1R},\ldots,k_{NR})$ and $(k'_{1R},\ldots,k'_{NR})$ are adjacent, in which case $(k'_{1R},\ldots,k'_{NR})=(k_{1R},\ldots,k_{NR})^{+nm}$ for some $1\leq n,m\leq N$, $n\neq m$.

Since L_R is identical to the state-space of a single-chain FCFS queueing network with customer population K_R and $a_{\alpha\beta}$ is nonzero only if $(k_{1R},\ldots,k_{NR})$ and $(k'_{1R},\ldots,k'_{NR})$ are adjacent, the reduced system A can be interpreted as the infinitesimal generator of a single-chain FCFS queueing network with chain population K_R. Denote such a system by $N_R(\underline{n}^{(1)})$. According to Step 2 in Section 2.3, $a_{\alpha\alpha_{ij}^{+}} = v_\alpha^* Q(\alpha,\alpha_{ij}^{+})\underline{1}^T$ where v_α^* is the distribution associated with $M(R-1,(k_{1R},\ldots,k_{NR}))$. This may be written in the form $a_{\alpha\alpha_{ij}^{+}} = p_{ij}t_{iR}^{-1}\mu_i(\underline{n}^{(1)}=(k_{1R},\ldots,k_{NR}))$

where $\mu_i(\underline{n}^{(1)})$ is to be regarded as the service rate of server i when the state of $N_R(\underline{n}^{(1)})$ is $\underline{n}^{(1)}$, $\underline{n}^{(1)} \in L_R$, and $P = [p_{ij}]$ is some arbitrarily chosen routing matrix for $N_R(\underline{n}^{(1)})$. The quantity $\mu_i(\underline{n}^{(1)})$ can be interpreted as the conditional rate of service for chain R customers at node i when the state of the chain R customers in $B^{(R)}(N,\underline{K})$ is $\underline{n}^{(1)} = (k_{1R},\ldots,k_{NR})$.

The single-level decomposition and aggregation by chain procedure for the PS network can now be represented in terms of a set of queueing network problems as depicted in Figure 4.2. The single-level procedure has decomposed the analysis of an R-chain network into the analysis of M_1 (R-1)-chain PS networks and one single-chain network $N_R(\underline{n}^{(1)})$. In the following Section it is shown that the analysis of $N_R(\underline{n}^{(1)})$ (Step 3), which would in general require solving an eigenvector problem, is simplified by virtue of its possessing a distribution expressible in the form of a product.

4.6 THE PRODUCT-FORM DISTRIBUTION FOR THE REDUCED SYSTEM

In general, a queueing network such as $N_R(\underline{n}^{(1)})$, whose servers have network state-dependent service rates $\mu_i(\underline{n}^{(1)})$, does not have a distribution which can be written in a simple form. However, it is to be shown that the network $N_R(\underline{n}^{(1)})$, arising from the procedure depicted in Figure 4.2, does have a distribution expressible in a product-form.

Theorem 4.3: The state-distribution for $N_R(\underline{n}^{(1)})$ is

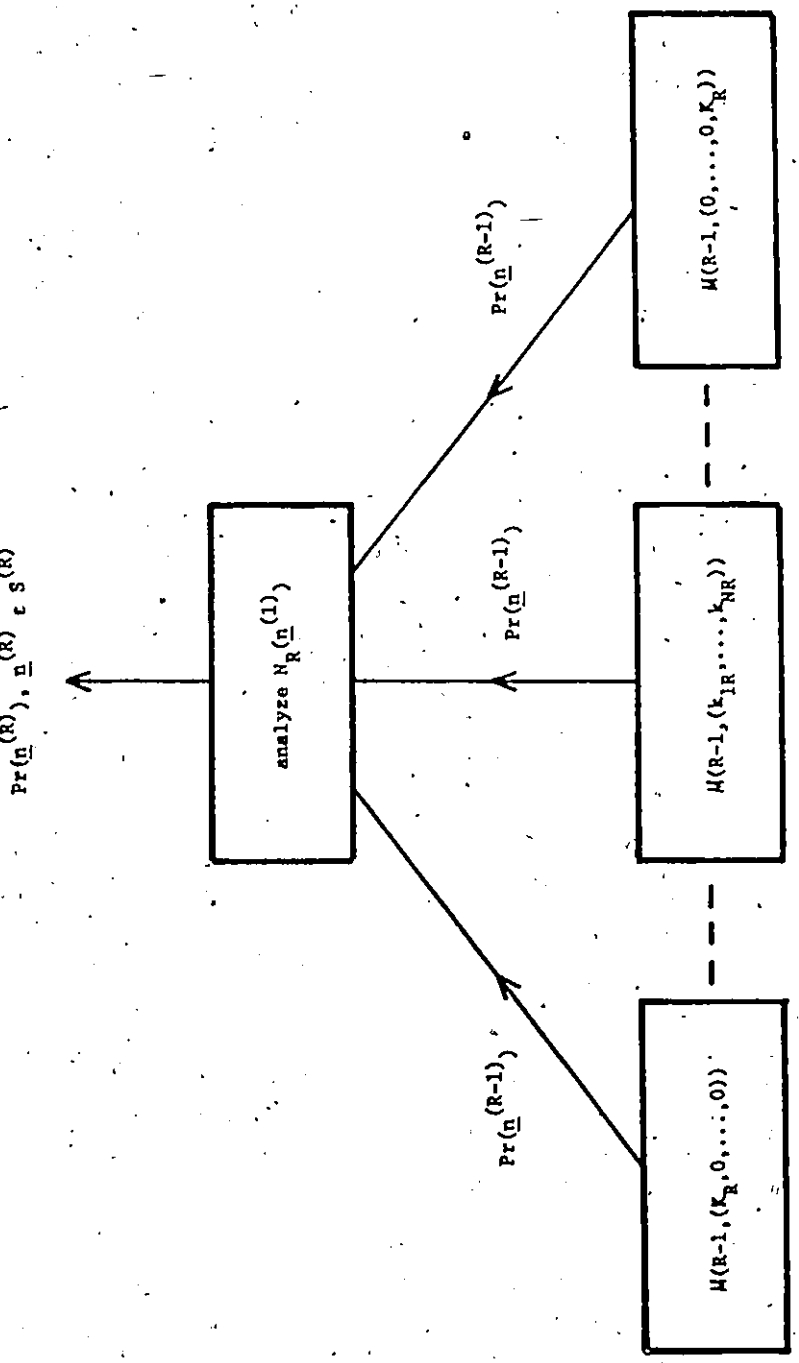

Figure 4.2 Queueing network representation of the single-level decomposition and aggregation procedure.

$$\Pr(\underline{n}^{(1)}=(k_{1R},\ldots,k_{NR})) = G^{-1}G_{R-1}(k_{1R},\ldots,k_{NR})\prod_{i=1}^{N} w_{iR}^{k_{iR}},\ \underline{n}^{(1)} \in L_R,$$

where $G_{R-1}(k_{1R},\ldots,k_{NR})$ is the normalization constant associated with the distribution for $M(R-1,(k_{1R},\ldots,k_{NR}))$ and

$$G = \sum_{(k_{1R},\ldots,k_{NR})\in L_R} G_{R-1}(k_{1R},\ldots,k_{NR}) \prod_{i=1}^{N} w_{iR}^{k_{iR}}.$$

Proof: The above result is part of a slightly more general one (Theorem 6.1) which is to be proved in Section 6.2.

4.7 PARAMETRIC ANALYSIS BY CHAIN

Having developed an exact decomposition and aggregation by chain procedure, a computationally efficient parametric analysis with respect to the routing of a particular type of customers may be made.

Consider the single-level exact decomposition and aggregation procedure summarized as follows and shown pictorially in Figure 4.2.

Step A: For all $(k_{1R},\ldots,k_{NR}) \in L_R$: Determine the distribution $\Pr(\underline{n}^{(R-1)})$, $\underline{n}^{(R-1)} \in S^{(R-1)}$, associated with $M(R-1,(k_{1R},\ldots,k_{NR}))$.

Step B: Find the probabilities $\Pr(\underline{n}^{(1)})$ for the system $N_R(\underline{n}^{(1)})$ using Theorem 4.3.

Step C: Compute the probabilities $\Pr(\underline{n}^{(R)})$ for the R-chain network as

$$\Pr(\underline{n}^{(R)}) = \Pr(\underline{n}^{(R-1)} \text{ in } M(R-1,(n_{1R},\ldots,n_{NR}))) \times \Pr((n_{1R},\ldots,n_{NR}) \text{ in } N_R(\underline{n}^{(1)})).$$

It is to be noticed that, in the above procedure, the values of e_{iR}, $1 \leq i \leq N$, only enter into steps B and C. Having, carried out Step A, an efficient parametric analysis with respect to the routing of chain R customers may, therefore, be made. The marginal distribution and the mean performance measures of interest with respect to chain R may be computed for various routing matrices without having to repetitively analyze the entire R-chain queueing network. The parametric analysis proceeds as described below and consists of two portions. The first, in which a set of networks $M(R-1,(k_1,\ldots,k_N))$ is analyzed and the second, in which $N_R(\underline{n}^{(1)})$ is analyzed for the various choices of $P^{(R)}$.

Preliminary analysis:

For all $(k_1,\ldots,k_N) \in L_R$:

For the network $M(R-1,(k_1,\ldots,k_N))$:

Using the convolution algorithm, obtain the normalization constant $G_{R-1}(k_1,\ldots,k_N)$ and the conditional (conditioned on $(k_1,\ldots,k_N)$, the distribution of the chain R customers) throughputs $T_{1r}(k_1,\ldots,k_N)$, $1 \leq r \leq R-1$, of the chain r customers at node 1. Store $G_{R-1}(k_1,\ldots,k_N)$. Compute the overall conditional throughput at node 1 as $T_1(k_1,\ldots,k_N) = \sum_{r=1}^{R-1} T_{1r}(k_1,\ldots,k_N)$. Compute the conditional throughput of the chain R customers at node 1 as

$$T_{1R}(k_1,\ldots,k_N) = \begin{cases} 0, & \text{if } k_1=0, \\ t_1^{-1}-T_1(k_1,\ldots,k_N), & \text{if } k_1 > 1. \end{cases}$$

Store $T_{1R}(k_1,\ldots,k_N)$ if $k_1 > 1$. (As described in Section 4.4, in the

network $M(R-1,(k_1,\ldots,k_N))$ the customers of chain R are fixed at the nodes in the distribution $(k_1,\ldots,k_N)$ and cannot depart).

Parametric analysis of chain R:

For each choice of $P^{(R)}$:

For all $(k_1,\ldots,k_N) \in L_R$:

Compute the state probability

$$\Pr(\underline{n}^{(1)} = (k_1,\ldots,k_N)) = G^{-1} G_{R-1}(k_1,\ldots,k_N) \prod_{i=1}^{N} w_{iR}^{k_i}.$$

Store $\Pr(\underline{n}^{(1)} = (k_1,\ldots,k_N))$.

For nodes i=1,2,...,N:

Compute the unconditional throughput, T_{iR}, of the chain R customers as

$$T_{iR} = \begin{cases} \sum_{\substack{(k_1,\ldots,k_N)\in L_R \\ k_1 \neq 0}} \Pr(\underline{n}^{(1)} = (k_1,\ldots,k_N))\, T_{1R}(k_1,\ldots,k_N), & \text{if } i=1, \\ T_{1R} e_{iR}/e_{1R}, & \text{if } i \neq 1. \end{cases}$$

Compute the unconditional mean queue length, Q_{iR}, of the chain R customers as

$$Q_{iR} = \sum_{\substack{(k_1,\ldots,k_N)\in L_R \\ k_i \neq 0}} k_i \Pr(\underline{n}^{(1)} = (k_1,\ldots,k_N)).$$

Compute the utilization, U_{iR}, and the mean waiting time, W_{iR}, of the

chain R customers as $U_{iR} = T_{iR}t_i$ and $W_{iR} = Q_{iR}/T_{iR}$.

In an implementation of the above procedure one needs a method to step through the elements of the set L_R. A simple way of doing this is illustrated in Appendix 4.1. An example parametric analysis of a simple store-and-forward network model, using the procedure described above, is given in Section 4.11.

4.8 COMPUTATIONAL COMPLEXITY OF THE PARAMETRIC ANALYSIS

In this Section, the time and space requirements of the parametric analysis procedure are derived and compared with the requirements of a straightforward repetitive analysis of the network using the convolution algorithm. (The computational requirements of the MVA algorithm are, in general, higher than those of convolution.)

Space requirement

The preliminary analysis requires the use of the convolution algorithm which has a space requirement, in number of array elements, equal to $2\prod_{r=1}^{R-1}(K_r+1)$. In addition, one requires $\binom{K_R+N-1}{N-1}$ elements to store $G_{R-1}(k_1,\ldots,k_N)$ for all $(k_1,\ldots,k_N)\in L_R$ and $\binom{K_R+N-2}{N-1}$ elements to store $T_{1R}(k_1,\ldots,k_N)$ for all $(k_1,\ldots,k_N)\in L_R$, $k_1\neq 0$. In the parametric analysis portion one requires $\binom{K_R+N-1}{N-1}$ elements to store the distribution $Pr(\underline{n}^{(1)})$. Hence, the total storage requirement is

$$2\prod_{r=1}^{R-1}(K_r+1) + 2\binom{K_R+N-1}{N-1} + \binom{K_R+N-2}{N-1}. \qquad (4.1)$$

A straightforward repetitive analysis of the network, using the convolution algorithm, would require a storage space, in number of array elements, equal to $2 \prod_{r=1}^{R} (K_r + 1)$.

Time requirement

Let us begin by evaluating the time requirement, in number of operations (additions and multiplications), of the preliminary analysis. The preliminary analysis consists of computing the normalization constants of the networks $M(R-1,(k_1,\ldots,k_N))$, $(k_1,\ldots,k_N) \in L_R$ using the convolution algorithm. Some of the elements of the vector $(k_1,\ldots,k_N)$ may be zero so that not all nodes in the network $M(R-1,(k_1,\ldots,k_N))$ have state-dependent servers. As a result, in the computation of $G_{R-1}(k_1,\ldots,k_N)$, for those nodes i where $k_i=0$ one may use the version of the convolution algorithm which applies to constant-speed servers (eq. 3.3) while for the other nodes one uses the version which applies to state-dependent servers (eq. 3.2).

Consider the set of vectors L_R. The number of vectors in L_R with i nonzero elements is $\binom{K_R-1}{i-1}\binom{N}{i}$. The number of operations used in the convolution algorithm for a constant-speed server in an (R-1)-chain network is (see eq. 3.4)

$$2(R-1) \prod_{r=1}^{R-1} (K_r+1)$$

while for a state-dependent server it is of the order of (see eq. 3.6)

$$\prod_{r=1}^{R-1} (K_r+1)(K_r+2)/2.$$

The time requirement of the preliminary analysis is, therefore, of the

order of (assuming that node i=1 is designated as a state-dependent node)

$$\sum_{i=1}^{\min\{K_R,N\}} \binom{K_R-1}{i-1}\binom{N}{i}\left[(i-1)\prod_{r=1}^{R-1}(K_r+1)(K_r+2)/2 + 2(N-i)(R-1)\prod_{r=1}^{R-1}(K_r+1)\right].$$

The above simplifies to

$$\binom{K_R+N-2}{N-2}\left[(K_R-1)\left[\prod_{r=1}^{R-1}(K_r+1)(K_r+2)/2\right] + 2N(R-1)\prod_{r=1}^{R-1}(K_r+1)\right]. \tag{4.2}$$

The throughputs $T_{1r}(k_1,\ldots,k_N)$, $1\leq r\leq R-1$, are readily obtained in the course of computing $G_{R-1}(k_1,\ldots,k_N)$ using the convolution algorithm (see eq. 3.7). The number of operations to obtain $T_{1R}(k_1,\ldots,k_N)$ will be ignored since it is relatively small.

The time requirement of each parametric analysis is proportional to the cardinality of the set L_R which is $\binom{K_R+N-1}{N-1}$. One requires of the order of $(K_R+3)\binom{K_R+N-1}{N-1}$ operations to obtain the entire marginal state distribution for chain R. The computation of the throughputs T_{iR}, $1\leq i\leq N$, then requires of the order of $2\binom{K_R+N-2}{N-1}$ operations and the computation of the mean queue lengths requires of the order of $2(N-1)\binom{K_R+N-2}{N-1}$ operations. The number of operations to obtain the utilizations U_{iR} and the waiting times W_{iR} will be ignored since it is relatively small. The time requirement of each parametric analysis is, therefore, of the order of

$$(K_R+3)\binom{K_R+N-1}{N-1} + 2N\binom{K_R+N-2}{N-1}. \qquad (4.3)$$

A straightforward repetitive analysis of the network, for each choice of $P^{(R)}$, would have a time requirement, in number of operations, equal to

$$2R(N-1)\prod_{r=1}^{R}(K_r+1). \qquad (4.4)$$

An advantage of the procedure developed here is, therefore, that the time requirement of each parametric analysis is independent of the number of chains R in the network. The disadvantage is, of course, the large time requirement of the preliminary analysis. It is to be noted, however, that if the routing of customers of chains 1,2,...,R-1 has sparsity or locality properties, then the analysis of the networks $M(R-1,(k_1,\ldots,k_N))$ may be made using the Tree Convolution algorithm [32] in which case the space and time requirements of the preliminary analysis would be less than those given by eq. 4.1 and eq. 4.2, respectively, and the time requirement of each parametric analysis would be unchanged from eq. 4.3.

4.9 MULTIPLE-LEVEL DECOMPOSITION BY CHAIN

The single-level decomposition of Section 4.4 is extended readily into an (R-1)-level decomposition for an R-chain PS network. With $S(\prod_{i=R}^{d}(k_{1i},\ldots,k_{Ni})) = \{\underline{n}^{(R)}: \underline{n}^{(R)} \in S^{(R)}; n_{1s}=k_{1s},\ldots,n_{Ns}=k_{Ns}$ for $d<s<R\}$ the multiple-level decomposition is conveniently presented in

the form of a tree as in Figure 4.3.

Let $Q(\prod_{i=R-d+1}^{1} (\alpha_i, \alpha_i))$ be associated with $S(\prod_{i=R}^{d} (k_{1i}, \ldots, k_{Ni}))$. This submatrix is, except for the diagonal elements, identical to the infinitesimal generator of a (d-1)-chain PS network with state-space $S^{(d-1)}$ whose servers have state-dependent service rates $\mu_i(a_i^{(d-1)}) = a_i^{(d-1)}/(a_i^{(d-1)} + \sum_{r=d}^{R} k_{ir})$. Such a system is denoted by

$$M(d-1, (\sum_{r=d}^{R} k_{1r}, \ldots, \sum_{r=d}^{R} k_{Nr})).$$

The submatrix $Q((\alpha,\alpha) \prod_{i=R-d+1}^{1} (\alpha_i, \alpha_i))$ associated with $S(\prod_{i=R}^{d-1} (k_{1i}, \ldots, k_{Ni}))$ is, except for the diagonal elements, identical to the infinitesimal generator of a (d-2)-chain PS network with state space $S^{(d-2)}$, whose servers have state-dependent service rates $\mu_i(a_i^{(d-2)}) = a_i^{(d-2)}/(a_i^{(d-2)} + \sum_{r=d-1}^{R} k_{ir})$ and is denoted by $M(d-2, (\sum_{r=d-1}^{R} k_{1r}, \ldots; \sum_{r=d-1}^{R} k_{Nr}))$. Hence Q is decomposable into a hierarchy of subsystems associated with subsets of the routing chains.

Theorem 4.4: With the decomposition of Q according to Figure 4.3, exact results are obtained by the multiple-level decomposition and aggregation procedure.

Proof: The proof utilizes the same line of reasoning as that for Theorem 4.2.

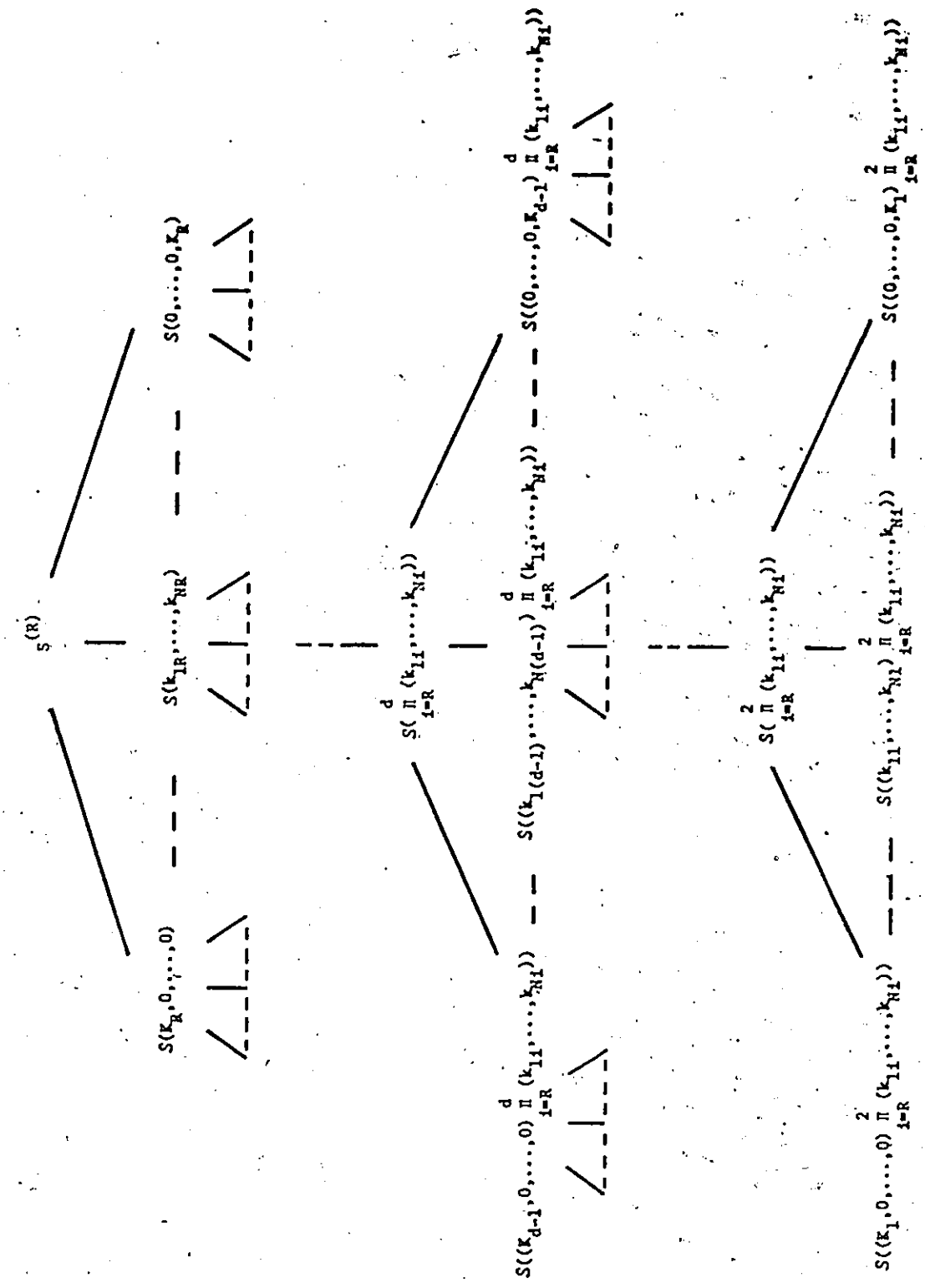

Figure 4.3 Multiple-level decomposition of the PS network

4.10 MULTIPLE-LEVEL AGGREGATION BY CHAIN

Using an argument like that in Section 4.5 it is seen that the analysis of the reduced system $A(\prod_{i=R-d+1}^{1} \alpha_i)$ can also be interpreted as the analysis of a single-chain queueing network of the type defined by $N_{d-1}(\underline{n}^{(1)})$. The procedure may then be viewed as the solution to a tree of single-chain queueing network problems, as depicted in Figure 2.3, where each box is now to be interpreted as the analysis of a queueing network of type $N(\underline{n}^{(1)})$.

Consider the analysis of a single portion of the tree, as given by Figure 4.4. This may be represented in terms of a set of queueing network problems, as shown in Figure 4.5, in light of Section 4.9. In Figure 4.5, the state distribution for $N_{d-1}(\underline{n}^{(1)})$ is

$$\Pr(\underline{n}^{(1)}=(n_{1(d-1)},\cdots,n_{N(d-1)})) =$$

$$G^{-1}G_{d-2}(n_{1(d-1)}+\sum_{r=d}^{R}k_{1r},\cdots,n_{N(d-1)}+\sum_{r=d}^{R}k_{Nr})\cdot$$

$$\cdot\prod_{i=1}^{N}\frac{(n_{i(d-1)}+\sum_{r=d}^{R}k_{ir})!}{n_{i(d-1)}!(\sum_{r=d}^{R}k_{ir})!}(w_{i(d-1)})^{n_{i(d-1)}},$$

where G is a normalization constant. This can be derived in a similar manner to that for the result of Theorem 4.3. Finally, in Figure 4.5

$$\Pr(\underline{n}^{(d-1)} = ((n_{11},\cdots,n_{1(d-1)})\cdots(n_{N1},\cdots,n_{N(d-1)}))) =$$

$$\Pr(\underline{n}^{(d-2)} \text{ in } M(d-2),(n_{1(d-1)}+\sum_{i=d}^{R}k_{1i},\cdots,n_{N(d-1)}+\sum_{i=d}^{R}k_{Ni})))$$

$$\times\Pr((n_{1(d-1)},\cdots,n_{N(d-1)}) \text{ in } N_{d-1}(\underline{n}^{(1)})).$$

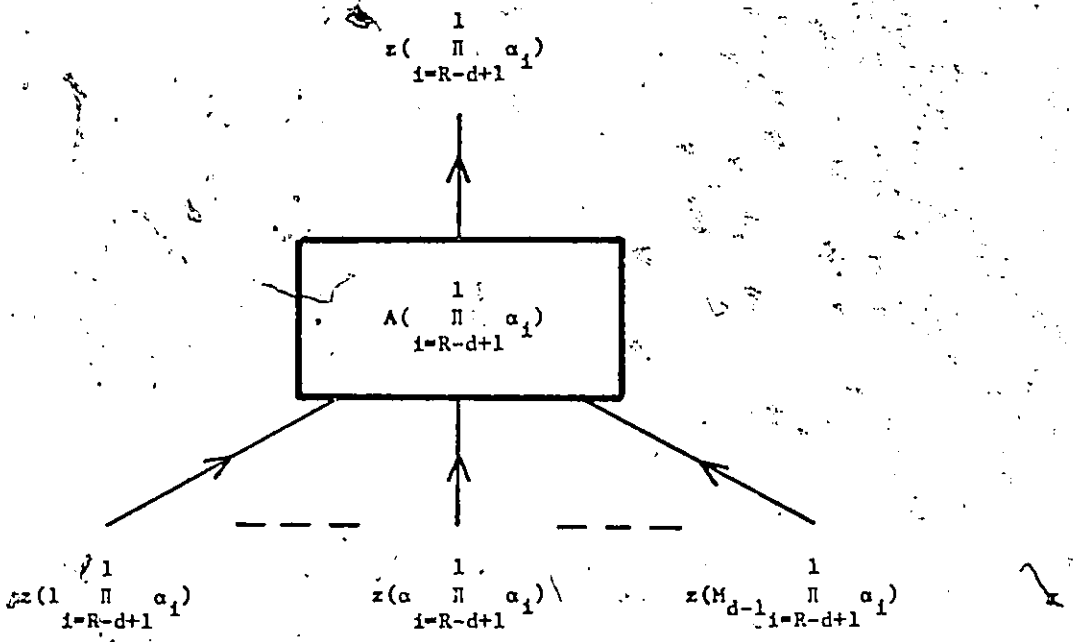

Figure 4.4 Portion of the solution tree for multiple-level decomposition and aggregation

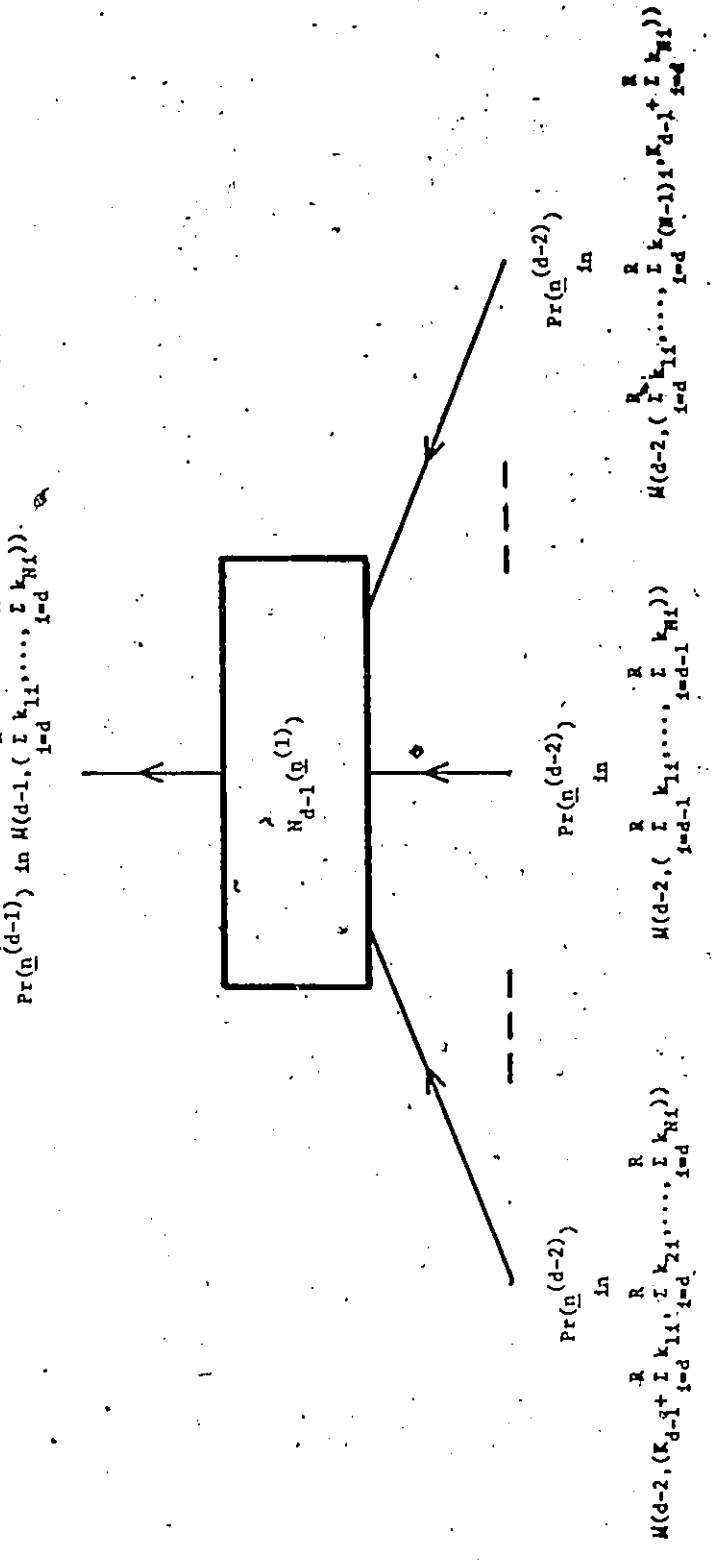

Figure 4.5 Queueing network representation of Figure 4.4

4.11 EXAMPLE OF PARAMETRIC ANALYSIS

This Section illustrates the parametric analysis of a window flow-controlled store-and-forward network model with respect to the routing of a particular type of customers. The model is illustrated in Figure 4.6 and is similar to one appearing in [48]. There are 4 virtual channels and 4 half duplex links. The parameters for the network considered here are given in Table 4.1. The routing of messages belonging to chains 1, 2, and 3 is deterministic while the routing of the chain 4 messages is random. Messages belonging to chain 4 are routed through links 4 and 1 with probability p and through links 3 and 2 with probability (1-p).

