General Concepts and Algorithm of Discrete-Event Simulation

IEEM 313, Spring 2005
L. Jeff Hong
Dept of IEEM, HKUST
The case of Bit Bucket Computers

Bit Bucket Computers specializes in installing and maintaining highly reliable computer systems. One of its standard configurations is to install a primary computer, an identical backup computer that is idle until needed, and provide a service contract that guarantees complete repair of a failed computer within 48 hours (if it has not fixed a computer within 48 hours, then it replaces the computer).
The case of Bit Bucket Computers

Computers systems are rated in terms of their “time to failure” (TTF). In our case, the failure represents that both computers are down. The company wants to determine the TTF rating for the computer system.

The system is supposed to be failure free for at least 2 years. Therefore, running physical tests is not feasible.

What can we do?
The case of Bit Bucket Computers

• We can create a model for the computer system, and then simulate the failures and find the corresponding TTFs.

• Logic:
  – Use one computer and use the other as backup
  – Replace the backup computer once the computer in use fails
  – Repair the failed computer and set it as a backup

• Inputs:
  – 
  – 
  –
System state

• System state is the collection of variables necessary to describe the system at any time.
  – In the case:

• To fully characterize the sample path, we need to know \( Y_t \), the state at any time \( t \geq 0 \)
System state

• Let \( \{S_n; n=0,1,2,\ldots\} \) represent the sequence of state changes.
• Let \( \{T_n; n=0,1,2,\ldots\} \) represent the times of the state changes (“event epochs”)
• For the plot of last slide
  
  
• Therefore, we have
Discrete/continuous system

• For all systems we studied so far, system state changes only at a discrete set of points in time. The systems are called discrete-event systems.

• If the state of a system can change continuously, then the system is a continuous system, e.g., the oil level in oil tank.

• The simulation of discrete system is called discrete-event system simulation.
Event

- An event is defined as an instantaneous occurrence that may change the state of the system.
- In our example, we have three events (that can change the system state)
The case of Bit Bucket Computers

• Simulation
  – Generate a sequence of TTF of a computer from the Weibull distribution:
    877, 1041, 612, 36, 975…
  – Generate a sequence of repair time of a computer from the uniform distribution:
    17, 8, 39, 9…
The case of Bit Bucket Computers

<table>
<thead>
<tr>
<th>Event Counter</th>
<th>Event Time</th>
<th>State</th>
<th>Failure Clock</th>
<th>Repair Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>I</td>
<td>Tn</td>
<td>Sn</td>
<td>C1</td>
</tr>
</tbody>
</table>
Simulation algorithm

• Step 1: build event functions for all events

• Step 2: create event scheduling (control structure) to select the next event

• Step 3: add a stopping rule to get the simulation stopped

• Step 4: replicate the simulation for many times and calculate performance measures
The case of Bit Bucket Computers

- Event functions

\[ e_0( ) \quad \text{% Installation} \]

1. \[ S_0 \leftarrow 0 \quad \text{% initially no computers down} \]

2. \[ C_1 \leftarrow \text{Weibull r.v.} \quad \text{% set clock for first computer TTF} \]
   \[ C_2 \leftarrow \infty \quad \text{% indicate no pending repair} \]
The case of Bit Bucket Computers

• Event functions

\[ e_1() \]

1. \[ S_{n+1} \leftarrow S_n + 1 \] % one more computer down

2. If \( \{S_{n+1}=1\} \), then % if one up and one down
   \[ C_1 \leftarrow T_{n+1} + \text{Weibull r.v.} \] % set clock for next computer TTF
   \[ C_2 \leftarrow T_{n+1} + \text{uniform r.v.} \] % set clock for end of repair
End If

% computer fails
The case of Bit Bucket Computers

• Questions on $e_1( )$
  – Why don’t we consider $S_{n+1}=0$ (no computer down)?

