Mean Value Analysis

Mean value analysis (MVA) is an efficient algorithm that allow us to analyze product form queueing networks and obtain mean values for queue lengths and response times, as well as throughputs. The efficiency comes with a price – MVA does not compute the joint probability distribution for queue lengths. However, in many if not most performance evaluation situations, the mean values are the performance metrics of interest.

Single Class Systems

We begin with systems in which there is a single job class (job classes are usually referred to as “chains” in queueing network theory). These systems may either be open or closed.

1. Open systems (arrival rate = λ)

   D_m is the total demand of a single customer for queue m. V_m is the visit ratio for queue m, with queue 0 (the "outside world") the reference queue; i.e., V_m is the average number of visits a single customer makes to queue m. μ_m is the service rate at queue m.

   $$D_m = V_m \cdot \frac{1}{\mu_m}$$

   The maximum throughput for the system occurs at the value of the arrival rate which saturates the queue with the largest demand.

   $$\text{maximum throughput} = \frac{1}{\max\limits_{1 \le m \le M}(D_m)} = \lambda_{sat}$$

   Throughput for queue m at $\lambda < \lambda_m$ is $\lambda_m = \lambda V_m = X_m(\lambda)$

   Utilization of queue m at $\lambda < \lambda_m$ is $\rho_m = \lambda_m \cdot \frac{1}{\mu_m} = \lambda D_m$

   Response time at queue m at $\lambda < \lambda_m$ is $R_m(\lambda)$.

   • for an IS queue:

   $$R_m(\lambda) = V_m \cdot \frac{1}{\mu_m} = D_m$$

   • for a FCFS, LCFSPR, or PS queue:

   $R_m(\lambda)$ is the time spent in service + the time spent waiting for other customers to complete service.

time spent in service $= V_m \cdot \frac{1}{\mu_m} = D_m$

time spent waiting for other customers to complete service

$$= V_m \cdot A_m(\lambda) \cdot \frac{1}{\mu_m} = A_m(\lambda) D_m$$

where $A_m(\lambda)$ is the average number of customers at queue m as seen by an arriving job.

This is intuitively correct for a FCFS queue with exponential service time, and hence must also be true for LCFSPR and PS queues in a product-form network.

$$\Rightarrow R_m(\lambda) = \begin{cases} D_m & \text{, IS queue} \\ D_m\left(1 + A_m(\lambda)\right) & \text{, FCFS, PS, or LCFSPR queue} \end{cases}$$

Also, for product form networks with a single open class,

$$A_m(\lambda) = Q_m(\lambda)$$

where $Q_m(\lambda)$ is the expectation of the length of queue m when the arrival rate is λ. That is, the average number of customers at queue m as seen by an arrival is exactly equal to the average queue length at queue m (averaged over all time, not just arrival instants).

$\Rightarrow R_m(\lambda) = D_m\left(1 + Q_m(\lambda)\right)$ for a queueing center.

Since $Q_m(\lambda) = \lambda R_m(\lambda)$,

$$R_m(\lambda) = D_m\left(1 + \lambda R_m(\lambda)\right)$$
$$R_m(\lambda) - \lambda D_m R_m(\lambda) = D_m$$
$$R_m(\lambda) = \frac{D_m}{1 - \lambda D_m} = \frac{D_m}{1 - \rho_m(\lambda)}$$

This looks reasonable, since

$$\rho_m(\lambda) \to 0 \Rightarrow R_m(\lambda) \to D_m$$
$$\rho_m(\lambda) \to 1 \Rightarrow R_m(\lambda) \to \infty$$

Queue lengths:

$$Q_m(\lambda) = \lambda R_m(\lambda)$$
$$= \begin{cases} \rho_m(\lambda) & \text{(IS)} \\ \dfrac{\rho_m(\lambda)}{1-\rho_m(\lambda)} & \text{(FCFS, LCFSPR, PS)} \end{cases}$$

System response time:

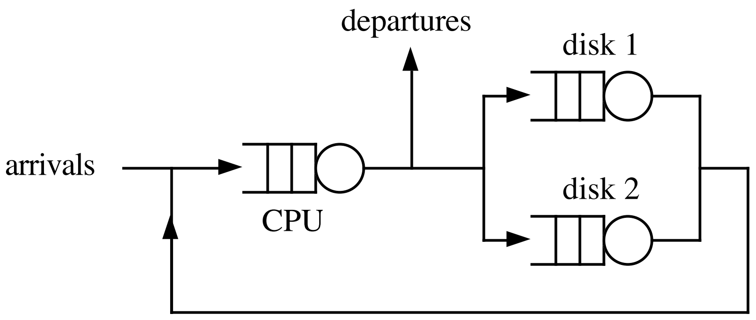
$$R(\lambda) = \sum_{m=1}^{M} R_m(\lambda)$$

Average number of customers in system:

$$Q(\lambda) = \lambda R(\lambda) = \sum_{m=1}^{M} Q_m(\lambda)$$

---

ex.



Given:

$V_{cpu} = 121$ $\quad V_{disk1} = 70$ $\quad V_{disk2} = 50$

$\dfrac{1}{\mu_{cpu}} = .005$ $\quad \dfrac{1}{\mu_{disk1}} = .030$ $\quad \dfrac{1}{\mu_{disk2}} = .027$

$\Downarrow \quad \Downarrow \quad \Downarrow$

$D_{cpu} = .605$ $\quad D_{disk1} = 2.1$ $\quad D_{disk2} = 1.35$

$\lambda = .3$

Model outputs:

$$\lambda_{sat} = \frac{1}{D_{max}} = \frac{1}{D_{disk1}} = 0.476 \text{ jobs/sec}$$

$$X_{cpu}(\lambda) = \lambda V_{cpu} = 0.3(121) = 36.3$$
$$X_{disk1}(\lambda) = \lambda V_{disk1} = 0.3(70) = 21$$
$$X_{disk2}(\lambda) = \lambda V_{disk2} = 0.3(50) = 15$$

$$R_{cpu}(\lambda) = \frac{D_{cpu}}{1-\rho_{cpu}(\lambda)} = \frac{D_{cpu}}{1-\lambda D_{cpu}} = \frac{0.605}{1-0.3(0.605)} \approx 0.740$$
$$R_{disk1}(\lambda) = \frac{D_{disk1}}{1-\lambda D_{disk1}} = \frac{2.1}{1-0.3(2.1)} \approx 5.676$$
$$R_{disk2}(\lambda) = \frac{D_{disk2}}{1-\lambda D_{disk2}} = \frac{1.35}{1-0.3(1.35)} \approx 2.269$$

$$Q_{cpu}(\lambda) = \lambda R_{cpu}(\lambda) \approx 0.3(0.740) = 0.222$$
$$Q_{disk1}(\lambda) = \lambda R_{disk1}(\lambda) \approx 0.3(5.676) = 1.7028$$
$$Q_{disk2}(\lambda) = \lambda R_{disk2}(\lambda) \approx 0.3(2.269) = 0.6807$$

$$R(\lambda) = R_{cpu}(\lambda) + R_{disk1}(\lambda) + R_{disk2}(\lambda) \approx 8.685$$
$$Q(\lambda) = Q_{cpu}(\lambda) + Q_{disk1}(\lambda) + Q_{disk2}(\lambda) \approx 2.6055$$

2. Closed systems

We have three basic quantities of interest – X, R, and Q – and we have three equations relating them:

$$X(N) = \frac{N}{\sum_{m=1}^{M} R_m(N)}$$

$$Q_m(N) = X(N)R_m(N)$$

$$R_m(N) = \begin{cases} D_m & \text{(IS)} \\ D_m(1+A_m(N)) & \text{(FCFS, LCFSPR, PS)} \end{cases}$$

where $R_m(N)$ is the response time at center *m* when the total number of customers in the system is N, and similarly for the other quantities.