The set L_R consists of the different ways the 2 messages of chain 4 may be fixed at nodes 1, 2, 3, 4, and 8 (chain 4 messages do not visit queues 5, 6 or 7). The result of the preliminary analysis is given in Table 4.2. The time requirement of the initial analysis is, using eq. 4.2, of the order of 42,336 operations. The result of a parametric analysis, with p=0.2, is given in Table 4.3. The time requirement of each parametric analysis is, according to eq. 4.3, of the order of 125 operations. A straightforward analysis using the convolution algorithm would require, according to eq. 4.4, of the order of 4536 operations.

4.12 EXAMPLE OF DECOMPOSITION BY CHAIN

This Section illustrates a single-level decomposition of the generator matrix for a queueing network.

Consider a FCFS network with 4 nodes, 2 closed routing chains and one customer in each chain. The infinitesimal generator for the equivalent PS network is shown in Figure 4.7. In the Figure, $u_i = t_i^{-1}$. The single-level decomposition by chain is indicated by the bold lines. The state vector is

$$(n_{11}, n_{21}, n_{31}, n_{41}, n_{12}, n_{22}, n_{32}, n_{42}).$$

APPENDIX 4.1

The elements of the set L_R may be generated, in an efficient manner, as described below in FORTRAN (with node = N, popr = K_R, k( ) = $\underline{k}$).

```
(initialization)      do 23 i=1,node
                      k(i)=0
                   23 continue
                      k(1)=popr
                      go to 3

(generation of L_R)   99 nz=0
                      do 1 i=1,node
                      if (k(i).gt.0) then
                      nz=i
                      go to 2
                      end if
                    1 continue
                    2 k(nz+1)=k(nz+1)+1
                      k(1)=k(nz)-1
                      if (nz.gt.1) k(nz)=0
                    3 continue

                      [PROGRAM STATEMENTS]

                      go to 99
```

Table 4.1 Parameters of the example network.

node, i	Visit ratios e_{ir}				mean service time t_i
	chain 1 r=1	chain 2 r=2	chain 3 r=3	chain 4 r=4	
1	1	0	1	p	2
2	1	1	1	(1-p)	1
3	1	1	0	(1-p)	1
4	0	1	0	p	2
5	1	0	0	0	1
6	0	1	0	0	2
7	0	0	1	0	2
8	0	0	0	1	4

All chain populations equal to 2.

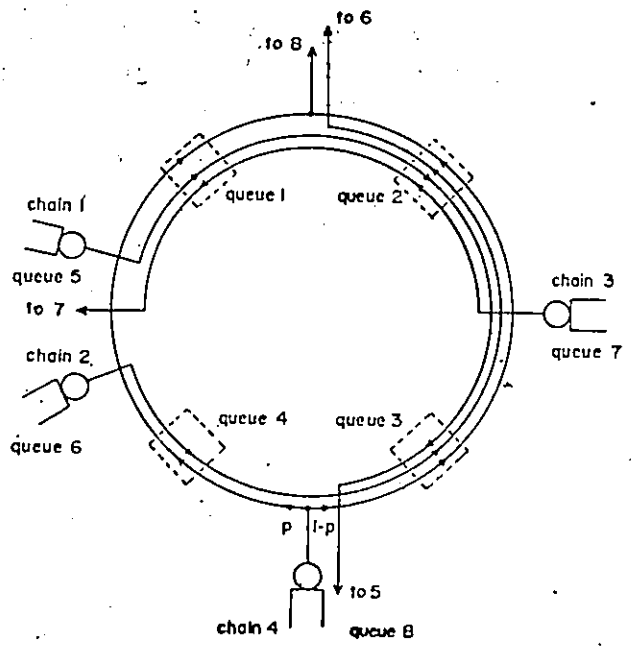

Figure 4.6 Example store-and-forward network model. The dotted boxes represent FCFS queues which model half duplex links. Queues 5 to 8 are source/sinks. The chains stand for virtual channels.

Table 4.2 Results of the preliminary analysis.

$(k_1,\ldots,k_8)\epsilon L_2$	$G_1(k_1,\ldots,k_8)$	$T_1(k_1,\ldots,k_8)$	$T_{1R}(k_1,\ldots,k_8)$
20000000	107,880	.2799	.2201
11000000	97,732	.3073	.1927
02000000	78,986	.3470	0
10100000	74,676	.3250	.175
01100000	58,181	.3709	0
00200000	40,679	.4013	0
10010000	76,952	.3362	.1638
01010000	56,172	.3863	0
00110000	39,516	.4150	0
00020000	36,712	.4305	0
10000001	47,488	.3334	.1666
01000001	37,660	.3813	0
00100001	26,140	.4111	0
00010001	25,216	.4276	0
00000002	15,820	.4230	0

Table 4.3 Results of a parametric analysis.

$(n_1,\ldots,n_8)$	$\prod_{i=1}^{8} (e_{i4}\, t_i)^{n_i}$	$G_1(n_1,\ldots,n_8) \prod_{i=1}^{8} (e_{i4}\, t_i)^{n_i}$	$Pr(\underline{n}^{[1]}=(n_1,\ldots,n_8))$
20000000	.16	17,260.8	.0213
11000000	.32	31,274.2	.0387
02000000	.64	50,551.04	.0625
10100000	.32	23,896.32	.0296
01100000	.64	37,235.84	.0460
00200000	.64	26,034.56	.0322
10010000	.16	12,312.32	.0152
01010000	.32	17,975.04	.0222
00110000	.32	12,645.12	.0156
00020000	.16	5,873.92	.0073
10000001	1.6	75,980.8	.0940
01000001	3.2	120,512	.1490
00100001	3.2	83,648	.1034
00010001	1.6	40,345.6	.0500
00000002	16	253,120	.3130

Mean performance measures for chain 4 (with p=0.2):

T_{84}=.177, Q_{84}=1.022,

T_{14}=T_{44}=.0355, Q_{14}=.220, Q_{44}=.118,

T_{24}=T_{34}=.1419, Q_{24}=.381, Q_{34}=.259.

Figure 4.7 Infinitesimal generator for example queueing network.

CHAPTER 5 - A NEW METHOD FOR COMPUTING THE NORMALIZATION CONSTANT

5.1 INTRODUCTION

In Section 2.4 we have seen that $B^{(R)}(n,\underline{k})$ can be decomposed into the set of networks $\{B^{(R)}(n-1,\underline{x}): \underline{0}\leq\underline{x}\leq\underline{k}\}$. The relationship between the normalization constants of these networks is provided by eq. 3.1. This relationship is the foundation of the convolution algorithm. In the previous Chapter, we have seen that $B^{(R)}(N,\underline{K})$ can be decomposed into a hierarchy of systems of the type $M(r,(k_1,\ldots,k_N))$. In Figure 4.5, we see that the network

$$M(d-1,(\sum_{i=d}^{R} k_{1i},\ldots,\sum_{i=d}^{R} k_{Ni}))$$

can be decomposed into the set of networks

$$\{M(d-2,(k_{1(d-1)}+\sum_{i=d}^{R} k_{1i},\ldots,k_{N(d-1)}+\sum_{i=d}^{R} k_{Ni})):$$

$$(k_{1(d-1)},\ldots,k_{N(d-1)}) \in L_{d-1}\}.$$

It is then fairly natural to ask what the relationship between the normalization constants of these networks is, and whether such a relationship might be helpful in computing the normalization constant.

In this Chapter, we establish the desired relationship formally and show how it defines a recursive expression for computing the normalization constant G. We evaluate the computational complexity to obtain G, using this new recursion, and show that it is polynomial in the number of closed routing chains R. When there are many chains in a network, this new method for computing G is substantially more efficient than the convolution algorithm, whose complexity in R is exponential. Our new algorithm, therefore, extends the range of

queueing networks whose normalization constants can be evaluated by efficient means.

In the following Section, we first make certain general remarks concerning the construction of algorithms to compute normalization constants. These remarks will serve to situate the convolution algorithm and our new algorithm together within a general framework.

5.2 ON THE DESIGN OF COMPUTATIONAL ALGORITHMS FOR NORMALIZATION CONSTANTS

It has been observed in Section 4.2 that we may apply almost arbitrary decompositions to $B^{(R)}(N,\underline{K})$ and obtain exact results by the decomposition and aggregation procedure if the necessary transformations are made so that reversibility will hold. Suppose that these transformations have been made and we decompose the generator Q, associated with $B^{(R)}(N,\underline{K})$, according to an arbitrary multiple-level decomposition, as illustrated in Fig. 2.2. The normalization constant for Q can be related to the normalization constants of the subsystems Q^*_α, $1 \leq \alpha \leq M_1$. Furthermore, the normalization constants of these subsystems can, in turn, be related to those of the subsystems at the next higher level in the decomposition, and so on. In this way, we may view the normalization constant for Q as being defined by an L-level tree of normalization constants. The number of subsystems at the ℓth level of the tree is $M_1 M_2 \ldots M_\ell$. However, it is quite possible, with the decomposition we have adopted, that a number of the subsystems at a particular level are identical to one another, so that the number of distinguishable subsystems D_ℓ, at the ℓth level, may be considerably

less than $M_1 M_2 \ldots M_\ell$. To minimize the number of operations for computing the normalization constant of Q we should, therefore, decompose Q in such way that $\sum_{\ell=1}^{L} D_\ell$ is minimized. To minimize the storage space requirement we should attempt to minimize $\underset{1\leq\ell\leq L}{\text{Max}} \{D_\ell\}$.

In conclusion, there are many ways to compute the normalization constant. By clever decomposition of $B^{(R)}(N,\underline{K})$ we can construct efficient computational algorithms. In the decomposition of Courtois, $D_\ell = \prod_{r=1}^{R} (K_r+1)$ for each level ℓ, hence the term $\prod_{r=1}^{R} (K_r+1)$ in the storage space requirement of the convolution algorithm (eq. 3.5). If $B^{(R)}(N,\underline{K})$ is decomposed by chain, as described in Section 4.9, then $D_\ell = \binom{\sum_{s=R-\ell+1}^{R} K_s+N-1}{N-1}$ $1\leq\ell\leq R$, this being the cardinality of the set

$$\{(C_1,\ldots,C_N): \quad C_i = \sum_{s=R-\ell+1}^{R} k_{is} \text{ for } 1\leq i\leq N; \quad \sum_{i=1}^{N} k_{is} = K_s \text{ for } (R-\ell+1)\leq s\leq R; \; k_{is} > 0\}.$$

The expression $\binom{R+N-1}{N-1}$ can be written as a polynomial in R. This explains informally why our new method, for computing the normalization constant, has a polynomial complexity in R. It is interesting to speculate as to the existence of other useful computational procedures.

5.3 PRELIMINARY RESULTS

In this Section, we establish formally the new recursive expression for computing the normalization constant of $B^{(R)}(N,\underline{K})$. As in the previous Chapter, $G_r(\underline{c})$, where $\underline{c} = (c_1,\ldots,c_N)$, is used to denote the normalization constant for $M(r,(c_1,\ldots,c_N))$. In Chapter 4,

it was assumed that the queueing discipline at all the nodes is PS so that we could take advantage of the notion of reversibility. In the following, however, the queueing disciplines may be any among FCFS, LCFSPR, PS or IS. In this situation, the system $M(r,(c_1,\ldots c_N))$ is to be interpreted as a system of the type $B^{(r)}(N,(K_1,\ldots,K_r))$ but with state-dependent servers having service rate functions of the form $\mu_i(n) = n/(n+c_i)$, if node i is FCFS, LCFSPR or PS. If node i is IS, then $\mu_i(n) = 1$, irrespective of the value of c_i. The physical interpretation is that there are c_i customers that cycle around continuously at node i and consume, on the average, a fraction $c_i/(c_i + n)$ of the server capacity when the population of the freely circulating customers at node i is n. At IS centers, $\mu_i(n) = 1$ since we have an unlimited number of servers to begin with.

The quantity we are interested in computing is, clearly, $G = G_R(\underline{0})$ since if $\underline{c} = \underline{0}$, then $\mu_i(n) = 1$ for $1 \leq i \leq N$. We reiterate that at FCFS centers it is required that $t_{ir} = t_i$ for $1 \leq r \leq R$.

Theorem 5.1:

$G_R(\underline{0})$ is given recursively by

$$G_r(\underline{v}_r) = \sum_{\underline{\ell} \in L_r} g_r(\underline{v}_r,\underline{\ell}) G_{r-1}(\underline{v}_r+\underline{\ell}), \text{ for } 1 \leq r \leq R, \ \underline{v}_r \in I_r, \quad (5.1)$$

where

$$\underline{v}_r = (v_{1r},\ldots,v_{Nr}),$$

$$\underline{\ell} = (\ell_1,\ldots,\ell_N),$$

$$I_r = \begin{cases} \{\underline{v}_r: \ v_{ir} > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} v_{ir} = \sum_{s=r+1}^{R} K_s\}, & \text{if } 0 \leq r \leq R-1, \\ \{\underline{0}\}, & \text{if } r = R, \end{cases}$$

$$L_r = \{\underline{\ell}: \ \ell_i > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} \ell_i = K_r\},$$

$$g_r(\underline{v}_r, \underline{\ell}) = \prod_{i=1}^{N} h_{ir}(\underline{v}_r, \underline{\ell}),$$

$$h_{ir}(\underline{v}_r, \underline{\ell}) = \begin{cases} \binom{\ell_i + v_{ir}}{v_{ir}} w_{ir}^{\ell_i}, & \text{if center } i \text{ is FCFS, LCFSPR or PS,} \\ w_{ir}^{\ell_i} / \ell_i!, & \text{if center } i \text{ is IS} \end{cases}$$

and the initial conditions are $G_0(\underline{v}_0) = 1$ for all $\underline{v}_0 \in I_0$.

Proof:

$G_r(\underline{c})$ is the normalization constant of a queueing network of type $B^{(r)}(N,(K_1,\ldots,K_r))$ but with state-dependent service rate functions $\mu_i(n) = n/(n+c_i)$ at those centers which do not have an IS discipline. Let the centers be enumerated so that centers $1,\ldots,p$ are the ones with a FCFS, LCFSPR or PS discipline and $p+1,\ldots,N$ are the IS centers. The aggregate system state distribution for this system with state dependent service rate functions is, according to eq. 1.3,

$$\Pr(\underline{n}^{(r)}) = G_r(\underline{c})^{-1} \prod_{i=1}^{p} \left(\left\{ \prod_{a=1}^{n_i^{(r)}} (a+c_i) \right\} \prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right) \prod_{i=p+1}^{N} \left(\prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right)$$

where, by definition,

$$G_r(\underline{c}) = \sum_{\underline{n}^{(r)} \in S^{(r)}} \prod_{i=1}^{p} \Big(\prod_{a=1}^{n_i^{(r)}} (a+c_i) \Big) \prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \prod_{i=p+1}^{N} \Big(\prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \Big).$$

But

$$S^{(r)} = \bigcup_{\underline{\ell} \in L_r} S^{(r)}(\underline{\ell})$$

where L_r is defined in Theorem 5.1 and

$$S^{(r)}(\underline{\ell}) = \{\underline{n}^{(r)}: \ \underline{n}^{(r)} \in S^{(r)}; n_{ir} = \ell_i \text{ for } 1 \leq i \leq N\}$$

so that

$$G_r(\underline{c}) = \sum_{\underline{\ell} \in L_r} \sum_{\underline{n}^{(r)} \in S^{(r)}(\underline{\ell})} \prod_{i=1}^{p} (\cdot) \prod_{i=p+1}^{N} (\cdot).$$

Now $\prod_{a=1}^{n_i^{(r-1)}+\ell_i} (a+c_i) = (1+c_i)\ldots(\ell_i+c_i)(\ell_i+c_i+1)\ldots(n_i^{(r-1)}+\ell_i+c_i)$

$$= \frac{(\ell_i+c_i)!}{c_i!} \prod_{a=1}^{n_i^{(r-1)}} (a+\ell_i+c_i).$$

Hence

$$G_r(\underline{c}) = \sum_{\underline{\ell} \in L_r} \prod_{i=1}^{p} ((\ell_i+c_i)!\, w_{ir}^{\ell_i}/c_i!\,\ell_i!) \prod_{i=p+1}^{N} (w_{ir}^{\ell_i}/\ell_i!).$$

$$\cdot \sum_{\underline{n}^{(r)} \in S^{(r)}(\underline{\ell})} \prod_{i=1}^{p} \left(\prod_{a=1}^{n_i^{(r-1)}} (a+\ell_i+c_i) \right) \prod_{s=1}^{r-1} w_{is}^{n_{is}}/n_{is}!) \cdot$$

$$\cdot \prod_{i=p+1}^{N} \left(\prod_{s=1}^{r-1} w_{is}^{n_{is}}/n_{is}! \right).$$

We now see that the sum on the far right is the definition of $G_{r-1}(\underline{c}+\underline{\ell})$ itself, so that

$$G_r(\underline{c}) = \sum_{\underline{\ell} \in L_r} \left(\prod_{i=1}^{p} \binom{\ell_i+c_i}{c_i} w_{ir}^{\ell_i} \prod_{i=p+1}^{N} w_{ir}^{\ell_i}/\ell_i! \right) G_{r-1}(\underline{c}+\underline{\ell}).$$

The initial conditions, $G_0(\underline{c})$, are normalization constants of networks that contain no customers. The only network state is the empty state and hence $G_0(\underline{c}) = 1$.

The above expressions are valid for any $\underline{c} \in C$ where $C = \{\underline{c}: \ c_i > 0 \text{ for } 1 \leq i \leq N\}$. They are therefore valid for $\underline{c} = \underline{v}_r$, where $\underline{v}_r \in I_r$, since $I_r \subset C$. Hence,

$$G_r(\underline{v}_r) = \sum_{\underline{\ell} \in L_r} \left(\prod_{i=1}^{N} h_{ir}(\underline{v}_r, \underline{\ell}) \right) G_{r-1}(\underline{v}_r+\underline{\ell})$$

where $h_{ir}(\underline{v}_r, \underline{\ell})$ is defined in Theorem 5.1.

It now remains to be shown that the domain of $G_r(\underline{v}_r)$, in the computation of $G_R(\underline{0})$, is:

$$I_r = \begin{cases} \{\underline{v}_r: \ v_{ir} > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} v_{ir} = \sum_{s=r+1}^{R} K_s\}, & \text{for } 0 \leq r \leq R-1; \\ \{\underline{0}\}, & \text{for } r=R. \end{cases} \tag{5.2}$$

We show this by induction on r.

Clearly, $I_R = \{\underline{0}\}$ since we only wish to determine $G_R(\underline{v}_R)$ for $\underline{v}_R = \underline{0}$. Now

$$I_{R-1} = \bigcup_{\underline{v}_R \in I_R} \left\{ \bigcup_{\underline{\ell} \in L_R} \{\underline{v}_R + \underline{\ell}\} \right\} = \bigcup_{\underline{\ell} \in L_R} \{\underline{\ell}\} = L_R$$

so that eq. 5.2 is true for r=R-1.

We now assume that eq. 5.2 is true for r=t and show that it is true for r=t-1. We have

$$I_{t-1} = \bigcup_{\underline{v}_t \in I_t} \left\{ \bigcup_{\underline{\ell} \in L_t} \{\underline{v}_t + \underline{\ell}\} \right\} = \bigcup_{\underline{\ell} \in L_t} \left\{ \bigcup_{\underline{v}_t \in I_t} \{\underline{v}_t + \underline{\ell}\} \right\}$$

$$= \bigcup_{\underline{\ell} \in L_t} \left\{ (\underline{v}_t + \underline{\ell}): \ v_{it} > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} v_{it} = \sum_{s=t+1}^{R} K_s \right\}$$

Now $\underline{\ell} \in L_t$ so that $\ell_i > 0$ for $1 \leq i \leq N$ and $\sum_{i=1}^{N} \ell_i = K_t$. Hence,

$$I_{t-1} = \{(\underline{v}_t + \underline{\ell}): \ v_{it} > 0 \text{ for } 1 \leq i \leq N; \ \ell_i > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} v_{it} = \sum_{s=t+1}^{R} K_s; \ \sum_{i=1}^{N} \ell_i = K_t\}.$$

$$= \{(\underline{v}_t + \underline{\ell}): \ (v_{it} + \ell_i) > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} (v_{it} + \ell_i) = \sum_{s=t}^{R} K_s\}.$$

Defining a new variable $\underline{v}_{t-1} = (\underline{v}_t + \underline{\ell})$, we see that this agrees in form with eq. 5.2.

Theorem 5.1 gives a new recursive expression for computing the normalization constant $G = G_R(\underline{0})$. In effect, the normalization constant $G_r(\underline{v}_r)$ of a network with r chains is decomposed into a sum of normalization constants for networks having (r-1) chains and in which the customers of chain r are fixed at the nodes and cannot depart. The number of networks involved in this sum is the number of distinct ways K_r customers of chain r may be distributed over N nodes.

5.4 A NEW METHOD TO COMPUTE G

The new algorithm for computing G consists of evaluating $G = G_R(\underline{0})$ using an efficient formulation of the recursion given in Theorem 5.1. Rather than taking $G_0(\underline{v}_0) = 1$ for all $\underline{v}_0 \in I_0$ as initial conditions, it is more efficient for the initial conditions to be $G_1(\underline{v}_1)$, where $\underline{v}_1 \in I_1$. These are normalization constants of single-chain product-form queueing networks with state-dependent servers and can be computed efficiently using the version of the convolution algorithm which applies to single-chain networks with state-dependent service rates (see eq. 3.2 with R=1). The term $g_r(\underline{v}_r,\underline{\ell})$ can itself be computed efficiently in a recursive manner using a recursive expression which relates $g_r(\underline{v}_r,\underline{\ell})$ to $g_r(\underline{v}_r-\underline{1}_w+\underline{1}_x,\underline{\ell}-\underline{1}_y+\underline{1}_z)$ where $\underline{1}_i$ is a unit vector pointing in the ith direction, $1 \leq w,x,y,z \leq N$, and the initial condition is $g_r(\underline{v}'_r,\underline{\ell}'_r)$ where $(\underline{v}'_r,\underline{\ell}'_r)$ is some pair of vectors such that $\underline{v}'_r \in I_r$ and $\underline{\ell}'_r \in L_r$. Such a recursive computation of $g_r(\underline{v}_r,\underline{\ell})$ may be done so long as we step through the elements of I_r and L_r in such a way that the previous pair of vectors, denoted by $(\underline{v}^P_r,\underline{\ell}^P)$, can always be written in terms of the current pair $(\underline{v}_r,\underline{\ell})$ as $(\underline{v}^P_r,\underline{\ell}^P)=(\underline{v}_r-\underline{1}_w+\underline{1}_x,\underline{\ell}-\underline{1}_y+\underline{1}_z)$, for

some w, x, y and z such that $1 \leq w,x,y,z \leq N$. A way of doing this is illustrated in Appendix 5.1. The recursive expression relating $g_r(\underline{v}_r, \underline{\ell})$ to $g_r(\underline{v}_r^P, \underline{\ell}^P)$ is formally $g_r(\underline{v}_r, \underline{\ell}) = \psi(\underline{v}_r, \underline{\ell}, w, x, y, z) g_r(\underline{v}_r^P, \underline{\ell}^P)$ where $\psi(\underline{v}_r, \underline{\ell}, w, x, y, z)$ is the product-form expression obtained after algebraic simplification of the expression

$$(g_r(\underline{v}_r, \underline{\ell}) / g_r(\underline{v}_r - \underline{1}_w + \underline{1}_x, \underline{\ell} - \underline{1}_y + \underline{1}_z)).$$

We conclude this Section with a 'programme-like' specification of the new algorithm.

A new method for computing G:

```
{
 (initialization)
 for(all v_1 ε I_1)
  {compute G_1(v_1) using the single-chain
   convolution algorithm}
 for (r=2,...,R)
  { compute g_r(v_r', ℓ_r') }
 (main recursion)
 for (r=2,...,R)
  {
   v_r^P = v_r';
   ℓ^P = ℓ_r';
   for (all v_r ε I_r)
    {
     sum=0;
     for (all ℓ ε L_r)
      {
```

$g_r(\underline{v}_r,\underline{\ell})=\psi(\underline{v}_r,\underline{\ell},w,x,y,z)g_r(\underline{v}_r^P,\underline{\ell}^P)$;

sum=sum+$g_r(\underline{v}_r,\underline{\ell})\times G_{r-1}(\underline{v}_r+\underline{\ell})$;

$\underline{\ell}^P=\underline{\ell}$;

}

$G_r(\underline{v}_r)$=sum;

$\underline{v}_r^P=\underline{v}_r$;

}

}

$G=G_R(\underline{0})$;

}

In the following Section we determine the time and space requirements of this algorithm.

