A note on RECAL: the recursion by chain algorithm

van Doremalen, J.B.M.

Published: 01/01/1986

Citation for published version (APA):
Memorandum COSOR 86-02

A note on RECAL:
the recursion by chain algorithm

by

J.B.M. van Doremalen

Eindhoven, Netherlands
April 1986
A note on RECAL: the recursion by chain algorithm.

Jan B.M. van Doremalen
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands

ABSTRACT

This note considers an alternative derivation of the basic recursive relation in the recently developed RECAL or recursion by chain algorithm for the evaluation of the normalization constant of a product-form closed multichain queueing network model. The technique is based on a $z$-transform representation of the set of normalization constants.

1. Introduction

In Conway and Georganas [1] a new and promising recursive algorithm has been presented to compute the normalization constant of a closed multichain product-form queueing network model with queuelength dependent service rates: a recursion by chain algorithm named RECAL. This paper considers an alternative derivation of the main recursive relation of the algorithm by applying a $z$-transform approach.

One of the major problems in analyzing closed product-form queueing networks forms the evaluation of a normalization constant for the equilibrium distribution. In the literature two main lines can be discerned to solve this problem: the convolution algorithm and the mean value analysis. The convolution algorithm, cf. Buzen [2] and Reiser and Kobayashi [3], is based on a set of recursive relations for normalization constants at varying population sizes. The mean value analysis, cf. Reiser and Lavenberg [4], Zahorjan and Wong [5] and Bruell, Balbo and Afshari [6], uses direct recursive relations between system characteristics of interest.

The problem in evaluating these algorithms arises when many closed customer chains are involved: the computational complexity and storage requirements grow exponentially in the number of chains. A new and convolution-like algorithm has been proposed in [1]. It is based on a set of recursive relations. The recursion is in the number of customers rather than in the number of workstations. Especially for smaller systems with many closed customer chains the algorithm forms an interesting alternative for the classical convolution algorithm.

We show that the basic recursive relation of this algorithm can be derived using a $z$-transform approach. In Section 2 the problem of evaluating the normalization constant is
formulated. In Section 3 the z-transform approach is applied to derive the main recursive relation. And in Section 4 some concluding remarks are made.

2. Normalization constants and z-transforms

Consider a queueing network model with \( N \) workstations, numbered \( n = 1, \ldots, N \) and \( R \) closed customer chains, numbered \( r = 1, \ldots, R \). The service rate of the single service unit at workstation \( n \) is \( \mu_n(k) \) if \( k \) customers are present. The \( K_r \) customers of chain \( r \) proceed through the network in accordance with a Markov routing. The so-called visit ratios \( f_{n,r} \) indicate the relative number of visits a customer of chain \( r \) brings to workstation \( n \). The expected service demand of a customer of chain \( r \) at workstation \( n \) equals \( w_{n,r} \). It is assumed that the network characteristics are such that a product-form solution exists. cf. Basket, Chandy, Muntz and Palacios [7] for more details on the precise conditions.

We introduce an aggregated state description by an \( N \)-dimensional vector \( k = (k_1, \ldots, k_N) \). Here \( k_n \) is an \( R \)-dimensional vector \( k_n = (k_{n,1}, \ldots, k_{n,R}) \) with \( k_{n,r} \) denoting the number of customers of chain \( r \) present at workstation \( n \). The set of states is denoted by \( S(K) \), where \( K \) emphasizes the dependence on the populations of the closed customer chains, i.e. \( K = (K_1, \ldots, K_R) \). As state space we use

\[
S(K) = \{ k \in \mathbb{N}_0^{NR} \mid \sum_{n=1}^N k_{n,r} = K_r, \; n = 1, \ldots, N \}.
\]  

Note that this set in general covers non-feasible states. If customers of a certain chain \( r \) do not visit workstation \( n \), then \( k_{n,r} \) will be 0 by definition. Fortunately, this causes no problems as the corresponding equilibrium probabilities will appear to be zero whatsoever.

