Introduction to Queueing Theory

Prof. N. Selvaraju
Department of Mathematics
Indian Institute of Technology Guwahati, India

Lecture - 34
Mean-Value Analysis Algorithm

Hi and hello, everyone. What we have been seeing so far was the analysis of the closed Jackson network, in which we first defined what a closed Jackson network or Gordon Newell network is. We gave the solution as a product form solution even in this case. And the main complexity while computing the joint distribution arises in the computation of the normalization constant, which was $G(N)$. And to compute the normalization constant in an efficient manner, we gave an algorithm which was called as Buzen's algorithm or convolution algorithm. Because the terms, as you can recall, the terms involved in the recursive computation are much like the convolution function. So that is why this is also called a convolution algorithm. So, in all these processes so far, whatever we have used requires the computation of this $G(N)$, which is the normalization constant. Now, from the normalization constant, we get the joint probability distribution. And once we have the joint probability distribution, we can then obtain marginal distributions and the expected value measures. So, what we are going to see next is an algorithm that can be employed if you are interested only in the mean value quantities or the expected value measures rather than the joint distributions and marginal distributions. So this method, this procedure is called mean value analysis.

And the mean value analysis algorithm or MVA algorithm is an approach for the calculation of expected value measures directly without involving or computing the normalization constant in the case of this convolution algorithm or directly in a naive computation way if you want to compute the joint probability distribution and hence marginal distributions. So, what this algorithm will give us is the excepted value measures which are a mean number in any particular node, and the mean sojourn time in any particular node is what we are going to compute directly.

So, that is what is called as the mean value analysis, which is an approach for calculating the expected value measures directly, which is the mean number mean sojourn time. In fact, marginal distributions can also be computed, which we can look at it in a simpler setting, but we are not going to look at it, but it is possible that marginal distributions can also be obtained using this algorithm. And this is now we are not computing this normalization constant or state probability distributions in the generality. And the objective is to compute the mean value measures directly. How it does is that it recursively computes by incrementing the number of customers until the desired number of customer is reached, which mean that you start with zero customers, assuming that there are zero customers, you do the certain computation, then use that idea to get the quantities corresponding to one customer in the whole of network and two customers in the whole of network and so on. So, suppose if 100 is your number, then you increment up to 100 in each step up to 100 steps. So, then you will get at the $100th$ step, you are going to get the required performance measures of the closed queueing network that is under consideration with that 100 customers in this case.

But this is based on two principles, and that is the first one is called either the mean value theorem or the arrival theorem, which we have seen in such scenarios; when you have a finite source and things like that, we have already seen it, which is also relevant here. What is that? The queue length observed by an arriving customer is the same as the general time queue length in a closed Jackson network with one less customer that one less customer is probably himself or herself itself, so to say. So, that is what? In a network with N customers, the arrival point probabilities of finding the system in N would be the same as the arbitrary time probabilities of finding the system in n when there are $N-1$, i.e., $a_n(N) = p_n(N-1)$ customers in the network. So, this is what is called as the arrival theorem or the mean value theorem probably easier to remember and an appropriate word arrival theorem because it talks about the arrival and what it sees in equilibrium. This is the first principle that we apply. The second principle that we apply is that Little's formula is applicable throughout the network; I mean, for combined as a network, also this Little's formula is applicable and each node also it is applicable that is what we assume. So, it happens so, but it is basically these are the two principles upon which the algorithm is built. But this algorithm, one has to be careful that this algorithm is applicable only to systems where the service times at each node are exponentially distributed. And also, each node follows FCFS discipline. Of course, it can also be expanded to more general queuing disciplines, but you have to look at it more carefully. In certain cases, the exact thing will be applicable; for example, single server when you have each node, even LCFS discipline will be applicable, and some other discipline, whether it is applicable or not there has been if you want more detail or deeper ones then you have to look at that, but we are assuming for simplicity that FCFS discipline for which this is applicable here.

So, that is the mean value algorithms background. Now, the two principles give us basically the steps of how to compute the quantities, which are mean number and mean sojourn time in a recursive form for each node. Now, let us look at the first principle, which is about the arrival theorem. So, the queue length observed by an arrival arriving customer in an N customer node network would be the same as the queue length observed at an arbitrary time with $N-1$ customers in the whole of the network; that is what is the case.