5.5 COMPUTATIONAL COMPLEXITY

In the following we will restrict our attention to the case where there are no IS centers in the network. When there are IS centers, however, the computational requirements are reduced slightly since the expression $\psi(\underline{v}_r,\underline{\ell},w,x,y,z)$ will involve fewer operations. We will determine the number of operations (additions, divisions, multiplications) and the storage (number of array elements) necessary to obtain G.

Let us begin by evaluating the number of operations which are necessary to obtain $G_1(\underline{v}_1)$ for all $\underline{v}_1 \in I_1$. The evaluation of $G_1(\underline{v}_1)$ requires the use of the convolution

$$C_m(x) = \sum_{k=0}^{x} f_m(k)C_{m-1}(x-k) \qquad (5.3)$$

for $2 \le m \le N$, where $C_N(K_1)=G_1(\underline{v}_1)$, $f_m(k)=w_{m1}^k / \prod_{a=1}^{k} u_m(a)$, $u_m(a)=a/(a+v_{m1})$ and the initial conditions are $C_1(k)=f_1(k)$ for $1 \le k \le K_1$. The quantity $f_m(k)$ can itself be computed recursively as

$$f_m(k)=f_m(k-1)w_{m1}(k+v_{m1})/k \qquad (5.4)$$

with initial condition $f_m(0)=1$.

The evaluation of a single step of eq. 5.4 requires 4 operations. The number of operations to evaluate eq. 5.3 for $x=K_1$ is therefore

$$\sum_{x=0}^{K_1}(6x+5)=3K_1^2+8K_1+5.$$

The number of operations to obtain $C_N(K_1)$ is therefore $(N-1)(3K_1^2+8K_1+5)$. Hence, the number of operations to compute $G_1(\underline{v}_1)$ for all $\underline{v}_1 \in I_1$ is

$$(N-1)(3K_1^2+8K_1+5)\binom{\sum_{r=2}^{R} K_r+N-1}{N-1};$$

the combinatorial term being the cardinality of the set I_1. We will ignore the number of operations necessary to obtain $g_r(\underline{v}_r',\underline{\ell}_r')$, for $2 \le r \le R$, since this is a small part of the initialization procedure.

In the determination of the number of operations necessary to obtain $G_R(\underline{0})$ we will need the number of operations used to update $g_r(\underline{v}_r,\underline{\ell})$ whenever $\underline{v}_r$ or $\underline{\ell}$ is altered. We see in Appendix 5.1 that only one of $\underline{v}_r$ or $\underline{\ell}$ need be changed at a time. When $\underline{\ell}$ is changed and $\underline{v}_r$ is fixed the next value of $g_r(\underline{v}_r,\underline{\ell})$ may be obtained using the expression

$$g_r(\underline{v}_r,\underline{\ell})=g_r(\underline{v}_r,\underline{\ell}-\underline{1}_y+\underline{1}_z)\, w_{yr}(\ell_y+v_{yr})(\ell_z+1)/\ell_y w_{zr}(\ell_z+v_{zr}+1)$$

Assuming that the ratio w_{yr}/w_{zr} has been precomputed and that the quantity (ℓ_z+1) is available as part of the control of the programme, the next value for $g_r(\underline{v}_r,\underline{\ell})$ may be obtained using 2 additions and 5 multiplications (divisions).

When $\underline{v}_r$ is changed and $\underline{\ell}$ is fixed the next value for $g_r(\underline{v}_r,\underline{\ell})$ may be obtained using the recursive expression

$$g_r(\underline{v}_r,\underline{\ell})=g_r(\underline{v}_r-\underline{1}_w+\underline{1}_x,\underline{\ell})(\ell_w+v_{wr})(v_{xr}+1)/(\ell_x+v_{xr}+1)v_{wr}.$$

Assuming that the quantity $(v_{xr}+1)$ is available as part of the control of the programme, the next value for $g_r(\underline{v}_r,\underline{\ell})$ may be obtained using 2 additions and 4 multiplications. Let us say then that 7 operations are used whenever $g_r(\underline{v}_r,\underline{\ell})$ is updated.

The number of operations to evaluate a single step of eq. 5.1 is therefore

$$9\binom{K_r+N-1}{N-1}-1;$$

the combinatorial term being the cardinality of the set L_r. To evaluate $G_r(\underline{v}_r)$ for all $\underline{v}_r \in I_r$, where $2\leq r<R$, requires then

$$\left(9\binom{K_r+N-1}{N-1}-1\right)\left(\sum_{s=r+1}^{R}\binom{K_s+N-1}{N-1}\right)$$

operations; the right hand term being the cardinality of the set I_r.

The total number of operations to obtain $G_R(\underline{0})$ is therefore

$$9\binom{K_R+N-1}{N-1}-1+\sum_{i=2}^{R-1}\left(9\binom{K_i+N-1}{N-1}-1\right)\left(\binom{\sum_{s=i+1}^{R}K_s+N-1}{N-1}+(N-1)(3K_1^2+8K_1+5)\binom{\sum_{r=2}^{R}K_r+N-1}{N-1}\right) \tag{5.5}$$

A straightforward implementation of eq. 5.1 requires a storage space, in number of elements, equal to the sum of the cardinalities of the sets I_1 and I_2. Hence, the storage requirement is

$$\binom{\sum_{r=2}^{R}K_r+N-1}{N-1}+\binom{\sum_{r=3}^{R}K_r+N-1}{N-1}. \tag{5.6}$$

In the following, we shall compare the number of operations and the storage space requirements of the new algorithm, given by eqs. 5.5 and 5.6, respectively, with those of the convolution algorithm.

In order to carry out a comparison of the complexity of the new algorithm with that of the convolution algorithm it will be assumed that $K_r=K$ for $1<r\leq R$, in which case eq. 5.5 simplifies to

$$\left(9\binom{K+N-1}{N-1}-1\right)\left(1+\sum_{i=3}^{R}\binom{(R-i+1)K+N-1}{N-1}\right)+(N-1)(3K^2+8K+5)\binom{(R-1)K+N-1}{N-1}, \tag{5.7}$$

eq. 5.6 simplifies to

$$\binom{(R-1)K+N-1}{N-1}+\binom{(R-2)K+N-1}{N-1}, \tag{5.8}$$

eq. 3.4 simplifies to

$$2R(N-1)(K+1)^R, \tag{5.9}$$

and eq. 3.5 becomes

$$2(K+1)^R. \qquad (5.10)$$

If we consider K and N fixed, then eqs. 5.7 and 5.8 are polynomials in R of degree N and (N-1), respectively. Eqs. 5.9 and 5.10 are exponential in R. Hence, if R is sufficiently large, then the new algorithm will be more efficient than convolution. In Figures 5.1 (a), (b) and (c) we compare the number of operations given by eq. 5.7 and eq. 5.9 for various values of N, K and R. In Figure 5.2 (a), (b) and (c) we compare the storage requirements specified by eq. 5.8 and eq. 5.10. From the Figures we arrive at the general conclusion that the usefulness of the new algorithm becomes pronounced as the number of routing chains is increased. When the number of nodes is small (i.e. 2, 3 or 4) the new algorithm is seen to be especially useful.

5.6 AN HYBRID ALGORITHM

It is possible to reduce the computational complexity of the new algorithm a certain amount by employing an hybrid type algorithm in which we begin by computing $G_h(\underline{v}_h)$, for all $\underline{v}_h \in I_h$, using the version of the convolution algorithm which applies to multiple-chains and state-dependent servers (eq. 3.2). We then go on to compute $G_R(\underline{0})$ using eq. 5.1 and the stored values of $G_h(\underline{v}_h)$, $\underline{v}_h \in I_h$. The value of h, where $1 \leq h \leq R$, is chosen to be that which minimizes the time and space requirements of the hybrid algorithm. Obviously, the hybrid algorithm

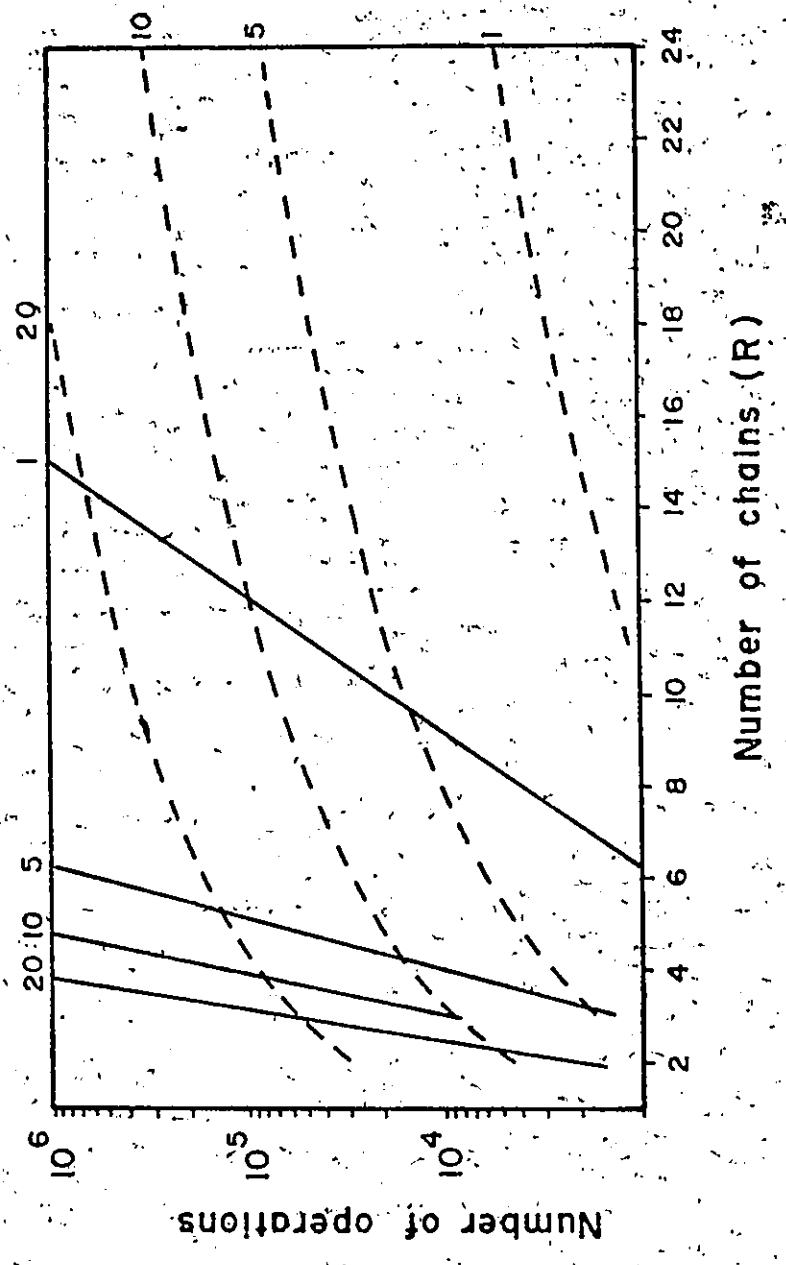

Figure 5.1 (a). Comparison of the number of operations for convolution and the new algorithm, for N=2 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

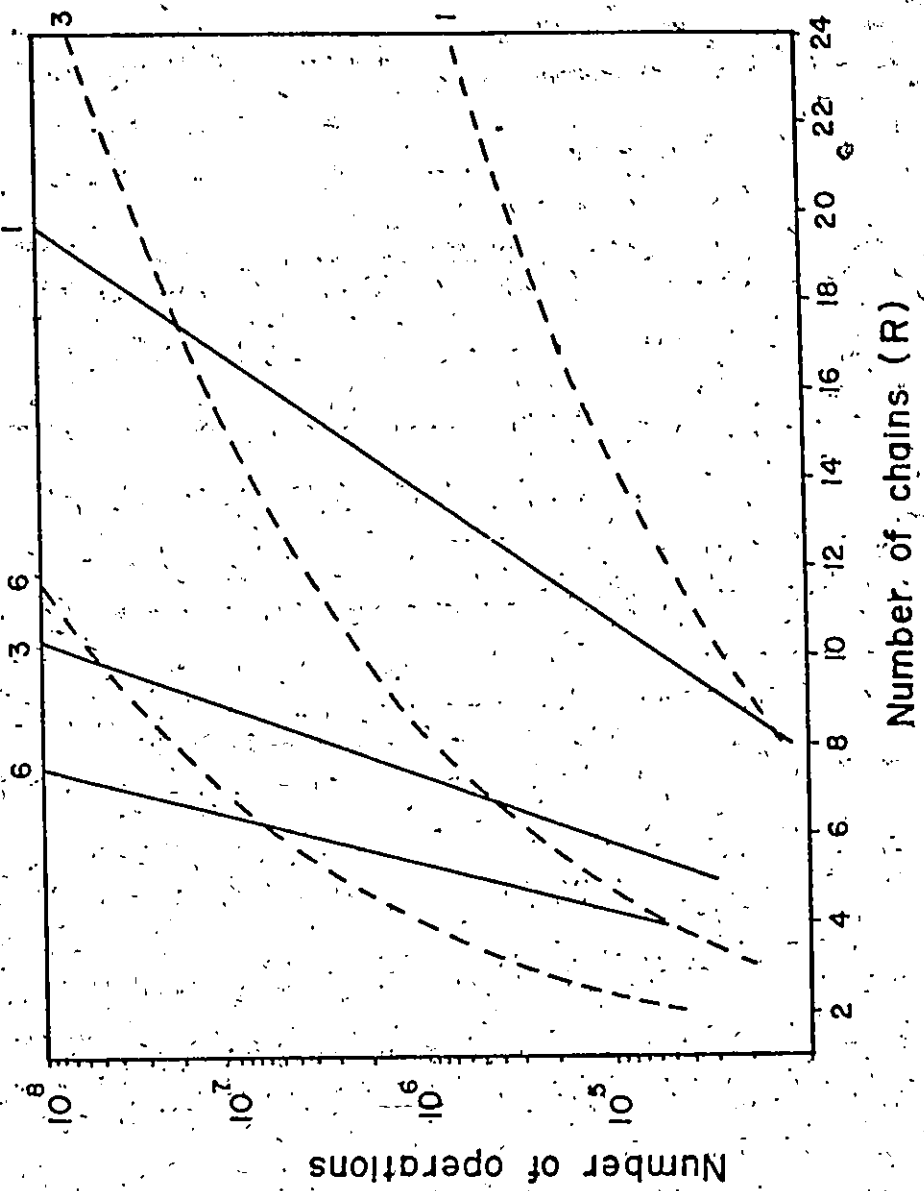

Figure 5.1 (b). Comparison of the number of operations for convolution and the new algorithm, for N=4 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

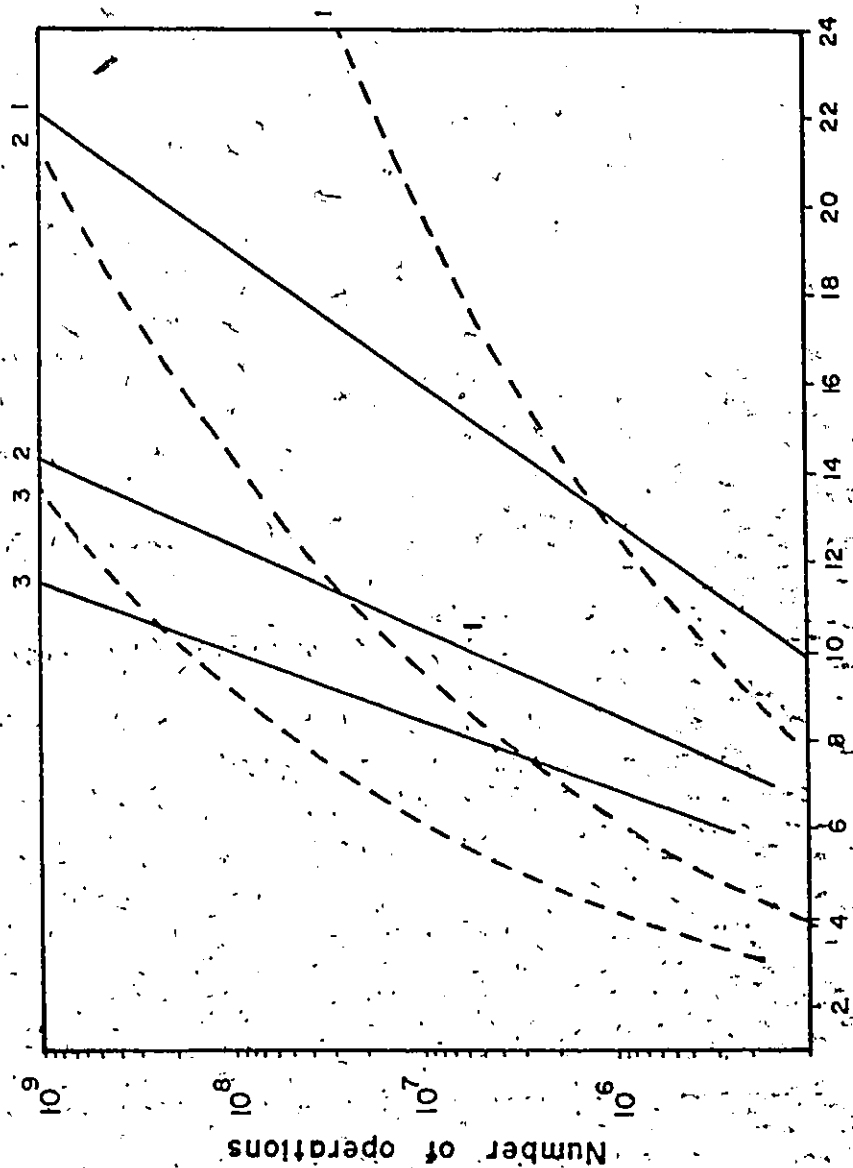

Figure 5.1 (c). Comparison of the number of operations for convolution and the new algorithm, for N=6 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

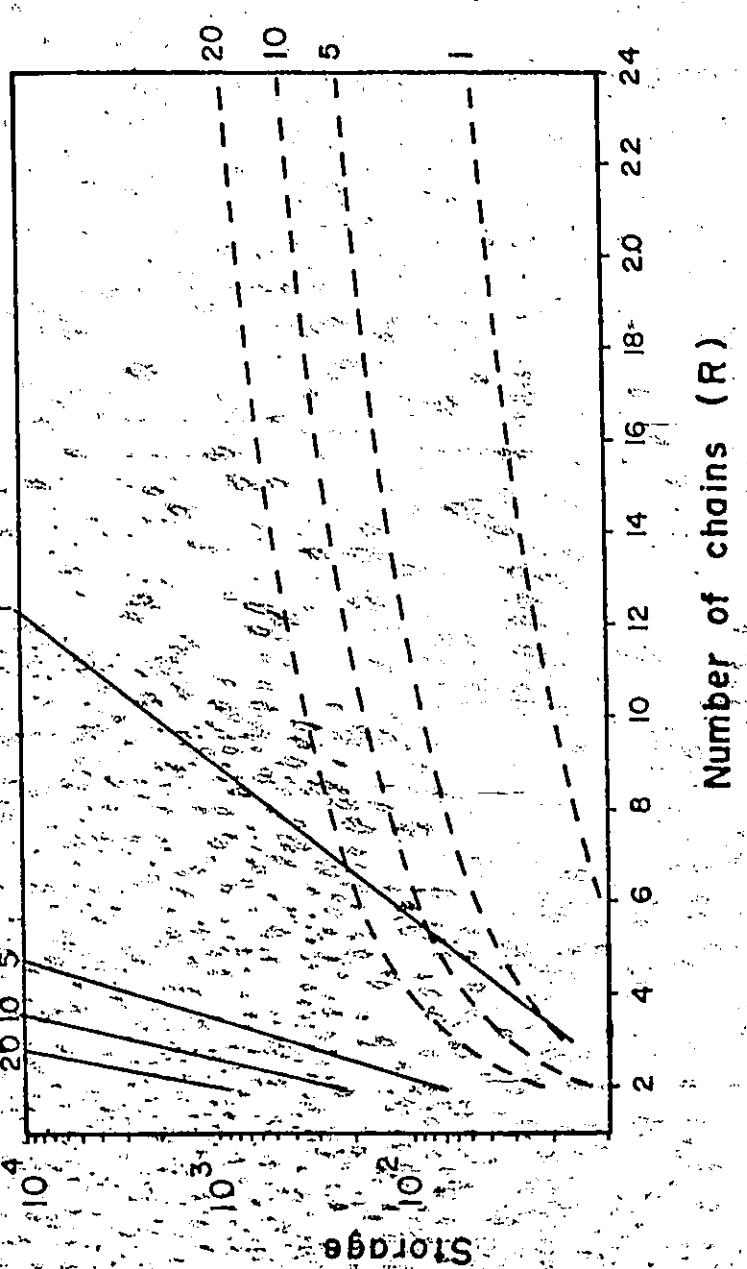

Figure 5.2 (a). Comparison of the storage space requirements for convolution and the new algorithm, for N=2 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

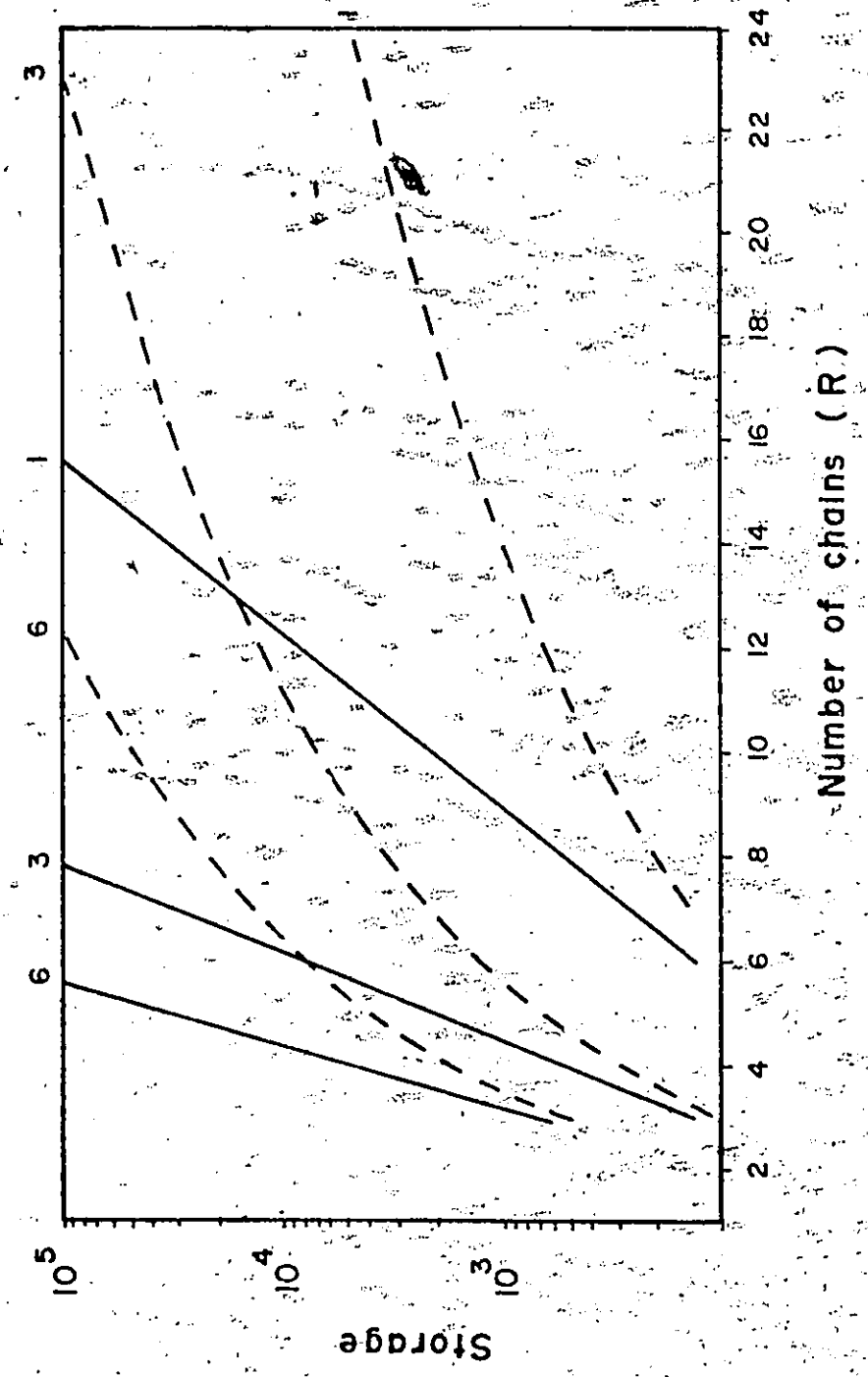

Figure 5.2 (b). Comparison of the storage space requirements for convolution and the new algorithm, for N=4 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

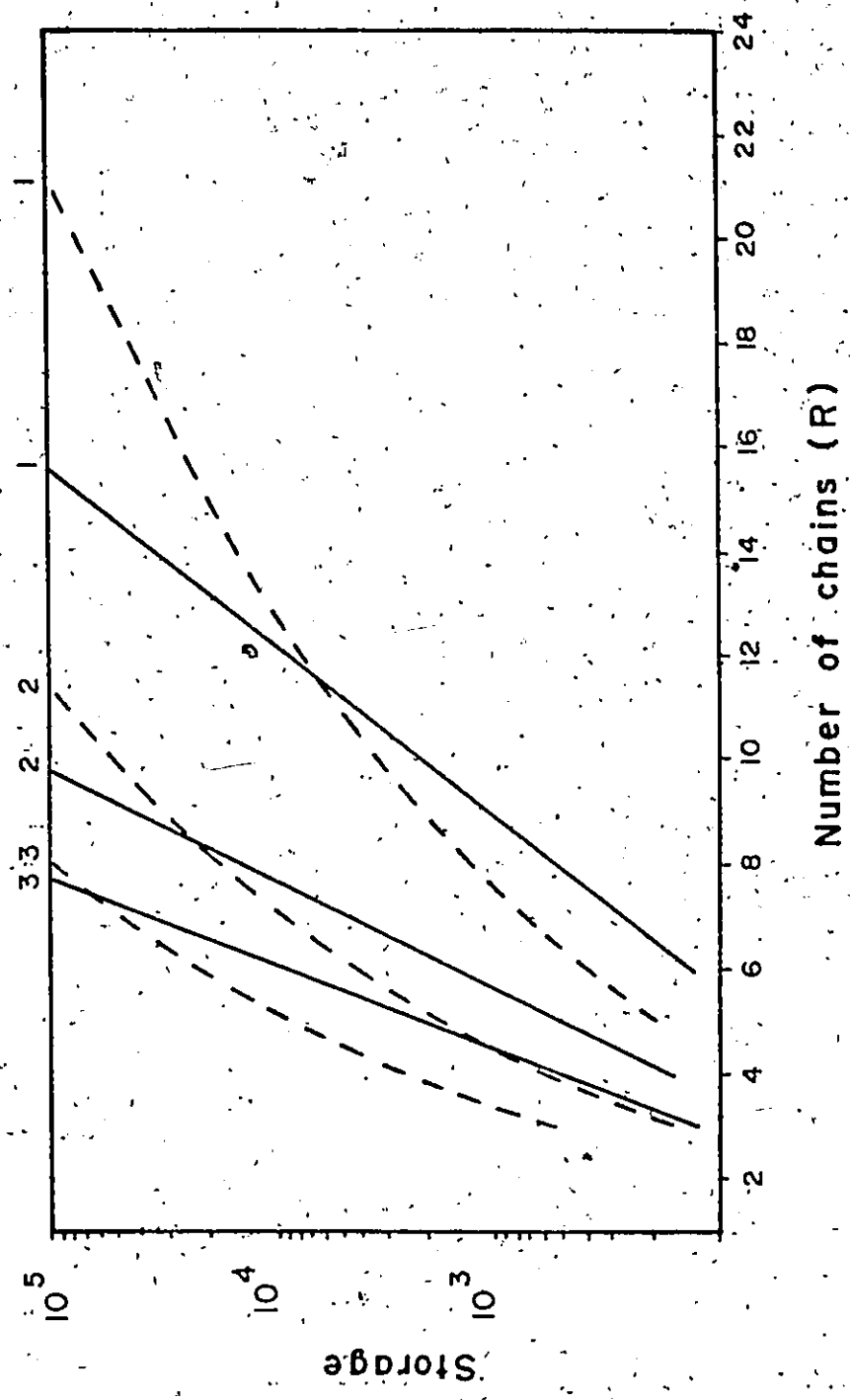

Figure 5.2 (c). Comparison of the storage space requirements for convolution and the new algorithm, for N=6 and various values of K. The chain population K is indicated at the ends of the curves. Solid line is convolution, broken line is for the new algorithm.

will never be inferior to both convolution and the new algorithm, but the improvement over either is, usually, only marginal (in the range of roughly 0-15%). For this reason we have not pursued the matter further.