If we knew $A_m(N)$, we could compute $R_m(N)$, which would allow us to compute $X(N)$. This would allow us to compute $Q_m(N)$.

For open models, we used

$$Q_m(\lambda) = A_m(\lambda)$$

In the closed model case, we cannot use $Q_m(N) = A_m(N)$, however. The following simple example with two servers and a single job illustrates this.

$$D_1 = D_2 \Rightarrow \rho_1 = \rho_2 = 0.5 \Rightarrow Q_1 = Q_2 = 0.5$$

but obviously $A_1 = A_2 = 0$ since there is only one customer in the system.

The problem in general is that at an arrival instant the arriving customer cannot itself already be in the queue.

However, for product form networks, we have the relation

$$A_m(N) = Q_m(N-1)$$

i.e., an arriving customer sees (on average) the steady-state number of customers that there would have been if there were one fewer customer in the system.

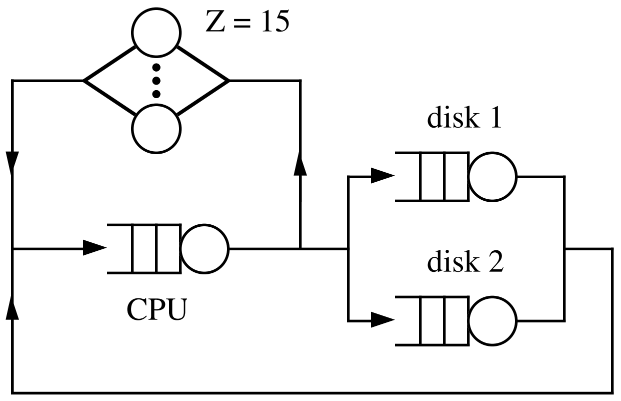
We also know that $Q_m(0) = 0 \ \forall \ m$.

This gives us an iterative algorithm (called *Mean Value Analysis*) that allows us to solve the closed system.

```
for m=0 to M do Qm = 0
for n=1 to N do begin
    for m=1 to M do $R_m = \begin{cases} D_m & , \text{IS} \\ D_m(1+Q_m) & , \text{FCFS, LCFSPR, PS} \end{cases}$
    $X = \frac{n}{\sum_{m=1}^{M} R_m}$
    for m=1 to M do Qm = XRm
end
```

The time complexity of this algorithm is $O(NK)$, and the space complexity is $O(K)$.

ex.



$$\left.\begin{aligned} D_{cpu} &= 0.605 \\ D_{disk1} &= 2.1 \\ D_{disk2} &= 1.35 \end{aligned}\right\} \text{as before}$$

Solve for $N = 3$

		n=0	n=1	n=2	n=3
	CPU	-	.605	.624	.644
R	disk1	-	2.1	2.331	2.605
	disk2	-	1.35	1.446	1.551
X		-	.0525	.1031	.1515
	CPU	0	.0318	.0643	.0976
Q	disk1	0	.1102	.2403	.3947
	disk2	0	.0709	.1491	.2350

Multiple Class Systems

In multiple class systems, each job class may have its own demand for each queue. The routing of jobs between queues and the per-visit service demand may also be class-dependent.

1. Open systems

C = number of classes

λ_c = arrival rate for class c

$\underline{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_C)$

$\mu_{c,m}$ = service rate for a class c job at queue m

$V_{c,m}$ = visit ratio for a class c job at queue m

$D_{c,m}$ = average total demand of a class c job for service at queue m

$\rho_{c,m}(\underline{\lambda})$ = utilization of queue m by class c jobs

$\rho_m(\underline{\lambda})$ = utilization of queue m by all job classes

$$= \sum_{c=1}^{C} \lambda_c D_{c,m}$$

$R_{c,m}(\underline{\lambda})$ = average residence time of a class c job at queue m

$Q_{c,m}(\underline{\lambda})$ = average number of class c jobs at queue m

$\max\limits_{1 \le m \le M} \left(\rho_{\mathrm{m}}(\underline{\lambda})\right) < 1$ for stability