- First principle $\Longrightarrow$

  ▶ Recall, for $M/M/1$, $W = \dfrac{1+L}{\mu}$. That is, the average time an arriving customer must wait is the average time to serve the queue size as seen by an arriving customer plus itself.

  So, this is basically $1/\mu$ plus L/μ. So, what is the Sojourn time of a customer there. It is his service time plus the service time of the average number of customers in the system. So, if there are L is the average number of customers in the system, each service time is $1/\mu$. So, L times $1/\mu$ is basically the average time to serve the queue size that he observes. And plus his own service time, which is $1/\mu$, this is an expression, but you know the expressions for L and W in this particular case very easily. So, $L = \frac{\rho}{1-\rho}$, and $W = \frac{1}{\mu(1-\rho)}$. That is what you were seeing. So, you can see that satisfies this relationship, but one can argue also like how and what is the meaning of that relationship, is basically you can think of this as $1/\mu$ plus L times $\frac{1}{\mu}$. And here because in $M/M/1$, $a_n = p_n$. So, basically, arriving customers or arbitrary time both are one and the same. But here, there is a difference. So, that is what you need to keep in mind.

  ▶ For $M/M/c$ also no adjustment is needed as L is based on $a_n = p_n$.

  ▶ The equivalent equation for our closed network, assuming that $c_i = 1 \quad \forall i$, is

$$W_i(N) = \frac{1 + L_i(N-1)}{\mu_i},$$

where

$$\begin{aligned} W_i(N) &= \text{ mean waiting time at node } i \text{ for a network with } N \text{ customers} \\ \mu_i &= \text{ mean service rate for the singer server at node } i \\ L_i(N-1) &= \text{ mean number at node } i \text{ in a network with } N-1 \text{ customers} \end{aligned}$$

So, the first principle, which is the arrival theorem, gives us this relationship for the close queueing network. This is quite easy, but the second principle is what we are assuming that Little's law is applicable throughout the network.

- Second principle $\implies$

$$L_i(N) = \lambda_i(N)W_i(N).$$

- If we can find $\lambda_i(N)$, which is the throughput at node i or the arrival rate at node i with a network with N customer, then we have a method of calculating L_i and W_i starting with an empty network and building up to one with N customers $(L_i(0) = 0, W_i(1) = \dfrac{1}{\mu_i})$

To compute $\lambda_i(N)$:

- Let $D_i(N)$ represent the average delay per customer between successive visits to node i for a network with N customers. Then per customer arrival rate would be as per the laws of conservation it will be $1/D_i(N)$, and you have seen in Markov chain and elsewhere to the similar analysis now, since the network has N customers. So, then

$$\lambda_i(N) = \frac{N}{D_i(N)}$$

- To get $D_i(N)$: Take traffic equations, set $\nu_i = \mu_i\rho_i$. Then $\nu_i = \sum_{j=1}^{k} \nu_j r_{ji}$.

  Now, we know that this set of equations is a linearly independent system.
  ▶ Set one ν_i to 1, say, $\nu_l = 1$ and solve for others.
  ▶ Then ν_i are relative throughput through node i, i.e., $\nu_i = \dfrac{\lambda_i}{\lambda_l}$.
  ▶ We can write $D_l(N) = \sum_{i=1}^{k} \nu_i W_i(N)$.

  So, $D_l(N)$ is weighted by this ν_i, which is the relative throughput.

So, basically, what do we have? $D_l(N) = \sum_{i=1}^{k} \nu_i W_i(N)$ is the expression that you need to keep in mind. So, this is the weighted average of the average delays at each node weighted by the relative throughputs, which is basically this ν_i here or equivalently, you can think of this as weighted by the expected number of visits to each node prior to returning to this node normalized node which is l here, l is the normalized node that we are taking. Because you can think you can imagine this ν_i can also be interpreted as the expected number of visits to node i after leaving node l prior to returning to node l. So this visit count quantities we have not defined for this network, but that is the interpretation that we can give for this.

