Performance of Computer Systems
M. Arato, A. Butrimenko, E. Gelenbe (eds.)

A COMPUTATIONAL ALGORITHM FOR QUEUE DISTRIBUTIONS VIA THE PÓLYA THEORY OF ENUMERATION

Hisashi Kobayashi
Computer Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

We present a new computational algorithm for evaluating the queue distribution in a general Markovian queuing network, based on the Pólya theory of counting. We formulate queue size vectors as equivalence classes relative to a symmetric group. The normalization constant of the queue-distribution then corresponds to the pattern inventory in the Pólya theory. A central server model is discussed as an application example of this new algorithm.

I. INTRODUCTION

A "network of queues" representation provides a basic framework in dealing with the performance analysis of multiple resource systems, in which different resources process jobs asynchronously to each other. The class of models for which we find a simple closed solution of the equilibrium queue distribution is the so-called "Markovian queuing network" [1-4]. For this class the equilibrium distribution is given in "product" form. This expression, however, includes a normalization constant, and determination of the normalization constant presents a computationally nontrivial task.

A number of authors have proposed various algorithms designed to evaluate efficiently the normalization constant, and related performance measures - utilization, throughput, moments of queue size, average response time, etc. In the present paper we propose a new algorithm that is derived based on the Pólya theory of enumeration - a well-discussed subject in books on combinatorial mathematics [5-8]. The Pólya theory of enumeration influenced the research in finding minimal cost networks for the realization of switching functions, as treated by Slepian [9] and Harrison [10]. The problem of evaluating the normalization factor of queue distribution is a bona fide combinatorial problem, thus it is quite natural to investigate possible applications of the Pólya theory to queuing theory.

II. STATEMENTS OF THE PROBLEM

Consider a closed* queuing network which consists of M service stations arbitrarily connected to each other. Let us define the following set of nomenclature concerning the analysis of such network:

$$M = \{1,2,3,\ldots,M\}: \quad \text{the set of service stations} \qquad (2.1)$$

$$N = \sum_{i \in M} n_i: \quad \text{the network population} \qquad (2.2)$$

*In the original paper [14] a more general class of queuing networks is discussed.

$$F(N) = \{\underline{n} \mid n_i > 0 \text{ for all } i \in M \text{ and } \sum_{i \in M} n_i = N\}: \quad \text{the set of feasible queue vectors} \tag{2.3}$$

$C_i(n)$ = the processing rate of server i, when its local queue size is n, $i \in M$ (2.4)

$$\beta_i(n) = \prod_{k=1}^{n} \frac{1}{C_i(k)}, \quad i \in M \tag{2.5}$$

W_i = the expected total work (or service) a job demands from server i during this job's entire life in the network. (2.6)

In order to obtain the equilibrium state distribution of the queue-size vector $p(\underline{n})$, we make a set of fairly general assumptions (see [2,3,4] for details) concerning (i) the routing behavior, (ii) service (or work) distribution, (iii) service (or processing) rates, and (iv) queue disciplines. We can then obtain the following product form solution:

$$p(\underline{n}) = \begin{cases} c \prod_{i \in M} f_i(n_i), & \text{if } \underline{n} \in F(N) \\ 0, & \text{if } \underline{n} \notin F(N) \end{cases} \tag{2.7}$$

where the functions $f_i(n_i)$ are themselves given in the following product form:

$$f_i(n_i) = \beta_i(n_i) W_i^{n_i}, \quad i \in M \tag{2.8}$$

the scalar constant c of Equation (2.7) is the normalization factor referred to in Section I and is given by

$$c = 1/g(M,N) \tag{2.9}$$

where

$$g(M,N) = \sum_{\underline{n} \in F(N)} \prod_{i \in M} \beta_i(n_i) W_i^{N_i} \tag{2.10}$$

thus the problem is reduced to that of evaluating $g(M,N)$ for a given pair (M,N).

