Advanced Simulation (2DI66) – Assignment 3
Asset Maintenance Planning

Deadline: April 18, 2021 at 23:59

1 Introduction

In this assignment, we focus on a different application area. Namely, on the maintenance of capital assets. Nowadays, we expect capital assets such as wind turbines, trains, hospital equipment or airplanes to experience as little downtime as possible. This drastically increases the operational costs of such assets. The ideal way of insuring no downtime at the lowest possible cost is to maintain the asset just prior to failure. To this end, two challenges complicate the problem at hand: the failure mechanism of an asset is random (this is captured by the degradation process) and such assets are parts of a fleet (network dynamics).

In Assignment 1, we used stochastic simulation to determine smart strategies for a specific board game and in Assignment 2, we used the same technique for a different purpose: to obtain numerical values for the performance measures of a queuing system that is generally very complicated to analyse. What both assignments have in common, is that we look for *policies* which optimize some performance criterion. Often, such decision problems are modelled as a *Markov Decision Process (MDP)*[1]. In this last Assignment, you are asked to use stochastic simulation to investigate such an MDP. For a stylized base model assigned to you, you are asked to investigate the performance of two simple policies. Then, you are asked to repeat this performance analysis for a model variation where we account for "degradation load sharing" in case of machine failures.

An informal introduction to MDPs is enclosed with this Assignment in Appendix A as a hand-out. Please study the hand-out to familiarize yourself with this abstract mathematical model for decision-making under uncertainty before reading the remainder of this assignment text.

2 Mathematical model description

Formally, we assume a network with $M \in \mathbb{N}_+$ *machines*, denoted by $\mathcal{M} = \{1, \ldots, M\}$. Each machine is located at a unique location (see Section 2.1). Each machine in the network is subject to degradation (see Section 2.2). Initially, we assume that the degradation processes of the machines are independent one of the other. The machines can be preventively or correctively maintained. The latter can only occur when a machine fails. We assume that a single *field service engineer* (FSE) travels the network and maintains the machines.

Essentially, when the FSE is idle (which constitutes a *decision epoch*), the FSE has to decide whether to travel to some other location (a different location than the one the FSE is currently at), start a maintenance activity at the current location or simply wait at the current location.

The cost structure in this assignment is as follows: if the machine is preventively maintained (PM), then a lower cost is paid for PM. If the machine is correctively maintained (CM), then a higher cost for CM is paid. Moreover, a downtime penalty is incurred only applicable to machines which are non-operational. A machine is non-operational when it is being repaired (either CM or PM), or when it has failed. The downtime penalty is proportional to the length of the downtime period, thus creating an incentive for the FSE to repair the machine as soon as possible. We store these costs in the sets $\{c_m^{\mathrm{PM}} \mid m \in \mathcal{M}\}$, $\{c_m^{\mathrm{CM}} \mid m \in \mathcal{M}\}$ and $\{c_m^{\mathrm{DT}} \mid m \in \mathcal{M}\}$ respectively.

[1] See `https://www.math.leidenuniv.nl/~kallenberg/Lecture-notes-MDP.pdf` for an introduction to MDP's, as well as many illustrative examples of their applications.

Finally, the objective is as follows: We consider an infinite time horizon and the goal is to determine a policy which minimizes the *average cost criterion*, aka the long-run rate of cost. The average cost criterion is a common optimality criterion used in MDPs.

Important details can be found below.

2.1 Network of machines

The network layout, represented in Figure 1, is as follows. Each machine is located at a unique location and it takes a deterministic time $\theta_{ij} \in \mathbb{R}_+$ for the FSE to travel between two machines $i, j \in \mathcal{M}$ where $i \neq j$. The travel times are stored in a (symmetric) matrix $\Theta \in \mathbb{R}_+^{M \times M}$. By definition, $\theta_{ii} = 0$ for all $i \in \mathcal{M}$.