Say, for example, if we have a two-node network with $\nu_1 = 1$ and $\nu_2 = 2$ since the arrival rate at 2 is twice that of node 1, that is what we call relative throughputs relative arrival rates; ν_1 equal to 1 ν_2 equal to 2 means that whatever the arrival rate is at node 1 the arrival rate at node 2 is twice of that. So, the expected number of visits to node 2 after leaving node 1 and prior to returning to node 1 must also be 2; that is how it will be satisfying. So, once we have this ν_i so; that means that $D_l(N)$, I can compute. Then I can compute this quantity $\lambda_i(N)$. So, the lambda i of N, I can we can compute; this is the background behind this algorithm. So, now we can write down the algorithm. Many a time, the algorithm is directly written, but you can give a little bit of understanding of what the step that you have involved is.

Mean-Value Analysis (MVA) Algorithm:

Objective: To find $L_i(N)$ and $W_i(N)$ in a k-node, single-server-per-node network with routing probability matrix $R = (r_{ij})$.

1. Solve $\nu_i = \sum_{j=1}^{k} \nu_j r_{ji}, \quad i = 1, 2, \ldots, k$, setting one of the ν_i's (say, ν_l) equal to 1.

2. Initialize $L_i(0) = 0, \quad i = 1, 2, \ldots k$

3. For $n = 1$ to N, calculate

   a) $W_i(n) = \dfrac{1 + L_i(n-1)}{\mu_i}, \qquad i = 1, 2, \ldots k$

   b) $\lambda_l(n) = \dfrac{n}{\sum_{i=1}^{k} \nu_i W_i(n)} \qquad$ (assume $\nu_l = 1$)

   c) $\lambda_i(n) = \lambda_l(n)\nu_i, \qquad i = 1, 2, \ldots k \quad i \neq l$

   Now, once I get this $\lambda_i(n)$, then I can use the second principle that we have used. So, the first principle is used in a) the second principle is used actually in d); to lead to d), we are computing b) and c) the steps.

   d) $L_i(n) = \lambda_i(n) W_i(n) \qquad i = 1, 2, \ldots k$

This is for $n = 1$. Now, that is for that means that you are computing for 1 customer. Now, $n = 2$, which means that you are computing it for $n = 2$ customers with the corresponding $W_i(n)$ and $L_i(n)$. So, this is how you iterate until up to N, which is the objective that you wanted to compute with N customers in the network. So, this is what is the MVA algorithm in a single server node network; this is what we are seeing here. //

Now, we can take up the previous example, but with a slight modification, there we have assumed two machines. If you go, recall the examples that we have considered earlier, which is basically

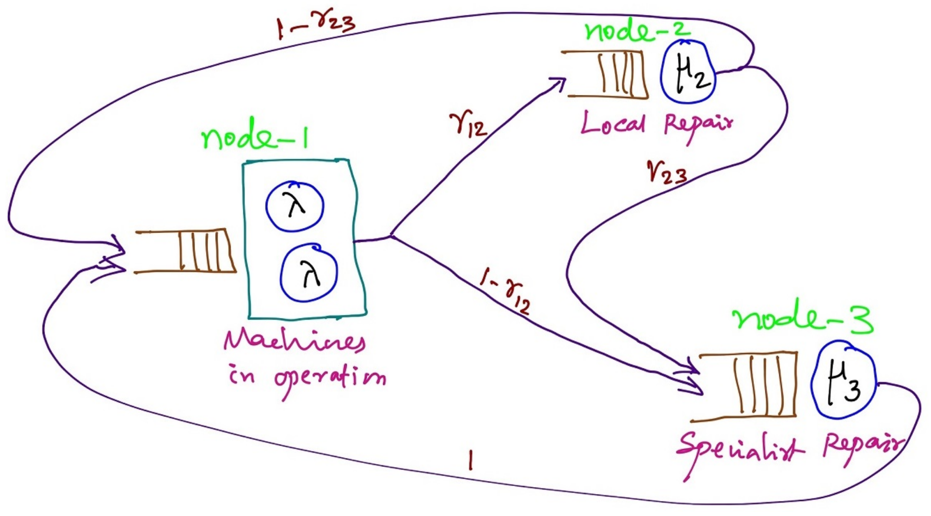

diagram if you can recall. So, now we assume two machines, and then there is local repair and specialist repair. Now, we will assume this one machine; here is the change that we will make. So, this will become a single server at each node network, but the values that we will pick it up are again the same values.