$X_{c,m}(\underline{\lambda}) = \lambda_c V_{c,m}$ is the throughput for a class c job at queue m

$$\rho_{c,m}(\underline{\lambda}) = X_{c,m}(\underline{\lambda}) \cdot \frac{1}{\mu_{c,m}} = \lambda_c D_{c,m}$$

$$R_{c,m}(\underline{\lambda}) = \begin{cases} D_{c,m} & \text{(IS)} \\ D_{c,m}\left(1 + A_{c,m}(\underline{\lambda})\right) & \text{(FCFS, LCFSPR, PS)} \end{cases}$$

$A_{c,m}(\underline{\lambda})$ is the expected number of jobs at queue m at an arrival instant for a job of class c.

For open classes in separable networks,

$$A_{c,m}(\underline{\lambda}) = Q_m(\underline{\lambda}) = \sum_{j=1}^{C} Q_{j,m}(\underline{\lambda})$$

$\therefore$ for FCFS, LCFSPR, and PS queues

$$X_c(\underline{N}) = \frac{N_c}{\sum_{m=1}^{M} R_{c,m}(\underline{N})}$$

Note that this expression uses $D_{c,m}$, the average total service demand for a class c customer at queue c'. However, the arriving customer of class c may see customers of any class already in the queue, and the "intuitive" guess would be that the arriving customer would be delayed by $D_{c',m}$ for each customer of class c' already in the queue. We will discuss this further in the section on closed systems.

Since the expression in parentheses multiplying $D_{c,m}$ does not depend on c, we can simplify this expression.

$$\frac{R_{c,m}(\underline{\lambda})}{R_{j,m}(\underline{\lambda})} = \frac{D_{c,m}}{D_{j,m}} \quad \Rightarrow \quad R_{j,m}(\underline{\lambda}) = \frac{D_{j,m}}{D_{c,m}} R_{c,m}(\underline{\lambda})$$

$$\Rightarrow \quad R_{c,m}(\underline{\lambda}) = D_{c,m}\left(1 + \sum_{j=1}^{C} \lambda_j \frac{D_{j,m}}{D_{c,m}} R_{j,m}(\underline{\lambda})\right)$$

$$R_{c,m}(\underline{\lambda})\left[1 - \sum_{j=1}^{C} \lambda_j D_{j,m}\right] = D_{c,m}$$

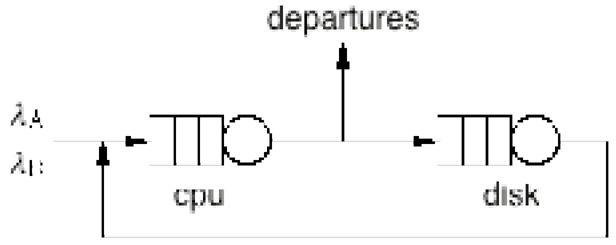
$$R_{c,m}(\underline{\lambda}) = \frac{D_{c,m}}{1 - \sum_{j=1}^{C} \lambda_j D_{j,m}} = \frac{D_{c,m}}{1 - \sum_{j=1}^{C} \rho_{j,m}(\underline{\lambda})}$$

$$= \frac{D_{c,m}}{1 - \rho_m(\underline{\lambda})}$$

$$Q_{c,m}(\underline{\lambda}) = \lambda_c R_{c,m}(\underline{\lambda})$$

etc.

ex. two classes of customers *A* and *B*



$$D_{A,cpu} = 1 \quad V_{A,cpu} = 10 \quad \lambda_A = \tfrac{3}{19}$$
$$D_{B,cpu} = 2 \quad V_{B,cpu} = 5 \quad \lambda_B = \tfrac{2}{19}$$
$$D_{A,disk} = 3 \quad V_{A,disk} = 9$$
$$D_{B,disk} = 4 \quad V_{B,disk} = 4$$