The convolutional algorithm of Buzen [11] and Reiser and Kobayashi [12,13] is essentially the following recursive formula:

$$g(M,N) = \sum_{k=0}^{N} g(M-1,N-k) \beta_M(k) W_M^k, \quad M \geq 1, \ N \geq 1 \tag{2.11}$$

with the boundary conditions

$$g(M,0) = 1, \quad \text{for } M \geq 0, \tag{2.12a}$$

and

$$g(0,N) = \begin{cases} 1, & \text{for } N=0, \\ 0, & \text{for } N\geq 1. \end{cases} \tag{2.12b}$$

for a fixed value of M, the sequence $\{g(M,i);\ 0\leq i\leq N\}$ is the convolutional sum of the sequence $\{\beta_M(i)W_M^i;\ 0\leq i\leq N\}$ and $\{g(M-1,i);\ 0\leq i\leq N\}$. The computation of $\{g(M,i);\ 0\leq i\leq N\}$ given the value of $\{g(M-1,i);\ 0\leq i\leq N\}$ requires $\frac{N(N+1)}{2}$ multiplications and additions. Thus, for a given value of the pair (M,N) the evaluation of $g(M,N)$ requires, in total, $\frac{(M-1)}{2}\sum_{n'=1}^{N} n'(n'+1) = \frac{(M-1)N(N+1)(N+2)}{6}$ multiplications and additions.

Under the special condition of constant service rates of the form for all $i\in M$:

$$C_i(n) = \begin{cases} C_i & \text{for } n\geq 1 \\ 0 & \text{for } n=0 \end{cases} \tag{2.13}$$

we find the following simple recurrence algorithms for the two-dimensional array $\{g(M,N)\}$:

$$g(M,N) = g(M-1,N) + \tau_M g(M,N-1), \quad M\geq 1,\ N\geq 1 \tag{2.14}$$

with the boundary conditions (2.12). The parameter τ_i is the expected <u>total</u> service time given to a job by server i during the job's lifetime within the network, and is given by

$$\tau_i = \frac{W_i}{C_i}, \quad i\in M \tag{2.15}$$

The evaluation of $g(M,N)$ requires, for this special case, $(M-1)N$ multiplications and additions.

III. <u>A NEW COMPUTATIONAL ALGORITHM</u>

We now introduce a new algorithm for evaluating the normalization constant $g(M,N)$. This algorithm is restricted to a network with exponential servers all of which have fixed service rates, i.e., the case where Equation (2.13) is true for all $i\in M$. Then certainly we could use the recursive formula (2.14) throughout the entire steps, starting with the boundary condition (2.12). The evaluation of $\{g(m,n);\ 1\leq n\leq N,\ 1\leq m\leq M\}$ would require only $(M-1)N$ multiplications and additions. However, the computational formula to be discussed below is sometimes more convenient, especially when N is small.

The assumption of the constant service rates of (2,13) allows us to write $g(M,N)$ of (2.10) as

$$g(M,N) = \sum_{\underline{n}\in\underline{F}(N)} \prod_{i\in M} \tau_i^{n_i} \tag{3.1}$$

where τ_i was defined by (2.15). Let us define the set of stations

$$M = \{1,2,\ldots,M\} \tag{3.2}$$

and the set of N jobs

$$N = \{1,2,\ldots,N\} \tag{3.3}$$

Consider then a set of functions that have N and M as their domain and range, respectively:

$$F = \{f \mid f:N \to M\} \tag{3.4}$$

A function f in the set F represents a way of placing N jobs into M service stations. We write, for example,

$$f(j) = i, \quad j \in N, \quad i \in M \tag{3.5}$$

which implies that job j is placed in station i.

Consider a permutation π defined over N, and let S_N be the set of all permutations defined over N:

$$S_N = \{\pi \mid \pi:N \to N\} \tag{3.6}$$

The elements of S_N form a symmetric group of degree N. For a given function $f_1 \in F$ and permutation $\pi \in S_N$, we can define another function f_2 by

$$f_2(j) = f_1(\pi(j)), \quad j \in N \tag{3.7}$$

Clearly the function f_2 is also a member of F. However, such functions f_1 and f_2 correspond to the same queue size vector $\underline{n}$

$$\underline{n} = [n_1,n_2,\ldots,n_M] \tag{3.8}$$

since we do not distinguish the individual jobs. Therefore, we say that the functions f_1 and f_2 are equivalent relative to the permutation group S_N. Distinct values of $\underline{n} \in F(N)$ correspond to distinct equivalence classes.

We interpret the parameter τ_i of (2.15) as the weight of element i in the set M, and thus

$$\sum_{i \in M} \tau_i \tag{3.9}$$

represents the inventory of the set M. If a function f belongs to the equivalence class $\underline{n}$ (3.8), then the weight $W(f)$ of the function f is

$$W(f) = \prod_{i \in M} \tau_i^{n_i}, \quad \text{for all} \quad f \in \underline{n} \tag{3.10}$$

which is called the weight of the equivalence class $\underline{n}$. Then the pattern inventory of F - the sum of weights of distinct equivalence classes relative to the permutation group S_N - is

$$\sum_{\underline{n} \in F(N)} W(f) \tag{3.11}$$

which is nothing but $g(M,N)$ of (3.1)! This observation immediately calls our attention to the celebrated Pólya theorem:

Theorem (Pólya): The pattern inventory $g(M,N)$ of the set of the equivalence classes of functions from the domain N to the range M is

$$g(M,N) = Z_{S_N}\left(\sum_{i\in M} \tau_i,\ \sum_{i\in M} \tau_i^2,\ \ldots,\ \sum_{i\in M} \tau_i^N\right) \tag{3.12}$$

where $Z_{S_N}(x_1,x_2,\ldots,x_N)$ is the cyclic index polynomial of the permutation group S_N.

