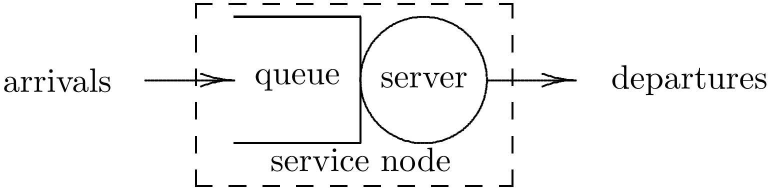
A Single-Server Queue

Section 1.2

Discrete-Event Simulation: A First Course

Section 1.2: A Single-Server Queue



- *Single-sever service node* consists of a server plus its queue
- If only one service technician, the machine shop model from section 1.1 is a single-server queue

Queue Discipline

Queue discipline: the algorithm used when a job is selected from the queue to enter service

- *FIFO* – first in, first out
- *LIFO* – last in, first out
- *SIRO* – serve in random order
- *Priority* – typically shortest job first (SJF)

Assumptions

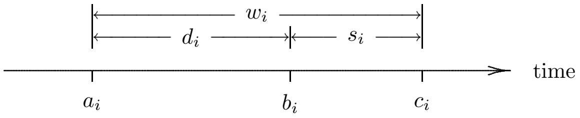
- FIFO is also known as first come, first serve (FCFS)
  - The order of arrival and departure are the same
  - This observation can be used to simplify the simulation
  - Unless otherwise specified, assume FIFO with infinite queue capacity.
- Service is *non-preemptive*
  - Once initiated, service of a job will continue until completion
- Service is *conservative*
  - Server will never remain idle if there is one or more jobs in the service node

  0-13-142917-5

Specification Model

For a job i:

- The *arrival time* is a_i
- The *delay* in the queue is d_i
- The time that service begins is $b_i = a_i + d_i$
- The *service time* is s_i
- The *wait* in the node is $w_i = d_i + s_i$
- The *departure time* is $c_i = a_i + w_i$

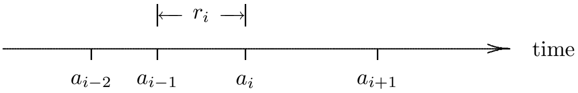


Arrivals

- The *interarrival time* between jobs $i-1$ and i is

$$r_i = a_i - a_{i-1}$$

where, by definition, $a_0 = 0$



- Note that $a_i = a_{i-1} + r_i$ and so (by induction)

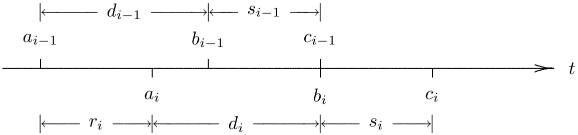
$$a_i = r_1 + r_2 + \ldots + r_i \qquad\qquad i = 1, 2, 3, \ldots$$

Algorithmic Question

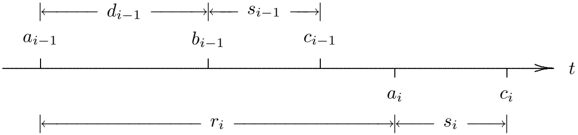
- Given the arrival times and service times, can the delay times be computed?
- For some queue disciplines, this question is difficult to answer
- If the queue discipline is FIFO,
  - d_i is determined by when a_i occurs relative to c_{i-1}.
- There are two cases to consider:

Cases

- If $a_i < c_{i-1}$, job i arrives before job $i-1$ completes:



- If $a_i \geq c_{i-1}$, job i arrives after job $i-1$ completes:



Calculating Delay for Each Job

Algorithm 1.2.1

$c_0 = 0.0;$ `/* assumes that` $a_0 = 0.0$ `*/`
$i = 0;$
`while (` *more jobs to process* `) {`
    i++;
    a_i = `GetArrival();`
    `if (`$a_i < c_{i-1}$`)`
        $d_i = c_{i-1} - a_i;$
    `else`
        $d_i = 0.0;$
    s_i = `GetService();`
    $c_i = a_i + d_i + s_i;$
`}`
$n = i;$
`return` $d_1, d_2, \ldots, d_n;$

