CS 547 Lecture 21: Mean-Value Analysis

Daniel Myers

Mean-value analysis (MVA) is our main tool for solving closed queueing models. It produces exact results for product-form queueing networks and produces reasonably accurate approximations for many other types of networks.

After developing the basic algorithm, we'll talk about approximate versions that relax the strict product-form assumptions.

The Iterative Solution Method

Recall our master equation for $\overline{R}_k(n)$, which we defined to be the residence time at center k when there are n total customers in the system.

$$\overline{R}_k(n) = \overline{s}_k + \overline{A}_k(n)\overline{s}_k$$

By the Arrival Theorem, $\overline{A}_k(n) = \overline{Q}_k(n-1)$.

$$\overline{R}_k(n) = \overline{s}_k + \overline{Q}_k(n-1)\overline{s}_k$$

This may not seem helpful. After all, we still don't know how to compute $\overline{Q}_k(n-1)$. Suppose, however, that we *did* know $\overline{Q}_k(n-1)$ for some value of n and for all the queueing centers in our network, $k = 1 \ldots K$.

We could use the equation to caluclate $\overline{R}_k(n)$ for all $k = 1 \ldots K$. Then, once we know the average residence time at each queueing center, we can use the forced-flow law to determine the total average residence time at *all* queueing centers, taking into account the average number of visits made to each center.

$$\overline{R}(n) = \sum_{k=1}^{K} \overline{V}_k \overline{R}_k(n)$$

Next, we can calculate the throughput, using n, $\overline{R}(n)$, and the average think time, $\overline{Z}$.

$$\Lambda(n) = \frac{n}{\overline{R}(n) + \overline{Z}}$$

Finally, we can use Little's result and the forced-flow law to calculate the expected value of $\overline{Q}_k(n)$ for all $k = 1 \ldots K$.

$$\overline{Q}_k(n) = \Lambda(n)\overline{V}_k\overline{R}_k(n)$$

Therefore, given some starting value of $\overline{Q}_k(n-1)$, we can solve the model for n customers, then derive $\overline{Q}_k(n)$, which allows us to solve the model for $n+1$ customers. All we need is starting value and we can use this iterative method to solve the model for any desired number of customers N.

To find the starting value, think about a closed network with only one customer. Whenever this one customer arrives to a queueing center, it can never find any other customer waiting, because there are no other customers present in the system!

$$\overline{A}_k(1) = \overline{Q}_k(0) = 0$$

This is the starting value that we need.

To solve a closed queueing network with N customers, we simply start with $n = 1$ and apply the iterative method N times. This is the mean value analysis algorithm

MVA Code

Here's a pseudocode implementation of the basic MVA algorithm. It uses arrays `Q` and `R` to keep track of the queue lengths and residence times at each queueing center. It overwrites these values on each loop iteration. A simple modification would keep track of the results for each value of n.

Note that I'm using `X` to represent the throughput rather than Λ.

```
ALGORITHM MVA

input: service time array s,
       visit count array V,
       number of customers N,
       think time Z

output: average residence times R,
        total residence time R_total,
        average queue lengths Q,
        system throughput X

# Intialize queueing centers to empty.
for k = 1 to K
  Q(k) = 0
end

# Loop for 1 to N customers in the system.
for n = 1 to N

  # Calculate residence times using queue lengths
  # for n-1 customers in the system.
  for k = 1 to K
    R(k) = s(k) * (1 + Q(k))
  end

  # Total residence time at all queueing centers,
  # taking into account the average number of visits
  # made to each center.
  R_total = sum(V(k) * R(k)) for all k

  # System throughput, using R_total and think time.
  X = n / (R_total + Z)
```

```
  # Calculate new queue lengths for n customers
  # in the system.
  for k = 1 to K
    Q(k) = X * V(k) * R(k)
  end
end
```

Scherr's Thesis (1965)

Scherr's thesis was the first analytic performance model.

In the early 1960's, MIT developed the Compatible Time-Sharing System (CTSS), a landmark project that finally allowed multiple users to interact with a computer in real time. Prior to the CTSS, computers were treated as "batch" devices. An individual programmer would submit a program – usually in the form of a box of punched cards – and the computing center's staff would load the program onto the department's machine and execute it. The programmer could return to pick up his results the next day. Hopefully they would be correct, as any error required another 24 hour cycle to submit and run the debugged program.

Everyone understood that this wasted a lot of programmer time, but the only alternative was to let an individual monopolize the machine to write and debug a program. This was unacceptable, as most of that time would be spent coding and inputting instructions, rather than actually running programs on the expensive mainframe computer.

The solution to this problem was *time-sharing*. While an individual user might spend most of their time coding and only a little time executing programs, a large group of users could supply enough work to keep a shared machine just as busy as a batch system. This approach created several technical challenges, since it required sharing the processor, memory, and I/O devices among many different programs in a fair and safe way. The techniques developed to solve these problems – context switching, virtual memory, multiplexing – are now a standard part of every computer system.

Allan Scherr was a graduate student at MIT writing his dissertation on the then-new CTSS systems. As part of his project, he collected workload data from the system and compared it to detailed simulation results. He thought this was enough work, but his advisor demanded that Scherr include more mathematics. He decided to apply techniques he had learned in an operations research course to analyze the CTSS.

Physically, the CTSS consisted of an IBM mainframe with a several attached terminals. Users sat at the terminals and submitted requests that would actually be executed on the mainframe.[1] Scherr's model was a simple two-element closed system with a think node and single queueing center. Users alternated between "thinking" at their terminals and submitting requests for the CTSS to process. He modeled the CTSS system as a single-server queue and assumed the service times were exponentially distributed.[2]

Miraculously, Scherr's simple model – using only the average request service time, the think time, and the number of customers, and assuming an exponential distribution – was just as accurate as his detailed simulations. This was the first result showing that abstract analytic models could be a useful tool in performance analysis.

[1] Compare this model of remote execution to our current ideas about cloud computing.

[2] Scherr had to solve his model the old-fashioned way: by designing a solving a complex Markov chain. He couldn't use MVA, because it wasn't known until the early 1980's. He also couldn't use any asymptotic bounds – which would have been ideal for this problem – since they weren't known until the 1970's.