APPENDIX

An Elevator Simulation Algorithm

We will define the terms used in the following algorithm and explain some of its underlying logic. Because the algorithm is complex, this approach should be more revealing than using some hypothetical numbers and taking you step by step through the algorithm. (This is a difficult program to write if you are not using GPSS or another simulation language.)

During the simulation there is a TIME clock that keeps track of the time (given in seconds). Initially, the value of TIME is 0 sec (at 7:50 a.m.), and the simulation ends when TIME reaches 4800 sec (at 9:10 a.m.). Each customer is assigned a number according to the order of his or her arrival: The first customer is labeled 1, the second customer 2, and so forth. Whenever another customer arrives at the lobby, the time between the customer's arrival and the time when the immediately preceding customer arrived is added to the TIME clock. This time between successive arrivals of customers i and $i-1$ is labeled $between_i$ in the algorithm, and the arrival time of customer i is labeled $arrive_i$. Initially, for the customer arrival submodel, we assume that all values between 0 and 30 have an equal likelihood of occurring.

All four elevators have their own availability times, called $return_j$ for elevator j. If elevator j is currently available at the main floor, its time is the current time, so $\text{return}_j =$ TIME. If an elevator is in transit, its availability time is the time at which it will return to the main floor. Passengers enter an available elevator in the numerical order of the elevators: first elevator 1 (if it is available), next elevator 2 (if it is available), and so on. Maximum occupancy of an elevator is 12 passengers.

Whenever another customer arrives in the lobby of the building, two possible situations exist. Either an elevator is available for receiving passengers or no elevator is available and a queue is forming as customers wait for one to become available. Once an elevator becomes available, it is ''tagged'' for loading, and passengers can enter *only* that elevator until either it is fully occupied (with 12 passengers) or the 15-sec time delay is exceeded before the arrival of another customer. After loading, the elevator departs to deliver all of its passengers. It is assumed that, even if fully loaded with 12 passengers, the elevator waits 15 sec to load the last passenger, allow floor selection, and get under way.

To keep track of which floors have been selected during the loading period of an elevator and the number of times a particular floor has been selected, the algorithm sets up two one-dimensional arrays (having a component for each of the floors 1–12). (Although no one selects floor 1, the indexing is simplified with its inclusion.) These arrays are called $selvec_j$ and $flrvec_j$ for the tagged elevator j. If a customer selects floor 5, for instance, then a 1 is entered into the fifth component position of selvec_j and also into the

fifth component of flrvec_j. If another customer selects floor 5, then the fifth component of flrvec_j is updated to a 2, and so forth. For example, suppose the passengers in elevator j have selected floor 3 twice, floor 5 twice, and floors 7, 8, and 12 once. Then for this elevator, $\mathrm{selvec}_j = (0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1)$ and $\mathrm{flrvec}_j = (0, 0, 2, 0, 2, 0, 1, 1, 0, 0, 0, 1)$. These arrays are then used to calculate the transport time of elevator j so we can determine when it will return to the main floor. As stated previously, that return time is designated $return_j$. The arrays are also used to calculate the delivery times of each passenger in elevator j. Initially, we assume that a customer chooses a floor with equal likelihood. We assume that it takes 10 sec for an elevator to travel between floors, 10 sec to open *and* close its doors, and 3 sec for each passenger to disembark. We also assume that it takes 3 sec for each passenger in a queue to enter the next available elevator.