- Assume $\lambda = 2,\ \mu_2 = 1,\ \mu_3 = 3,\ r_{12} = \frac{3}{4},\ r_{23} = \frac{1}{3}$. Then, the joint distribution is

$$p_{n_1,n_2,n_3} = \frac{1}{G(N)}\left(\frac{2}{3}\right)^{n_1}\frac{1}{a_1(n_1)}\left(\frac{2}{9}\right)^{n_3},$$

  where $G(2)$ is computed to be

$$G(2) = \left(\frac{2}{3}\right)^2\frac{1}{2} + 1 + \left(\frac{2}{9}\right)^2 + \frac{2}{3} + \frac{2}{3}\frac{2}{9} + \frac{2}{9} = \frac{187}{81} = 2.3086.$$

$\bar{n}$	(2,0,0)	(0,2,0)	(0,0,2)	(1,1,0)	(1,0,1)	(0,1,1)
$p_{\bar{n}}$	0.0962	0.4332	0.0214	0.2888	0.0642	0.0962

Now, with that understanding, we will take up the same example two-machine three-node network, but now with one machine.

Example. *[One-machine three-node network]*
Consider the previously considered two-machines three-node network, but now with only one machine (to make it as a singer-server-at-all-nodes network).

The traffic equations are $(\nu_1, \nu_2, \nu_3) = (\nu_1, \nu_2, \nu_3)\begin{pmatrix} 0 & 3/4 & 1/4 \\ 2/3 & 0 & 1/3 \\ 1 & 0 & 0 \end{pmatrix}$.

Choosing $\nu_2 = 1$ *(i.e.,* $l = 2$ *here), we get* $\nu_1 = 4/3$ *and* $\nu_3 = 2/3$.

Now, $i = 1, 2, 3$ and $n = N = 1$.

For step-3(a), $W_1(1) = (1 + L_1(0))/\lambda = 1/\lambda = 1/2, W_2(1) = 1/\mu_2 = 1, W_3(1) = 1/\mu_3 = 1/3$.

For step-3(b), $\lambda_2(1) = \frac{1}{\sum_{i=1}^{3}\nu_i W_i(1)} = \frac{9}{17}$.

For step-3(c), $\lambda_1(1) = \nu_1 \lambda_2(1) = 12/17, \;\; \lambda_3(1) = \nu_3 \lambda_2(1) = 6/17$.
For step-3(d), $L_1(1) = \lambda_1(1) W_1(1) = 6/17, L_2(1) = 9/17, L_3(1) = 2/17$.
Since here $N = 1$, we are done.

Verification: Recall that the solution of traffic equations was $\rho_1 = 2/3, \rho_2 = 1, \rho_3 = 2/9$ and $p_{n_1,n_2,n_3} = (1/G(1))(2/3)^{n_1}(2/9)^{n_3}$. This implies that $G(1) = 17/9$ and $p_{100} = 6/17, p_{010} = 9/17, p_{001} = 2/17$ which in turn gives the above values of L_i's and W_i's.

Now, you see from the joint distribution from here; I can obtain these L's; it is simple here; it is not a complex here, but in a more complex situation, obtaining this is not very easy because obtaining this G of this is not easy, but if you wanted to compute the joint distribution at any cost you have no choice, you have to compute it. But, if you are interested only in the mean values, then one can directly apply the MVA algorithm, which might be very simple to apply because you are directly computing the mean values directly here. These are verification that you can do; you can look at them. Now, this algorithm can also be used to get the marginal distributions, but we are not looking at that part. But, what we will just state along a similar line is that what is the MVA algorithm in the multi-server case; this is the more general one that you can take.

Mean-Value Analysis (MVA) Algorithm (Multi-Server Case):

Denote the marginal probability of j at node i in an n-customer system by $p_i(j,n)$.

And, set $\alpha_i(j) = \begin{cases} j, & j \le c_i \\ c_i, & j \ge c_i \end{cases}$.

Objective: To find $L_i(N)$ and $W_i(N)$ in a k-node, c_i-servers-at-node-i network with routing probability matrix $R = (r_{ij})$.

1. Solve $\nu_i = \sum_{j=1}^{k} \nu_j r_{ji}, \quad i = 1, 2, \ldots, k$, setting one of the ν_i's (say, ν_l) equal to 1.

2. Initialize $L_i(0) = 0, p_i(0,0) = 1, p_i(j,0) = 0, j \neq 0, \quad i = 1, 2, \ldots k$

   We are saying $p_i(j,0)$ must be equal to 0 because the total number of customers is 0, and you find j customer in the node, it has to be 0. So, $p_i(j,0)$ is 0 when there is 0 for finding 0 in the system; the probability is 1; that is what you are getting here. These are the initial conditions or the initials starting condition or state.

