Advanced Practical 2022/2023
Operations Research Case
Lecture: Discrete Event Simulation

Guanlian Xiao

Department Operations Analytics
Vrije Universiteit Amsterdam

June 2023

Discrete-Event Simulation (DES)

What is DES?

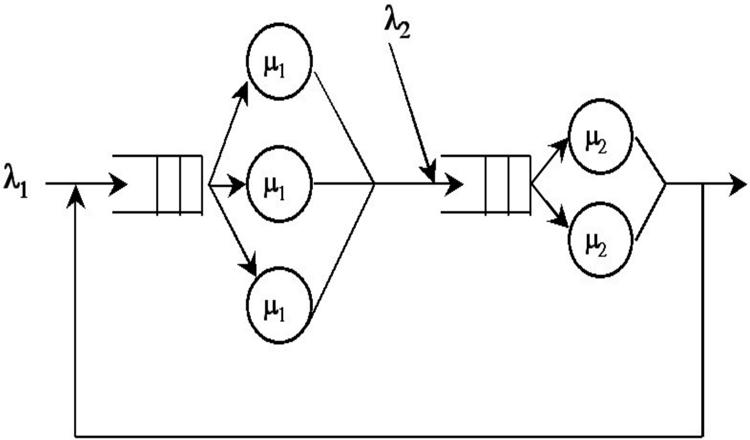
Discrete-event simulation is a computer simulation model/program of a stochastic system that evolves dynamically in time via state information which changes at discrete time epochs.

The components of a DES are

- Entities
- Time variable (or simulation clock)
- System state variables
- Events and Event list (or calendar)
- Global variables
- Statistics collectors (or counter variables)

Reference Model

As a reference model we consider a queueing network model with feedback.



Modeling aspects

- The queueing model consists of two stations ('queues') Q_1 and Q_2.
- At Q_i jobs arrive according to a Poisson(λ_i) process.
- There are three servers at Q_1 and two servers at Q_2.
- The service times of the servers at Q_i have distribution function $G_i(\cdot)$.
- After service completion at Q_1 the jobs enter Q_2.
- After service completion at Q_2 a job leaves the system with probability p or re-enters ('feeds back' to) Q_1 with probability $1 - p$.
- Both stations have infinite waiting spaces.
- Waiting jobs are served in order of arrival (FCFS).

Performance measures

Typically we are interested in

- average waiting times or system times at the two stations;
- mean waiting lines or system lines at two stations;
- average time spend in the system (sojourn time);
- throughput (per station or from the whole system);
- utilization;
- ...

Transient vs Steady-State

We have to specify

- whether we wish to estimate these performance measures for a finite period; for instance the queueing system operates daily, opening empty at 8.00 hr in the morning, closing at 18.00 hr in the evening.
- or whether we wish to estimate steady-state averages; then we assume that the system operates for an infinite time.

Entities

- Actually, a system is defined to be a collection of entities, e.g., people or cars or machines, that act and interact together.
- Without entities, nothing would happen.
- Entities have attributes, usually given as data values.
- In our example the entities are (attributes between brackets)
  - –> jobs or customers (arrival time);
  - –> servers (idle/busy).
- Most often in a simulation study, we do not bother too much with entities.

System state

- The system state is a collection of variables necessary to describe a system at a particular time.
- The set of all states is denoted by $\mathcal{X}$; a specific state by $\boldsymbol{x} \in \mathcal{X}$; and the (random) state at time t by $\boldsymbol{X}(t) \in \mathcal{X}$.
- In our example the state comprises
  - –> the number of jobs (x_1, x_2) present at the two stations;
  - –> two vectors a_1, a_2 of the arrival times of these waiting jobs;
  - –> two vectors b_1, b_2 specifying the status of the servers (idle/busy).

Events

- An event is an instantaneous occurrence that may change the state or trigger a state transition.
- The set of all possible events is denoted by $\mathcal{E}$; a specific event by e; the events active in state $\boldsymbol{x} \in \mathcal{X}$ by $E(\boldsymbol{x}) \subset \mathcal{E}$.
- In our example events are
  - –> the arrival of a new job (at Q_1 or at Q_2);
  - –> a service completion at one of the five servers.