Example 1.2.2

- Algorithm 1.2.1 used to process $n = 10$ jobs

	i	1	2	3	4	5	6	7	8	9	10
read from file	a_i	15	47	71	111	123	152	166	226	310	320
from algorithm	d_i	0	11	23	17	35	44	70	41	0	26
read from file	s_i	43	36	34	30	38	40	31	29	36	30

- For future reference, note that for the last job
  - $a_n = 320$
  - $c_n = a_n + d_n + s_n = 320 + 26 + 30 = 376$

  0-13-142917-5

Output Statistics

- The purpose of simulation is insight — gained by looking at statistics
- The importance of various statistics varies on perspective:
  - Job perspective: wait time is most important
  - Manager perspective: utilization is critical
- Statistics are broken down into two categories
  - *Job-averaged* statistics
  - *Time-averaged* statistics

  0-13-142917-5

Job-Averaged Statistics

Job-averaged statistics: computed via typical arithmetic mean

- *Average interarrival time*:

$$\bar{r} = \frac{1}{n} \sum_{i=1}^{n} r_i = \frac{a_n}{n}$$

  - $1/\bar{r}$ is the *arrival rate*

- *Average service time*:

$$\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$$

  - $1/\bar{s}$ is the *service rate*

Example 1.2.3

- For the 10 jobs in Example 1.2.2
  - average interarrival time is

$$\bar{r} = a_n/n = 320/10 = 32.0 \text{ seconds per job}$$

  - average service is $\bar{s} = 34.7$ seconds per job
  - arrival rate is $1/\bar{r} \approx 0.031$ jobs per second
  - service rate is $1/\bar{s} \approx 0.029$ jobs per second
- The server is not quite able to process jobs at the rate they arrive on average.

Job-Averaged Statistics

- The *average delay* and *average wait* are defined as

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i \qquad\qquad \bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i$$

- Recall $w_i = d_i + s_i$ for all i

$$\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i = \frac{1}{n}\sum_{i=1}^{n}(d_i + s_i) = \frac{1}{n}\sum_{i=1}^{n} d_i + \frac{1}{n}\sum_{i=1}^{n} s_i = \bar{d} + \bar{s}$$

  - Sufficient to compute any two of $\bar{w}, \bar{d}, \bar{s}$

Example 1.2.4

- From the data in Example 1.2.2, $\bar{d} = 26.7$

  From Example 1.2.3, $\bar{s} = 34.7$

  Therefore $\bar{w} = 26.7 + 34.7 = 61.4$.

- Recall verification is one (difficult) step of model development

- *Consistency check*: used to verify that a simulation satisfies known equations
  - Compute $\bar{w}$, $\bar{d}$, and $\bar{s}$ *independently*
  - Then verify that $\bar{w} = \bar{d} + \bar{s}$

Time-Averaged Statistics

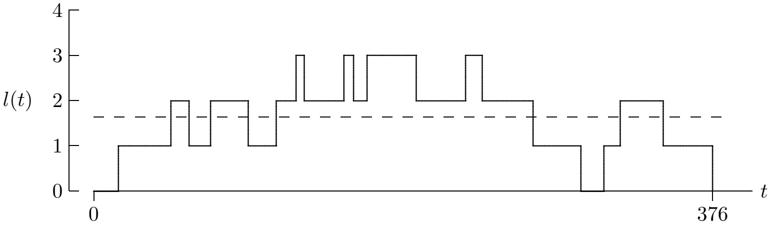
- *Time-averaged statistics*: defined by area under a curve (integration)
- For SSQ, need three additional functions
  - $l(t)$: number of jobs in the service node at time t
  - $q(t)$: number of jobs in the queue at time t
  - $x(t)$: number of jobs in service at time t
- By definition, $l(t) = q(t) + x(t)$.
- $l(t) = 0, 1, 2, \ldots$