Summary of Elevator Simulation Algorithm Terms

$between_i$	time between successive arrivals of customers i and $i-1$ (a random integer varying between 0 and 30 sec)
$arrive_i$	time of arrival from start of clock at $t = 0$ for customer i (calculated only if customer enters a queue waiting for an elevator)
$floor_i$	floor selected by customer i (a random integer varying between 2 and 12)
$elevator_i$	time customer i spends in an elevator
$wait_i$	time customer i waits before stepping into an elevator (calculated only if customer enters a queue waiting for an elevator)
$delivery_i$	time required to deliver customer i to destination floor from time of arrival, including any waiting time
$selvec_j$	binary 0, 1 one-dimensional array representing the floors selected for elevator j, not counting the number of times a particular floor has been selected
$flrvec_j$	integer one-dimensional array representing the number of times each floor has been selected for elevator j for the group of passengers currently being transported to their respective floors
$occup_j$	number of current occupants of elevator j
$return_j$	time from start of clock at $t = 0$ that elevator j returns to the main floor and is available for receiving passengers
$first_j$	an index, the customer number of the first passenger who enters elevator j after it returns to the main floor
$quecust$	customer number of the first person waiting in the queue
$queue$	total length of current queue of customers waiting for an elevator to become available
$startque$	clock time at which the (possibly updated) current queue commences to form
$stop_j$	total number of stops made by elevator j during the entire simulation
$eldel_j$	total time elevator j spends in delivering its current load of passengers
$operate_j$	total time elevator j operates during the entire simulation
$limit$	customer number of the last person to enter an available elevator before it commences transport

max	largest index of a nonzero entry in the array $selvec_j$ (highest floor selected)
remain	number of customers left in the queue after loading next available elevator
quetotal	total number of customers who spent time waiting
TIME	current clock time in seconds, starting at $t = 0$
DELTIME	average delivery time of a customer to reach destination floor from time of arrival, including any waiting time
ELEVTIME	average time a person spends in an elevator
MAXDEL	maximum time required for a customer to reach his or her floor of destination from time of arrival
MAXELEV	maximum time a customer spends in an elevator
QUELEN	number of customers waiting in the longest queue
QUETIME	average time a customer who must wait spends in a queue
MAXQUE	longest time a customer spends in a queue

Elevator System Simulation Algorithm

Input None required.

Output Number of passengers serviced, DELTIME, ELEVTIME, MAXDEL, MAXELEV, QUELEN, QUETIME, MAXQUE, stop_j, and the percentage time each elevator is in use.

STEP 1 Initially set the following parameters to zero: DELTIME, ELEVTIME, MAXDEL, MAXELEV, QUELEN, QUETIME, MAXQUE, quetotal, remain.

STEP 2 For the first customer, generate time between successive arrivals and floor destination and initialize delivery time:

$$i = 1$$

Generate between_i and floor_i

$$\text{delivery}_i = 15$$

STEP 3 Initialize clock time, elevator available clock times, elevator stops, and elevator operating times. Also, initialize all customer waiting times.

$$\text{TIME} = \text{between}_i$$

$$\text{For } k = 1\text{–}4 : \text{return}_k = \text{TIME} \quad \text{and} \quad \text{stop}_k = \text{operate}_k = 0$$

$$\text{For } k = 1\text{–}400 : \text{wait}_k = 0$$

(The number 400 is an upper-bound guess for the total number of customers)

STEP 4 While TIME $\leq$ 4800, do Steps 5–32.

STEP 5 Select the first available elevator:

If TIME $\geq \text{return}_1$, then $j = 1$ else
If TIME $\geq \text{return}_2$, then $j = 2$ else
If TIME $\geq \text{return}_3$, then $j = 3$ else
If TIME $\geq \text{return}_4$, then $j = 4$

ELSE (no elevator is currently available) GOTO Step 19.

STEP 6 Set as an index the customer number of the first person to occupy tagged elevator, and initialize the elevator occupancy floor selection vectors:

$$\text{first}_j = i, \qquad \text{occup}_j = 0$$
For $k = 1$–12: $\text{selvec}_j[k] = \text{flrvec}_j[k] = 0$

STEP 7 Load current customer on elevator j by setting the floor selection vectors and incrementing elevator occupancy:

$$\text{selvec}_j[\text{floor}_i] = 1$$
$$\text{flrvec}_j[\text{floor}_i] = \text{flrvec}_j[\text{floor}_i] + 1$$
$$\text{occup}_j = \text{occup}_j + 1$$

STEP 8 Get next customer and update clock time:

$$i = i + 1$$
Generate between_i and floor_i
$$\text{TIME} = \text{TIME} + \text{between}_i$$
$$\text{delivery}_i = 15$$

STEP 9 Set all available elevators to current clock time:

For $k = 1$–4:
If $\text{TIME} \geq \text{return}_k$, then $\text{return}_k = \text{TIME}$.
Else leave return_k as is.