5.7 EXTENSIONS FOR STATE-DEPENDENT SERVICE CENTERS

In the previous Sections we have assumed, for the sake of simplicity in presentation, that the servers at the service centers operate at a constant speed. In this Section, we extend the new method of computing G to accommodate the situation in which there may be state-dependent service rates. An attractive feature of the new method is that this extension may be made with little additional complexity. As can be seen from eqs. 3.3 and 3.2, such an extension for the convolution algorithm is not so trivial.

Consider a queueing network of type $B^{(R)}(N;\underline{K})$ but with state-dependent servers at the service centers. Let the service rate function for center i depend on the total number of customers at the center and be denoted by $\beta_i(n)$. At IS centers, with the definition of $f_i(\underline{n}_i^{(R)})$ that has been adopted in Section 1.3, we have $\beta_i(n)=1$. With state-dependent servers, the marginal state distribution for the network is, for $\underline{n}^{(R)} \in S^{(R)}$,

$$\Pr(\underline{n}^{(R)}) = G^{-1} \prod_{i=1}^{N} (f_i(\underline{n}_i^{(R)}) / \prod_{a=1}^{n_i^{(R)}} \beta_i(a)). \qquad (5.11)$$

Without loss of generality we need not explicitly consider the more

general situation, which has been considered in [5], in which there may be different service rate functions, $\beta_{ir}(n)$, for each chain r since we may write

$$\prod_{r=1}^{R}\prod_{a=1}^{n_i(R)}\beta_{ir}(a) = \prod_{a=1}^{n_i(R)}\beta_i(a).$$

The extended version of Theorem 5.1 is the following.

Theorem 5.2: (We use notation introduced in Theorem 5.1)

$G_R(\underline{0})$ is given recursively by

$$G_r(\underline{v}_r) = \sum_{\underline{\ell} \in L_r} g_r(\underline{v}_r,\underline{\ell})G_{r-1}(\underline{v}_r+\underline{\ell}), \text{ for } 1 \leq r \leq R, \ \underline{v}_r \in I_r, \quad (5.12)$$

where $h_{ir}(\underline{v}_r,\underline{\ell}) = \begin{cases} \binom{\ell_i+v_{ir}}{\ell_i} w_{ir}^{\ell_i} / \prod_{a=1}^{\ell_i}\beta_i(a+v_{ir}), & \text{if center i is FCFS,} \\ & \text{LCFSPR or PS,} \\ w_{ir}^{\ell_i}/\ell_i!, & \text{if center i is IS.} \end{cases}$

and the initial conditions are $G_0(\underline{v}_0)=1$ for all $\underline{v}_0 \in I_0$.

Proof: (We use notation introduced in the Proof of Theorem 5.1)

The quantity $G_r''(\underline{c})$ is now the normalization constant of an r-chain queueing network with state-dependent service rate functions of the form

$$\mu_i(n) = n\beta_i(n+c_i)/(n+c_i).$$

Clearly $G=G_R(\underline{0})$ since if $\underline{c} = \underline{0}$, then $\mu_i(n)=\beta_i(n)$. The aggregate system state distribution for this system is, according to eq. 5.11,

$$\Pr(\underline{n}^{(r)}) = G_r(\underline{c})^{-1} \prod_{i=1}^{p} \left(\left\{ \prod_{a=1}^{n_i^{(r)}} (a+c_i)/\beta_i(a+c_i) \right\} \prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right) \cdot$$

$$\cdot \prod_{i=p+1}^{N} \left(\prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right)$$

where, by definition,

$$G_r(\underline{c}) = \sum_{\underline{n}^{(r)} \in S^{(r)}} \prod_{i=1}^{p} \left(\left\{ \prod_{a=1}^{n_i^{(r)}} (a+c_i)/\beta_i(a+c_i) \right\} \prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right) \cdot$$

$$\cdot \prod_{i=p+1}^{N} \left(\prod_{s=1}^{r} w_{is}^{n_{is}}/n_{is}! \right).$$

Now

$$\prod_{a=1}^{n_i^{(r-1)}+\ell_i} \beta_i(a+c_i) = \prod_{a=1}^{\ell_i} \beta_i(a+c_i) \prod_{a=1}^{n_i^{(r-1)}} \beta_i(a+c_i+\ell_i),$$

hence

$$G_r(\underline{c}) = \sum_{\underline{\ell} \in L_r} \prod_{i=1}^{p} \left(\binom{\ell_i+c_i}{\ell_i} w_{ir}^{\ell_i} \Big/ \prod_{a=1}^{\ell_i} \beta_i(a+c_i) \right) \cdot$$

$$\cdot \prod_{i=p+1}^{N} (w_{ir}^{\ell_i}/\ell_i!) \cdot \sum_{\underline{n}^{(r)} \in S^{(r)}(\underline{\ell})} \prod_{i=1}^{p} \left(\prod_{a=1}^{n_i^{(r-1)}} (a+\ell_i+c_i)/\beta_i(a+\ell_i+c_i) \right) \cdot$$

$$\cdot \prod_{s=1}^{r-1} w_{is}^{n_{is}}/n_{is}!) \prod_{i=p+1}^{N} \left(\prod_{s=1}^{r-1} w_{is}^{n_{is}}/n_{is}! \right).$$

We now see that the sum of the far right is the definition of $G_{r-1}(\underline{c} + \underline{\ell})$ itself so that eq. 5.12 is proved. The domain I_r in the computation of $G_r(\underline{v}_r)$ is established in the Proof of Theorem 5.1.

The algorithm to compute the normalization constant $G=G_R(\underline{0})$, in the case of state-dependency, is unchanged from that given in Section 5.4 except that the recursive computation of $g_r(\underline{v}_r,\underline{\ell})$ should, of course, take into account the term $\prod_{a=1}^{\ell_i} \beta_i(a+v_{ir})$ which appears in Theorem 5.2. Furthermore, in the computation of $G_1(\underline{v}_1)$, using eq. 5.3, we now have $u_m(a)=a\beta_m(a+v_{m1})/(a+v_{m1})$. We shall not derive the number of operations required to compute G in the case of state-dependency since the derivation is analogous to that of Section 5.5 and the result is essentially the same as eq. 5.5. The storage requirement, in the case of state-dependency, is unchanged from eq. 5.6, except, of course, for the space required to store the network parameters $\beta_i(n)$, which we will ignore.

The computation of G, in the case of state-dependency, using the convolution algorithm, requires the use of the R-dimensional convolution of eq. 3.2. The number of operations involved in arriving at $G=C_N(\underline{K})$, using eq. 3.2, is of the order of (see eq. 3.6)

$$(N-1)\prod_{r=1}^{R}(K_r+1)(K_r+2)/2.$$

The space requirement is unchanged from eq. 3.5. The time requirement of the convolution algorithm is, therefore, increased significantly when there are state-dependent service rates. As a result, the new algorithm is useful when there are state-dependent service rates and relatively many chains in the network.

5.8 NUMERICAL STABILITY

It is well-known that in the conventional convolution algorithm there may arise floating-point overflows and underflows in the course of computing the normalization constant. Such potential difficulties also exist in the new algorithm. To alleviate these problems in the convolution algorithm, a dynamic scaling procedure has been developed [31]. Such scaling procedures can also be applied within the framework of the algorithm presented here.

Let s_r be the scaling factor for chain r ($s_r > 0$). If $\underline{e}_r$ is a left eigenvector, associated with eigenvalue one, of the routing matrix $P^{(r)}$ for chain r, then so is $s_r \underline{e}_r$. If we substitute $(s_r e_{ir})$ for e_{ir} in eq. 5.12, then we may write

$$s_r^{K_r} G_r(\underline{v}_r) = \sum_{\underline{\ell} \in L_r} \Big(\prod_{i=1}^{p} \binom{\ell_i + v_{ir}}{v_{ir}} (s_r w_{ir})^{\ell_i} / \prod_{a=1}^{\ell_i} \beta_i(a+v_{ir}) \Big) \cdot \Big(\prod_{i=p+1}^{N} (s_r w_{ir})^{\ell_i} / \ell_i! \Big) \, G_{r-1}(\underline{v}_r + \underline{\ell}).$$

Hence, we may use s_r as a control to avoid underflows and overflows when we compute $G_r(\underline{v}_r)$ for all $\underline{v}_r \in I_r$. If an overflow occurs in the computation of at least one of the $G_r(\underline{v}_r)$, where $\underline{v}_r \in I_r$, s_r should be made smaller and the computation of $G_r(\underline{v}_r)$, for all $\underline{v}_r \in I_r$, should be recommenced. If an underflow occurs, s_r should be made larger. As with the algorithm in [31], such a dynamic scaling procedure only reduces the possibility that the floating-point range will be exceeded. It does not represent a complete solution to the problem of numerical stability.

APPENDIX 5.1

Here we show by means of an example how we may step through the elements of I_r and L_r in such a way that the current pair of vectors $(\underline{v}_r, \underline{\ell})$ can always be written in terms of the previous pair $(\underline{v}_r^P, \underline{\ell}^P)$ as $(\underline{v}_r^P, \underline{\ell}^P) = (\underline{v}_r - \underline{1}_w + \underline{1}_x, \underline{\ell} - \underline{1}_y + \underline{1}_z)$, for some w, x, y and z such that $1 \leq w,x,y,z \leq N$. Suppose N=3,

$$L_r = \{\underline{\ell}: \ \ell_i > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} \ell_i = 3\}$$

and

$$I_r = \{\underline{i}: \ i_j > 0 \text{ for } 1 \leq j \leq N; \ \sum_{j=1}^{N} i_j = 1\};$$

then, with $\underline{v}_r'$ and $\underline{\ell}_r'$ chosen to be (1,0,0) and (3,0,0) respectively, we may step through L_r and I_r as shown in Table 5.1.

$\underline{\ell}$	$\underline{v_r}$
300	100
210	100
120	100
030	100
021	100
111	100
201	100
102	100
012	100
003	100
003	010
012	010
102	010
201	010
111	010
021	010
030	010
120	010
210	010
300	010
300	001
210	001
120	001
030	001
021	001
111	001
201	001
102	001
012	001
003	001

Table 5.1 Illustration of a method to step through L_r and I_r.

CHAPTER 6 - RECAL: A NEW EFFICIENT ALGORITHM FOR COMPUTING THE MEAN PERFORMANCE MEASURES OF MULTIPLE-CHAIN CLOSED QUEUEING NETWORKS

6.1 INTRODUCTION

In the previous Chapter we have presented a new method to compute the normalization constant. Although this quantity is of interest in itself, the quantities that are of most interest in practice are the mean performance measures. Equations for these measures, in terms of normalization constants, are given in eq. 3.7 for the case where all servers are of the constant-speed type. We could, therefore, in principle, compute all the required normalization constants and substitute them in eq. 3.7 to obtain T_{ir}, U_{ir}, Q_{ir} and W_{ir}. This, however, would involve applying the new algorithm $(K_1+\ldots+K_R+1)$ times. Furthermore, in the case of state-dependency, matters are complicated further. In this latter situation, the mean queue lengths must be computed using eq. 3.8. As a result, in order to obtain the required normalization constants to obtain Q_{ir} for $1 \leq r \leq R$, $1 \leq i \leq N$, we would have to repeat the new algorithm $(N-1) \prod_{r=1}^{R} (K_r+1)$ times. This would defeat the purpose of having an algorithm whose complexity in R is polynomial. Clearly, what we require is a more direct way of obtaining the mean performance measures from the normalization constants which arise in the recursions of eqs. 5.1 and 5.12, rather than resorting to equations that have been developed within the context of the convolution algorithm.

In this Chapter, we present an efficient algorithm, by the name of RECAL (Recursion by Chain Algorithm), to obtain all the mean performance measures. Although we have been unable to obtain expressions for the performance measures in terms of the normalization constants that arise in eqs. 5.1 and 5.12, we have devised a means of circumventing this difficulty which, at the same time, simplifies considerably the recursive expression. The main idea in RECAL is to break down each closed routing chain, containing K_r customers, into K_r identical routing chains that contain one customer each. For this artificial network, in which there is one customer in each routing chain, we have developed simple equations for the mean performance measures in terms of the normalization constants that arise in the computation of $G_R(\underline{0})$ using eq. 5.1 or 5.12. As a result, we need no longer resort to eqs. 3.7. The other useful consequence of breaking down the routing chains is that eqs. 5.1 and 5.12 are then simplified. The artifice of breaking down the routing chains certainly alters the state space and hence the normalization constant of the network under consideration, but it leaves the performance characteristics (i.e. nodal throughputs, server utilizations, waiting times, queue lengths) unchanged.

The Chapter is organized as follows. In the following Section, we present certain new results which form the basis for the proposed algorithm. The algorithm is then developed in Section 6.3. The computational complexity of the algorithm is derived in Section 6.4. In Section 6.5, we compare this complexity with that of the convolution

and MVA algorithms and demonstrate the situations in which the proposed algorithm is useful. We shall see that it is substantially more efficient than the hitherto adopted methods when there are many chains in the network. In Section 6.6, we extend the algorithm to accommodate the situation in which there are state-dependent servers. In Section 6.7, we discuss how one may handle mixed networks, with constant-speed or limited queue-dependent servers, and networks in which there is customer-class switching. In Section 6.8, we present a simple dynamic scaling procedure, which can be incorporated easily within the framework of RECAL, to reduce the possibility of exceeding the floating point range of a machine. Finally, in Section 6.9, we present a FORTRAN implementation of RECAL.

6.2 PRELIMINARY RESULTS

In this Section, we present certain new relationships among the normalization constants of multiple-chain closed queueing networks and new expressions for the mean performance measures in terms of these normalization constants. They will provide the mathematical basis for the algorithm to be developed in the following Section.

Corollary 6.1: If $K_r = 1$ for $1 \leq r \leq R$, then $G_R(\underline{0})$ is given recursively by

$$G_r(\underline{v}_r) = \sum_{i=1}^{N} (1+v_{ir}\delta_i) w_{ir} G_{r-1}(\underline{v}_r - \underline{1}_i), \text{ for } 1 \leq r \leq R, \ \underline{v}_r \in I_r,$$

where $\underline{1}_i$ is a unit vector pointing in the ith direction,

$$\delta_i = \begin{cases} 1, & \text{if center } i \text{ is FCFS, LCFSPR or PS,} \\ 0, & \text{if center } i \text{ is IS,} \end{cases}$$

and the initial conditions are $G_0(\underline{v}_0) = 1$ for all $\underline{v}_0 \in I_0$, where now

$$I_0 = \{\underline{v}_0: \ v_{i0} > 0 \text{ for } 1 \leq i \leq N; \ \sum_{i=1}^{N} v_{i0} = R\}.$$

Proof: The Corollary follows directly from Theorem 5.1 when $K_r = 1$ for $1 \leq r \leq R$.

Theorem 6.1:

$$\Pr(\underline{n}_R = \underline{k}) = G_R(\underline{0})^{-1} G_{R-1}(\underline{k}) \prod_{i=1}^{N} b_i(k_i)$$

where $\underline{n}_R = (n_{1R}, \ldots, n_{NR})$,

$\underline{k} = (k_1, \ldots, k_N)$

and $b_i(k_i) = \begin{cases} w_{iR}^{k_i}, & \text{if center } i \text{ is FCFS, LCFSPR or PS,} \\ w_{iR}^{k_i}/k_i!, & \text{if center } i \text{ is IS.} \end{cases}$

Proof: (We use some notation and definitions found in the Proof of Theorem 5.1)

By definition

$$\Pr(\underline{n}_R = \underline{k}) = \sum_{\underline{n}^{(R)} \in S^{(R)}(\underline{k})} \Pr(\underline{n}^{(R)}).$$

Now, using eq. 1.3, we may write

$$\Pr(n_{1R}=k_1,\ldots,n_{NR}=k_N) = G^{-1} \prod_{i=1}^{N} w_{iR}^{k_i}/k_i!\cdot$$

$$\sum_{\underline{n}^{(R)} \in S^{(R)}(\underline{k})} \prod_{i=1}^{p} \binom{n_i^{(R-1)}+k_i}{k_i} \prod_{r=1}^{R-1} w_{ir}^{n_{ir}}/n_{ir}! \prod_{i=p+1}^{N} \prod_{r=1}^{R-1} w_{ir}^{n_{ir}}/n_{ir}!)$$

$$= G^{-1} \prod_{f=1}^{p} w_{iR}^{k_i} \prod_{i=p+1}^{N} w_{iR}^{k_i}/k_i!\cdot$$

$$\sum_{\underline{n}^{(R)} \in S^{(R)}(\underline{k})} \prod_{i=1}^{p} \binom{n_i^{(R-1)}}{k_i} \prod_{a=1}^{k_i} (a+k_i) \prod_{r=1}^{R-1} w_{ir}^{n_{ir}}/n_{ir}! \prod_{i=p+1}^{N} \prod_{r=1}^{R-1} w_{ir}^{n_{ir}}/n_{ir}!).$$

From the definition of $\theta_r(\underline{c})$ given in the Proof of Theorem 5.1, we see that the sum on the far right is the definition of $G_{R-1}(\underline{k})$ itself. Furthermore, $G=G_R(\underline{0})$, so that

$$\Pr(n_{1R}=k_1,\ldots,n_{NR}=k_N) = G_R^{-1}(\underline{0}) G_{R-1}(\underline{k}) \prod_{i=1}^{p} w_{iR}^{k_i} \prod_{i=p+1}^{N} w_{iR}^{k_i}/k_i!.$$

Theorem 6.2: (a) If $K_R=1$ and there are no IS centers in the network, then

$$G_{R-1}(\underline{0}) = \sum_{\ell=1}^{N} G_{R-1}(\underline{1}_\ell) / (N+K-1)$$

where $K = \sum_{r=1}^{R} K_r$.

(b) If x is an IS center, then $G_{R-1}(\underline{0}) = G_{R-1}(\underline{1}_x)$.

Proof:

Proof of Theorem 6.2 (a):

When there are no IS centers in the network, using the definition of $G_r(\underline{c})$ in the Proof of Theorem 5.1, we may write

$$G_{R-1}(\underline{1}_\ell) = \sum_{\underline{n}^{(R-1)} \in S^{(R-1)}} (n_\ell^{(R-1)} + 1) \prod_{i=1}^{N} (n_i^{(R-1)}! \left(\prod_{r=1}^{R-1} w_{ir}^{n_{ir}} / n_{ir}! \right)).$$

Therefore

$$G_{R-1}(\underline{1}_\ell) = G_{R-1}(\underline{0})\, E(n_\ell^{(R-1)}) + G_{R-1}(\underline{0})$$

where $E(\cdot)$ denotes expectation. Hence, when $K_R = 1$,

$$\sum_{\ell=1}^{N} G_{R-1}(\underline{1}_\ell) = G_{R-1}(\underline{0})(K-1) + NG_{R-1}(\underline{0})$$

since

$$\sum_{\ell=1}^{N} E(n_\ell^{(R-1)}) = \sum_{r=1}^{R-1} K_r = K-1.$$

Proof of Theorem 6.2 (b):

Consider the definition of $G_r(\underline{c})$ in the Proof of Theorem 5.1. When $\underline{c} = \underline{1}_x$ and x is an IS center, we have $\underline{c}_i = 0$ for all $1 \leq i \leq p$ in which case $G_r(\underline{1}_x) = G_r(\underline{0})$. Hence, $G_{R-1}(\underline{0}) = G_{R-1}(\underline{1}_x)$.

We now give some results concerning the mean performance measures for chain R when $K_R = 1$.

Corollary 6.2: If $K_R = 1$, then

$$T_{iR} = \begin{cases} \left(\frac{w_{iR}}{t_{iR}}\right) \sum_{\ell=1}^{N} G_{R-1}(\underline{1}_\ell)/G_R(\underline{0})(N+K-1), & \text{if there are no IS centers in the network,} \\ \left(\frac{w_{iR}}{t_{iR}}\right) G_{R-1}(\underline{1}_x)/G_R(\underline{0}), & \text{if there are IS centers and x is any one of them,} \end{cases}$$

$$U_{iR} = t_{iR} T_{iR},$$

$$Q_{iR} = G_R(\underline{0})^{-1} G_{R-1}(\underline{1}_i) w_{iR},$$

and

$$W_{iR} = Q_{iR} / T_{iR}.$$

Proof: It is known (see eqs. 3.7) that, in terms of the notation adopted here, when $K_R = 1$, $T_{iR} = e_{iR} G_{R-1}(\underline{0})/G_R(\underline{0})$. The expressions for T_{iR} then follow from Theorem 6.2. The expression for Q_{iR} follows directly from its definition and Theorem 6.1. The expressions for U_{iR} and W_{iR} are well-known and based on Little's result [38].

6.3 THE RECURSION BY CHAIN ALGORITHM - RECAL

The new algorithm for multiple-chain networks is based directly on Corollaries 6.1 and 6.2, and the introduction of the artifice of breaking down each chain r, with K_r customers, into K_r identical sub-chains, with one customer each. The basic idea of employing this artifice is to create an artificial network in which $K_r = 1$, for all chains, so that we may apply the simpler recursion of Corollary 6.1 and

obtain the mean performance measures using Corollary 6.2. The artificial network has, in general, a different state space and normalization constant from the original network, but we may nevertheless employ the artifice and obtain the correct values for the performance measures since it alters in no way the physical characteristics of the original network.

Let the network created from $B^{(R)}(N,\underline{K})$, by breaking down each chain, be denoted by $B^{(R)}(N,\underline{K})^{o}$ and abbreviated as B^{o}. (We use a dot (o) to differentiate the notation associated with B^{o}.) The total population K^{o} of B^{o} is $K^{o} = \sum_{r=1}^{R} K_r$. The number of chains R^{o} in B^{o} is, by the definition of B^{o}, K^{o} and the population of chain r is $K_r^{o} = 1$ for $1 \leq r \leq K^{o}$. Let the customers in B^{o} be enumerated as $1,2,\ldots,K^{o}$ and let the chain to which customer k belongs in $B^{(R)}(N,\underline{K})$ be r(k). We then have, using Corollary 6.1, that $G^{o}_{R^{o}}(\underline{0})$ is given recursively by

$$G_k^{o}(\underline{v}_k) = \sum_{i=1}^{N} (1+v_{ik}\delta_i) w_{ir(k)} G_{k-1}^{o}(\underline{v}_k + \underline{1}_i) \qquad (6.1)$$

for $1 \leq k \leq K^{o}$, $\underline{v}_k \in I_k^{o}$, where $I_k^{o} = \{\underline{v}_k : v_{ik} \geq 0 \text{ for } 1 \leq i \leq N; \sum_{i=1}^{N} v_{ik} = K^{o}-k\}$. The initial conditions are $G_0^{o}(\underline{v}_0) = 1$ for all $\underline{v}_0 \in I_0^{o}$.

Hence, by breaking down each chain in $B^{(R)}(N,\underline{K})$ into constituent sub-chains we can apply Corollary 6.1 and circumvent, in particular, the need to compute the terms $g_r(\underline{v}_r,\ell)$ which appear in the expression of Theorem 5.1. We note that, in general, $G^{o}_{R^{o}}(\underline{0}) \neq G_R(\underline{0})$ since, in general, $S^{o(R^{o})} \neq S^{(R)}$. Hence, we are no longer able to compute the

normalization constant $G = G_R(\underline{0})$ which is supposed to be of central interest. Nevertheless, using Corollary 6.2, we may write

$$T^{o}_{iR^o} = \begin{cases} \left(\frac{w_{ir(R^o)}}{t_{ir(R^o)}}\right) \sum_{\ell=1}^{N} G^{o}_{R^o-1}(\underline{1}_\ell) \;/\; G^{o}_{R^o}(\underline{0})(N+K^{\sigma}-1), & \text{if there are no IS centers in the network,} \\ \left(\frac{w_{ir(R^o)}}{t_{ir(R^o)}}\right) G^{o}_{R^o-1}(\underline{1}_x) / \; G^{o}_{R^o}(\underline{0}), & \text{if there are IS centers and x is any one of them,} \end{cases}$$

$$U^{o}_{iR^o} = t_{ir(R^o)} T^{o}_{iR^o},$$

$$Q^{o}_{iR^o} = G^{o}_{R^o}(\underline{0})^{-1} G^{o}_{R^o-1}(\underline{1}_i) w_{ir(R^o)}$$

and $$W^{o}_{iR^o} = Q^{o}_{iR^o} / \; T^{o}_{iR^o}. \qquad (6.2)$$

The above expressions give the mean performance measures with respect to a single customer who belongs to chain $r(R^o)$ in $B^{(R)}(N,\underline{K})$. The normalization constants in eqs. 6.2 are obtained in the computation of $G^{o}_{R^o}(\underline{0})$ using eq. 6.1. The key point remaining in the algorithm is that all customers who belong to a particular chain in $B^{(R)}(N,\underline{K})$ are statistically indistinguishable in equilibrium, so we have finally for the network $B^{(R)}(N,\underline{K})$:

$$T_{ir(R^o)} = K_{r(R^o)} T^{o}_{iR^o},$$

$$U_{ir(R^o)} = K_{r(R^o)} U^{o}_{iR^o},$$

$$Q_{ir(R^o)} = K_{r(R^o)} Q^{o}_{iR^o}$$

and $$W_{ir(R^o)} = W^{o}_{iR^o}. \qquad (6.3)$$

Hence, we are able to obtain the mean performance measures for a particular chain $r(R^o)$, using eq. 6.1, even though the value for G itself is never obtained.

The mean performance measures for chains other than $r(R^o)$ may be obtained by reenumerating the customers in B^o so that $r(R^o)$ is changed to another chain for which one wishes to compute performance measures using eqs. 6.1, 6.2 and 6.3. More specifically, suppose that the customers in B^o are enumerated initially so that the initial assignment of the customers in B^o to chains in $B^{(R)}(N,\underline{K})$ is $r^{(1)}(k)$, as illustrated in Table 6.1 and given by

$$r^{(1)}(k) = \begin{cases} 1 \text{ for } 1 \leq k \leq K_1 - 1, \\ i \text{ for } 1 + \sum_{r=1}^{i-1} (K_r - 1) \leq k \leq \sum_{r=1}^{i} (K_r - 1), \quad 2 \leq i \leq R, \\ R - k + 1 + \sum_{r=1}^{R} (K_r - 1) \text{ for } 1 + \sum_{r=1}^{R} (K_r - 1) \leq k \leq K^o. \end{cases} \tag{6.4}$$

Having arrived at $G_R^o o(\underline{0})$, using eq. 6.1, we may compute the mean performance measures for chain 1 in $B^{(R)}(N,\underline{K})$ using eqs. 6.2 and 6.3. We then reenumerate the customers in B^o to obtain a new assignment of the customers in B^o to chains in $B^{(R)}(N,\underline{K})$. Let the new assignment be $r^{(2)}(k)$ as illustrated in Table 6.1 and given by

$$r^{(2)}(k) = \begin{cases} r^{(1)}(k) \text{ for } 1 \leq k < \sum_{r=1}^{R} (K_r - 1), \\ 1 \text{ for } k = 1 + \sum_{r=1}^{R} (K_r - 1), \\ r^{(1)}(k) + 1 \text{ for } 2 + \sum_{r=1}^{R} (K_r - 1) \leq k \leq K^o. \end{cases}$$

Customer number in B^o		Assignment of customer k to chains in $B^{(R)}(N,\underline{K})$					
k		$r^{(1)}(k)$	$r^{(2)}(k)$	$r^{(s)}(k)$	$r^{(s+1)}(k)$	$r^{(R-1)}(k)$	$r^{(R)}(k)$
K^o		1	2	s	s+1	R-1	R
K^o-1		2	3	s+1	s+2	R	R-1
K^o-2		3	4	s+2	s+3	R-2	R-2
•		•	•	•	•	•	•
•		•	•	•	R	•	•
•		•	•	R	s	•	•
•		•	•	s-1	s-1	•	•
•		•	•	•	•	•	•
K^o-R+3		R-2	R-1	3	3	3	3
K^o-R+2		R-1	R	2	2	2	2
K^o-R+1		R	1	1	1	1	1
K^o-R		R	same as $r^{(1)}(k)$				
•		•					
•		•					
$R-1+\sum_{r=1}^{R-1}(K_r-1)$		R					
$\sum_{r=1}^{R-1}(K_r-1)$		R-1					
•		•					
•		•					
K_1+K_2-1		3					
K_1+K_2-2		2					
•		•					
•		•					
K_1		2					
K_1-1		1					
•		•					
1		1					

Table 6.1 - Assignments of the customers in B^o to the chains in $B^{(R)}(N,\underline{K})$.