Time or clock variable

- Events occur at some point in time.
- For this we need a variable representing the current time of the simulation.
- This is also called the simulation clock or system time that measures the elapsed simulation time.
- Clearly we need to specify its unit size (second or minute or ...); then we set it to zero at the start of a simulation and update it every time an event occurs.

Event list or calendar

- The calendar for the simulation is a list of the events that are currently scheduled to occur.
- There is only one event list and it consists of the scheduled event times, sorted by the earliest scheduled time first.
- The event list at time t is denoted by $L_{\mathrm{ev}}(t)$.
- In our example the event list comprises
  - –> the next arrival time of a customer at Q_1;
  - –> the next arrival time of a customer at Q_2;
  - –> for any busy server the next time he is ready completing the job.

A Set of Alarm Clocks

- Another view of the event list is that it is a collection of alarm clocks, one for each scheduled event. The clocks are preset at different (random) times in the future. For instance at some t there are 3 events scheduled:

$L_{\mathrm{ev}}(t) = \{$

arrival Q_1

arrival Q_2

service completion Q_2

$\}$

The Event-Scheduling Approach

- The discrete-event simulation runs as follows.
  - –> The simulation clock is advanced forward to the earliest time on the event list (when the first alarm goes off).
  - –> The alarm belongs to a particular event that triggers several activities in the system.
  - –> These activities make that there will be changes in the system state and in the event list; these need to be updated.
  - –> Then the simulation clock is advanced again, etc.

A walk through DES

In our queueing example we let time be measured in seconds, starting at 08:00. This is system time 0.

- Suppose current time is t_{sim} and state $\boldsymbol{x} \in \mathcal{X}$.
- Each active event $e_i \in E(\boldsymbol{x})$ has an associated scheduled alarm time t_i.
- We denote the type-time pair of events by $[\text{'A1'}, \texttt{<time>}]$ for the arrival event at time $\texttt{<time>}$ at Q_1 (and similarly for arrival event at Q_2);
- and by $[\text{'D1'}, 1, \texttt{<time>}]$ for departure events due to service completion at server 1 of Q_1; similarly for the other servers, and the other queue.
- We start $t_{\text{sim}} = 0$ with an empty system and idle servers:

$$\boldsymbol{x} = \{x_1 = 0, x_2 = 0, a_1 = [], a_2 = [], b_1 = (0, 0, 0), b_2 = (0, 0)\}.$$

- There are two active events: arrivals at the queues; the associated event times are drawn by our random number generator, resulting. Thus

$$L_{\text{ev}} = \{[\text{'A2'}, 0.817], [\text{'A1'}, 1.284]\}.$$

The 1-st event

- The simulation time is advanced to the earliest event time 0.817 belonging to the event of an arriving job at Q_2;
- Immediately, this job enters service with server 1 of Q_2;
- We draw a sample of the service time S_2, say 1.693, to schedule a new event ‘D2’ with departure time $0.817 + 1.693 = 2.510$;
- Also we realise a new sample of the interarrival time A_2, say 1.226, and update the scheduled time of event ‘A2’ to $0.817 + 1.226 = 2.043$;

Thus the new situation is:

$$\begin{aligned} t_{\text{sim}} &= 0.817 \\ \boldsymbol{x} &= \{x_1 = 0, x_2 = 1, a_1 = [], a_2 = [], b_1 = (0,0,0), b_2 = (1,0)\} \\ L_{\text{ev}} &= \{[\text{‘A1’}, 1.284], [\text{‘A2’}, 2.043], [\text{‘D2’}, 1, 2.510]\} \end{aligned}$$

The 2-nd event

- The simulation time is advanced to the earliest event time 1.284 belonging to the event of an arriving job at Q_1.
- This job enters service with server 1 of Q_1 .
- We draw a sample of the service time S_1, say 2.613, to schedule a new event 'D1' with departure time $1.284 + 2.613 = 3.987$;
- We realise a new sample of the interarrival time A_1, say 0.577, and update the scheduled time of event 'A1' to $1.284 + 0.577 = 1.861$;

Thus the new situation is:

$$\begin{aligned} t_{\text{sim}} &= 1.284 \\ \boldsymbol{x} &= \{x_1 = 1, x_2 = 1, a_1 = [], a_2 = [], b_1 = (1,0,0), b_2 = (1,0)\} \\ L_{\text{ev}} &= \{[\text{'A1'}, 1.861], [\text{'A2'}, 2.043], [\text{'D2'}, 1, 2.510], [\text{'D1'}, 1, 3.987]\} \end{aligned}$$