STEP 10 If $\text{between}_i \leq 15$ and $\text{occup}_j < 12$, then increase the delivery times for each customer on the tagged elevator j:

For $k = \text{first}_j$ to $i - 1$:
$$\text{delivery}_k = \text{delivery}_k + \text{between}_i$$
and GOTO Step 7 to load current customer on the elevator and get the next customer.
Else (send off the tagged elevator):
Set limit $= i - 1$ and GOTO Step 11.

The sequence of Steps 11–18 implements delivery of all passengers on the currently tagged elevator j:

STEP 11 For $k = \text{first}_j$ to limit, do Steps 12–16.

STEP 12 Calculate time customer k spends in elevator:

$$N = \text{floor}_k - 1 \text{ (an index)}$$
$$\text{elevator}_k = \text{travel time up to floor} + \text{time to drop off previous customers} + \text{customer } k \text{ drop off time} + \text{open/close door times on previous floors} + \text{open door on current floor}$$
$$= 10N + 3\sum_{m=1}^{N} \text{flrvec}_j[m] + 3 + 10\sum_{m=1}^{N} \text{selvec}_j[m] + 5$$

STEP 13 Calculate delivery time for customer k:
$$\text{delivery}_k = \text{delivery}_k + \text{elevator}_k$$

STEP 14 Sum to total delivery time for averaging:
$$\text{DELTIME} = \text{DELTIME} + \text{delivery}_k$$

STEP 15 If $\text{delivery}_k > \text{MAXDEL}$, then $\text{MAXDEL} = \text{delivery}_k$.
Else leave MAXDEL as is.

STEP 16 If $\text{elevator}_k > \text{MAXELEV}$, then $\text{MAXELEV} = \text{elevator}_k$.
Else leave MAXELEV as is.

STEP 17 Calculate total number of stops for elevator j, its time in transit, and the time at which it returns to the main floor:

$$\text{stop}_j = \text{stop}_j + \sum_{m=1}^{12} \text{selvec}_j[m]$$

$$\text{Max} = \text{index of largest nonzero entry in selvec}_j \text{ (i.e., the highest floor visited)}$$

$$\text{eldel}_j = \text{travel time} + \text{passenger drop-off time} + \text{door time}$$

$$= 20(\text{Max} - 1) + 3\sum_{m=1}^{12} \text{flrvec}_j[m] + 10\sum_{m=1}^{12} \text{selvec}_j[m]$$

$$\text{return}_j = \text{TIME} + \text{eldel}_j$$

$$\text{operate}_j = \text{operate}_j + \text{eldel}_j$$

STEP 18 GOTO Step 5.

The sequence of Steps 19–32 is taken when no elevator is currently available and a queue of customers waiting for elevator service is set up.

STEP 19 Initialize queue:

$$\text{quecust} = i \text{ (number for first customer in queue)}$$

$$\text{startque} = \text{TIME (starting time of queue)}$$

$$\text{queue} = 1$$

$$\text{arrive}_i = \text{TIME}$$

STEP 20 Get the next customer and update clock time:

$$i = i + 1$$

Generate between_i and floor_i

$$\text{TIME} = \text{TIME} + \text{between}_i$$

$$\text{arrive}_i = \text{TIME}$$

$$\text{queue} = \text{queue} + 1$$

STEP 21 Check for elevator availability:

If TIME $\geq \text{return}_1$, then $j = 1$ and GOTO Step 22 else
If TIME $\geq \text{return}_2$, then $j = 2$ and GOTO Step 22 else
If TIME $\geq \text{return}_3$, then $j = 3$ and GOTO Step 22 else
If TIME $\geq \text{return}_4$, then $j = 4$ and GOTO Step 22
ELSE (no elevator is available yet) GOTO Step 20.