The mean performance measures for chain 2 in $B^{(R)}(N,\underline{K})$ may then be obtained after having recomputed $G^{o}_{R}o(\underline{0})$. In general then, after having computed the performance measures for chain s in $B^{(R)}(N,\underline{K})$, we reenumerate the customers in B^{o} so that

$$r^{(s+1)}(k) = \begin{cases} r^{(s)}(k) \text{ for } 1 \leq k \leq s-1+\sum_{r=1}^{R}(K_r-1), \\ s \text{ for } k=s+\sum_{r=1}^{R}(K_r-1), \\ r^{(s)}(k)+1 \text{ for } s+1+\sum_{r=1}^{R}(K_r-1) \leq k \leq K^{o}. \end{cases} \tag{6.5}$$

We then recompute $G^{o}_{R}o(\underline{0})$ and obtain the measures for chain (s+1). This procedure of reassignment and recomputation of $G^{o}_{R}o(\underline{0})$ is rendered efficient by storing $G^{o}_{x-1}(\underline{v}_{x-1})$ for all $\underline{v}_{x-1} \in I^{o}_{x-1}$, where

$$x = s + \sum_{r=1}^{R}(K_r-1)$$

so that, in the determination of the performance measures for chain (s+1) in $B^{(R)}(N,\underline{K})$, we need not to reevaluate eq. 6.1 for all $1 \leq k \leq K^{o}$ but <u>only</u> for $x \leq k \leq K^{o}$.

We now summarize the algorithm which has been developed.

Step 1: Initialize $G_0^o(\underline{v}_0) = 1$ for all $\underline{v}_0 \in I_0^o$.

Step 2: Enumerate the customers in B^o so that the assignment of customers in B^o to chains in $B^{(R)}(N,\underline{K})$ is $r^{(1)}(k)$, given by eq. 6.4, for $1 \leq k \leq K^o$.

Step 3: Compute and store $G_x^o(\underline{v}_x)$ for all $\underline{v}_x \in I_x^o$, where $x=K^o-R$, using eq. 6.1.

Step 4: For each chain s in $B_R(N,\underline{K})$, $1 \leq s \leq R$:

Step 4a: Determine $G_K^o o(\underline{0})$ using eq. 6.1 and the stored values of $G_x^o(\underline{v}_x)$, $\underline{v}_x \in I_x^o$.

Step 4b: Compute the mean performance measures for chain s using $G_K^o o(\underline{0})$, $G_K^o o_{-1}(\underline{1}_i)$ for $1 \leq i \leq N$, eqs. 6.2 and 6.3. Stop if $s=R$.

Step 4c: Reenumerate the customers in B^o so that the assignment of customers in B^o to chains in $B^{(R)}(N,\underline{K})$ is $r^{s+1}(k)$, as defined by eq. 6.5.

Step 4d: Increment x by 1.

Step 4e: Compute and store $G_x^o(\underline{v}_x)$ for all $\underline{v}_x \in I_x^o$ using eq. 6.1 and the stored values of $G_{x-1}^o(\underline{v}_{x-1})$, $\underline{v}_{x-1} \in I_{x-1}^o$.

6.4 COMPUTATIONAL COMPLEXITY

In this Section we determine the time and space requirements of the algorithm. We derive the number of operations (additions and multiplications) and the storage space (number of elements) required. For simplicity we assume that there are no IS centers in the network. When there are IS centers, however, the computations are simplified slightly since $\delta_i=0$ when i is an IS center.

Time requirement

Let us begin by evaluating the number of operations involved in Step 3. The evaluation of the summation in eq. 6.1 requires (4N-1) operations. The number of operations to obtain $G_1^o(\underline{v}_1)$ for all $\underline{v}_1 \in I_1^o$ is

$$(4N-1)\binom{K^o+N-2}{N-1};$$

the combinatorial term being the cardinality of I_1^o. The total number of operations to obtain $G_x^o(\underline{v}_x)$ for all $\underline{v}_x \in I_x^o$, where $x=K^o-R$, is therefore

$$\sum_{i=1}^{K^o-R} (4N-1)\binom{K^o-i+N-1}{N-1}. \qquad (6.6)$$

With s = 1, the number of operations to carry out Step 4a is

$$\sum_{i=K^o-R+1}^{K^o} (4N-1)\binom{K^o-i+N-1}{N-1}.$$

In Step 4b we require (N+2) operations to compute T_{iR}^o using eq. 6.2 (assuming that there are no IS centers in the network and that the constant $(N+K^o-1)$ has been precomputed). We then require six more operations to obtain $T_{ir(R^o)}$, $U_{ir(R^o)}$, $Q_{ir(R^o)}$ and $W_{ir(R^o)}$ using eqs. 6.2 and 6.3. The number of operations involved in Step 4b is, therefore, (N+8). We loop on Step 4 R times, so that the total number of operations consumed in Step 4b is

$$R(N+8). \qquad (6.7)$$

With s = 1, the number of operations required to carry out Step 4e is

$$(4N-1)\binom{R-1+N-1}{N-1}.$$

In general, the number of operations required in Step 4a is

$$\sum_{i=K^o-R+s}^{K^o} (4N-1)\binom{K^o-i+N-1}{N-1} \tag{6.8}$$

and the number of operations involved in Step 4e is

$$(4N-1)\binom{R-s+N-1}{N-1}. \tag{6.9}$$

The total number of operations required by the algorithm is, using eqs. 6.6, 6.7, 6.8 and 6.9,

$$\sum_{i=1}^{K^o-R} (4N-1)\binom{K^o-i+N-1}{N-1} + \sum_{s=1}^{R} \sum_{i=K^o-R+s}^{K^o} (4N-1)\binom{K^o-i+N-1}{N-1} + \sum_{s=1}^{R-1} (4N-1)\binom{R-s+N-1}{N-1} + R(N+8)$$

which simplifies to

$$(4N-1)\left(\binom{K^o+N-1}{N} + \binom{R+N-1}{N+1} + \binom{R+N-1}{N} - 1\right) + R(N+8). \tag{6.10}$$

Now suppose that $K_r = \kappa$ for $1 \leq r \leq R$. Then $K^o = \kappa R$. If we consider κ and N fixed, then eq. 6.10 is a polynomial in R of degree (N+1). When R is large, eq. 6.10 is of the order of

$$\frac{(4N-1)}{(N+1)!} R^{N+1}$$

If we now consider R and κ fixed, then eq. 6.10 is a polynomial in N of degree (κR). When N is large, eq. 6.10 is of the order of

$$\begin{cases} 8N^R/(R-1)!, & \text{if } \kappa = 1, \\ 4N^{\kappa R}/(\kappa R-1)!, & \text{if } \kappa > 2. \end{cases}$$

Space requirement

The space requirement is dictated by the maximum storage space which is required at any one time in the implementation of eq. 6.1. It is assumed, for the sake of simplicity in implementation, that we compute $G_k^o(\underline{v}_k)$ for all $\underline{v}_k \in I_k^o$ before incrementing k.

Suppose that $G_{k-1}^o(\underline{v}_{k-1})$ has been computed and stored for all $\underline{v}_{k-1} \in I_{k-1}^o$. Furthermore, suppose that the value for $G_{k-1}^o(\underline{v}_{k-1})$ is held in an array at location $M_{k-1}(\underline{v}_{k-1})$, where $M_x(\underline{d})$ is a mapping that leads to increasing values of the storage location index to vectors $\underline{d}$ that give increasing values for the sum

$$\sum_{i=1}^{N} d_i (K^o - x)^{i-1}. \tag{6.11}$$

Such a mapping has been introduced in [13, Section 4.2]. The domain of

$M_x(\underline{d})$ is I_x^o and the range is $\{1,2,\ldots,\binom{K_o-x+N-1}{N-1}\}$, the combinatorial term being the cardinality of the set I_x^o. An example of the mapping is given in Table 6.2.

If we now compute the values of $G_k^o(\underline{v}_k)$, $\underline{v}_k \in I_k^o$, in the order which gives increasing values for the sum

$$\sum_{i=1}^{N} v_{ik}(K^o-k)^{i-1},$$

then a useful consequence is that after having computed $G_k^o(\underline{v}_k)$ using eq. 6.1 we may store the result at location $M_k(\underline{v}_k)$ of the <u>same</u> array which was used to store $G_{k-1}^o(\underline{v}_{k-1})$ for all $\underline{v}_{k-1} \in I_{k-1}^o$. This mechanism is illustrated in Figure 6.1. This storage procedure may be adopted since the value of $G_{k-1}^o(\underline{v}_{k-1})$ at location $M_k(\underline{v}_k)$ is never required, after $G_k^o(\underline{v}_k)$ has been computed, for any of the computations of $G_k^o(\underline{\gamma})$, where $\underline{\gamma}$ is any vector such that $\underline{\gamma} \in I_k^o$ and $M_k(\underline{\gamma}) > M_k(\underline{v}_k)$. This point is proved in Appendix 6.1. As a result, the storage space required to implement eq. 6.1 is only the cardinality of I_0^o and not the sum of the cardinalities of I_0^o and I_1^o. The cardinality of I_0^o is $\binom{K^o+N-1}{N-1}$. In addition, Step 3 requires a storage space equal to the cardinality of $I_{K^o-R}^o$ which is $\binom{R+N-1}{N-1}$. The values obtained in Step 4e may be stored in the same storage space as that used for Step 3. The total storage space required for the algorithm is then

$$\binom{K^o+N-1}{N-1} + \binom{R+N-1}{N-1}. \qquad (6.12)$$

$\underline{d}$	$M_x(\underline{d})$
300	1
210	2
120	3
030	4
201	5
111	6
021	7
102	8
012	9
003	10

Table 6.2 - Example of the mapping $M_x(\underline{d})$

($N=3$, $K^o=3$, $x=0$)

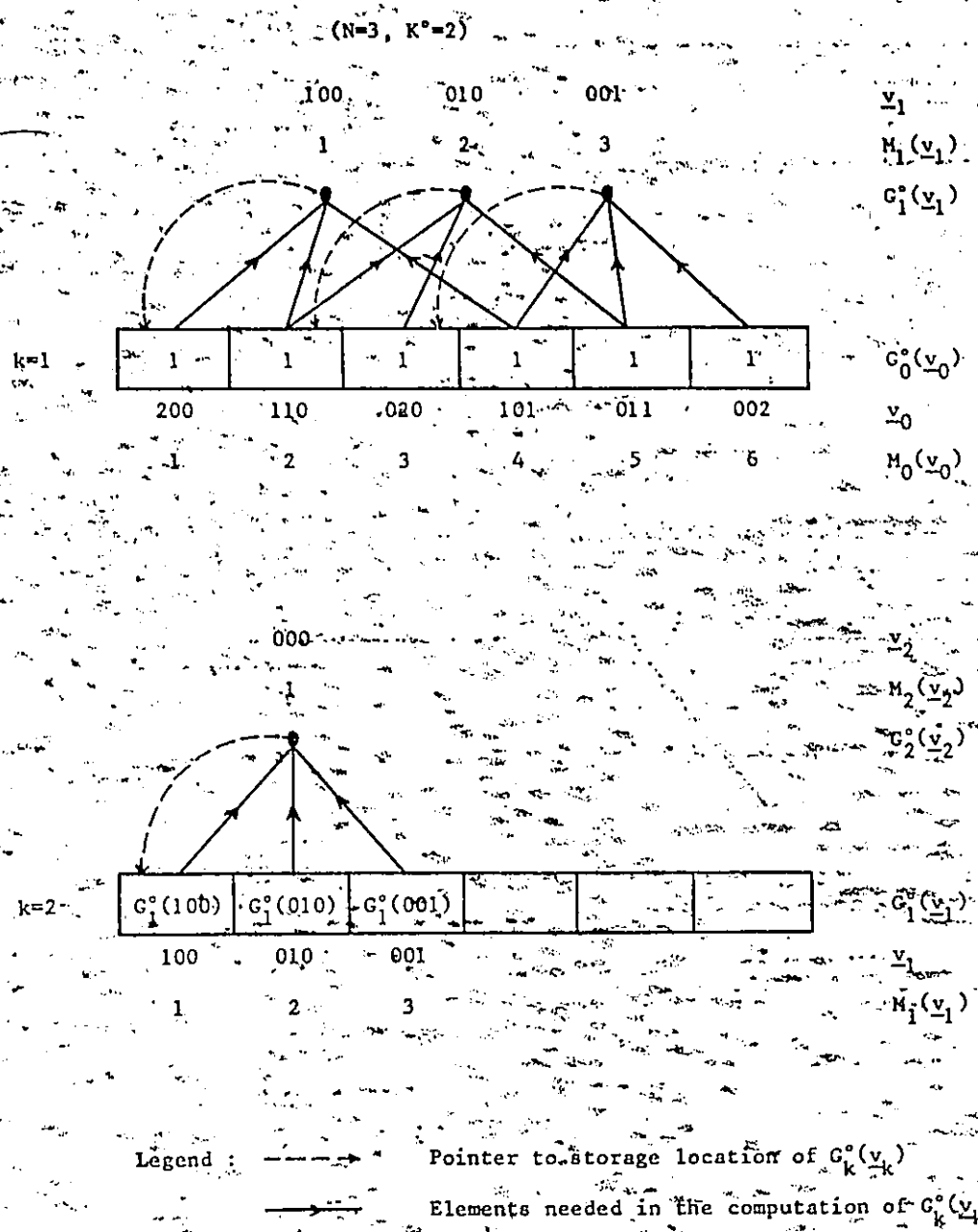

Figure 6.1 - Illustration of the implementation of eq. 6.1

With $K_r = \kappa$ for $1 \leq r \leq R$ and with N and κ considered fixed, eq. 6.12 is a polynomial in R of degree (N-1). When R is large, eq. 6.12 is of the order of

$$\frac{(\kappa^{N-1}+1)}{(N-1)!} R^{N-1}$$

If we now consider R and κ fixed, then eq. 6.12 is a polynomial in N of degree (κR). When N is large, eq. 6.12 is of the order of

$$\begin{cases} 2N^R/R!, & \text{if } \kappa=1, \\ N^{\kappa R}/(\kappa R)!, & \text{if } \kappa \geq 2. \end{cases}$$

6.5 COMPARISONS IN COMPLEXITY

In this Section, we compare the time and space requirements of RECAL with those of the convolution and MVA algorithms.

If we use the version of the convolution algorithm which applies to networks with constant-speed servers, then the total number of operations required to arrive at G is (eq. 3.4)

$$2R(N-1) \prod_{r=1}^{R} (K_r+1) \qquad (6.13)$$

and the required storage space is

$$2 \prod_{r=1}^{R} (K_r+1) \qquad (6.14)$$

The MVA algorithm involves approximately the same time requirement as eq. 6.13 and the required storage space is (eq. 3.10)

$$N \prod_{r=1}^{R} (K_r+1). \qquad (6.15)$$

We show in Appendix 6.2 that no computational advantage can be obtained in the convolution or MVA algorithms by breaking down the chains in $B^{(R)}(N,\underline{K})$ into sub-chains.

In the previous Section, we have seen that with N and K considered fixed, the time and space requirements of RECAL are polynomial in R. Hence, if R is sufficiently large, then the time and space requirements given by eqs. 6.10 and 6.12 respectively, will be less than those given by eqs. 6.13 and 6.14 or 6.15.

We see from eqs. 6.12, 6.14 and 6.15 that RECAL has a smaller space requirement than either convolution or MVA when

$$\binom{\sum_{r=1}^{R} K_r + N-1}{N-1} + \binom{R+N-1}{N-1} < 2 \prod_{r=1}^{R} (K_r+1).$$

From eqs. 6.10 and 6.13 we see that RECAL has a smaller time requirement when

$$(4N-1)\left(\binom{\sum_{r=1}^{R} K_r + N-1}{N} + \binom{R+N-1}{N+1} + \binom{R+N-1}{N} - 1\right) + R(N+8) < 2R(N-1) \prod_{r=1}^{R} (K_r+1).$$

In Figure 6.2 we compare, for the purposes of illustration, the number of operations given by eq. 6.10 with eq. 6.13 in the case where $N = 4$ and $K_r = \kappa$ for $1 \leq r \leq R$, for various values of κ and R. In Figure 6.3 we compare the storage requirements given by eq. 6.12 with eq. 6.14 in the case where $N = 4$. In Figure 6.4 we determine, for various values of κ, the region in the space of queueing networks $(N \times R)$ in which the number of operations required by RECAL (eq. 6.10) is less than that required with convolution (eq. 6.13). In Figure 6.5 we determine the region in which the storage space requirement of RECAL (eq. 6.12) is less than that required with convolution (eq. 6.14). In Figures 6.6 and 6.7 we give, respectively, several iso-time and iso-storage curves in the space $(N \times R)$ for RECAL and convolution in the case where $\kappa = 3$. We see from Figures 6.2-6.7 that RECAL is useful when there are many chains in the network. We finally note that when $N = 2$ the storage requirement of RECAL, given by eq. 6.12, is linear in R.

6.6 EXTENSION TO STATE-DEPENDENT SERVERS

The algorithm in its present form can be readily extended to handle state-dependent servers with service rate functions of the form $\beta_i(n)$. In the previous Chapter, we have already extended Theorem 5.1 in Theorem 5.2. It follows from Theorem 5.2 that the extended version of eq. 6.1 is

$$G_k^o(\underline{v}_k) = \sum_{i=1}^{N} (1+v_{ik}\delta_i) w_{ir(k)} G_{k-1}^o(\underline{v}_k - \underline{1}_i) / \beta_i(1+v_{ik}). \qquad (6.16)$$

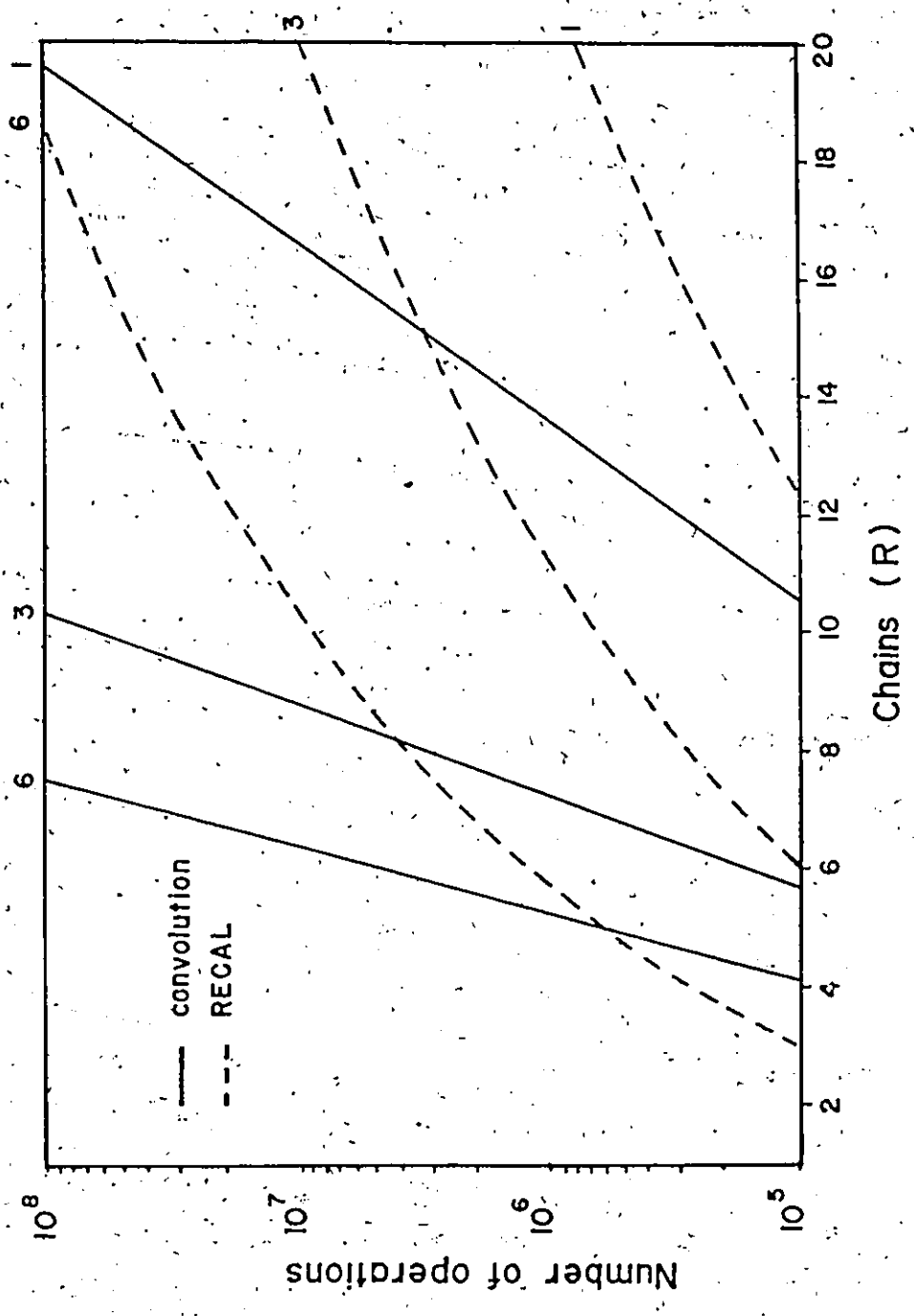

Figure 6.2 - Comparison of the number of operations in RECAL and convolution, for N=4 and κ=1, 3 and 6. (The values for κ are indicated at the ends of the curves).

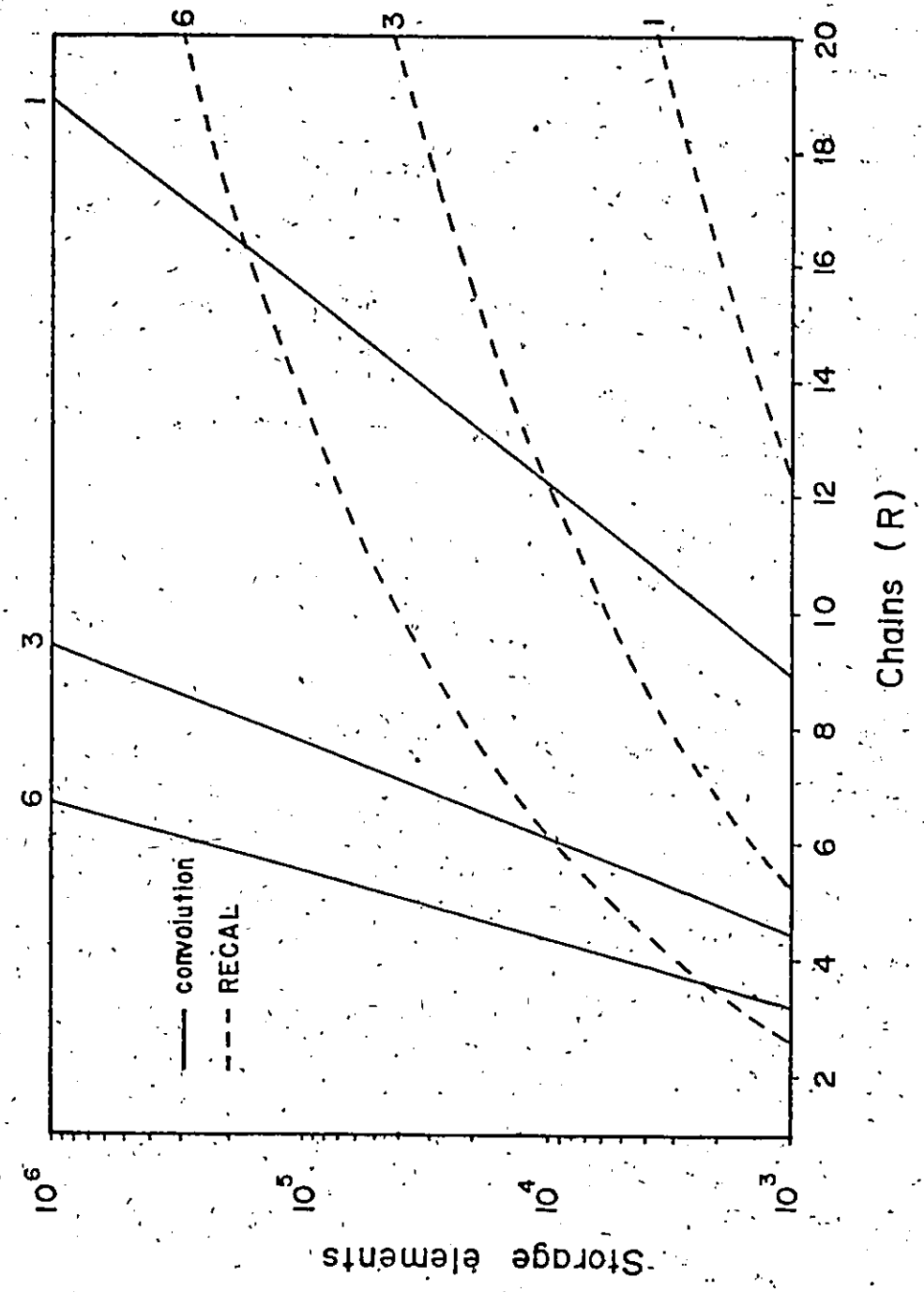

Figure 6.3 - Comparison of the storage space requirement in RECAL and convolution, for N=4 and κ=1,3 and 6. (The values for κ are indicated at the ends of the curves).

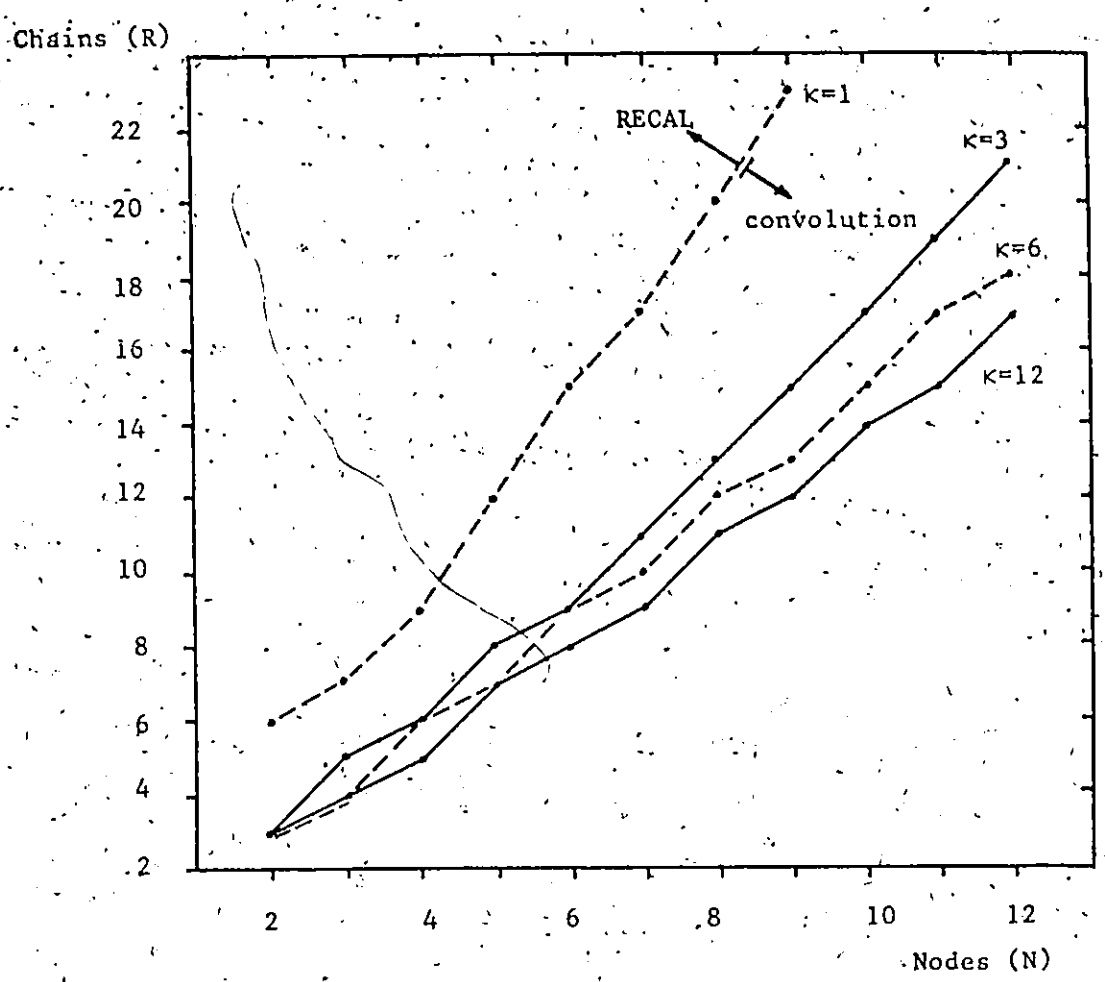

Figure 6.4 - Regions in the space (N×R) in which the time requirement (number of operations) of RECAL is less than that for convolution, for K=1,3,6 and 12. (On or above the curves, the time requirement of RECAL is less than for convolution).