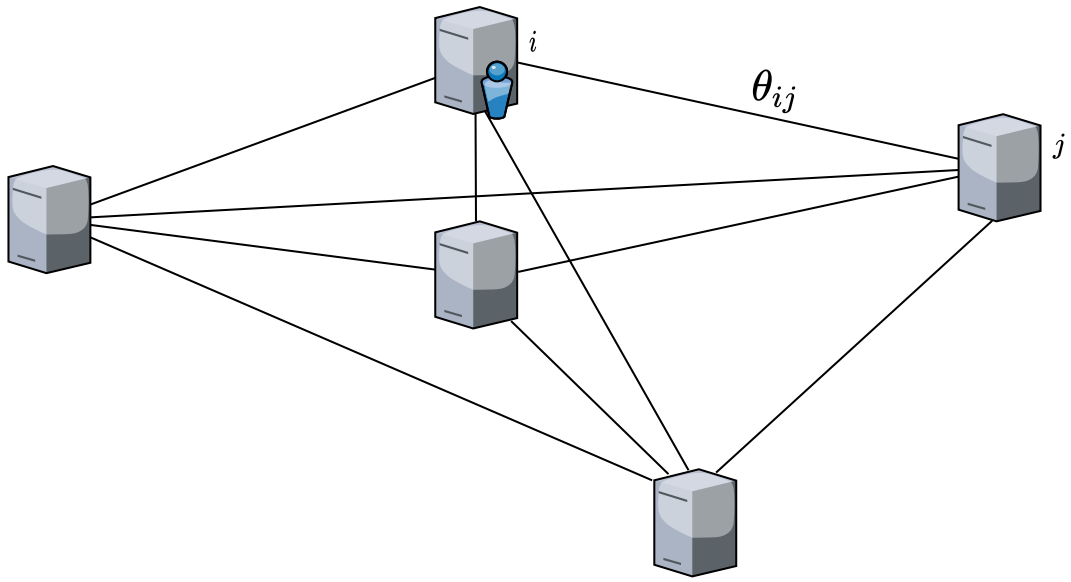

Figure 1: Schematic overview of the layout of a network consisting of $M = 5$ machines.

Note that for the purpose of this assignment, by construction, the network is always fully connected.

2.2 Degradation of machines

Each machine $m \in \mathcal{M}$ is assumed to degrade (independently of the other machines in the network) according to a non-decreasing degradation process. The degradation process is modelled as follows: For each machine $m \in \mathcal{M}$, we define a degradation process $\{X_m(t), t \in \mathbb{N}\}$ where $X_m(0) = 0$ and $X_m(t) \in \mathbb{R}_+$ for all $t \geq 0$. Without a maintenance intervention by the FSE, the degradation processes $\{X_m(t)\}_{m \in \mathcal{M}}$ are assumed to be non-decreasing: $X_m(t+1) \geq X_m(t)$. In what follows, we assume that the degradation process has a special structure. Namely, a *Compound Poisson Process*, i.e.,

$$X_m(t) = \sum_{i=1}^{N_m(t)} D_m^{(i)},$$

with $N_m(t)$ denoting a Poisson Process with arrival intensity $\lambda_m > 0$. Note that the Poisson processes $\{N_m(t)\}_{m \in \mathcal{M}}$ are independent one from another. The sequence of random variables $D_m^{(1)}, D_m^{(2)}, \ldots \sim D_m$ is assumed to be positive and i.id., specifically, we assume that the *jump size distribution* follows a Beta distribution, i.e., $D_m \sim \text{Beta}(\alpha_m, \beta_m)$, where $\alpha_m, \beta_m > 0$ are called the shape parameters. The probability density function of the Beta distribution is defined as

$$f(x; \alpha_m, \beta_m) = \frac{1}{\mathbf{B}(\alpha_m, \beta_m)} x^{\alpha_m - 1}(1-x)^{\beta_m - 1}, \text{ for } x \in [0, 1],$$

where the beta function, $\mathbf{B}$, is a normalization constant to ensure that the total probability equals 1. Note that the Poisson processes $\{N_m(t)\}_{m\in\mathcal{M}}$ are independent from the jumps $\{D_m^{(i)}\}_{i\in\mathbb{N}_+,m\in\mathcal{M}}$.

Failure mechanism:
We assume that when the degradation processes $X_m(t)$ exceeds the *failure threshold* $\xi_m \in \mathbb{R}_+$, the degradation is capped at ξ_m. In other words, if $X_m(t) \geq \xi_m$, then set $X_m(t) = \xi_m$. For simplicity, we assume that, for all $m \in \mathcal{M}$, $\xi_m \equiv \xi \in \mathbb{R}_+$. The machine will thus reside in this state until a corrective maintenance action is performed by the FSE.

Performing maintenance:
As stated in the introduction, the FSE can perform two types of maintenance:

CM: If $X_m(t) = \xi$ and the FSE starts a CM action, then a cost c_m^{CM} is incurred *at the start of the corrective maintenance action*. The FSE again becomes available after t_m^{CM} units of time.