  $q(t) = 0, 1, 2, \ldots$

  $x(t) = 0, 1$

Time-Averaged Statistics

- All three functions are *piece-wise constant*



  - Figures for $q(\cdot)$ and $x(\cdot)$ can be deduced

$$q(t) = 0 \text{ and } x(t) = 0 \text{ if and only if } l(t) = 0$$

Time-Averaged Statistics

- Over the time interval $(0, \tau)$:

$$\text{time-averaged number in the node:} \quad \bar{l} = \frac{1}{\tau}\int_0^\tau l(t)dt$$

$$\text{time-averaged number in the queue:} \quad \bar{q} = \frac{1}{\tau}\int_0^\tau q(t)dt$$

$$\text{time-averaged number in service:} \quad \bar{x} = \frac{1}{\tau}\int_0^\tau x(t)dt$$

- Since $l(t) = q(t) + x(t)$ for all $t > 0$

$$\bar{l} = \bar{q} + \bar{x}$$

  - Sufficient to calculate any two of $\bar{l}, \bar{q}, \bar{x}$

Example 1.2.5

- From Example 1.2.2 (with $\tau = c_{10} = 376$),

$$\bar{l} = 1.633 \qquad \bar{q} = 0.710 \qquad \bar{x} = 0.923$$

- The average of numerous *random* observations (samples) of the number in the service node should be close to $\bar{l}$.
  - Same holds for $\bar{q}$ and $\bar{x}$
- *Server utilization*: time-averaged number in service ($\bar{x}$)
  - $\bar{x}$ also represents the probability the server is busy

Little's Theorem

How are job-averaged and time-average statistics related?

Theorem (Little, 1961)

If (a) queue discipline is FIFO,
(b) service node capacity is infinite, and
(c) server is idle both at $t = 0$ and $t = c_n$
then

$$\int_0^{c_n} l(t)dt = \sum_{i=1}^n w_i \qquad \text{and}$$

$$\int_0^{c_n} q(t)dt = \sum_{i=1}^n d_i \qquad \text{and}$$

$$\int_0^{c_n} x(t)dt = \sum_{i=1}^n s_i$$

Little's Theorem Proof

Proof.

For each job $i = 1, 2, \ldots,$ define an *indicator function*

$$\psi_i(t) = \begin{cases} 1 & a_i < t < c_i \\ 0 & \text{otherwise} \end{cases}$$

Then

$$l(t) = \sum_{i=1}^{n} \psi_i(t) \qquad\qquad 0 < t < c_n$$

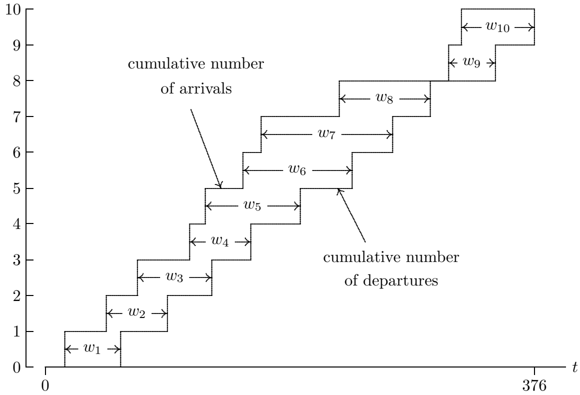
and so

$$\int_0^{c_n} l(t)dt = \int_0^{c_n} \sum_{i=1}^{n} \psi_i(t)dt = \sum_{i=1}^{n} \int_0^{c_n} \psi_i(t)dt = \sum_{i=1}^{n}(c_i - a_i) = \sum_{i=1}^{n} w_i$$

□

The other two equations can be derived similarly.

Example 1.2.6



$$\int_0^{376} l(t)dt = \sum_{i=1}^{10} w_i = 614$$

Little's Equations

- Using $\tau = c_n$ in the definition of the time-averaged statistics, along with Little's Theorem, we have

$$c_n\bar{l} = \int_0^{c_n} l(t)dt = \sum_{i=1}^{n} w_i = n\bar{w}$$

- We can perform similar operations and ultimately have

$$\bar{l} = \left(\frac{n}{c_n}\right)\bar{w} \quad \text{and} \quad \bar{q} = \left(\frac{n}{c_n}\right)\bar{d} \quad \text{and} \quad \bar{x} = \left(\frac{n}{c_n}\right)\bar{s}$$