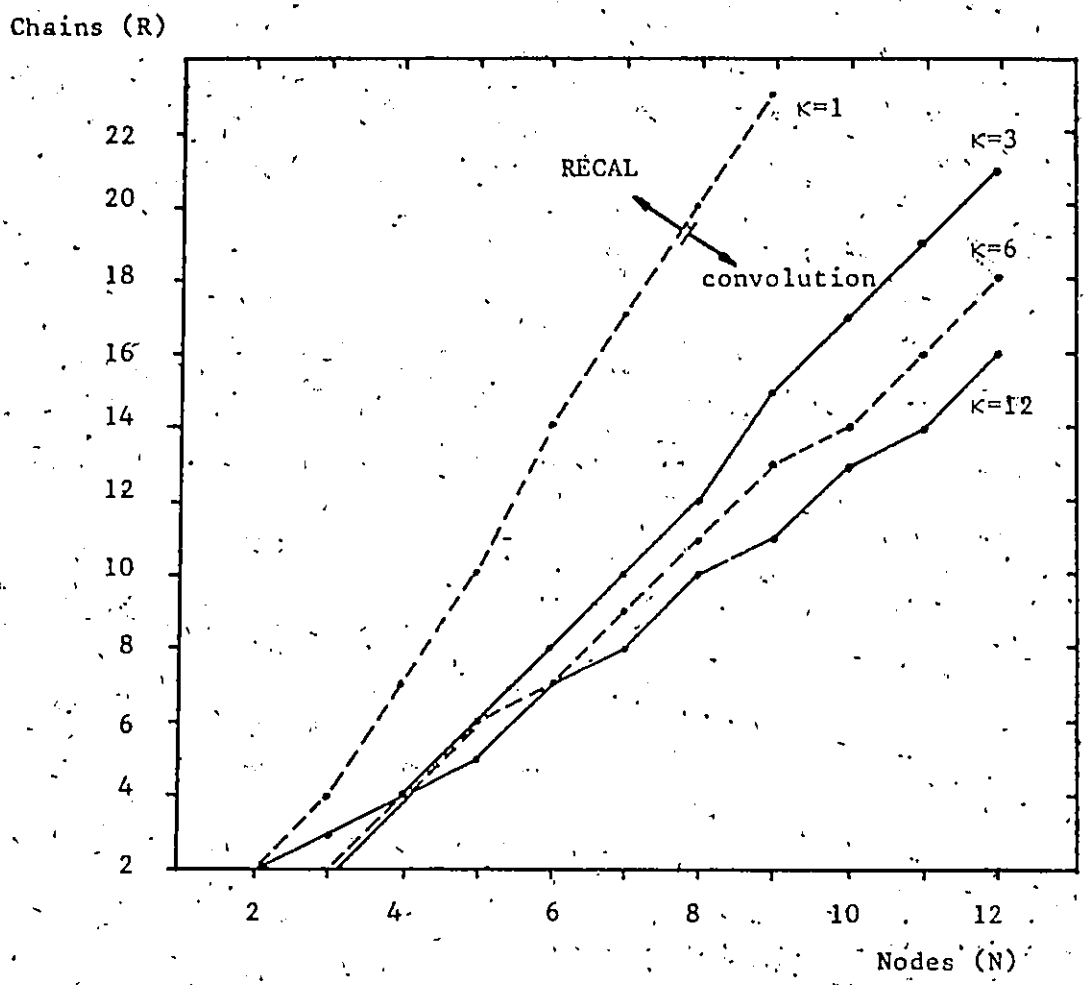

Figure 6.5 - Regions in the space (N×R) in which the space requirement (number of elements) of RECAL is less than that for convolution, for κ=1,3,6 and 12. (On or above the curves, the space requirement of RECAL is less than for convolution).

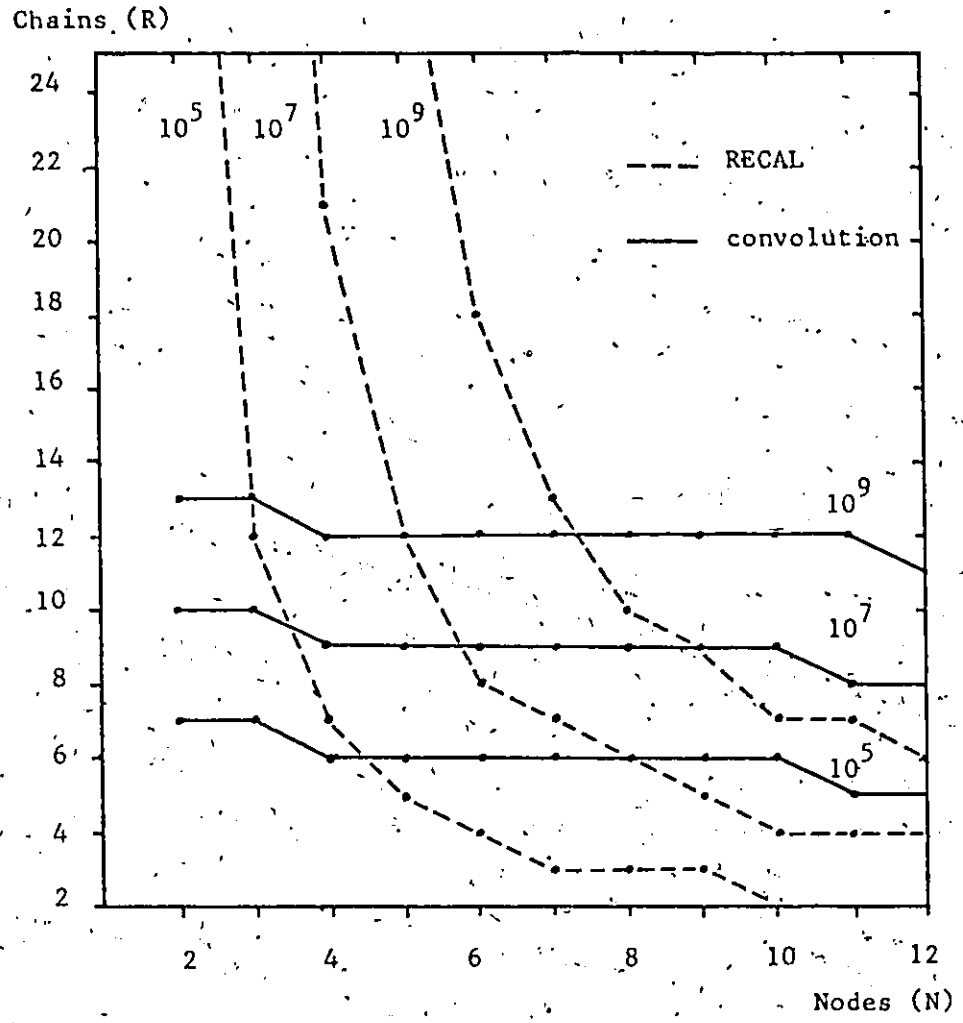

Figure 6.6 - Curves in the space (N×R) on and above which the time requirement (number of operations) is equal to or greater than the number at the end of the curve, for $\omega=3$.

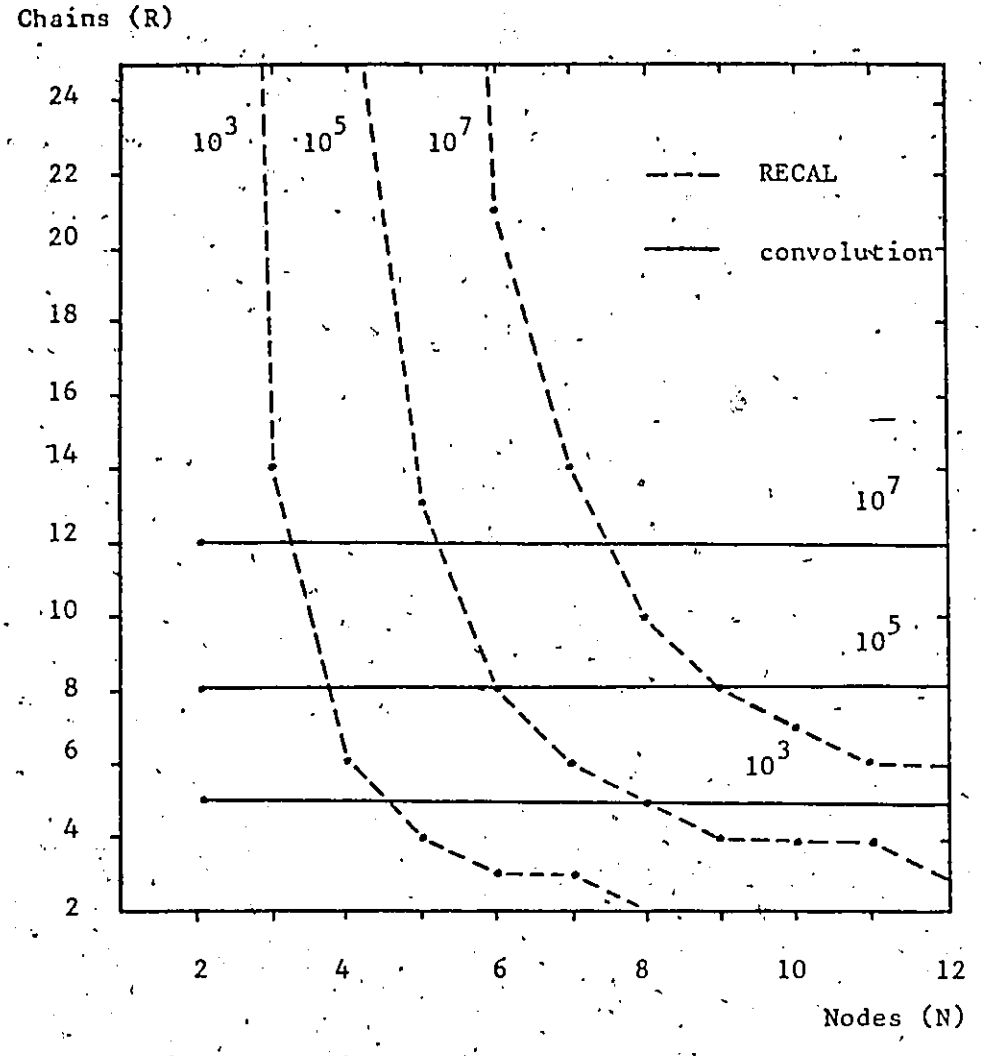

Figure 6.7 - Curves in the space (N×R) on and above which the space requirement (number of elements) is equal to or greater than the number at the end of the curve, for κ=3.

Comparing eq. 6.16 with eq. 6.1, we see that little additional complexity has been introduced to accommodate the situation of state-dependency.

The expression for the marginal distribution with respect to a particular chain R, in the case of state-dependency, is unchanged from Theorem 6.1 except that if center i is FCFS, LCFSPR or PS then

$$b_i(k_i) = w_{iR}^{k_i} \Big/ \prod_{a=1}^{k_i} \beta_i(a).$$

As a result, if $K_R = 1$, then $Q_{iR} = G_R(\underline{0})^{-1} G_{R-1}(\underline{1}_i) w_{iR}/\beta_i(1)$, so that

$$Q^o_{iR}o = G^o_R o(\underline{0})^{-1} G^o_R o_{-1}(\underline{1}_i) w_{ir(R^o)}/\beta_i(1). \qquad (6.17)$$

Under the additional condition that there is at least one IS center in the network we have, in the case of state-dependency,

$$T^o_{iR}o = e_{ir(R^o)} G^o_R o_{-1}(\underline{1}_x)/G^o_R o(\underline{0}) \qquad (6.18)$$

where x is any one of the IS centers. This expression for the throughput follows from the identity $G_{R-1}(\underline{0}) = G_{R-1}(\underline{1}_x)$ (Theorem 6.2b) and the result that, when $K_R = 1$, $T_{iR} = e_{iR} G_{R-1}(\underline{0})/G_R(\underline{0})$ (see eq. 3.7) which is known to hold <u>even</u> in the case of state-dependency [8, page 71]. We also have, using Little's result [38],

$$W^o_{iR}o = Q^o_{iR}o/T^o_{iR}o. \qquad (6.19)$$

If we assume that there is one IS center in the network and that there is state-dependency at all service centers, then the evaluation of the summation in eq. 6.16 requires (5N-4) operations. The total number of operations required by the algorithm, to obtain the queue lengths, throughputs and waiting times is then

$$(5N-4)\left(\binom{K^{o}+N-1}{N} + \binom{R+N-1}{N+1} + \binom{R+N-1}{N} - 1\right) + 8R.$$

This result can be derived in a manner analogous to that of eq. 6.10. When there is more than one IS center, the number of operations is reduced slightly since $\delta_i = 0$ when i is an IS center. The required storage space, in the case of state-dependency, is unchanged from eq. 6.12 except, of course, for the additional space required to store the network parameters $\beta_i(n)$, which will be ignored.

We finally mention that it does not appear possible for RECAL to accommodate state-dependent servers whose service rate functions are of the form $\beta_i(\underline{n}_i^{(R)})$.

6.7 MIXED NETWORKS AND CLASS SWITCHING

RECAL can be used to analyze queueing networks having certain other more general features than the ones we have considered up until now. It may be used to analyze the class of stable mixed queueing networks in which there are constant-speed or limited queue-dependent servers (defined in Section 3.5). This may be done since it is known that a mixed network can, for the purposes of analysis, be transformed

to a closed network (see Section 3.5). We may then obtain all the mean performance measures for the closed chains in the mixed network, by analyzing this closed network using RECAL, if in the original mixed network there are constant-speed servers. If there are limited queue-dependent servers in the mixed network, then we should use the version of RECAL that applies to networks with state-dependent servers (Section 6.6). The mean queue lengths can be obtained using eqs. 6.17 and 6.3. The throughputs and waiting times can be obtained using eqs. 6.18, 6.19 and 6.3, assuming that there is at least one IS center in the original mixed network. The mean queue lengths and waiting times for the open chains can be obtained readily, using eq. 3.11, if in the original mixed queueing network we have constant-speed servers. The open chain throughputs are obtained from the conservation of flow equations.

RECAL can also be used to analyze networks with customer class switching, since the mean performance measures for the individual classes can be obtained directly from the measures associated with the closed routing chains (see eq. 3.7).

6.8 DYNAMIC SCALING IN RECAL

In Section 5.8, we have seen that we may use scaling factors to avoid floating point overflows/underflows in the computation of G. We may also introduce such scaling factors into eq. 6.1, or 6.16, to avoid the same difficulties in RECAL. Let s_r^o be a scaling factor.

According to eq. 6.1 we may write

$$(s^{o}_{r(k)}G^{o}_{k}(\underline{v}_{k})) = \sum_{i=1}^{N}(1+v_{ik}\delta_{i})(s^{o}_{r(k)}w_{ir(k)})G^{o}_{k-1}(\underline{v}_{k}+\underline{1}_{i}).$$

Hence, we may scale the quantities $w_{ir(k)}$, $1 \leq i \leq N$, to reduce the possibility of encountering an overflow or underflow when computing $G^{o}_{k}(\underline{v}_{k})$ for all $\underline{v}_{k} \in I^{o}_{k}$. In the following, we shall describe the details of a scaling procedure. We assume, for the sake of simplicity, that we have a network of constant-speed servers. The same ideas, however, can be carried over to the state-dependent case.

The scaling procedure we propose for RECAL is the following. We initially (statically) scale the quantities w_{ir}, where $1 \leq i \leq N$ and $1 \leq r \leq R$, so that $\underset{i}{\text{Min}}\{w_{ir}\}=1$ for $1 \leq r \leq R$. The <u>dynamic</u> scaling to be introduced into eq. 6.1 is to scale $w_{ir(k)}$, for $1 \leq i \leq N$, as follows, <u>prior</u> to computing $G^{o}_{k}(\underline{v}_{k})$ for all $\underline{v}_{k} \in I^{o}_{k}$. If

$$\underset{\underline{v}_{k-1} \in I^{o}_{k-1}}{\text{Min}} \{G^{o}_{k-1}(\underline{v}_{k-1})\} < 1,$$

multiply $w_{ir(k)}$, for $1 \leq i \leq N$, by $10^{\zeta-\alpha}$, where ζ is the smallest exponent of the machine,

$$\alpha = \left[\log_{10} \underset{\underline{v}_{k-1} \in I^{o}_{k-1}}{\text{Min}} \{G^{o}_{k-1}(\underline{v}_{k-1})\} \right]$$

and $[x]$ is the integer portion of x. Otherwise, multiply $w_{ir(k)}$, for

$1 \leq i \leq N$, by 10^{ζ} and then by $10^{-\alpha}$. After having computed $G_k^b(\underline{v}_k)$, for all $\underline{v}_k \in I_k^o$, we then remultiply $w_{ir(k)}$, for $1 \leq i \leq N$, by $10^{-\zeta}$ and then by 10^{α} so that, once again, $\text{Min}_i\{w_{ir}\}=1$ for $1 \leq r \leq R$. If we are about to commence Step 4b, then we should leave this remultiplication until after Step 4b so that the scaling factor will cancel out properly in the computation of the mean performance measures using eqs. 6.2 and 6.3. An attractive feature of this dynamic scaling procedure is that there is no need to store any scaling factor for future use.

6.9 FORTRAN 77 IMPLEMENTATION

The version of RECAL that applies to constant speed servers (Section 6.3) has been programmed in FORTRAN 77. A listing is provided in Appendix 6.3. The program incorporates the dynamic scaling procedure described in the previous Section. It is presented in the form of a subroutine so that it could be installed in existing modeling/performance-evaluation software packages (e.g. ONAP2 [45]).

APPENDIX 6.1:

We need to show that the value of $G_{k-1}^o(\underline{y}_{k-1})$ at location $M_k(\underline{v}_k)$ is not required in the computation of $G_k^o(\underline{\gamma})$, where $\underline{\gamma}$ is any vector such that $\underline{\gamma} \in I_k^o$ and $M_k(\underline{\gamma}) > M_k(\underline{v}_k)$.

Proof:

As can be seen from eq. 6.1, the computation of $G_k^o(\underline{\gamma})$ requires the values of $G_{k-1}^o(\underline{\gamma}+\underline{1}_i)$ for $1 \leq i \leq N$. We need to show that the value of

$G^{o}_{k-1}(\underline{v}_{k-1})$ at location $M_k(\underline{v}_k)$ is not one of these. Hence, we need to show that $M_{k-1}(\underline{\gamma}+\underline{1}_i) > M_k(\underline{v}_k)$ for any $1 \le i \le N$ and $\underline{\gamma}_i$ such that $\underline{\gamma} \in I^{o}_{k}$ and $M_k(\underline{\gamma}) > M_k(\underline{v}_k)$.

We assume that $M_{k-1}(\underline{\gamma}+\underline{1}_i) \le M_k(\underline{v}_k)$ and show that this leads to a contradiction.

By the definition of $\underline{\gamma}$, $M_k(\underline{\gamma}) > M_k(\underline{v}_k)$ so that $M_{k-1}(\underline{\gamma}+\underline{1}_i) < M_k(\underline{\gamma})$. Therefore, using eq. 6.11, we have

$$(K^{o}-k+1)^{i-1} + \sum_{j=1}^{N}{}' \gamma_j (K^{o}-k+1)^{j-1} < \sum_{j=1}^{N} \gamma_j (K^{o}-k)^{j-1} < \sum_{j=1}^{N} \gamma_j (K^{o}-k+1)^{j-1}$$

Hence $(K^{o}-k+1)^{i-1} < 1$. Now $1 \le k \le K^{o}$ and $1 \le i \le N$ so there is a contradiction.

APPENDIX 6.2:

Suppose, for the sake of simplicity, that $K_r = K$ for $1 \le r \le R$. When we break down the chains in $B^{(R)}(N,\underline{K})$, so that each sub-chain consists of one customer, the number of operations required to arrive at G^{o}, using the convolution algorithm is, according to eq. 3.4, $2RK(N-1)2^{RK}$. Now

$$\frac{2RK(N-1)2^{RK}}{2R(N-1)(K+1)^{R}} = \frac{K(2^{K})^{R}}{(K+1)^{R}} > K > 1$$

since $2^{K} > K+1$. Hence no advantage is obtained in the number of operations.

The storage space required in the convolution algorithm to obtain G^{o} is, using eq. 3.5, $2^{R\kappa+1}$. Now

$$2^{R\kappa+1} / 2(\kappa+1)^{R} = (2^{\kappa})^{R}/(\kappa+1)^{R} > 1.$$

Hence, no advantage in the storage space is obtained either.

APPENDIX 6.3:

```
      subroutine recal(nnode,nchain,e,t,ipop,ind,rmir,
     *sg,u,utim,space,store,lestate,imax,ig,igdim,istdim,
     *thru,qlen,wait,util)
c*****************************************************************
c                                                                *
c  Subroutine recal computes the mean performance measures for   *
c  product-form multiple chain closed queueing networks          *
c  with constant speed servers. The subroutine incorporates      *
c  a cyclic scaling algorithm.                                   *
c  (the theory of recal is given in INRIA report 373, March      *
c  1985)                                                         *
c                                                                *
c  input:                                                        *
c                                                                *
c  nnode       - number of service centers in the network        *
c  nchain      - number of closed routing chains                 *
c  e(i,r)      - visit ratio for chain r customers at node i     *
c  t(i,r)      - mean service time for chain r customers at      *
c                node i                                          *
c  ipop(r)     - population of chain r                           *
c  ind(i)      - indicator function                              *
c                ind(i)=1 if node i is FCFS, LCFSPR or PS        *
```

```
c                 inc(i)=0 if node i is IS                              *
c   rmin         - the smallest real of the machine running            *
c                 real                                                  *
c                                                                       *
c   workspace used by recal:                                           *
c                                                                       *
c   gg            - vector of normalization constants with            *
c                   dimension iggdim                                    *
c   w             - workspace of dimension (nnode,nchain)             *
c   wmin          - workspace of dimension (nchain)                   *
c   space         - workspace of dimension (nnode)                    *
c   store         - workspace of dimension (istdim)                   *
c   iustor        - workspace of dimension (nnode+1)                  *
c   imax          - workspace of dimension (nchain)                   *
c   ix            - workspace of dimension (nnode)                    *
c   iggdim        - the dimension of gg                               *
c                   iggdim=(totalnop+nnode-1)                         *
c                                 combination (nnode-1)               *
c                   where totalnop=the population of the network      *
c   istdim        - the dimension of store                            *
c                   istdim=(nchain+nnode-1)                           *
c                                 combination (nnode-1)               *
c                                                                       *
c   output:                                                             *
c                                                                       *
c   thru(i,r)     - throughput of chain r customers at node i           *
c   clen(i,r)     - mean number of chain r customers at node i          *
c                   (in queue or in service)                            *
c   wait(i,r)     - mean waiting time (queueing+service) for            *
c                   chain r customers at node i                         *
c   util(i,r)     - utilization of the server at node i by              *
c                   customers of chain r                                *
c                                                                       *
c***********************************************************************
      real e(nnode,nchain),t(nnode,nchain),thru(nnds,nchain)
      real clen(nnode,nchain),wait(nnode,nchain),gg(iggdim),rmin
      real util(nnode,nchain),w(nnode,nchain),x,min9,sum9
      real wmin(nchain),sum,space(nnode),store(istdim)
      integer iuser,iuse(nchain),ino(nnode),nnode,nchain,ir
      integer ipp,iustor(nnode+1),itmop,ixxx,ie
      integer ic,io,imax(nchain),ip,inc,i,j,k,iflag,ix(nnode),k3
      integer itmop9,ic9,j9,ixxx9,k9,il,izzz9,iggdim,istdim
      ip=0
      do 1 i=1,nnode
      if(ino(i).eq.1) then
      ip=ip+1
      do 2 j=1,nchain
      w(ip,j)=e(i,j)*t(i,j)
2     continue
      end if
1     continue
      if(ip.lt.nnode) then
      k=ip
      do 3 i=1,nnode
      if(ino(i).eq.0) then
      k=k+1
```

```
      do 4 j=1,nchair
      a(k,j)=e(i,j)*t(i,j)
    4 continue
      end if
    3 continue
      end if
      ipp=ip+1
      itpop=0
      do 44 i=1,nchair
      itpop=itpop+ipop(i)
   44 continue
      npop=itpop
      ixxx=iccnb(itpop+nnode-1,nnode-1)
      do 5 i=1,ixxx
      gg(i)=1.0
    5 continue
      do 6 j=1,nchair
      do 7 k=1,nnode
      if(a(k,j).gt.0.0) then
      amin(j)=a(k,j)
      goto 8
      end if
    7 continue
    8 do 9 k=1,nnode
      if(a(k,j).lt.amin(j).and.a(k,j).gt.0.0) amin(j)=a(k,j)
    9 continue
      do 10 k=1,nnode
      a(k,j)=a(k,j)/amin(j)
   10 continue
    6 continue
      do 11 i=1,nchair
      imax(i)=ipop(i)-1
   11 continue
      ir=1
   12 continue
      if(ir.gt.nchair) goto 5555
      ic=1
   13 continue
      if(ic.gt.imax(ir)) goto 6666
      icpc=itpop
      itpop=itpop-1
      iflag=1
      goto 7615
  341 continue
      ic=ic+1
      goto 13
 6666 continue
      ir=ir+1
      goto 12
 5555 continue
      ixxx=iccnb(nchair+nnode-1,nnode-1)
      do 14 ic=1,ixxx
      store(ic)=gg(ic)
   14 continue
      ic=1
   99 continue
```

```
      if(ic.gt.nchain) goto 9999
      iepo=nchain-ic+1
      iux=ic+1
   16 continue
      if(iux.gt.nchain) goto 9999
      ir=nchain+ic+1-iux
      iflag=2
      goto 7613
  342 continue
      iepo=iepo-1
      iux=iux+1
      goto 16
 9999 continue
      do 17 k=1,nnode
      space(k)=gg(k)
   17 continue
      ir=ic
      iflag=3
      goto 7613
  343 continue
      sum=0.0
      do 18 i=1,nnode
      sum=sum+space(i)
   18 continue
      do 19 k=1,nnode
      if(k.gt.ip) then
      ia=0
      do 22 j=1,nnode
      if(ino(j).eq.1) ia=ia+1
      if(ia.eq.k) then
      i=j
      goto 222
      end if
   22 continue
      end if
      if(k.gt.ip) then
      ia=ip
      do 23 j=1,nnode
      if(ino(j).eq.0) ia=ia+1
      if(ia.eq.k) then
      i=j
      goto 222
      end if
   23 continue
      end if
  222 continue
      if(ip.eq.nnode) then
      thru(i,ir)=e(i,ir)*sum/(gg(1)*umin(ir)*float(nnop+nnode-1))
      end if
      if(ip.lt.nnode) then
      thru(i,ir)=e(i,ir)*space(inp)/(gg(1)*umin(ir))
      end if
      util(i,ir)=thru(i,ir)*t(i,ir)
      clen(i,ir)=u(k,ir)*space(k)/gg(1)
      x=(rmir/exp(float(iepo)*log(10.0)))*inop(ir)
      util(i,ir)=util(i,ir)*x
```

```
      qlen(i,ir)=clen(i,ir)*x
      thru(i,ir)=thru(i,ir)*x
      if(thru(i,ir).gt.0.0) then
      wait(i,ir)=qlen(i,ir)/thru(i,ir)
      else
      wait(i,ir)=0.0
      end if
49    continue
      iopc=nchain-ic+1
      ixxx=iconb(iopc+nnode-1,nnode-1)
      do 20 k=1,ixxx
      ss(k)=store(k)
20    continue
      iflag=4
      goto 7615
344   continue
      ixxx=iconb(iopc+nnode-2,nnode-1)
      do 21 k=1,ixxx
      store(k)=ss(k)
21    continue
      ic=ic+1
      goto 99
7615  itmpp=iopc-1
      ixxx=iconb(itmpp+nnode-1,nnode-1)
      izzz=iconb(iopc+nnode-1,nnode-1)
      do 51 k=1,izzz
      if(ss(k).gt.0.0) then
      min=ss(k)
      goto 52
      end if
51    continue
52    do 53 k=1,izzz
      if(ss(k).gt.0.0.and.ss(k).lt.min) min=ss(k)
53    continue
      ispow=int(log10(rnir))
      do 54 k=1,izzz
      ss(k)=ss(k)*(rnir/exp(float(ispow)*log(10.0)))
54    continue
      iq=0
      do 55 k=1,nnode
      i(k)=0
55    continue
      i(1)=itpop
      goto 1465
246   nz=0
      do 56 k=1,nnode
      if(i(k).gt.0) then
      nz=k
      goto 57
      end if
56    continue
57    i(nz+1)=i(nz+1)+1
      i(1)=i(nz)-1
      if(nz.gt.1) i(nz)=0
1465  iq=iq+1
      sum=0.0
```

```
      do 59 k9=1,nnoce
      iustate(k9)=i9(k9)
59    continue
      iustate(1)=iustate(1)+1
      call count(iopc,nnoce,iustate,ic)
      sum9=sum9+float(i9(1)+1)*u(1,ir)*gg(ic)
      if(ip.gt.1) then
      do 510 j9=2,ip
      iustate(j9)=iustate(j9)+1
      iustate(j9-1)=iustate(j9-1)-1
      call count(iopc,nnoce,iustate,ic)
      sum9=sum9+float(i9(j9)+1)*u(j9,ir)*gg(ic)
510   continue
      end if
      if(ip.le.nnoce) then
      do 511 j9=ip,nnoce
      iustate(j9)=iustate(j9)+1
      iustate(j9-1)=iustate(j9-1)-1
      call count(iopc,nnoce,iustate,ic)
      sum9=sum9+u(j9,ir)*gg(ic)
511   continue
      end if
      gg(iq9)=sum9
      if(iq9.eq.ixxx9) goto 999
      goto 248
999   continue
      goto(341,342,343,344),iflag
9999  continue
      return
      end
      function icomb(n1,n2)
      integer n1,n2,i
      real x
      x=1.0
      if(n1.ge.2*n2) then
      do 1 i=n1-n2+1,n1
      x=x*float(i)
1     continue
      icomb=int((x/fact(n2))+0.111)
      else if(n1.eq.n2) then
      icomb=1
      else
      do 2 i=n2+1,n1
      x=x*float(i)
2     continue
      icomb=int((x/fact(n1-n2))+0.111)
      end if
      return
      end
      function fact(n)
      integer n,i
      real x
      x=1.0
      if(n.gt.1) then
      do 1 i=1,n
      x=x*float(i)
```

```
1     continue
      fact=x
      else
      fact=1.0
      end if
      return
      end
      subroutine count(iopc,nnode,iustate,iu)
      real prod,c,sum1
      integer n,r,ia,ix,iz,isum,izzz,k
      integer iopc,nnode,iustate(nnode),iu
      n=nnode+1
      do 1 k=1,nnode+1
      j=nnode+2-k
      if(iustate(j).ec.0) n=j-1
      if(iustate(j).gt.0) goto 2
1     continue
2     if(n.ec.1) then
      iu=1
      goto 999
      end if
      if(n.ge.2) then
      prod=1.0
      do 3 ia=1,iustate(n)
      ix=iopc-iustate(n)+ia
      prod=prod*float(ix+n-1)/float(ix)
3     continue
      prod=prod-1.0
      d=prod*float(idcmb(iopc-iustate(n)+n-1,n-1))
      if(n.ec.2) then
      iu=int(c+1.1)
      goto 999
      end if
      sum1=0.0
      do 4 iz=2,n-1
      isum=0
      do 5 ia=iz+1,n
      isum=isum+iustate(ia)
5     continue
      izzz=isum
      prod=1.0
      if(iustate(iz).gt.0) then
      do 6 ia=1,iustate(iz)
      ix=iopc-izzz-iustate(iz)+ia
      prod=prod*float(ix+iz-1)/float(ix)
6     continue
      end if
      prod=prod-1.0
      prod=prod*float(idcmb(iopc-izzz-iustate(iz)+iz-1,iz-1))
      sum1=sum1+prod
4     continue
      iu=int(c+sum1+1.11)
      end if
999   continue
      return
      end
```