$$X_{A,cpu}(\underline{\lambda}) = \lambda_A V_{A,cpu} = \tfrac{3}{19} \cdot 10 = 1.579$$
$$\rho_{A,cpu}(\underline{\lambda}) = \lambda_A D_{A,cpu} = \tfrac{3}{19} \cdot 1 = 0.1579$$
$$\rho_{B,cpu}(\underline{\lambda}) = \lambda_B D_{B,cpu} = \tfrac{2}{19} \cdot 2 = 0.2105$$
$$\rho_{A,disk}(\underline{\lambda}) = \lambda_A D_{A,disk} = \tfrac{3}{19} \cdot 3 = 0.4737$$
$$\rho_{B,disk}(\underline{\lambda}) = \lambda_B D_{B,disk} = \tfrac{2}{19} \cdot 4 = 0.4211$$

$$R_{A,cpu}(\underline{\lambda}) = \frac{D_{A,cpu}}{1-\rho_{cpu}(\underline{\lambda})} = \frac{1}{1-\frac{7}{19}} = 1 \cdot \frac{19}{12} = 1.583$$
$$R_{B,cpu}(\underline{\lambda}) = \frac{D_{B,cpu}}{1-\rho_{cpu}(\underline{\lambda})} = \frac{2}{1-\frac{7}{19}} = 2 \cdot \frac{19}{12} = 3.167$$
$$R_{A,disk}(\underline{\lambda}) = \frac{D_{A,disk}}{1-\rho_{disk}(\underline{\lambda})} = \frac{3}{1-\frac{17}{19}} = 3 \cdot \frac{19}{2} = 28.5$$
$$R_{B,disk}(\underline{\lambda}) = \frac{D_{B,disk}}{1-\rho_{disk}(\underline{\lambda})} = \frac{4}{1-\frac{17}{19}} = 4 \cdot \frac{19}{2} = 38$$
$$R_A(\underline{\lambda}) = R_{A,cpu}(\underline{\lambda}) + R_{A,disk}(\underline{\lambda}) = 30.083$$

etc.

2. Closed systems

N_c = number of customers in class c
$\underline{N}$ = $(N_1, N_2, \ldots, N_C)$

As in the case for single class closed systems, there are three equations:

$$X_c(\underline{N}) = \frac{N_c}{\sum_{m=1}^{M} R_{c,m}(\underline{N})}$$

$$Q_{c,m}(\underline{N}) = X_c(\underline{N}) R_{c,m}(\underline{N})$$

$$R_{c,m}(\underline{N}) = \begin{cases} D_{c,m} & \text{, IS} \\ D_{c,m}\left(1 + A_{c,m}(\underline{N})\right) & \text{, FCFS, LCFSPR, PS} \end{cases}$$

Notice once again that we are using $D_{c,m}$ in the expression for queues with contention, for all customers in the queue at the arrival instant, regardless of class. Some explanation seems in order.

That the above expression applies to a FCFS queue is easy to explain. FCFS queues have the additional restriction that all classes of customers must see the same (exponential) service time distribution; thus, $D_{c,m} = D_{c',m}$ for all classes c and c'. Also, the distribution of the residual lifetime of an exponential random variable is identical to the original distribution, so that it does not matter how long a job has been in service at the arrival instant.

We can wave our hands at a PS queue and at least make the expression sound plausible. Recall that in a PS queue, all jobs in the queue receive simultaneous service. If there are N_m jobs in queue m on average while a job is being serviced, the time to complete that job's service requirement of $D_{c,m}$ should be $N_m D_{c,m}$. It remains to be proven that the average number of jobs in the queue while a job is being serviced is 1 + the average number of jobs in the queue at the job's arrival instant.

Similar to the single class MVA algorithm, we have the chain of computations $A_{c,m}(\underline{N}) \to R_{c,m}(\underline{N}) \to X_c(\underline{N}) \to Q_{c,m}(\underline{N})$. The question is how to find $A_{c,m}(\underline{N})$.

Let $\underline{N} - 1_c = \left(N_1, N_2, \ldots, N_{c-1}, N_c - 1, N_{c+1}, \ldots, N_c\right)$.

