Buzen's algorithm

From Wikipedia, the free encyclopedia

In queueing theory, a discipline within the mathematical theory of probability, **Buzen's algorithm** (or **convolution algorithm**) is an algorithm for calculating the normalization constant G(*N*) in the Gordon–Newell theorem. This method was first proposed by Jeffrey P. Buzen in 1973.[1] Computing G(*N*) is required to compute the stationary probability distribution of a closed queueing network.[2]

Performing a naïve computation of the normalising constant requires enumeration of all states. For a system with *N* jobs and *M* states there are $\binom{N+M-1}{M-1}$ states. Buzen's algorithm "computes G(1), G(2), ..., G(*N*) using a total of *NM* multiplications and *NM* additions." This is a significant improvement and allows for computations to be performed with much larger networks.[1]

Contents [hide]

Problem setup [edit]

Consider a closed queueing network with *M* service facilities and *N* circulating customers. Write $n_i(t)$ for the number of customers present at the *i*th facility at time *t*, such that $\sum_{i=1}^M n_i = N$. We assume that the service time for a customer at the *i*th facility is given by an exponentially distributed random variable with parameter μ_i and that after completing service at the *i*th facility a customer will proceed to the *j*th facility with probability p_{ij}.[2]

It follows from the Gordon–Newell theorem that the equilibrium distribution of this model is

$$\mathbb{P}(n_1, n_2, \cdots, n_M) = \frac{1}{\mathrm{G}(N)} \prod_{i=1}^{M} (X_i)^{n_i}$$

where the X_i are found by solving

$$\mu_j X_j = \sum_{i=1}^{M} \mu_i X_i p_{ij} \quad \text{ for } j = 1, \ldots, M.$$

and *G*(*N*) is a normalizing constant chosen that the above probabilities sum to 1.[1]

Buzen's algorithm is an efficient method to compute G(*N*).[1]

Algorithm description [edit]

Write g(*N*,*M*) for the normalising constant of a closed queueing network with *N* circulating customers and *M* service stations. The algorithm starts by noting solving the above relations for the X_i and then setting starting conditions[1]

$$g(0, m) = 1 \text{ for } m = 1, 2, \cdots, M$$
$$g(n, 1) = (X_1)^n \text{ for } n = 0, 1, \cdots, N.$$

The recurrence relation[1]

$$g(n, m) = g(n, m-1) + X_m g(n-1, m).$$

is used to compute a grid of values. The sought for value G(*N*) = g(*N*,*M*).[1]

Marginal distributions, expected number of customers [edit]

The coefficients g(*n*,*m*), computed using Buzen's algorithm, can also be used to compute marginal distributions and expected number of customers at each node.

$$\mathbb{P}(n_i = k) = \frac{X_i^k}{G(N)}[G(N-k) - X_i G(N-k-1)] \quad \text{for } k = 0, 1, \ldots, N-1,$$

$$\mathbb{P}(n_i = N) = \frac{X_i^N}{G(N)}[G(0)].$$

the expected number of customers at facility *i* by

$$\mathbb{E}(n_i) = \sum_{k=1}^{N} X_i^k \frac{G(N-k)}{G(N)}.$$

Implementation [edit]

It will be assumed that the X_m have been computed by solving the relevant equations and are available as an input to our routine. Although *g* is in principle a two dimensional matrix, it can be computed in a column by column fashion starting from the leftmost column. The routine uses a single column vector *C* to represent the current column of *g*.

```
C[0] := 1
for n := 1 step 1 until N do
   C[n] := 0;

for m := 1 step 1 until M do
for n := 1 step 1 until N do
   C[n] := C[n] + X[m]*C[n-1];
```

At completion, *C* contains the desired values *G(0), G(1)* to *G(N)*. [1]

References [edit]

1. ^ a b c d e f g h Buzen, J. P. (1973). "Computational algorithms for closed queueing networks with exponential servers" (PDF). *Communications of the ACM* **16** (9): 527. doi:10.1145/362342.362345.
2. ^ a b Gordon, W. J.; Newell, G. F. (1967). "Closed Queuing Systems with Exponential Servers". *Operations Research* **15** (2): 254. doi:10.1287/opre.15.2.254. JSTOR 168557.

- Jain: The Convolution Algorithm (class handout)
- Menasce: Convolution Approach to Queueing Algorithms (presentation)

v · t · e	Queueing theory [hide]
Single queueing nodes	D/M/1 queue · M/D/1 queue · M/D/c queue · M/M/1 queue (Burke's theorem) · M/M/c queue · M/M/∞ queue · M/G/1 queue (Pollaczek–Khinchine formula · Matrix analytic method) · M/G/k queue · G/M/1 queue · G/G/1 queue (Kingman's formula · Lindley equation) · Fork–join queue · Bulk queue
Arrival processes	Poisson process · Markovian arrival process · Rational arrival process
Queueing networks	Jackson network (Traffic equations) · Gordon–Newell theorem (Mean value analysis · **Buzen's algorithm**) · Kelly network · G-network · BCMP network
Service policies	FIFO · LIFO · Processor sharing · Shortest job first · Shortest remaining time
Key concepts	Continuous-time Markov chain · Kendall's notation · Little's law · Product-form solution (Balance equation · Quasireversibility · Flow-equivalent server method) · Arrival theorem · Decomposition method · Beneš method
Limit theorems	Fluid limit · Mean field theory · Heavy traffic approximation (Reflected Brownian motion)
Extensions	Fluid queue · Layered queueing network · Polling system · Adversarial queueing network · Loss network · Retrial queue
Information systems	Data buffer · Erlang (unit) · Erlang distribution · Flow control (data) · Message queue · Network congestion · Network scheduler · Pipeline (software) · Quality of service · Scheduling (computing) · Teletraffic engineering
Category	

Categories: Stochastic processes | Queueing theory | Statistical algorithms