Computational Model

- The ANSI C program `ssq1` implements Algorithm 1.2.1
- Data is read from the file `ssq1.dat` consisting of arrival times and service times in the format

$$\begin{array}{cc} a_1 & s_1 \\ a_2 & s_2 \\ \vdots & \vdots \\ a_n & s_n \end{array}$$

- Since queue discipline is FIFO, no need for a queue data structure

Example 1.2.8

- Running program `ssq1` with `ssq1.dat`

$$1/\bar{r} \approx 0.10 \quad \text{and} \quad 1/\bar{s} \approx 0.14$$

- If you modify program `ssq1` to compute $\bar{l}, \bar{q}$, and $\bar{x}$

$$\bar{x} \approx 0.28$$

- Despite the significant idle time, $\bar{q}$ is nearly 2.

Traffic Intensity

- *Traffic intensity*: ratio of arrival rate to service rate

$$\frac{1/\bar{r}}{1/\bar{s}} = \frac{\bar{s}}{\bar{r}} = \frac{\bar{s}}{a_n/n} = \left(\frac{c_n}{a_n}\right)\bar{x}$$

- Assuming c_n/a_n is close to 1.0, the traffic intensity and utilization will be nearly equal

Case Study

Sven and Larry's Ice Cream Shoppe

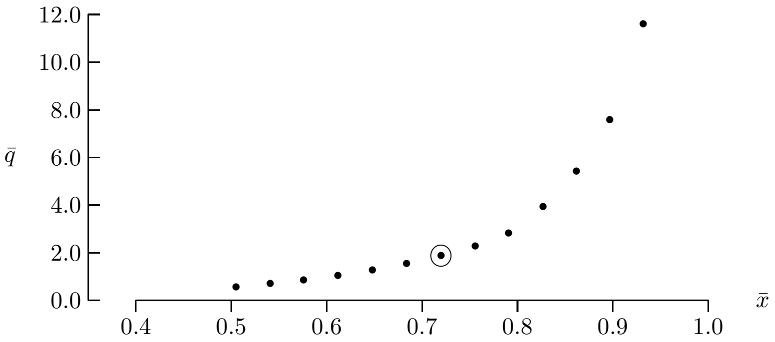
- owners considering adding new flavors and cone options
- concerned about resulting service times and queue length

Can be modeled as a single-sever queue

- `ssq1.dat` represents 1000 customer interactions
- Multiply each service time by a constant
    - In the following graph, the circled point uses unmodified data
    - Moving right, constants are 1.05, 1.10, 1.15, ...
    - Moving left, constants are 0.95, 0.90, 0.85, ...

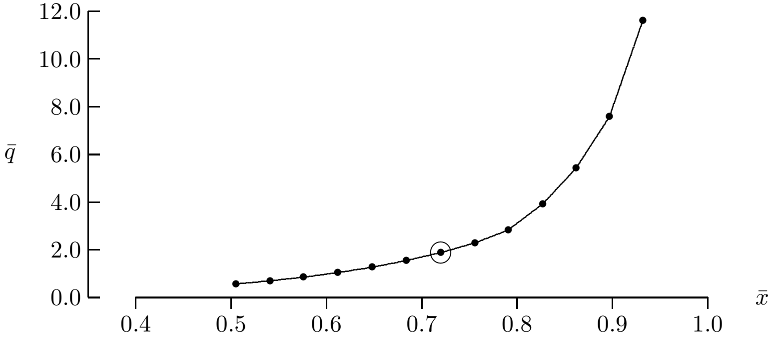
  0-13-142917-5

Sven and Larry



- Modest increase in service time produces significant increase in queue length
  - Non-linear relationship between $\overline{q}$ and $\overline{x}$
- Sven and Larry will have to assess the impact of the increased service times

Graphical Considerations



- Since both $\overline{x}$ and $\overline{q}$ are continuous, we could calculate an "infinite" number of points
- Few would question the validity of "connecting the dots"

Guidelines

- If there is essentially no uncertainty and the resulting interpolating curve is smooth, connecting the dots is OK
  - Leave the dots as a reminder of the data points
- If there is essentially no uncertainty but the curve is not smooth, more dots should be generated
- If the dots correspond to uncertain (noisy) data, then interpolation is not justified
  - Use approximation of a curve or do not superimpose at all
- Discrete data should *never* have a solid curve