For closed product-form networks,

$$A_{c,m}(\underline{N}) = Q_m(\underline{N} - 1_c)$$

This gives us the following *multiple class MVA algorithm*:

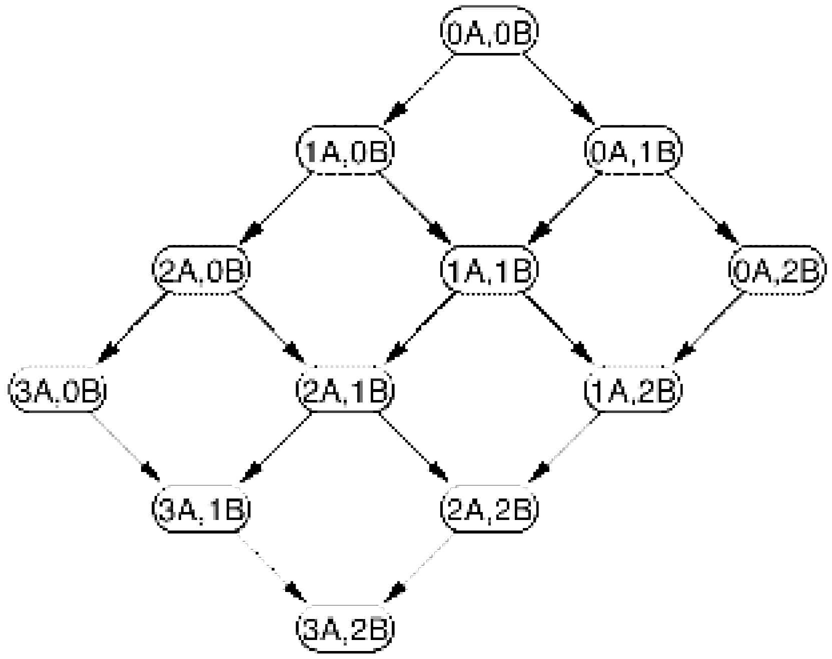
```
for m=1 to M do Q_m(0) = 0
for n=1 to $\sum_{c=1}^{C} N_c$ do
    for each feasible population n = (n_1,…,n_C) s.t.
    $n = \sum_{c=1}^{C} n_c \quad , n_c \ge 0$

    do begin
        for c=1 to C do
            for m=1 to M do
                $R_{c,m}(\underline{n}) = \begin{cases} D_{c,m} & \text{, IS} \\ D_{c,m}(1 + A_{c,m}(\underline{n})) & \text{, not IS} \end{cases}$

        for c=1 to C do $X_c = \frac{n_c}{\sum_{m=1}^{M} R_{c,m}(\underline{n})}$

        for m=1 to M do $Q_m(\underline{n}) = \sum_{c=1}^{C} X_c R_{c,m}$
    end
end
```

A feasible population with *n* jobs total is a set of jobs such that the number of jobs within each class c is between 0 and N_c, and the sum of the number of jobs in all classes is n. As you can see, the algorithm sets the total number of jobs in the system and then computes results for each feasible population with that many jobs before beginning computations for one more job total in the system. The following diagram illustrates the precedence of feasible states for a system with two classes *A* and *B*, $N_A = 3$, and $N_B = 2$.



The time complexity of the algorithm is $CM\prod_{c=1}^{C}\left(N_c+1\right)$, where CM is the time complexity of the computations for one feasible population, and the product term is the total number of feasible populations. The space complexity is $M\prod_{\substack{c=1 \\ c\neq c_{\max}}}^{C}(N_c+1)$.

ex. $D_{A,cpu} = 1$ $D_{A,disk} = 3$ $N_A = 1$
$D_{B,cpu} = 2$ $D_{B,disk} = 4$ $N_B = 1$