The equilibrium probability \( p(k,K) \) that the system is in state \( k \) with the population vector being \( K \), attains the following product-form:

\[
p(k,K) = \frac{1}{G(K)} \prod_{n=1}^N \frac{k_n!}{b_n(k_n)} \prod_{r=1}^R x_{n,r}^{k_{n,r}}.
\]  

where

\[
x_{n,r} = f_{n,r} w_{n,r},
\]

\[
k_n = \sum_{r=1}^R k_{n,r},
\]

and

\[
b_n(k_n) = \prod_{k=1}^k \mu_n(k).
\]  

The normalization constant \( G(K) \) is given by

\[
G(K) = \sum_{k \in S(K)} \prod_{n=1}^N \frac{k_n!}{b_n(k_n)} \prod_{r=1}^R x_{n,r}^{k_{n,r}}.
\]
The evaluation of the normalization constant involves a summation over all states in the set $S(K)$. The cardinality of the set $S(K)$ prohibits the use of a direct enumeration method and one has to develop more efficient computational procedures. We shall use a $z$-transform approach to derive a recursive relation which defines a relatively efficient evaluation method.

The normalization constant can be interpreted as a coefficient of an $R$-dimensional $z$-transform. Define for all $z = (z_1, \ldots, z_R) \in \mathbb{C}^R$

$$H(z) = \sum_{k_1=0}^{\infty} \cdots \sum_{k_R=0}^{\infty} G(K) \prod_{r=1}^{R} z_r^{K_r}. \quad (7)$$

It is easily verified that the following two Lemmata hold.

**Lemma 1:**
For all $z \in \mathbb{C}^R$

$$H(z) = \prod_{n=1}^{N} H_n(z). \quad (8)$$

where, for $n=1,\ldots,N$,

$$H_n(z) = \sum_{k_{n,1}=0}^{\infty} \cdots \sum_{k_{n,R}=0}^{\infty} \frac{k_n!}{b_n(k_n)} \prod_{r=1}^{R} \frac{(x_{n,r}z_r)^{k_{n,r}}}{k_{n,r}!}. \quad (9)$$

**Lemma 2:**
For all $K \geq 0$ the normalization constant $G(K)$ satisfies

$$G(K) = \prod_{r=1}^{R} \frac{1}{K_r!} \left[ \frac{d^{K_1}}{dz_1} \cdots \frac{d^{K_R}}{dz_R} H(z) \right]_{z=0}. \quad (10)$$

3. The main recursive relation

In order to derive a recursive scheme for the computation of the normalization constant at a given population vector $K$ we shall apply the results of Lemma 1 and Lemma 2.

To describe the derivatives of the function $H(z)$ some notation is introduced. For $u \in \mathbb{N}_0^N$ and $z \in \mathbb{C}^R$

$$H(u,z) = \prod_{n=1}^{N} H_n(u_nz). \quad (11)$$

with, for $n=1,\ldots,N$.

$$H_n(u_nz) = \sum_{k_{n,1}=0}^{\infty} \cdots \sum_{k_{n,R}=0}^{\infty} \frac{(k_n + u_n)!}{b_n(k_n + u_n)} \prod_{r=1}^{R} \frac{(x_{n,r}z_r)^{k_{n,r}}}{k_{n,r}!}. \quad (12)$$
For $i = 1, \ldots, R$, $u \in \mathbb{N}_0^N$ and $z \in \mathbb{C}^R$

$$F_i(u, z) = \frac{d^{X_i}}{K_1! dZ_1^{X_i}} \cdots \frac{d^{X_i}}{K_i! dZ_i^{X_i}} H(u, z).$$

(13)

It is convenient to define $F_0(u, z)$ by

$$F_0(u, z) = H(u, z).$$

(14)

With these definitions it will be evident that the normalization constant is given by

$$G(K) = F_R(0, 0).$$

(15)

The main theorem can now be formulated.

Theorem 1:

For $i = 1, \ldots, R$, $u \in \mathbb{N}_0^N$ and $z \in \mathbb{C}^R$

$$F_i(u, z) = \sum_{l \in V(K_i)} \left( \prod_{n=1}^N \frac{x_{n,i}^{l_n}}{l_n!} \right) F_{i-1}(u + l, z).$$

(16)

where the set $V(K_i)$ is defined by

$$V(K_i) = \{ l \in \mathbb{N}_0^N | \sum_{n=1}^N l_n = K_i \}. $$

(17)

Proof:

Note that

$$\frac{1}{K_i!} \frac{d^{X_i}}{dZ_i^{X_i}} H(u, z) =$$

$$\frac{1}{K_i!} \frac{d^{X_i}}{dZ_i^{X_i}} \prod_{n=1}^N H_n(u_n, z) =$$

$$\frac{1}{K_i!} \sum_{l \in V(K_i)} \frac{K_i!}{l_1! \cdots l_N!} \prod_{n=1}^N \frac{d^{l_n}}{dZ_i^{l_n}} H_n(u_n, z) =$$

$$\sum_{l \in V(K_i)} \prod_{n=1}^N \frac{x_{n,i}^{l_n}}{l_n!} H_n(u_n + l_n, z) =$$

$$\sum_{l \in V(K_i)} \left( \prod_{n=1}^N \frac{x_{n,i}^{l_n}}{l_n!} \right) H(u + l, z).$$

where we have used the fact that

$$\frac{d^l}{dZ_i^l} H_n(u_n, z) =$$

$$\frac{d^l}{dZ_i^l} \sum_{k_n, r=0}^\infty \sum_{k_{n+1}=0}^\infty \frac{(k_n + u_n)!}{b_n(k_n + u_n)} \prod_{r=1}^l \frac{(x_n r z_r)^{k_n r}}{k_n r!} =$$
The theorem follows from relation (15).

We now can formulate a set of recursive relations which form the basis of an enumeration algorithm for the computation of the normalization constant at a given population vector.

Introduce for \( u \in N_0^N \) and \( i = 0,1, \ldots, R \)

\[
F_i(u) = F_i(u, 0)
\]  

(18)

Then the following corollary is an immediate consequence of Theorem 1.

**Corollary 1:**

For \( i = 1, \ldots, R \) and \( u \in N_0^N \)

\[
F_i(u) = \sum_{l \in V(K_i)} \left[ \prod_{n=1}^{N} \frac{x_{n,i}}{l_n!} \right] F_{i-1}(u + l)
\]

(19)

and for \( i = 0 \) and \( u \in N_0^N \)

\[
F_0(u) = \prod_{n=1}^{N} \frac{u_n!}{b_n(u_n)}
\]

(20)

To show that (19) and (20) correspond with the results in [1] a transformation of our scheme is needed. Let us therefore introduce for \( i = 0,1, \ldots, R \) and \( u \in N_0^N \)

\[
\Phi_i(u) = \prod_{n=1}^{N} \frac{b_n(u_n)}{u_n!} F_i(u)
\]

(21)

Then the following corollary is an immediate consequence of Corollary 1.

**Corollary 2:**

For \( i = 1, \ldots, R \) and \( u \in N_0^N \)

\[
\Phi_i(u) = \sum_{l \in V(K_i)} \left[ \prod_{n=1}^{N} E_{n,i}(u_n, l_n) \right] \Phi_{i-1}(u + l)
\]

(22)

and for \( i = 0 \) and \( u \in N_0^N \)

\[
\Phi_0(u) = 1
\]

(23)

where, for \( n = 1, \ldots, N \) and \( u_n, l_n \in N_0 \).
\begin{equation}
E_{n,i}(u_n,l_n) = \frac{b_n(u_n)}{b_n(u_n + l_n)} \begin{pmatrix} u_n + l_n \\ u_n \end{pmatrix}^{l_n}_{x_{n,i}}.
\end{equation}

These relations correspond with the results presented in [1].

4. Concluding remarks

We have shown that the recursive scheme introduced in [1] can be derived using a $z$-transform approach.

For some special cases the derivation becomes more simple. As an example we mention the case where all workstations have a fixed service rate, say $b_n(k) = 1$ for all $k$. Then the function $H(z)$ reduces to

\begin{equation}
H(z) = \prod_{n=1}^{N} \left( 1 - \sum_{r=1}^{R} x_{n,r} z_r \right)^{-1}
\end{equation}

One may verify that the proof of Theorem 1 is more straightforward for this particular case. Furthermore, the non-trivial recursion arises in a more natural way.

Another important simplification arises if we follow the line sketched in [1], where all customer chains are broken down into as many chains as there are customers. For the resulting model this implies that $K_r = 1$ for all $r$. In this case the recursion is defined by the following relations.

**Corollary 3:**

If $K_r = 1$ for $r = 1,...,R$, then for $i = 1,...,R$ and $u \in N_0^N$

\begin{equation}
F_i(u) = \sum_{n=1}^{N} x_{n,i} F_{i-1}(u + e_n)
\end{equation}

and for $i = 0$ and $u \in N_0^N$

\begin{equation}
F_0(u) = \prod_{n=1}^{N} \frac{u_n!}{b_n(u_n)}
\end{equation}

where $e_n$ denotes the n-th unit vector.

For a discussion on the implementation of the algorithm and notes on the time and space complexity of the algorithm we refer to [1].
5. References