The cycle index polynomial of S_N is given from Cauchy's formula

$$Z_{S_N}(x_1,x_2,\ldots,x_N) = \sum \frac{x_1^{\mu_1} x_2^{\mu_2} \cdots x_N^{\mu_N}}{\mu_1!\, 2^{\mu_2} \mu_2! \cdots N^{\mu_N} \mu_N!} \tag{3.13}$$

where the sum is taken over the set of distinct M tuples, $\langle \mu_i;\ i = 1,2,\ldots,M\rangle$ such that

$$\sum_{i\in M} i\mu_i = N \tag{3.14}$$

Table 1 tabulates (3.13) for $N = 1,2,\ldots,7$.

Table 1

Cycle Index Polynomials of Symmetric Groups

N	Z_{S_N}
1	x_1
2	$1/2(x_1^2 + x_2)$
3	$1/6(x_1^3 + 3x_1x_2 + 2x_3)$
4	$1/24(x_1^4 + 6x_1^2x_2 + 3x_2^2 + 8x_1x_3 + 6x_4)$
5	$1/120(x_1^5 + 10x_1^3x_2 + 15x_1x_2^2 + 20x_1^2x_3 + 20x_2x_3 + 30x_1x_4 + 24x_5)$
6	$1/720(x_1^6 + 15x_1^4x_2 + 45x_1^2x_2^2 + 15x_2^3 + 40x_1^3x_3 + 120x_1x_2x_3 + 40x_3^2 + 90x_1^2x_4 + 90x_2x_4 + 144x_1x_5 + 120x_6)$
7	$1/5040(x_1^7 + 21x_1^5x_2 + 70x_1^4x_3 + 105x_1^3x_2^2 + 210x_1^3x_4 + 420x_1^2x_2x_3 + 105x_1x_2^3 + 280x_1x_3^2 + 630x_1x_2x_3 + 504x_1^2x_5 + 840x_1x_6 + 210x_2^2x_3 + 504x_2x_5 + 420x_3x_4 + 720x_7)$

Thus all that is required is to compute the set of values

$$x_k = \sum_{i \in M} \tau_i^k, \quad k = 1,2,\ldots \tag{3.15}$$

and substitute them into the polynomial Z_{S_N}.

Alternatively, we recursively compute $g(m,n)$, $1 \le n \le N$, $1 \le m \le N$. We can derive the following expression for the cycle index polynomials:

$$Z_{S_n}(x_1,x_2,\ldots,x_n) = \begin{cases} \frac{1}{n} \sum_{k=0}^{n-1} x_{n-k} Z_{S_k}(x_1,x_2,\ldots,x_k), & \text{for } n \ge 1 \\ 1, & \text{for } n=1 \end{cases} \tag{3.16}$$

which leads to the recurrence relation of the sequence $g(M,n)$, $n = 1,2,3,\ldots$

$$g(M,n) = \frac{1}{n} \sum_{k=0}^{n-1} x_{n-k} g(M,k)$$

$$= \frac{1}{n} \sum_{k=1}^{n} x_k g(M,n-k) \tag{3.17}$$

with the initial condition

$$g(M,0) = 1, \quad M \ge 1 \tag{3.18}$$

Note that Equation (3.17) is also of a convolutional form: we can view the sequence $\{g(M,n): n = 1,2,\ldots\}$ as an autoregressive sequence with varying regressive coefficients $\{\frac{1}{n} x_{n-k}: k = 0, 1, \ldots, n-1\}$.

IV. AN APPLICATION EXAMPLE

The computational formulas presented above will be of practical interest when there are many servers in the network. The cost of computing the parameters $\{x_k, k = 1,2,\ldots\}$ of (3.15) is insignificant in many cases of practical interest. Consider, for example, a central server model in which the CPU station is followed by a number of I/O devices (disks and drums) with a number of independent access paths in parallel: if the traffic distribution to different paths is uniform (which is often assumed in the absence of detailed measurement data), then the model becomes a closed network with many independent servers, but with the same parameter value of $\{\tau_i\}$.