	0A,0B	1A,0B	0A,1B	1A,1B
$R_{A,cpu}$	-	1	-	$\frac{4}{3}$
$R_{A,disk}$	-	3	-	5
$R_{B,cpu}$	-	-	2	$\frac{5}{2}$
$R_{B,disk}$	-	-	4	7
X_A	-	$\frac{1}{4}$	-	$\frac{3}{19}$
X_B	-	-	$\frac{1}{6}$	$\frac{2}{19}$
$Q_{A,cpu}$	0	$\frac{1}{4}$	0	$\frac{4}{19}$
$Q_{A,disk}$	0	$\frac{3}{4}$	0	$\frac{15}{19}$
$Q_{B,cpu}$	0	0	$\frac{1}{3}$	$\frac{5}{19}$
$Q_{B,disk}$	0	0	$\frac{2}{3}$	$\frac{14}{19}$

Approximate MVA - multiple classes

The time and space complexities are proportional to the number of feasible populations, and this number quickly gets large for even relatively few classes and jobs per class. The following algorithm produces approximate solutions for the given class populations without having to iterate through all smaller feasible populations. Step 1 distributes the members of each class equally among all the queues.

1. $Q_{c,m}(\underline{N}) = \dfrac{N_c}{M} \quad \forall\, c,m$
2. $A_{c,m}(\underline{N}) = h_c\left(Q_{1,m}(\underline{N}), Q_{2,m}(\underline{N}), \ldots, Q_{C,m}(\underline{N})\right)$
3. Compute $R_{c,m}(\underline{N})$ given $A_{c,m}(\underline{N})$

   Compute $X_c(\underline{N})$ given $A_{c,m}(\underline{N})$

   Compute new $Q_{c,m}(\underline{N})$ given $X_c(\underline{N})$ and $R_{c,m}(\underline{N})$
4. If $\left|\dfrac{\text{new } Q_{c,m}(\underline{N}) - \text{old } Q_{c,m}(\underline{N})}{\text{old } Q_{c,m}(\underline{N})}\right| < \delta$, stop; else go to 2.

The trick here is in step 2, where we use an approximation for the number of jobs in queue m at a job class c arrival instant.

$$
\begin{aligned}
A_{c,m}(\underline{N}) &= Q_{c,m}(\underline{N} - 1_c) \\
&\approx h_c\left(Q_{1,m}(\underline{N}), Q_{2,m}(\underline{N}), \ldots, Q_{C,m}(\underline{N})\right) \\
&= \frac{N_c - 1}{N_c} Q_{c,m}(\underline{N}) + \sum_{\substack{l=1 \\ l \neq c}}^{C} Q_{l,m}(\underline{N})
\end{aligned}
$$

That is, we assume the number of class $c' \neq c$ jobs in queue m at a class c arrival instant is the steady state average number of class c' jobs in the queue, but the number of class c jobs must be adjusted by a factor reflecting that the arriving job could not already be in the queue.

Mixed Open and Closed Models

$\{O\}$ = set of all open classes
$\{C\}$ = set of all closed classes
$\underline{I}$ = load vector
= $(I_1, I_2, \ldots, I_C)$ where $I_c = \lambda_c$ if $c \in \{O\}$ and $I_c = N_c$ if $c \in \{C\}$

(1) Since the throughput for the open classes is already fixed, first compute the utilization of each queue m by <u>all</u> of the open classes.

$$\rho_{c,m}(\underline{I}) = \lambda_c D_{c,m} \quad \forall\, c \in \{O\}$$

$$\rho_{\{O\},m}(\underline{I}) = \sum_{c \in \{O\}} \rho_{c,m}(\underline{I})$$

(2) Inflate the demands for all the closed classes to account for the utilizations of the various queues by the open classes.

$$D^*_{c,m} = \frac{D_{c,m}}{1 - \rho_{\{O\},m}} \quad \forall\, c \in \{C\}$$

This is sometimes referred to as *load concealment*.

(3) Solve the model consisting only of the closed classes, using the inflated demands computed in step (2). Find the throughputs, average response times, and average queue lengths for each closed class at each queue.