The 3-rd event

- The simulation time is advanced to the earliest event time 1.861 belonging to the event of an arriving job at Q_1.
- This job enters service with server 2 of Q_1.
- We draw a sample of the service time S_1, say 0.811, to schedule a new event 'D1' with departure time $1.861 + 0.811 = 2.752$;
- We realise a new sample of the interarrival time A_1, say 1.428, and update the scheduled time of event 'A1' to $1.861 + 1.428 = 3.289$;

Thus the new situation is:

$$
\begin{aligned}
t_{\text{sim}} &= 1.861 \\
\boldsymbol{x} &= \{x_1 = 2, x_2 = 1, a_1 = [], a_2 = [], b_1 = (1,1,0), b_2 = (1,0)\} \\
L_{\text{ev}} &= \{[\text{'A2'}, 2.043], [\text{'D2'}, 1, 2.510], [\text{'D1'}, 2, 2.752], [\text{'A1'}, 3.289], [\text{'D1'}, 1, 3.987]\}
\end{aligned}
$$

The k-th event

Suppose that currently the situation is

$$
\begin{aligned}
t_{\text{sim}} &= 249.31 \\
\boldsymbol{x} &= \{x_1 = 4, x_2 = 8, a_1 = (240.82), a_2 = (238.71, 240.61, 241.01, 244.55, 246.91, 248.88), \\
&\quad b_1 = (1, 1, 1), b_2 = (1, 1)\} \\
L_{\text{ev}} &= \{[\text{`D2'}, 2, 250.28], [\text{`D2'}, 1, 251.43], [\text{`A1'}, 254.38], [\text{`D1'}, 2, 255.36], \\
&\quad [\text{`A2'}, 256.42], [\text{`D1'}, 1, 260.91], [\text{`D1'}, 3, 263.93]\}
\end{aligned}
$$

- The next event is service completion at Q_2 at time 250.28.
- There are jobs waiting at Q_2 to occupy the empty seat. We draw a sample of the service time S_2, say 0.83, and update the scheduled time of event 'D2', 2 to $250.28 + 0.83 = 251.11$.
- For the job leaving Q_2 we flip a coin (Bernoulli random variable) to decide a feed back; suppose the outcome is to loop back for entering Q_1 again.
- The looped job finds all servers busy at Q_1, and thus joins the queue.

The k-th event (cont'd)

Next situation:

$$
\begin{aligned}
t_{\text{sim}} &= 250.28 \\
\boldsymbol{x} &= \{x_1 = 5, x_2 = 7, a_1 = (240.82, 250.28), a_2 = (240.61, 241.01, 244.55, 246.91, 248.88), \\
&\quad b_1 = (1, 1, 1), b_2 = (1, 1)\} \\
L_{\text{ev}} &= \{[\text{'D2'}, 2, 251.11], [\text{'D2'}, 1, 251.43], [\text{'A1'}, 254.38], [\text{'D1'}, 2, 255.36], \\
&\quad [\text{'A2'}, 256.42], [\text{'D1'}, 1, 260.91], [\text{'D1'}, 3, 263.93]\}
\end{aligned}
$$

Trace

- This detailed walk through a system simulation is called a *trace*.
- In a trace a print is made of all the (important) simulation components after each event.
- It serves two objectives.
  - *(i).* In developing a simulation program it helps you to think about the events that may happen in the system, the activities they trigger and their logic.
  - *(ii).* After having written a computer program of the simulation, it provides a check whether the program is correct.

Simulation run

- A trace covers usually a short time interval or a small number of events.
- The actual execution of the computer program of the simulation is longer but should end at some time or after some number of events: *stopping criterion*.
- One such a simulation is called a *simulation run* which corresponds to the concept of sample path of a stochastic process.
- Later we will see that in most simulations studies you will simulate (many) more simulation runs in order to obtain more reliable estimates.