CHAPTER 7 - AN EFFICIENT ALGORITHM FOR SEMI-HOMOGENEOUS QUEUEING NETWORKS

7.1 INTRODUCTION

In Section 1.5, we have outlined how one may model distributed computer systems and local area networks of workstations using queueing network models. It is only quite recently that the analysis of such models has been undertaken. These recent modeling efforts give rise to queueing networks that contain both many service centers and closed routing chains, the latter lacking any sparsity or locality properties. Unfortunately, in these circumstances, the general purpose exact algorithms, for the solution of product-form queueing networks, are prohibitively expensive in terms of their time and space requirements. The convolution and MVA algorithms have requirements which are exponential in the number of routing chains. The tree convolution algorithm [32] cannot be used to advantage since there is no sparsity or locality in the routing. The recursion by chain algorithm (RECAL) has a complexity which is combinatorial in the number of service centers and routing chains. It is therefore impractical, in general, to analyze queueing network models of large distributed systems, or of local area networks, using the general purpose exact algorithms.

In view of these difficulties, one approach which has been adopted is to use approximate iterative techniques that are based on the exact MVA equations. Goldberg et al [18] make use of the Schweitzer

approximation [56] to analyze the model of a distributed system of computers by the name of LOCUS. In [17], the approximate iterative solution of a sequence of subnetworks is used to analyze models of the LOCUS type as well as queueing network models of packet-switching networks. Another approach, which has been applied to the analysis of product-form models of local area networks of personal workstations, has been to reduce the complexity of the exact solution techniques by exploiting the structure which has been imposed on the model.

Balbo et al [3] have defined the class of homogeneous queueing networks for modeling local area networks. In such models, we associate a queue with each workstation and each shared resource (i.e. bus, fileserver) in the network. Associated with each workstation is a closed routing chain containing a single customer. This customer visits all of the remote workstations in an homogeneous fashion. The demands placed on the various shared resources are homogeneous with respect to the customers. Balbo et. al [3] exploit the inherent symmetry of homogeneous queueing networks to reduce the exact analysis of this P-chain queueing network, by P-chain flow-equivalent aggregation (see Section 4.5), to simply that of a sequence of 3-chain flow-equivalent aggregation steps. The overall space and time requirements of this method are, respectively, of the order of P^4 and P^5, where P is the number of workstations [3]. This type of approach is an attractive alternative to approximate iterative techniques.

In this Chapter, we present an efficient algorithm for the exact solution of a class of queueing networks which is more general than

the homogeneous class considered in [3]. The assumptions of homogeneity of Balbo et al are relaxed to allow for each customer to visit the shared resources in a different fashion. We use the term sem.-homogeneous to describe this more general class of queueing networks. The motivation for this generalization stems from the fact that, in reality, the various users at the workstations in a local area network place different demands on the shared resources. Balbo et al have taken this point into consideration by extending their original algorithm to the case of multiple clusters of workstations [2]. In this case, only customers belonging to the same cluster need use the shared resources in an identical fashion. This extended algorithm has a computational complexity which is exponential in twice the number of clusters [2]. If, in the extreme, one associates an individual cluster with each workstation, so that each customer may visit the shared resources in a different fashion, then this extended algorithm is inferior to the convolution algorithm or MVA (it is superior when there are at least 3 stations in each cluster [2]). The algorithm to be presented here is, however, designed to accommodate this generality.

The mathematical basis of the new algorithm is provided by the equations which form the foundation of RECAL (Chapter 6). A direct application of RECAL, to the class of semi-homogeneous queueing networks, would in general entail time and space requirements greater than those of either convolution or MVA. The assumptions of semi-homogeneity, however, give rise to much redundancy in the recursive equation which is used by RECAL. We formally establish the

nature of this redundancy and exploit its structure to develop an efficient algorithm specifically tailored for the class of semi-homogeneous queueing networks. The computational complexity of our algorithm is closely related to the number of unordered partitions of P. We are able to show that the order of the complexity of our algorithm is less than exponential in $(P-1)^2$. This establishes the effectiveness of our procedure over the direct application of a general purpose algorithm.

The Chapter is organized as follows. In the following Section, we define the class of semi-homogeneous queueing networks. Section 7.3 presents certain preliminary results which will provide the mathematical basis of the new algorithm. The algorithm is then presented in Section 7.4. An analysis of the complexity follows in Section 7.5.

7.2 SEMI-HOMOGENEOUS QUEUEING NETWORKS

Balbo et al [3] have defined the class of homogeneous queueing networks for modeling local area networks with P workstations (processors) and S shared resources. In the class of homogeneous queueing networks, the following constraints are placed on the parameters of $B^{(R)}(N,\underline{K})$. It is implicitly assumed that the service disciplines at nodes 1,...,P are among FCFS, LCFSPR and PS (all these disciplines give rise to identical marginal state distributions). The service discipline at nodes P + 1,..., P + S may be any among FCFS, LCFSPR, PS or IS. The number of closed routing chains is R = P. The

visit ratios are

$$e_{ir} = \begin{cases} d_1 \text{ if } i \neq r,\ 1 \leq r \leq P,\ 1 \leq i \leq P, \\ d_2 \text{ if } i = r,\ 1 \leq r \leq P, \\ d_i \text{ if } P + 1 \leq i \leq P + S,\ 1 \leq r \leq P. \end{cases}$$

The mean service requirements are

$$t_{ir} = \begin{cases} \tau \text{ if } 1 \leq i \leq P,\ 1 \leq r \leq P, \\ \tau_i \text{ if } P + 1 \leq i \leq P + S,\ 1 \leq r \leq P. \end{cases}$$

Let $w_1 = d_1\tau$, $w_2 = d_2\tau$ and $s_{(i-P)} = d_i\tau_i$. Then, in an homogeneous queueing network, the relative traffic intensities are given by

$$w_{ir} = \begin{cases} w_1 \text{ if } i \neq r,\ 1 \leq r \leq P,\ 1 \leq i \leq P, \\ w_2 \text{ if } i = r,\ 1 \leq r \leq P, \\ s_{(i-P)} \text{ if } P + 1 \leq i \leq P + S,\ 1 \leq r \leq P. \end{cases} \tag{7.1}$$

In the local area network models considered by Balbo et al, it is further assumed that $K_r = 1$, for $1 \leq r \leq R$.

We now define the class of semi-homogeneous queueing networks.

Definition 7.1:

$B^{(R)}(N,K)$ is a semi-homogeneous queueing network if

$$w_{ir} = \begin{cases} w_1 \text{ if } i \neq r,\ 1 \leq r \leq P,\ 1 \leq i \leq P, \\ w_2 \text{ if } i = r,\ 1 \leq r \leq P, \\ s_{(i-P)r} \text{ if } P+1 \leq i \leq P+S,\ 1 \leq r \leq P, \end{cases}$$

R = P and centers 1,...,P have either a FCFS, LCFSPR or PS service discipline.

As can be seen from eq. 7.1 and Definition 7.1, semi-homogeneous networks only differ from homogeneous networks in the conditions imposed on the relative traffic intensities to the shared resources. The class of semi-homogeneous networks cannot be analyzed using the algorithm proposed by Balbo et al in [3]. Nor can it be analyzed in an efficient manner using the extended algorithm [2] that can accommodate multiple clusters of workstations. In the modeling of local area networks using semi-homogeneous queueing networks we shall assume, as in [3], that $K_r = 1$ for $1 \leq r \leq P$.

In the following Section, we present certain preliminary results which are applicable to semi-homogeneous queueing networks. These will provide the mathematical basis for the algorithm to be presented in Section 7.4.

7.3 RESULTS FOR SEMI-HOMOGENEOUS QUEUEING NETWORKS

Corollary 6.1 is the mathematical foundation of RECAL. The new algorithm, for the solution of semi-homogeneous queueing networks, is also based on Corollary 6.1. We exploit the redundancy which arises in the recursive expression when the assumptions of semi-homogeneity are introduced. More specifically, for a queueing network of type $B^{(R)}(N,\underline{K})$, in general, $G_r(\underline{v}_r)$ must be computed for all $\underline{v}_r \in I_r$. However, for semi-homogeneous networks, the domain in the computation of $G_r(\underline{v}_r)$ need only consist of those elements of I_r for which $G_r(\underline{v}_r)$ is distinct. The following two Theorems establish formally the structure of this redundancy.

Theorem 7.1: If $B^{(R)}(N,\underline{K})$ is a semi-homogeneous queueing network, $K_s = 1$ for $1 \leq s \leq P$ and $1 \leq r \leq P$, then

$$G_r(\underline{c}) = G_r(c_1,\ldots,c_r,u_1,\ldots,u_{P-r},c_{P+1},\ldots,c_N)$$

for all $(u_1,\ldots,u_{P-r}) \in U_r(c_{r+1},\ldots,c_P)$

where $\underline{c} = (c_1,\ldots,c_r,c_{r+1},\ldots,c_P,c_{P+1},\ldots,c_N)$ and $U_r(c_{r+1},\ldots,c_P) = \{(u_1,\ldots,u_{P-r}): u_i \geq 0 \text{ for } 1 \leq i \leq P-r; \sum_{i=1}^{P-r} u_i = \sum_{i=r+1}^{P} c_i\}$.

Proof:

According to eq. 3.1 $G_r(\underline{c}) = (f_1 * \ldots * f_N)(\underline{K}_r)$ where * is the r-dimensional convolution operation, $\underline{K}_r = (K_1,\ldots,K_r)$ and

$$f_i(\underline{n}_i^{(r)}) = \begin{cases} \left\{ \prod_{a=1}^{n_i^{(r)}} (a+c_i) \right\} \prod_{s=1}^{r} w_{is}^{n_{is}} / n_{is}!, & \text{if node } i \text{ is FCFS,} \\ & \text{LCFSPR or PS,} \\ \prod_{s=1}^{r} w_{is}^{n_{is}} / n_{is}!, & \text{if node } i \text{ is IS,} \end{cases}$$

where $\underline{n}_i^{(r)} = (n_{i1},\ldots,n_{ir})$, $n_i^{(r)} = \sum_{s=1}^{r} n_{is}$ and n_{is} is the number of customers at node i belonging to chain s. Now $n_{is} = 0$ or 1 since it is assumed that $K_s = 1$ for $1 \leq s \leq r$ and $w_{is} = w_1$ when $1 \leq s \leq r$ and $(r+1) \leq i \leq P$, so that if $(r+1) \leq i \leq P$, then

$$f_i(\underline{n}_i^{(r)}) = \left\{ \prod_{a=1}^{n_i^{(r)}} (a + c_i) \right\} \prod_{s=1}^{r} w_{is}^{n_{is}} / n_{is}!$$

$$= \prod_{a=1}^{n_i^{(r)}} (aw_1 + w_1 c_i) \qquad (7.2)$$

Let $g_i(\underline{\ell}) = \prod_{a=1}^{\ell} (aw_1 + w_1 c_i)$ and $h(\underline{\ell}) = (g_i * g_{i+1})(\underline{\ell})$ where $\underline{\ell} = (\ell_1,\ldots,\ell_r)$ and

$$\ell = \sum_{s=1}^{r} \ell_s.$$

Then

$$h(\underline{\ell}) = \sum_{k_r=0}^{\ell_r} \cdots \sum_{k_1=0}^{\ell_1} w_1^{\ell} \left\{ \prod_{a=1}^{(k_1+\ldots+k_r)} (a+c_i) \right\} \left\{ \prod_{b=1}^{(\ell-k_1-\ldots-k_r)} (b+c_{i+1}) \right\}.$$

Since the components of $\underline{\ell}$ are either zero or one, we may write

$$h(\underline{\ell}) = \sum_{k=0}^{\ell} \binom{\ell}{k} w_1^{\ell} \{\prod_{a=1}^{k} (a+c_i)\}\{\prod_{b=1}^{\ell-k} (b+c_{i+1})\}.$$

Simplifying [51], we obtain

$$h(\underline{\ell}) = w_1^{\ell}\, \ell! \sum_{k=0}^{\ell} \binom{k+c_i}{c_i} \binom{\ell-k+c_{i+1}}{c_{i+1}}$$

$$= w_1^{\ell}\, \ell! \binom{\ell+c_i+c_{i+1}+1}{\ell}$$

$$= \prod_{a=1}^{\ell} (aw_1 + w_1(c_i + c_{i+1} + 1)). \qquad (7.3)$$

Hence, using eqs. 7.2 and 7.3, we may write

$$(f_{r+1} * \ldots * f_p)(\underline{\ell}) = \prod_{a=1}^{\ell} (aw_1 + w_1(c_{r+1} + \ldots + c_p + p - r - 1)).$$

It now follows that

$$(f_{r+1} * \ldots * f_p)(\underline{\ell}) = \prod_{a=1}^{\ell} (aw_1 + w_1(u_1 + \ldots + u_{p-r} + p - r - 1))$$

for all $(u_1, \ldots, u_{p-r}) \in U_r(c_{r+1}, \ldots, c_p)$. The Theorem is now proved since, by the associativity and commutativity of convolution,

$$G_r(\underline{c}) = ((f_{r+1} * \ldots * f_p) * (f_1 * \ldots * f_r * f_{p+1} * \ldots * f_N))(\underline{K}_r).$$

Theorem 7.2: If $B^{(R)}(N,\underline{K})$ is a semi-homogeneous queueing network and $1 \leq r \leq R$, then

$$G_r(\underline{c}) = G_r(p_1,\ldots,p_r,c_{r+1},\ldots,c_N)$$

for all $(p_1,\ldots,p_r) \in P_r(c_1,\ldots,c_r)$ where $\underline{c} = (c_1,\ldots,c_r,c_{r+1},\ldots,c_N)$ and

$$P_r(c_1,\ldots,c_r) = \{(p_1,\ldots,p_r): (p_1,\ldots,p_r) \text{ is a permutation of } (c_1,\ldots,c_r)\}.$$

Proof: (We use some notation and definitions given in the Proof of Theorem 7.1). By the assumptions of semi-homogeneity, if $1 \leq i \leq r$, then

$$f_i(\underline{n}_i^{(r)}) = \Big\{\prod_{a=1}^{n_i^{(r)}} (a+c_i)\Big\} \prod_{s=1}^{r} w_{is}^{n_{is}} / n_{is}!$$

$$= w_2^{n_{ii}} w_1^{(n_i^{(r)}-n_{ii})} \prod_{a=1}^{n_i^{(r)}} (a+c_i).$$

Let $g_i^{(x)}(\ell) = w_2^{\ell_i} w_1^{\ell-\ell_i} \prod_{a=1}^{\ell} (a+x)$. It now follows that

$$(g_1^{(p_1)} *\ldots* g_r^{(p_r)})(\underline{\ell}) = (f_1*\ldots*f_r)(\underline{\ell})$$

for all $(p_1,\ldots,p_r) \in P_r(c_1,\ldots,c_r)$. The Theorem is now proved since $G_r(\underline{c}) = ((f_1*\ldots*f_r)*(f_{r+1}*\ldots*f_N))(\underline{K}_r)$.

Corollary 7.1: If $B^{(R)}(N,\underline{K})$ is a semi-homogeneous queueing network and $K_s = 1$ for $1 \leq s \leq r$, then

$$G_r(p_1,\ldots,p_r,u_1,\ldots,u_{P-r},c_{P+1},\ldots,c_N) = G_r(\underline{c})$$

for all $(p_1,\ldots,p_r) \in P_r(c_1,\ldots,c_r)$, $(u_1,\ldots,u_{P-r}) \in U_r(c_{r+1},\ldots,c_P)$ where $1 \leq r \leq P$.

Proof: The Corollary follows directly from Theorems 7.1 and 7.2.

The immediate consequence of Corollary 7.1 is that in the computation of $G_R(\underline{0})$, using Corollary 6.1, instead of computing $G_r(\underline{v}_r)$ for all $\underline{v}_r \in I_r$, we need only compute $G_r(\underline{v}_r)$ for those vectors $\underline{v}_r$ where, informally, either $(v_{(P+1)r},\ldots,v_{Nr})$ is different, or $\sum_{i=r+1}^{P} v_{ir}$ is different, or where $(v_{1r},\ldots,v_{rr})$ is a different unordered partition of the quantity $\sum_{i=1}^{r} v_{ir}$. In Theorem 7.3 below, we state formally the recursion which results when we take advantage of Corollary 7.1 to reduce the complexity of the recursion contained in Corollary 6.1. We first require some further notation and definitions.

Definition 7.2:

(a) An unordered partition of a positive integer y into n parts is [21] a vector of positive integers $(a_1,\ldots,a_n)$ such that $\sum_{i=1}^{n} a_i = y$ and $a_1 \geq \ldots \geq a_n \geq 1$.

(b) $\underline{\pi}_r = (\pi_1, \ldots, \pi_r)$

(c) $(\pi_1, \ldots, \pi_r)^- = (\pi_1, \ldots, \pi_{r-1})$

(d) $$B_r(y) = \begin{cases} \{\underline{\pi}_r : (\pi_1, \ldots, \pi_n) \text{ is an unordered partition of } y \text{ into } n \\ \qquad \text{parts; } 1 \leq n \leq r; (\pi_{n+1}, \ldots, \pi_r) = 0\}, & \text{if } y > 1, \\ \{\underline{0}\}, & \text{if } y = 0. \end{cases}$$

(e) $\underline{v}^*$ is the vector formed from the elements of $\underline{v}$ by arranging them in decreasing order.

(f) $H_r(\alpha, \underline{\beta}, \underline{\gamma}) = G_r(\beta_1, \ldots, \beta_r, \alpha, 0, \ldots, 0, \gamma_1, \ldots, \gamma_S)$
where $\underline{\beta} = (\beta_1, \ldots, \beta_r)$ and $\underline{\gamma} = (\gamma_1, \ldots, \gamma_S)$.

Theorem 7.3: If $B^{(R)}(N, \underline{K})$ is a semi-homogeneous queueing network, then $G_R(\underline{0}) = H_R(0, \underline{0}, \underline{0})$ is given recursively by

$$\begin{aligned} H_r(x, \underline{\pi}_r, \underline{d}) = {} & w_1 \sum_{i=1}^{r-1} (1 + \pi_i) H_{r-1}(x + \pi_r, ((\underline{\pi}_r + \underline{1}_i)^-)^*, \underline{d}) \\ & + (w_1(P - r + x) + w_2 + w_2 \pi_r) H_{r-1}(x + \pi_r + 1, \underline{\pi}_r^-, \underline{d}) \\ & + \sum_{i=1}^{S} (1 + d_i \delta_{i+P}) s_{ir} H_{r-1}(x + \pi_r, \underline{\pi}_r^-, \underline{d} + \underline{1}_i) \end{aligned} \tag{7.4}$$

for $1 \leq r \leq R$ and $(x, \underline{\pi}_r, \underline{d}) \in A_r$

where δ_i was defined in Corollary 6.1

$$A_r = \{(x, \underline{\pi}_r, \underline{d}): \ 0 \leq x \leq R-r;\ 0 \leq y \leq R-r-x \text{ if } r \geq 1;\ y=0 \text{ if } r=0;$$
$$\underline{\pi}_r \in B_r(y);\ d_i > 0 \text{ for } 1 \leq i \leq S;$$
$$\sum_{i=1}^{S} d_i = R-r-x-y\},$$

$\underline{d} = (d_1,\ldots,d_S)$, $S = N-P$, and the initial conditions are $H_0(x,\underline{\pi}_0,\underline{d}) = 1$ for all $(x,\underline{\pi}_0,\underline{d}) \in A_0$.

Proof:

By Corollary 6.1,

$$G_r(\underline{v}_r) = \sum_{i=1}^{N} (1+v_{ir}\delta_i)\, w_{ir}\, G_{r-1}(\underline{v}_r + \underline{1}_i)$$

where $\underline{v}_r \in I_r$ and $1 \leq r \leq R$. Expanding the summation and using Definition 7.1, we obtain

$$G_r(\underline{v}_r) = \sum_{i=1}^{r-1} (1+v_{ir})\, w_1\, G_{r-1}(\underline{v}_r+\underline{1}_i)$$

$$+ (1+v_{rr})\, w_2\, G_{r-1}(\underline{v}_r + \underline{1}_r)$$

$$+ \sum_{i=r+1}^{P} (1+v_{ir})\, w_1\, G_{r-1}(\underline{v}_r + \underline{1}_i)$$

$$+ \sum_{i=P+1}^{N} (1 + v_{ir}\delta_i)\, s_{(i-P)r}\, G_{r-1}(\underline{v}_r + \underline{1}_i). \qquad (7.5)$$

Let $x = \sum_{i=r+1}^{P} v_{ir}$, $\underline{d} = (v_{(P+1)r}, \ldots, v_{Nr})$ and $\underline{\pi}_r = (v_{1r}, \ldots, v_{rr})^*$.

Now $\underline{v}_r \in I_r$, therefore $(\pi_1, \ldots, \pi_r, x, 0, \ldots, 0, d_1, \ldots, d_S) \in I_r$.

By Corollary 7.1 and Definition 7.2f,

$$H_r(x, \underline{\pi}_r, \underline{d}) = G_r(\underline{v}_r),$$

$$H_{r-1}(x+\pi_r, ((\underline{\pi}_r + \underline{1}_i)^-)^*, \underline{d}) = G_{r-1}(\underline{v}_r + \underline{1}_i) \text{ for } 1 \leq i \leq r-1,$$

$$H_{r-1}(x+\pi_r+1, \underline{\pi}_r^-, \underline{d}) = G_{r-1}(\underline{v}_r + \underline{1}_i) \text{ for } r \leq i \leq P,$$

and $$H_{r-1}(x+\pi_r, \underline{\pi}_r^-, \underline{d}+\underline{1}_{i-P}) = G_{r-1}(\underline{v}_r + \underline{1}_i) \text{ for } (P+1) \leq i \leq N. \quad (7.6)$$

Substituting eqs. 7.6 into eq. 7.1, we obtain

$$H_r(x, \underline{\pi}_r, \underline{d}) = \sum_{i=1}^{r-1} (1+\pi_i) w_1 \, H_{r-1}(x+\pi_r, ((\underline{\pi}_r + \underline{1}_i)^-)^*, \underline{d})$$

$$+ (1+\pi_r) w_2 \, H_{r-1}(x+\pi_r+1, \underline{\pi}_r^-, \underline{d})$$

$$+ (P-r+x) \, w_1 \, H_{r-1}(x+\pi_r+1, \underline{\pi}_r^-, \underline{d})$$

$$+ \sum_{i=P+1}^{N} (1+d_{i-P}\delta_i) \, s_{(i-P)r} \, H_{r-1}(x+\pi_r, \underline{\pi}_r^-, \underline{d} + \underline{1}_{i-P}). \quad (7.7)$$

By Corollary 7.1, $G_r(\underline{c}) = H_r(\alpha, \underline{\beta}, \underline{\gamma})$ for all

$$\underline{c} \in \{\underline{c} : c_i > 0 \text{ for } 1 \leq i \leq N;\ \sum_{i=r+1}^{P} c_i = \alpha;\ (c_1,\ldots,c_r)^* = (\underline{\beta})^*;$$

$$(c_{P+1},\ldots,c_N) = \underline{\gamma}\}$$

so that $H_r(x,\underline{\pi}_r,\underline{d})$ need only be computed for all $(x,\underline{\pi}_r,\underline{d}) \in A_r$ to obtain $H_R(0,\underline{0},\underline{0})$ using the recursion given by eq. 7.1. Finally, since in Corollary 6.1 $G_0(\underline{v}_0)=1$ for all $\underline{v}_0 \in I_0$, it follows that $H_0(x,\underline{\pi}_0,\underline{d}) = 1$ for all $(x,\underline{\pi}_0,\underline{d}) \in A_0$.