3. For $n = 1$ to N, calculate

   a) $W_i(n) = \dfrac{1}{c_i \mu_i}\left(1 + L_i(n-1) + \sum_{j=0}^{c_i-2}(c_i - 1 - j)p_i(j, n-1)\right), \qquad i = 1, 2, \ldots k$

   b) $\lambda_l(n) = \dfrac{n}{\sum_{i=1}^{k} \nu_i W_i(n)} \qquad$ (assume $\nu_l = 1$)

   c) $\lambda_i(n) = \lambda_l(n)\nu_i, \qquad i = 1, 2, \ldots k \quad i \neq l$

   d) $L_i(n) = \lambda_i(n) W_i(n), \qquad i = 1, 2, \ldots k$

   e) $p_i(j,n) = \dfrac{\lambda_i(n)}{\alpha_i(j)\mu_i} p_i(j-1, n-1), \quad j = 1, 2, \ldots, n; \;\; i = 1, 2, \ldots, k$

So, that is what is this relationship; this is the usual recurrence relationship in an $M/M/c$ model between the steady-state probabilities of being in state k and state $k+1$. There is nothing here beyond that; it comes from there.

But one can usually show that also this is true in single-server cases and multi-server cases, and so on. But whatever it is, this is the algorithm; now, with this algorithm, you can now take up the original problem that example that we have considered which is basically two machines and three nodes problem closed network example. Now, you can apply this one to get the mean number of W_i's and L_i's from this. So, that is what happened.

So, you can again by computing the joint distribution; also, we have already computed it. Now, you can compute you can do the verification of the result also from the joint distribution; joint distribution is already computed; you do not need to compute it again, but from their marginal distributions and then the corresponding L i's and W i's you can compute and you can match with this MVA algorithm whatever value that you are given you can just do. So, this is all about MVA that we are going to see; we are not going to go beyond this level, but we may end with what can we say about these three processes; the brute force or naive computation of G directly and the convolution algorithm and the MVA algorithm. As you have seen, all three methods are easy for small problems for smaller k smaller n it is always possible that any of these you use you do not have a problem at least for our pen and paper calculation that we do all three methods are actually one and the same.

So, you really did not feel the need for these two things; one can do a naive computation directly with two customers, three customers, three-node, two-node, four-node like you can always do this computation very easily, it is not a problem at all, but in reality that is not the case. So, you have to implement it with a large number of nodes or a large number of customers or both; that is what we call simply as a larger network. In that case, this Buzen's and MVA algorithm or the convolution algorithm and MVA algorithm are computationally far superior over the naive computation with respect to efficiency, which is in terms of storage and speed, and the stability for larger problems, numerical errors, and all other stuff you can put together; for larger problems, these are computationally far superior. But, between these two, if you want to compare, what would be the best or better? Now, for a single server at all node network like the case that we have first considered within this MVA, MVA is superior if we want only the mean waiting times and mean system size at each node; we do not want the state distributions if we want only the mean then MVA is superior.

But if you want there either the joint, marginal distributions or even if you want only the mean, but you have multi server nodes, or if you want marginal probability distributions, it is not clear how much is better; if it is really better, maybe in some situations MVA might be better in some other situation convolution procedure might be better that is depends on the situation and what kind of things that you want and so on.

So, and hence both of these are essential in order to have a complete understanding of the closed queuing network, the Gordon Newell network that we have seen in this case. So, naive a brute force approach is applicable easy in the case of very small networks but larger networks since it is not clear which one will do better in which situation because you can always find a case where this is better than this; this has performed better than the other one in the kind of network that one has analyzed. So, it is not clear since it is not clear that it is which one is better. So, it is always better that you know about both. But a crude way is that if you are interested only in mean and mean values, you can directly go for the mean value algorithm rather than computing the convolution process. The convolution process gives you that $G(N)$.

But, the product form is there. $G(N)$ only you need it. So, that one alone you can compute through convolution process, but if you only want mean directly go for MVA algorithm that is what normally one can one would do. So, we will end our discussion on the closed queueing networks; this type of closed queuing network now, like we will take up a bit more on these ideas in the following lectures.
Thank you bye.