Programming a Run

- A simulation run of the system goes from event to event at discrete time epochs determined by realisations of the appropriate random variables.
- When you write a computer program of the simulation it is convenient to modularise your program into several subprograms or routines to clarify the logic and the interactions.
- Roughly a simulation run looks in pseudo-code as follows.

```
initialise;
REPEAT
    [e,t] = next_event_type_and_time;
    switch (e)
        case 'arrival': ...
        case 'departure': ...
        case '. . .': ...
UNTIL stopping_criterion;
```

Programming a Run (cont'd)

In the subroutines for the different events you program code for updating state, event list, and statistical counters. Similar as what you would do in the trace. For instance:

```
subroutine execute_arrival_at_Q1_event(t)
begin
    update_all_relevant_counter_variables();
    j = find_free_server();
    if (server(j)=='idle')
        server(j) = 'busy';
        s = generate_servicetime();
        add_to_eventlist('D1',j,t+s);
    else
        add_customer_to_queue(j,t);
    end
    a = generate_interarrivaltime();
    add_to_eventlist('A1',t+a);
end
```

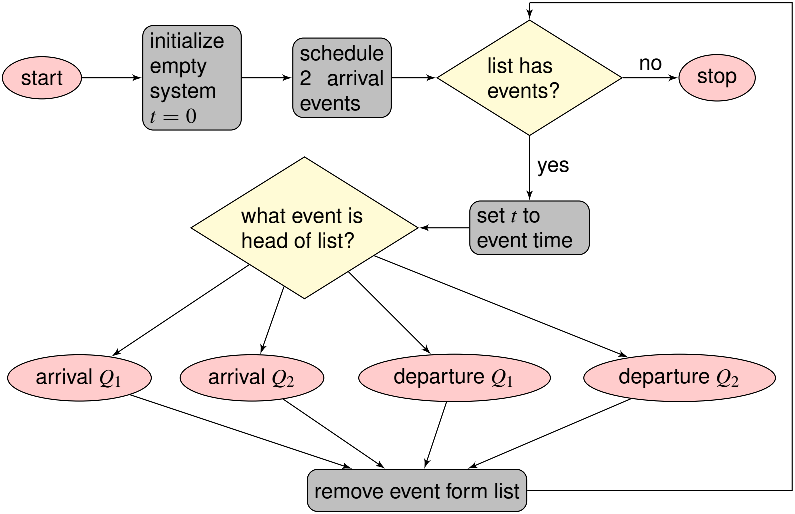
Flowchart

Visualization of the various steps in the simulation is done via flowcharts.

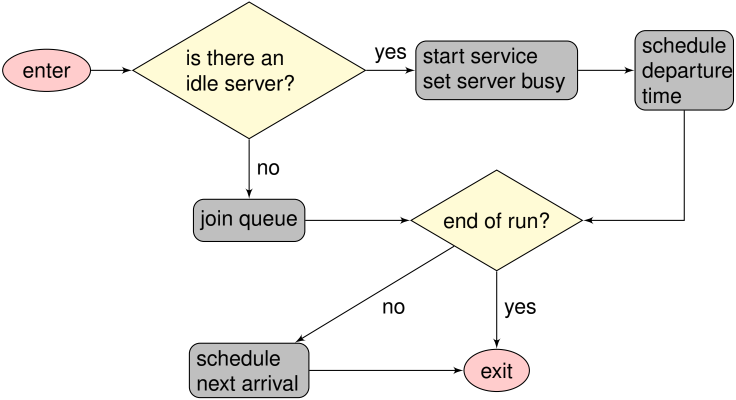
It helps to understand the process.

For large models you break up in several charts.

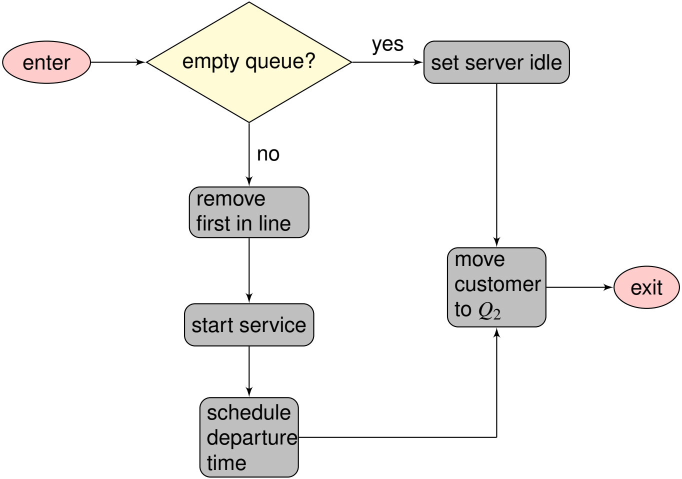
Main Loop


start
initialize empty system $t = 0$
schedule 2 arrival events
list has events?
no
stop
yes
set t to event time
what event is head of list?
arrival Q_1
arrival Q_2
departure Q_1
departure Q_2
remove event form list