Note that to get the actual utilization of a queue m by a closed class c, we need to use the actual rather than the inflated demand:

$$\rho_{c,m}(\underline{I}) = X_c(\underline{I}) D_{c,m} \quad \forall\, c \in \{C\}$$

(4) Calculate the open class response times and queue lengths.

$$R_{c,m}(\underline{I}) = D^*_{c,m}\left(1 + Q_{\{C\},m}(\underline{I})\right) \quad \forall\, c \in \{O\}$$
$$Q_{c,m}(\underline{I}) = \lambda_c R_{c,m}(\underline{I})$$

Note that $D^*_{c,m}$, the inflated service demand for an open class c customer at queue m, is computed as in step (2); i.e., using the utilization of the queue by all open classes. This seems unintuitive, but perhaps a better way of looking at this is to say that we are inflating the *queue length* rather than the service demand to account for the presence of the closed class customers.

---

ex. $D_{A,cpu} = \tfrac{1}{4}$ $\quad D_{B,cpu} = \tfrac{1}{2}$ $\quad D_{C,cpu} = \tfrac{1}{2}$ $\quad D_{D,cpu} = 1$

$D_{A,disk} = \tfrac{1}{6}$ $\quad D_{B,disk} = 1$ $\quad D_{C,disk} = 1$ $\quad D_{D,disk} = \tfrac{4}{3}$

$\lambda_A = 1$ $\quad \lambda_B = \tfrac{1}{2}$ $\quad N_C = 1$ $\quad N_D = 1$

$$\rho_{\{O\},cpu} = \lambda_A D_{A,cpu} + \lambda_B D_{B,cpu} = 1\left(\tfrac{1}{4}\right) + \tfrac{1}{2}\left(\tfrac{1}{2}\right) = \tfrac{1}{2}$$
$$\rho_{\{O\},disk} = \lambda_A D_{A,disk} + \lambda_B D_{B,disk} = 1\left(\tfrac{1}{6}\right) + \tfrac{1}{2}(1) = \tfrac{2}{3}$$

$$\Rightarrow \quad D^*_{C,cpu} = \frac{\tfrac{1}{2}}{1-\tfrac{1}{2}} = 1 \qquad D^*_{D,cpu} = \frac{1}{1-\tfrac{1}{2}} = 2$$
$$D^*_{C,disk} = \frac{1}{1-\tfrac{2}{3}} = 3 \qquad D^*_{D,disk} = \frac{\tfrac{4}{3}}{1-\tfrac{2}{3}} = 4$$

which are the same as the (uninflated) demands for the closed model example in the handout on solution techniques for multi-class networks.

$\Rightarrow \quad R_{C,cpu} = \tfrac{4}{3} \qquad R_{D,cpu} = \tfrac{5}{2}$

$R_{C,disk} = 5 \qquad R_{D,disk} = 7$

$X_C = \tfrac{3}{19} \qquad X_D = \tfrac{2}{19}$

$Q_{C,cpu} = \tfrac{4}{19} \qquad Q_{D,cpu} = \tfrac{5}{19}$

$Q_{C,disk} = \tfrac{15}{19} \qquad Q_{D,disk} = \tfrac{14}{19}$

$$R_{A,cpu} = \frac{D_{A,cpu}\left(1 + Q_{\{C\},cpu}\right)}{1-\rho_{\{O\},cpu}} = \frac{\tfrac{1}{4}\left(1+\tfrac{9}{19}\right)}{1-\tfrac{1}{2}} = \tfrac{14}{19}$$
$$R_{B,cpu} = \frac{D_{B,cpu}\left(1 + Q_{\{C\},cpu}\right)}{1-\rho_{\{O\},cpu}} = \frac{\tfrac{1}{2}\left(1+\tfrac{9}{19}\right)}{1-\tfrac{1}{2}} = \tfrac{28}{19}$$

$$Q_{A,cpu} = \lambda_A R_{A,cpu} = 1\left(\tfrac{14}{19}\right) = \tfrac{14}{19}$$
$$Q_{B,cpu} = \lambda_B R_{B,cpu} = \tfrac{1}{2}\left(\tfrac{28}{19}\right) = \tfrac{14}{19}$$

etc.