7.4 AN EFFICIENT ALGORITHM FOR SEMI-HOMOGENEOUS QUEUEING NETWORKS

The algorithm used to analyze semi-homogeneous queueing networks is essentially the same as RECAL except that we utilize eq. 7.4, rather than eq. 6.1. In RECAL, we make use of an artificial network, designated as B^o, in which the original chains in $B^{(R)}(N,\underline{K})$ are broken down into sub-chains that contain one customer each. In addition, there is an iterative procedure where we reenumerate the customers in B^o so that the assignment of the customers in B^o to chains in $B^{(R)}(N,\underline{K})$ is changed. This is done to obtain, in an efficient manner, the mean performance measures for all the individual routing chains in the original network $B^{(R)}(N,\underline{K})$. For the semi-homogeneous networks under consideration, however, this procedure is simplified considerably since we have, by assumption, already one customer in each chain.

In the following, we shall only specify how one may obtain the mean performance measures for chain R in $B^{(R)}(N,\underline{K})$. The measures for

other chains can be obtained by repeating the algorithm but with a different enumeration of the routing chains. As mentioned above, there is an efficient manner in which this repetition can be done. This has been described in detail in Chapter 6 and will not be repeated here.

An efficient algorithm for semi-homogeneous queueing networks

Step 1: Initialize $H_0(x,\underline{\pi}_0,\underline{d}) = 1$ for all $(x,\underline{\pi}_0,\underline{d}) \in A_0$.

Step 2: For $1 \leq r \leq R-1$:

Compute and store $H_r(x,\underline{\pi}_r,\underline{d})$ for all $(x,\underline{\pi}_r,\underline{d}) \in A_r$ using eq. 7.4 and the stored values of $H_{r-1}(x,\underline{\pi}_{r-1},\underline{d})$, where $(x,\underline{\pi}_{r-1},\underline{d}) \in A_{r-1}$.

Step 3: Compute and store $G_R(\underline{0}) = H_R(0,\underline{0},\underline{0})$ using eq. 7.4 and the stored values of $H_{R-1}(x,\underline{\pi}_{R-1},\underline{d})$ where $(x,\underline{\pi}_{R-1},\underline{d}) \in A_{R-1}$.

Step 4: Compute the mean performance measures for chain R in $B^{(R)}(N,K)$. (The following equations have been established in Corollary 6.2.)

$$T_{iR} = \begin{cases} e_{iR} \sum_{\ell=1}^{N} G_{R-1}(\underline{1}_\ell) \,/\, G_R(\underline{0})(N_i + R - 1), & \text{if there are no IS centers in the network,} \\ e_{iR}\, G_{R-1}(\underline{1}_x) \,/\, G_R(\underline{0}), & \text{if there are IS centers and x is any one of them,} \end{cases}$$

$U_{iR} = t_{iR}T_{iR}$, $Q_{iR} = G_R(\underline{0})^{-1}G_{R-1}(\underline{1}_i)\, w_{iR}$ and $W_{iR} = Q_{iR} / T_{iR}$ where, according to Corollary 7.1 and Definition 7.2f, $G_R(\underline{0}) = H_R(0,\underline{0},\underline{0})$ and

$$G_{R-1}(\underline{1}_i) = \begin{cases} H_{R-1}(0,(1,0,\ldots,0),\ \underline{0}) \text{ if } 1 \leq i \leq R-1, \\ H_{R-1}(1,\underline{0},\underline{0}) \text{ if } i = R, \\ H_{R-1}(0,\underline{0},\underline{1}_{i-R}) \text{ if } R+1 \leq i \leq N. \end{cases}$$

In an implementation of the above algorithm, an algorithm is needed to generate the unordered partitions of a positive integer y into n or fewer parts. Such an algorithm is well-known and can be found in [46] (algorithm 5.11).

7.5 COMPUTATIONAL COMPLEXITY

We first consider the space requirement (in number of array elements) of the algorithm. Let $[S]$ denote the cardinality of the set S. The required storage space, to obtain $H_R(0,\underline{0},\underline{0})$ and the performance measures for chain R, is

$$\underset{2 \leq r \leq R}{\text{Max}} \{[A_r] + [A_{r-1}]\}. \qquad (7.8)$$

In order to obtain the measures for all the other chains in $B^{(R)}(N,\underline{K})$, one may either simply repeat the algorithm of Section 7.4, in which case no additional storage is required over that given by eq. 7.8, or one may use the efficient method of repeating explained in Section 6.3, in which case additional storage space would be required equal to

$$\text{Max} \{[A_r]\}.$$
$$2 \leq r \leq R-1$$

For the sake of simplicity, in the following, we shall assume that we merely repeat the algorithm of Section 7.4.

Now

$$[A_r] = \sum_{x=0}^{P-r} \sum_{y=0}^{P-r-x} [B_r(y)][D(P-r-x-y,S)] \qquad (7.9)$$

where $$D(x,S) = \{\underline{d} : d_i \geq 0 \text{ for } 1 \leq i \leq S; \sum_{i=1}^{S} d_i = x\}$$

and $$[D(x,S)] = \binom{x + S - 1}{S - 1}.$$

In Appendix 7.1 we show that, with respect to P,

$$O(\text{Max}_{2 \leq r \leq R} \{[A_r] + [A_{r-1}]\}) \leq \frac{1}{2\sqrt{3}} \binom{P + S - 2}{S - 1} \frac{P^2}{(P-1)} 2^{3.71(P-1)^{\frac{1}{2}}} \qquad (7.10)$$

where $O(f(P)) = g(P)$ if $\lim_{P \to \infty} (f(P) / g(P)) = 1$. We note that the space requirement of the convolution algorithm, for the class of networks under consideration, is

$$2^{P+1}. \qquad (7.11)$$

Equation 7.10 therefore establishes the effectiveness of the new algorithm over a direct application of convolution. In Figure 7.1, the storage requirement (eq. 7.8) is compared with that of convolution,

(eq. 7.11), for various values of S (the number of shared resources). We see that the new algorithm offers a saving in the storage requirements. We note that in the computation of $[B_r(y)]$ in eq. 7.9 use is made of the recursive equation [21, eq. 4.1.5]

$$p_k(n) = \sum_{i=1}^{k} p_i(n-k)$$

with the initial condition $p_k(n) = 0$ for $n < k-1$ and $p_k(k) = 1$, where $p_k(n)$ is the number of unordered partitions of n into k parts.

We now consider the time requirement (in number of operations) of the algorithm. We assume that, to obtain the performance measures for the P chains, we merely repeat the algorithm P times. Assuming that $H_{r-1}(x, \underline{\pi}_{r-1}, \underline{d})$ has been stored for all $(x, \underline{\pi}_{r-1}, \underline{d}) \in A_{r-1}$, the computation of $H_r(x, \underline{\pi}_r, \underline{d})$ requires $(3r+4S+5)$ operations. The computation of $H_r(x, \underline{\pi}_r, \underline{d})$, for all $(x, \underline{\pi}_r, \underline{d}) \in A_r$, therefore requires $(3r+4S+5)[A_r]$ operations. Ignoring the operations involved in Step 4, the total number of operations required by the algorithm, to obtain the performance measures for all chains, is

$$P \sum_{r=1}^{P} (3r+4S+5)[A_r].$$

In Appendix 7.2 it is shown that, with respect to P,

$$O(P \sum_{r=1}^{P} (3r+4S+5)[A_r]) < \frac{1}{4\sqrt{3}} (3P+4S+5) \binom{P+S-2}{S-1} \frac{P^4}{(P-1)} 2^{3.71(P-1)^{\frac{1}{2}}}$$

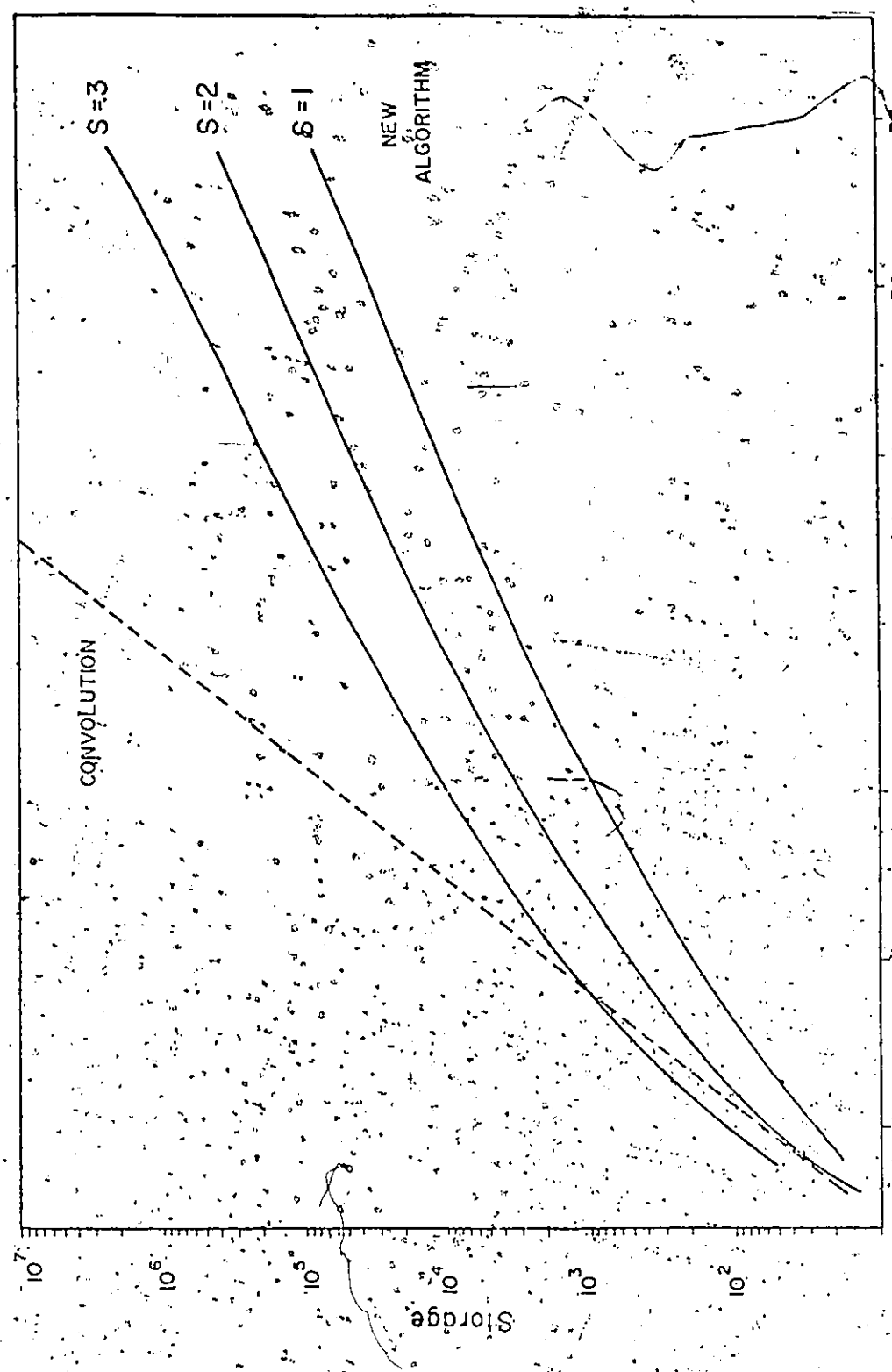

Figure 7.1 - Comparison of the storage requirements of the convolution algorithm with that of the new algorithm (eq. 7.8), for various numbers of shared resources (S).

The time requirement of the convolution algorithm would be

$$2R(N-1)\prod_{r=1}^{R}(K_r+1) = P(P+S-1)2^{P+1}$$

Since the time requirement of the new algorithm is of order less than exponential in $(P-1)$, and that of convolution is exponential in $(P+1)$, the new algorithm will be more efficient, at least when P is large relative to S. This, in general, will be the case since, in a local area network, the number of workstations is large relative to the number of shared resources.

APPENDIX 7.1

$$\max_{2\leq r\leq R}\{[A_r]+[A_{r-1}]\} \leq 2\max_{1\leq r\leq R}\{[A_r]\}$$

Now

$$[A_r] = \sum_{x=0}^{P-r}\sum_{y=0}^{P-r-x}[B_r(y)]\binom{P-r-x-y+S-1}{S-1}$$

$$\leq \binom{P-r+S-1}{S-1}\sum_{y=0}^{P-r}(P-r-y+1)[B_r(y)]$$

$$\leq \binom{P-r+S-1}{S-1}(P-r+1)^2[B_R(P-r)]$$

since $(P-r) \leq R-1$.

Therefore

$$2 \underset{1 \le r \le R}{\text{Max}} \{[A_r]\} \le 2 \underset{1 \le r \le R}{\text{Max}} \{\binom{P-r+S-1}{S-1} (P-r+1)^2 [B_R(P-r)]\}.$$

$$= 2P^2 \binom{P+S-2}{S-1} [B_R(P-1)].$$

Let $p(n)$ denote the number of unordered partitions of n. Then $[B_R(P-1)] = p(P-1)$ since $R = P$. From [21, eq. 4.2.8] we have that

$$O(p(n)) = \frac{1}{4n\sqrt{3}} e^{\pi(2n/3)^{\frac{1}{2}}}.$$

Therefore

$$O(\underset{2 \le r \le R}{\text{Max}} \{[A_r] + [A_{r-1}]\}) \le 2P^2 \binom{P+S-2}{S-1} \frac{1}{4(P-1)\sqrt{3}} e^{\pi(2(P-1)/3)^{\frac{1}{2}}}$$

$$= \frac{1}{2\sqrt{3}} \binom{P+S-2}{S-1} \frac{P^2}{(P-1)} 2^{3.71(P-1)^{\frac{1}{2}}}.$$

APPENDIX 7.2

(We use results contained in Appendix 7.1). We may write

$$P \sum_{r=1}^{P} (3r+4S+5)[A_r] \le P \sum_{r=1}^{P} (3r+4S+5) \binom{P-r+S-1}{S-1} (P-r+1)^2 [B_R(P-r)]$$

$$\le P^4 (3P+4S+5) \binom{P+S-2}{S-1} [B_R(P-1)].$$

Therefore

$$O(P \sum_{r=1}^{P} (3r+4S+5)[A_r]) \le \frac{1}{4\sqrt{3}} (3P+4S+5) \binom{P+S-2}{S-1} \frac{P^4}{(P-1)} 2^{3.71(P-1)^{\frac{1}{2}}}$$

CHAPTER 8 - CONCLUSION

8.1 SUMMARY

The new results, that form the basis of this thesis, can be summarized as follows:

(1) In Chapter 4, we have developed a new approach for decomposing multiple-chain closed queueing networks. Our decomposition by chain technique resolves the analysis of a multiple-chain network into a hierarchy of single-chain closed queueing network problems. It shows how a reduced system may be constructed around a particular routing chain of interest. The reduced system facilitates parametric studies and is a complementary result to 'Norton's Theorem' for queues.

(2) In Chapter 5, we have presented a new recursive expression for computing the normalization constant of multiple-chain closed queueing networks. The new recursion is much more efficient than the well-known convolution algorithm, when there are many routing chains in the network. It applies to both the constant-speed and state-dependent server situations.

(3) In Chapter 6, we have devised a means of obtaining the mean performance measures using the new recursion. Again, this algorithm is much more efficient than hitherto adopted algorithms, when there are

many chains in the network. It can be applied to networks containing constant-speed or state-dependent servers. Our algorithm extends the range of queueing networks whose mean performance measures can be obtained efficiently by exact means. We have also developed a dynamic scaling procedure, which can be incorporated easily within the framework of the algorithm to reduce the possibility of exceeding the floating-point range of a machine.

(4) Finally, in Chapter 7, we have introduced the class of semi-homogeneous queueing networks for the performance modeling of local area networks and have developed an efficient solution algorithm for their exact analysis.

All of the results presented in this dissertation are to do with the analysis of queueing networks in the steady-state. We have not been concerned with the transient analysis. For the purposes of system performance evaluation it is the equilibrium which is of most interest. For availability, reliability and 'performability' evaluation it is the transient analysis which is of more interest since, in general, it is not reasonable to assume that the measures of interest reach equilibrium values within the observation period (the lifetime of the system). The transient analysis of queueing networks is an open problem which has received little attention.

8.2 SUGGESTIONS FOR FURTHER RESEARCH

In Section 5.2, we have argued that there are many ways to

compute the normalization constant, our new method (Chapter 5) and the convolution algorithm being just two specific ways. We have also proposed a hybrid method (Section 5.6) that offers some improvement in efficiency. It appears then that it may be worthwhile searching for other network decompositions that lead to more efficient ways of computing the normalization constant. An interesting question is what the 'optimum' decomposition is. It appears that the optimum decomposition varies with the parameters (i.e. N, R, K_r) of the queueing network. An 'optimum' algorithm would, therefore, have to be an adaptive one.

There are as well certain other avenues that may be worth following. For networks with sparsity or locality in the routing, many of the visit ratios are zero so that the sum in eq. 5.1 need only range over those centers that chain r customers may actually visit. It seems then that the computational complexity of RECAL could, in this situation, be reduced, as in the tree convolution algorithm. Another question, is whether it may be possible to develop an MVA type algorithm whose complexity in R is polynomial. Such an algorithm would have the advantage of guaranteed numerical stability. Finally, we mention that there are many directions in which the algorithm for semi-homogeneous networks (Chapter 7) may be extended. For example, it may be useful if the chain populations are not restricted to unity.

REFERENCES

[1] A.O. Allen, "Queueing models of computer systems", IEEE Computer, April, pp. 13-24, 1980.

[2] G. Balbo, private communication, May 1985.

[3] G. Balbo, S.C. Bruell, and S. Ghanta, "The solution of homogeneous queueing networks with many job classes", in Proceedings of the International Workshop on Modeling and Performance Evaluation of Parallel Systems, Grenoble, France, pp. 385-417, Dec. 1984.

[4] S. Balsamo, and G. Iazeolla, "An extension of Norton's theorem for queueing networks", IEEE Trans. Soft. Eng., SE-8, 4, pp. 298-305, 1982.

[5] F. Baskett, K.M. Chandy, R.R. Muntz, and F. Palacios, "Open, closed, and mixed networks of queues with different classes of customers", J. Assoc. Comput. Mach., vol. 22, pp. 248-260, 1975.

[6] A. Brandwajn, "A model of a time-sharing system solved using equivalence and decomposition methods", Acta Informatica, 1, pp. 11-47, 1974.

[7] A. Brandwajn, "Fast approximate solution of multiprogramming models", in Proc. 1982 ACM SIGMETRICS Conf. Meas. Modeling Comput. Syst., pp. 141-149, 1982.

[8] S.C. Bruell, and G. Balbo, Computational Algorithms for Closed Queueing Networks, North Holland, Amsterdam, 1980.

[9] J.P. Buzen, "Queueing network models of multiprogramming", Ph.D. Dissertation, Div. Eng. Appl. Phys., Harvard Univ., Cambridge, MA, 1971.

[10] J.P. Buzen, "Computational algorithms for closed queueing networks with exponential servers", Commun. ACM, vol. 16, pp. 527-531, 1973.

[11] K.M. Chandy, U. Herzog, and L. Woo, "Parametric analysis of queueing networks", IBM J. Res. Develop., vol. 19, pp. 36-42, 1975.

[12] K.M. Chandy, and C.H. Sauer, "Computational algorithms for product-form queueing networks", Commun. ACM, vol. 23, pp. 573-583, 1980.

[13] P.J. Courtois, Decomposability: Queueing and Computer System Applications, Academic Press, New York, 1977.

[14] P.J. Courtois, "Exact aggregation in queueing networks", in Proc. First Meeting AFCET-SMF on Applied Mathematics, Ecole Polytech. Palaiseau (France), vol. 1, pp. 35-51, 1978.

[15] P.J. Courtois, "On time and space decomposition of complex structures", Commun. ACM, vol 28, 6, pp. 590-603, 1985.

[16] D.R. Cox, "A use of complex probabilities in the theory of stochastic processes", Proceedings Cambridge Philosophical Society, 51, pp. 313-319, 1955.

[17] E. de Souza e Silva, S.S. Lavenberg, and R.R. Muntz, "A perspective on iterative methods for the approximate analysis of closed queueing network", in Mathematical Computer Performance and Reliability, G. Iazeolla, P.J. Courtois, and A. Hordijk, Eds., North-Holland, Amsterdam, pp. 225-244, 1984.

[18] A. Goldberg, G. Popek, and S.S. Lavenberg, "A validated distributed system performance model", in PERFORMANCE '83, A.K. Agrawala and S.K. Tripathi, Eds., North-Holland, Amsterdam, pp. 251-268, 1983.

[19] W.J. Gordon, and G.F. Newell, "Closed queueing systems with exponential servers", Operations Research, p. 254-265, 1967.

[20] W.J. Gordon, and G.J. Newell, "Acknowledgement", Operations Research, page 1182, 1967.

[21] M. Hall, Combinatorial Theory, Blaisdell Publishing Company, Waltham, Massachusetts, 1967.

[22] P. Heidelberger, and S.S. Lavenberg, "Computer performance evaluation methodology", IEEE Trans. on Computers, vol. C-33, no. 12, pp. 1195-1220, Dec. 1984.

[23] J.R. Jackson, "Networks of waiting lines", Operations Research, 5, pp. 518-521, 1957.

[24] J.R. Jackson, "Jobshop-like queueing systems", Management Science, vol. 10, pp. 131-142, 1963.

[25] F.P. Kelly, "Networks of queues with customers of different types", J. of Applied Probability, 12, pp. 542-554, 1975.

[26] F.P. Kelly, "Networks of queues", Advances in Applied Probability, 8, pp. 416-432, 1976.

[27] F.P. Kelly, Reversibility and Stochastic Networks, Wiley, New York, 1980.

[28] P. Kritzinger, S. van Wyk, and A. Krzesinski, "A generalization of Norton's theorem for multiclass queueing networks", Perform. Eval., vol. 2, pp. 98-107, 1982.

[29] A. Kumar, "Equivalent queueing networks and their use in approximate equilibrium analysis", BSTJ, 62, 10, pp. 2893-2910, 1983.

[30] S.S. Lam, "Queueing networks with population size constraints", IBM J. Res. Develop., 21, pp. 370-378, 1977.

[31] S.S. Lam, "Dynamic scaling and growth behaviour of queueing network normalization constants", J. Assoc. Comput. Mach., 29, pp. 492-513, 1982.

[32] S.S. Lam, and Y.L. Lien, "A tree convolution algorithm for the solution of queueing networks", Commun. ACM, vol. 26, pp. 203-215, 1983.

[33] S.S. Lam, and J.W. Wong, "Queueing network models of packet switching networks, Part 2: Networks with population size constraints", Perform. Eval., 2, 3, pp. 161-180, 1982.

[34] S.S. Lavenberg, (Ed.), Computer Performance Modeling Handbook, Academic Press, New York, 1983.

[35] S.S. Lavenberg, and M.Reiser, "Stationary state probabilities at arrival instants for closed queueing networks with multiple types of customers", J. Appl. Prob., vol. 17, pp. 1048-1061, 1980.

[36] E.D. Lazowska, and J. Zahorjan, "Multiple class memory constrained queueing networks", in Proc. 1982 ACM SIGMETRICS Conf. Meas. Modeling Comput. Syst., pp. 130-140, 1982.

[37] A.J. Lemoine, "Networks of queues - a survey of equilibrium analysis", Management Science, 24, pp. 464-481, 1977.

[38] J.D.C. Little, "A proof of the queueing formula $L=\lambda W$", Operations Research, vol. 9, pp. 383-387, 1961.

[39] R.A. Marie, "An approximate analytical method for general queueing networks", IEEE Trans. Soft. Eng., SE-5, pp. 530-538, 1979.

[40] D. Neuse, and K.M. Chandy, "HAM: The heuristic aggregation method for solving general closed queueing network models of computer systems", Perform. Eval., 11, pp. 195-212, 1982.

[41] M.C. Pennotti, and M. Schwartz, "Congestion control in store-and-forward tandem links", IEEE Trans. Comm., COM-23, 12, pp. 1434-1443, 1975.

[42] B. Pittel, "Closed exponential networks of queues with saturation: The Jackson-type stationary distribution and its asymptotic analysis", Mathematics of Operations Research, vol 4, 4, pp. 357-378, 1979.

[43] M. Posner, and B. Bernholtz, "Closed finite queueing networks with time lags", Operations Research, 16, pp. 962-976, 1968.

[44] M. Posner, and B. Bernholtz, "Closed finite queueing networks with time lags and with several classes of units", Operations Research, 16, pp. 977-985, 1968.

[45] D. Potier, "New user's introduction to GNAP2", Rapports Techniques, No. 40, INRIA, Rocquencourt, France, 1984.

[46] E.M. Reingold, J. Nievergelt, and N. Deo, Combinatorial Algorithms: Theory and Practice, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

[47] M. Reiser, and H. Kobayashi, "Queueing networks with multiple closed chains: Theory and computational algorithms, "IBM J. Res. Develop., vol. 19, pp. 283-294, 1975.

[48] M. Reiser, "A queueing network analysis of computer communication networks with window flow control", IEEE Trans. Commun., COM-27, pp. 1199-1209, 1979.

[49] M. Reiser, and S.S. Lavenberg, "Mean-value analysis of closed multi-chain queueing networks", J. Assoc. Comput. Mach., vol. 27, pp. 313-322, 1980.

[50] M. Reiser, "Mean value analysis and convolutional method for queue-dependent servers in closed queueing networks", Perform. Eval., vol. 1, pp. 7-18, 1981.

[51] J. Riordan, Combinatorial Identities, R.E. Krieger Publishing Company, Huntingdon, New York, 1979.

[52] C.H. Sauer, "Computational algorithms for state-dependent queueing networks", ACM Trans. Comput. Syst., vol. 1, pp. 67-92, 1983.

[53] C.H. Sauer, "Corrigendum: Computational algorithms for state-dependent queueing networks", ACM Trans. Comput. Syst., 1, 4, page 369, 1983.

[54] C.H. Sauer, and K.M. Chandy, "Approximate analysis of central server models", IBM J. Res. and Dev., 19, 3, pp. 301-313, 1975.

[55] C.H. Sauer, and K.M. Chandy, Computer Systems Performance Modeling, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[56] P. Schweitzer, "Approximate analysis of multiclass closed networks of queues", in Proc. Int. Conf. Stochastic Control and Optimization, Amsterdam, 1979.

[57] K.C. Sevcik, and I. Mitrani, "The distribution of queueing network states at input and output instants", J. Assoc. Comput. Mach., vol. 28, pp. 358-371, 1981.

[58] H.A. Simon, and A. Ando, "Aggregation of variables in dynamic systems", Ecohometrica, 29, pp. 111-138; 1961.

[59] W.J. Stewart, "A comparison of numerical techniques in Markov modeling", Commun. ACM, 21, pp. 144-151, 1978.

[60] D.F. Towsley, "Queueing network models with state-dependent routing", J. Assoc. Comput. Mach., vol. 27, pp. 323-337, 1980.

[61] S. Tucci, and C.H. Sauer, "The tree MVA algorithm", Res. Rep. RC 9338, IBM Corp., Yorktown Heights, NY, 1982.

[62] H. Vantilborgh, "Exact aggregation in exponential queueing networks", J. Assoc. Comput. Mach., 25, 4, pp. 620-629, 1978.

[63] H. Vantilborgh, R.L. Garner, and E.D. Lazowska, "Near-complete decomposability of queueing networks with clusters of strongly interacting servers", in Proc. 7th IFIP W.G.7.3 Inter. Symp. on Computer Performance Modeling, Measurement and Evaluation, 1980.

[64] J.W. Wong, "Queueing network modeling of communication networks", Computing Surveys, vol. 10, 3, pp. 343-351, 1978.

[65] J.W. Wong, and S.S. Lam, "Queueing network models of packet switching networks, part 1: Open networks", Perform. Eval., 2, 1, p. 9-21, 1982.

[66] K.J. Zahorjan, E.D. Lazowska, and R.L. Garner, "A decomposition approach to modeling high service time variability", Perform. Eval., 3, pp. 35-54, 1983.