STEP 22 Elevator j is available. Initialize floor selection vectors and assess the length of the queue:

For $k = 1$–12: $\text{selvec}_j[k] = \text{flrvec}_j[k] = 0$

$$\text{remain} = \text{queue} - 12$$

STEP 23 If remain ≤ 0, then $R = i$ and $\text{occup}_j = \text{queue}$.
Else $R = \text{quecust} + 11$ and $\text{occup}_j = 12$.

STEP 24 Load customers onto elevator j:

For $k = \text{quecust}$ to R:

$$\text{selvec}_j[\text{floor}_k] = 1 \quad \text{and} \quad \text{flrvec}_j[\text{floor}_k] = \text{flrvec}_j[\text{floor}_k] + 1$$

STEP 25 If queue $\geq$ QUELEN, then QUELEN = queue.
Else leave QUELEN as is.

STEP 26 Update queuing totals:

$$\text{quetotal} = \text{quetotal} + \text{occup}_j$$

$$\text{QUETIME} = \text{QUETIME} + \sum_{m=\text{quecust}}^{R} [\text{TIME} - \text{arrive}_m]$$

STEP 27 If (TIME − startque) ≥ MAXQUE, then MAXQUE = TIME − startque.
Else leave MAXQUE as is.

STEP 28 Set index giving number of first customer to occupy tagged elevator:
$\text{first}_j = \text{quecust}$

STEP 29 Calculate delivery and waiting times for each passenger on the tagged elevator:

For $k = \text{first}_j$ to R:
$\text{delivery}_k = 15 + (\text{TIME} - \text{arrive}_k)$
$\text{wait}_k = \text{TIME} - \text{arrive}_k$

STEP 30 If remain ≤ 0, then set queue = 0 and GOTO Step 8 to get next customer.
Else set limit = R and, for $k = \text{first}_j$ to limit, do Steps 12–17. When finished, GOTO Step 31.

STEP 31 Update queue length and check for elevator availability:

queue = remain
quecust = $R + 1$
$\text{startque} = \text{arrive}_{R+1}$

STEP 32 GOTO Step 20.

The sequence of Steps 33–36 calculates output values for the morning rush-hour elevator simulation.

STEP 33 Output the following values:

$N = i -$ queue, the total number of customers served
DELTIME = DELTIME/N, average delivery time
MAXDEL, maximum delivery time of a customer

STEP 34 Output the average time spent in an elevator and the maximum time spent in an elevator:

$$\text{ELEVTIME} = \sum_{m=1}^{\text{limit}} \frac{\text{elevator}[m]}{\text{limit}} \quad \text{and MAXELEV}$$

STEP 35 Output the number of customers waiting in the longest queue, the average time a customer who waits in line spends in a queue, and the longest time spent in a queue:

$$\text{QUELEN}$$
$$\text{QUETIME} = \frac{\text{QUETIME}}{\text{quetotal}}$$
$$\text{MAXQUE}$$

STEP 36 Output the total number of stops for each elevator and the percentage time each elevator is in transport:

For $k = 1$–4: display stop_k and $\dfrac{\text{operate}_k}{4800}$

STOP

Note For ease of presentation in the elevator simulation, TIME is updated only as the next customer arrives in the lobby. Therefore, TIME is not an actual clock being updated every second. It is possible, when a queue has formed, that an elevator returns to the main floor during Step 20, before the next customer arrives. However, loading of the available elevator does not commence until that customer actually arrives. For this reason, the times spent waiting in a queue are slightly on the high side. In Problem 2 you are asked to modify Steps 20–32 in the algorithm so that loading commences immediately upon the return of the first available elevator.