For example, a model of an interactive system with multiprogramming in virtual storage can be decomposed into the outer model - a time-shared system model - and the inner model - a central server model [2,3]. Figure 1 shows a typical structure of the inner model with M=16: servers 1 through 10 represent magnetic drum sectors with independent access paths; servers 11 through 15 are magnetic disks with independent channels; and server 16 represents CPU. The multiprogramming level, N, varies as time changes. Usually the value N is controlled through the job scheduler. We assume the following workload parameters per interaction, where an interaction starts when an interactive user creates a

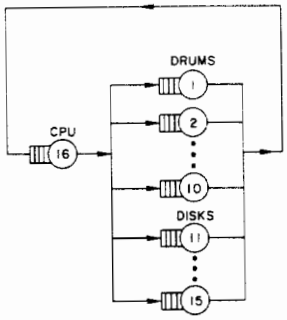

Figure 1. A Closed Queuing Network
Model With M=16 Stations

request (or job) and it ends when the job is processed by the system and its response is received by the user.

Average CPU work per interaction:	$W_{16} = 2.0$ sec.
Average number of drum accesses (reads) per interaction:	$R_{drm} = 80$
Average number of disk accesses (reads) per interaction:	$R_{dsk} = 20$
Average latency and transfer time per drum access:	20 msec.
Average seek, latency and transfer time per disk access:	100 msec.

In the absence of measurement data concerning how these drum reads and disk reads are distributed among the separate access paths, we assume the uniform distributions:

$$W_1 = \ldots = W_{10} = 20 \text{ msec} \times \frac{R_{drm}}{10} = 0.16 \text{ sec};$$

$$W_{11} = \ldots = W_{15} = 100 \text{ msec} \times \frac{R_{dsk}}{5} = 0.40 \text{ sec}.$$

Since the service (or work) is represented in time, the processing rate $\{C_i\}$ should be set to unity. Hence the parameters $\{\tau_i\}$ of (2.15) are the same as $\{W_i\}$:

$$\tau_1 = \ldots = \tau_{10} = 0.16 \text{ sec};$$

$$\tau_{11} = \ldots = \tau_{15} = 0.40 \text{ sec};$$

$$\tau_{16} = 2.0 \text{ sec}.$$

We first compute the parameters x_i's of (3.15)

$$x_1 = 10 \times 0.16 + 5 \times 0.4 + 2.0 = 5.6 \text{ sec};$$

$$x_2 = 10 \times 0.16^2 + 5 \times 0.4^2 + 2.0^2 = 5.056 \text{ sec}^2;$$

$$x_3 = 10 \times 0.16^3 + 5 \times 0.4^3 + 2.0^3 = 8.361 \text{ sec}^3;$$

$$x_4 = 10 \times 0.16^4 + 5 \times 0.4^4 + 2.0^4 = 16.135 \text{ sec}^4,$$

etc. Then from Formula (3.12) and the polynomials of Table 1 (alternatively from the recurrence formula (3.17)), we obtain

$$g(16,0) = 1$$

$$g(16,1) = x_1 = 5.6 \text{ sec};$$

$$g(16,2) = \frac{1}{2}(x_1^2 + x_2) = 18.2 \text{ sec}^2;$$

$$g(16,3) = \frac{1}{6}(x_1^3 + 3x_1x_2 + 2x_3) = 46.2 \text{ sec}^3;$$

$$g(16,4) = \frac{1}{24}(x_1^4 + 6x_1^2x_2 + 3x_2^2 + 8x_1x_3 + 6x_4) = 103.4 \text{ sec}^4;$$

etc. Utilization $\rho_i(N)$ of server i for the degree of multiprogramming N is given (see e.g., [2]) by

$$\rho_i(N) = W_i \frac{g(M,N-1)}{g(M,N)} \tag{4.1}$$

We can predict, for example, CPU utilization under different values of multiprogramming level, N, as follows:

$$\rho_{16}(1) = \frac{2.0}{5.6} = 0.36;$$

$$\rho_{16}(2) = 2.0 \times \frac{5.6}{18.2} = 0.62;$$

$$\rho_{16}(3) = 2.0 \times \frac{18.2}{46.2} = 0.79;$$

$$\rho_{16}(4) = 2.0 \times \frac{46.2}{103.4} = 0.89;$$

An alternative formula for utilization $\rho_M(N)$ for the M^{th} resource is given from Equations (2.14) and (4.1) as