  – Why don’t we consider $S_{n+1}=2$ (both computers down)?
The case of Bit Bucket Computers

- Event functions

\[ e2( ) \] % end of repair

1. \( S_{n+1} \leftarrow S_n - 1 \) % one fewer computer down

2. If \( \{S_{n+1}=1\} \), then % if one up and one down
   \[ C_1 \leftarrow T_{n+1} + \text{Weibull r.v.} \] % set clock for next computer TTF
   \[ C_2 \leftarrow T_{n+1} + \text{uniform r.v.} \] % set clock for end of repair
   End If
The case of Bit Bucket Computers

• Questions on $e_2( )$
  – Why don’t we consider $S_{n+1}=0$ (no computer down)?
  
  – Why don’t we consider $S_{n+1}=2$ (both computers down)?
The case of Bit Bucket Computers

- Event scheduling

Algorithm Bit_Bucket_simulation

1. \( n \leftarrow 0 \) \hspace{1cm} \% initialize system-event counter
   \( T_0 \leftarrow 0 \) \hspace{1cm} \% initialize event epoch
   \( e_0() \) \hspace{1cm} \% execute initial system event
2. \( T_{n+1} \leftarrow \min\{C_1,C_2\} \) \hspace{1cm} \% advance time to next pending system event
   \( I \leftarrow \arg\min\{C_1,C_2\} \) \hspace{1cm} \% find index of next system event
3. \( C_I \leftarrow \infty \) \hspace{1cm} \% event \( I \) no longer pending
4. \( e_I() \) \hspace{1cm} \% execute system event \( I \)
   \( n \leftarrow n + 1 \) \hspace{1cm} \% update event counter
5. Repeat 2
The case of Bit Bucket Computers

• Stopping rule

4.5 If \( \{S_n=2\} \) then

\[ D \leftarrow T_n \]

\% both computers down

\% record time of system failure

stop

\% terminate simulation

endif

• Replicate simulation

for \( r = 1 \) to \( m \)

do

algorithm Bit_Bucket_simulation

enddo
Algorithm for discrete-event simulation

1. $n \leftarrow 0$ % initialize system-event counter
   $T_0 \leftarrow 0$ % initialize event epoch
   $e_0()$ % execute initial system event

2. $T_{n+1} \leftarrow \min\{C_i\}$ % advance time to next pending system event
   $I \leftarrow \arg\min\{C_i\}$ % find index of next system event

3. $S_{0+1} \leftarrow S_0$ % temporarily maintain previous system state
   $C_I \leftarrow \infty$ % event I no longer pending

4. $e_I()$ % execute system event I
   $n \leftarrow n+1$ % update event counter

5. Repeat 2
The case of Bit Bucket Computers

• Calculating performance measures
  – TTF for the system: \( D = \min \{ t: Y_t = 2 \} \)
    • We simulated system 500 times (equivalent to installing 500 systems), we get \( D_1, D_2, \ldots, D_{500} \)
    • \( \overline{D} = 63 \) years!
  • But the user may also be interested in what is the probability that the system will fail within 2 years. How to find it?
The case of Bit Bucket Computers

• Calculating performance measures
  – What if we are interested in the average number of failed computers in a sample path? (more from the company’s point of view)
Simulation Software

• The logical approach we studied here (system events and control structure) is embedded in every simulation language, e.g., Arena, AutoMod, ProModel, SIMUL8, Witness

• In some case, like in financial industry, we may need to write simulation code using C++, Java or VBA. Then we can use this approach directly.
Bus Stop Case (An in-class exercise)

• Students arrive randomly to a bus stop. The interarrival-time distribution is exponential with average 17 seconds. The capacity of bus is 60 (take at most 60 passengers).

• Question: How often should a bus arrive so that the average waiting time of the customers is small?

• Note: More frequent bus arrivals will mean we need more buses, and buses are expensive.

• How to build a simulation model to study this?
The Case of Bus Stop

• Inputs:

• Logic:

• \{S_n; n=0,1,2,\ldots\}:

• \{T_n; n=0,1,2,\ldots\}:

• System events:
  –
  –
  –
The Case of Bus Stop

• If bus arrives at the bus stop every 15 minutes..
• We use Exprnd(17/60) denote the random interarrival time with mean $\frac{17}{60}$ minutes
• Event functions:
  – $e_0(\ )$
The Case of Bus Stop

- Event functions:
  - e1( )
  - e2( )
The Case of Bus Stop

• Control structure and data collection
The Case of Bus Stop

• How to estimate the average waiting time?