Table B.1 Results of elevator simulation for 15 consecutive days

Simulation number	Numbers of customers serviced	Average delivery time	Maximum delivery time	Average time in elevator	Maximum time in elevator	Number of customers in longest queue	Average time in a queue	Longest time in a queue	Total number of stops for each elevator				Percentage of total time each elevator is in transport			
									1	2	3	4	1	2	3	4
1	328	147	412	89	208	12	40	166	67	76	56	52	84	87	80	75
2	322	146	409	88	211	18	43	176	74	62	52	61	88	80	78	77
3	309	139	385	87	201	12	37	161	62	61	61	62	85	83	80	80
4	331	149	371	89	205	13	42	149	72	69	68	44	85	82	87	73
5	320	146	404	87	208	15	48	178	72	52	58	58	86	78	80	74
6	313	153	405	91	211	13	45	146	69	62	72	47	85	82	82	74
7	328	138	341	88	195	10	35	120	59	66	70	61	86	81	84	82
8	312	147	377	86	198	12	46	163	69	60	61	43	82	82	81	73
9	329	139	352	87	208	11	37	155	58	63	70	57	86	86	82	78
10	314	143	325	88	205	9	35	128	65	68	57	65	83	84	78	83
11	317	137	344	85	202	10	38	129	64	75	64	56	86	85	81	77
12	341	153	396	90	211	18	45	177	83	63	63	53	87	82	78	77
13	318	136	345	80	208	11	36	140	58	64	63	55	91	82	83	73
14	319	140	356	88	208	13	33	135	64	67	58	65	84	83	82	81
15	323	147	386	91	218	15	39	166	76	64	70	54	86	81	87	76
Averages (rounded)	322	144	374	88	206	13	40	153	67	65	63	56	86	83	82	77

Note: All times are measured in seconds and rounded to the nearest second.

Table B.1 gives the results of 15 independent simulations, representing 3 weeks of morning rush hour, according to the preceding algorithm.

B.1 PROBLEMS

1. Consider an intersection of two one-way streets controlled by a traffic light. Assume that between 5 and 15 cars (varying probabilistically) arrive at the intersection every 10 sec in direction 1, and that between 6 and 24 cars arrive every 10 sec going in direction 2. Suppose that 36 cars per 10 sec can cross the intersection in direction 1 and that 20 cars per 10 sec can cross the intersection in direction 2 if the traffic light is green. No turning is allowed. Initially, assume that the traffic light is green for 30 sec and red for 70 sec in

direction 1. Write a simulation algorithm to answer the following questions for a 60-min time period:

a. How many cars pass through the intersection in direction 1 during the hour?

b. What is the average waiting time of a car stopped when the traffic signal is red in direction 1? The maximum waiting time?

c. What is the average length of the queue of cars stopped for a red light in direction 1? The maximum length?

d. What is the average number of cars passing through the intersection in direction 1 during the time when the traffic light is green? What is the maximum number?

e. Answer Problems (a)–(d) for direction 2.

How would you use your simulation to determine the switching period for which the total waiting time in both directions is as small as possible? (You will have to modify it to account for the waiting times in direction 2.)

2. Modify Steps 20–32 in the elevator simulation algorithm so that loading of the first available elevator commences immediately upon its return. Thus, if TIME $>$ return$_j$ so that elevator j is available for loading, then loading commences at time return$_j$ rather than TIME. Consider how you will now process customer i in Step 20 who has not yet quite arrived on the scene.

B.1 PROJECT

1. Find a building in your local area that has from 4 to 12 floors that are serviced by 1–4 elevators. Collect data for the interarrival times (and, possibly, floor destinations) of the customers during a busy hour (e.g., the morning rush hour), and build the interarrival and destination submodels based on your data (by constructing the cumulative histograms). Write a computer program incorporating your submodels into the elevator system algorithm to obtain results such as those given in Table B.1.