$$\rho_M(N) = 1 - \frac{g(M-1,N)}{g(M,N)} \tag{4.2}$$

For the degree of multiprogramming $N=4$, for example, we need to calculate $g(15,4)$. For this purpose we compute the following parameters:

$$y_1 = 10 \times 0.16 + 5 \times 0.4 = 3.6 \text{ sec}$$

$$y_2 = 10 \times 0.16^2 + 5 \times 0.4^2 = 1.056 \text{ sec}^2$$

$$y_3 = 10 \times 0.16^3 + 5 \times 0.4^3 = 0.361 \text{ sec}^3$$

$$y_4 = 10 \times 0.16^4 + 5 \times 0.4^4 = 0.135 \text{ sec}^4$$

Then

$$g(15,4) = \frac{1}{24}(y_1^4 + 6y_1^2y_2 + 3y_2^2 + 8y_1y_3 + 6y_4) = 11.0.$$

Hence

$$\rho_{16}(4) = 1 - \frac{11.0}{103.4} = 0.89$$

which is, not surprisingly, the same as the value obtained earlier.

The k^{th} moment of the number of customers, n_i, is given [2] by

$$E[n_i^k] = \frac{1}{g(M,N)} \sum_{n=1}^{N} g(M,N-n)[n^k-(n-1)^k]\tau_i^n \qquad (4.3)$$

For instance, the average of CPU queue for the degree of multiprogramming N=4 is

$$E[n_{16}] = \frac{1}{g(16,4)} \sum_{n=1}^{4} g(16,4-n)2.0^n$$

$$= \frac{1}{103.4}(46.2 \times 2.0 + 18.2 \times 2.0^2 + 5.6 \times 2.0^3 + 1 \times 2.0^4$$

$$= 2.19$$

Similarly, we obtain the average queue sizes the the drums and disks:

$$E[n_1] = \ldots = E[n_{10}] = 0.076$$

$$E[n_{11}] = \ldots = E[n_{15}] = 0.210$$

We check that these values add up to N=4:

$$2.19 + 0.76 \times 10 + 0.210 \times 5 = 4.0$$

References

|1| L. Kleinrock (1975). Queueing Systems, Vol. I: Theory, John Wiley & Sons, New York.

|2| H. Kobayashi (1978). Modeling and Analysis: An Introduction to System Performance Evaluation Methodology, Addison-Wesley, Reading, Mass.

|3| H. Kobayashi (1978). "System Design and Performance Analysis Using Analytic Models" in Current Trends in Programming Methodology Vol. III: Software Modelling, (Ed. M. Chandy and R.T. Yeh), pp. 72-114, Prentice-Hall Inc., Englewood Cliffs, N.Y.

|4| H. Kobayashi and A.G. Konheim (1977). "Queuing Models for Computer Communications System Analysis" (Invited paper). IEEE Trans. on Communications , COM-25, No. 1, pp. 2-29.

|5| C.L. Liu (1968). Introduction to Combinatorial Mathematics, McGraw-Hill Book Co., New York.

|6| C. Berge (1971). Principle of Combinatorial Mathematics. Academic Press, New York.
|7| H.S. Stone (1973). Discrete Mathematical Structures and Their Applications, Science Research Associate, Inc., Chicago.
|8| F.P. Preparata and R.T. Yeh (1973). Introduction to Discrete Structures for Computer Science and Engineering, Addison-Wesley, Reading, Mass.
|9| D. Slepian (1953). "On the Number of Symmetry Types of Boolean Functions of n Variables", Can. J. Math., Vol. 5, No. 2, pp. 185-193.
|10| M.H. Harrison (1965). Introduction to Switching and Automata Theory, McGraw-Hill Book Co., New York.
|11| J.P. Buzen (1973). "Computational Algorithms for Closed Queuing Networks with Exponential Servers", Comm. of ACM, Vol. 16, No. 9, pp. 527-531.
|12| M. Reiser and H. Kobayashi (1973). "Recursive Algorithms for General Queuing Networks with Exponential Servers", IBM Research Report, RC-4254, IBM Research Center, Yorktown Heights, N.Y.
|13| M. Reiser and H. Kobayashi (1975). "Queuing Networks with Multiple Closed Chains: Theory of Computational Algorithms", IBM J. of Res. and Develop., Vol. 19, No. 3, pp. 283-294.
|14| H. Kobayashi (1976). "A Computational Algorithm for Queue Distribution via Pólya Theory of Enumeration", IBM Research Report, RC-6154, IBM Research Center, Yorktown Heights, N.Y.