Arrival Q_1 Event



Service Completion Q_1 Event



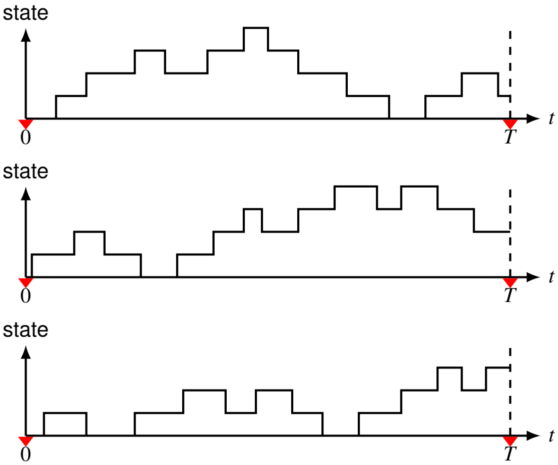
Finite-Horizon

Simulation

Procedure

- A run starts at time 0, at a specific specified state;
- A run ends at some stopping criterion, e.g.
  - *(i).* A time horizon T;
  - *(ii).* A number of events N_e;
  - *(iii).* A number of arrivals N_a;
  - *(iv).* Etc.
- Execute n runs, independent and (probabilistically) identical.
- In each run we keep track of certain variables (*counter variables*) that at the end of the run determine its outcomes or outputs.
- The performance measures are estimated by averaging these n outcomes.
- The counter variables are initialised at the start of a simulation run, and because nothing changes in between two consecutive events, it suffices to update them after an event occurs.

Illustration



run 1: compute output Y_1.

run 2: compute output Y_2.

run 3: compute output Y_3.

Example

- Consider Q_1 in the tandem network.
- Suppose the goal is to estimate the performance measure "mean average waiting time per customer from 08:00-18:00".
- This means:
  - –> let K be the (random) number of customers that has been served during these ten hours at Q_1;
  - –> let $W_1, \ldots, W_K$ be their corresponding (random) waiting times;
  - –> define the output $Y = (1/K) \sum_{j=1}^{K} W_j$;
  - –> then we wish to compute $\mathbb{E}[Y]$.

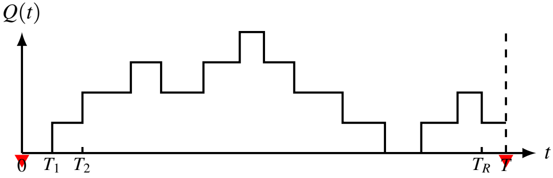
Example (cont'd)

- During a run in the simulation program you keep track of two variables:
  1. n_{serv1} = the current total number customers who went into service at Q_1;
  2. w_{total1} = the current total waiting time of these n_{serv1} customers.
- Whenever a new customer enters service at Q_1, these two variables are updated.
- At the end of the run you have a realisation $y = w_{\text{total1}}/n_{\text{serv1}}$ of the output Y.
- Repeat n times and do your statistics.
- For such finite-horizon simulation the statistics is similar as for static stochastic simulation (see the lecture on the Monte Carlo simulation).

Estimating the Average Queue Length

- How to compute (or estimate) the mean length at Q_1 during the interval 08:00-18:00?
- Let $Q(t)$ be the queue length at time t, $0 \leq t \leq T$, where $t = 0$ represents 08:00 and T represents 18:00; measured in some time-units;
- Goal: $\mathbb{E}[Y]$ for output $Y = \frac{1}{T} \int_0^T Q(t)\, dt$;
- Interpretation of the integral: area below graph of the function $Q(t)$.

Illustration



Let $0 = T_0 < T_1 < T_2 < \cdots < T_R < T \leq T_{R+1}$ be the consecutive event times.

$$Y = \frac{1}{T} \sum_{r=1}^{R} (T_r - T_{r-1}) Q(T_{r-1}) + \frac{1}{T} (T - T_R) Q(T_R)$$

Note: number of events R is random.

Example and Program Code

See the file `simnotes.pdf`