PM: If $X_m(t) < \xi$ and the FSE starts a PM action, then a cost c_m^{PM} is incurred *at the start of a preventive maintenance action*. When the FSE starts PM on machine m, we set $X_m(t) = \xi$ to indicate that the machine is non-operational. The FSE again becomes available after t_m^{PM} units of time.

When a machine m becomes non-operational (say at time $\tilde{t}_m$), a downtime penalty c_m^{DT} is incurred for each *full* time unit the machine has been non-operational, i.e, at timestamps $\tilde{t}_m + 1, \tilde{t}_m + 2, \ldots$, the FSE pays a cost c_m^{DT} until the machine is repaired. Note that $\tilde{t}_m$ has **two triggers**: 1) a machine m reaching the failed state (i.e., when $X_m(t)$ hits ξ) and 2) the start of a maintenance action by the FSE. Regardless of the action (PM or CM) and the state of the machine at the start of the maintenance action, after a maintenance action is completed, the machine's state $X_m(t)$ is reset to its initial condition $X_m(0) = 0$.

2.3 Policies

As the model dynamics are *Markovian*, the *decision epochs* coincide with the (transition) events in the model.

To help you get started, we ask you to investigate *at least* the following asset maintenance planning policies π:

- (**Reactive policy**)**:** At each decision epoch, the FSE will either: 1) travel to a failed machine (i.e., a machine m' which satisfies $X_{m'}(t) = \xi$); 2) repair a machine which has failed at the current location of the FSE; or 3) remain idle. In case there are multiple failed machines, the FSE will select the machine (from the set of failed machines) with the maximum downtime penalty (in terms of $\{c_m^{\mathrm{DT}}\}_{m\in\mathcal{M}}$), in case of ties, the FSE will pick a machine from this subset uniformly at random. Note that it can happen that the FSE arrives at the location of a failed machine, but may immediately decide to travel to another machine which has failed during this last travel time.
- (**Greedy policy**)**:** At each decision epoch, the FSE will either: 1) travel to the non-healthy machine with the biggest degradation, i.e., the machine m' which satisfies $m' = \underset{m\in\mathcal{M}:X_m(t)>0}{\arg\max} X_m(t)$; 2) repair the machine at the FSE's current location; or 3) remain idle. In case there are multiple machines with the biggest degradation, the FSE will select a machine from this set which is closest (in terms of Θ). Ties are handled as follows: the FSE will pick a machine from the the constructed subset of closest machines with the biggest degradation uniformly at random. Note that if the FSE is already present at such a location m', then the FSE will always start a maintenance action on this machine as $\theta_{m'm'} = 0$.

For **any policy**, we assume that both types of maintenance **cannot** be interrupted. Moreover, when the FSE selects the idle action, it will thus remain idle until the next state change (event) in the network.

3 Input file per student

Each student will receive two separate files, the first one will contain all model parameters and the second one will contain the travel time matrix. **We expect your program to be able to import the provided text files and read the required input parameters from these files!**

An example of both input files follows below.

Model parameters:
The model parameters are $\xi_m = \xi, \lambda_m, \alpha_m, \beta_m, t_m^{\text{PM}}, t_m^{\text{CM}}, c_m^{\text{PM}}$ and $c_m^{\text{CM}}, c_m^{\text{DT}}$ for $m \in \{1, \ldots, M\}$. Thus, the first line reads $\xi_1, \ldots, \xi_M$, the second line reads $\lambda_1, \ldots, \lambda_M$, the third line reads $\alpha_1, \ldots, \alpha_M$, the fourth line reads $\beta_1, \ldots, \beta_M$, the fifth line reads $t_1^{\text{PM}}, \ldots, t_M^{\text{PM}}$, the sixth line reads $t_1^{\text{CM}}, \ldots, t_M^{\text{CM}}$, the seventh line reads $c_1^{\text{PM}}, \ldots, c_M^{\text{PM}}$, the eight line reads $c_1^{\text{CM}}, \ldots, c_M^{\text{CM}}$ and lastly the ninth line reads $c_1^{\text{DT}}, \ldots, c_M^{\text{DT}}$. So,

```
10 10 10 10
1.56 1.93 1.72 1.48
0.57 0.69 0.35 0.45
0.75 0.45 0.58 0.7
1.52 1.26 1.96 1.54
1.94 1.6 2.5 1.96
2 1.65 1.11 1.04
7.69 6.37 4.27 3.99
7.84 9.81 8.73 8.31
```

$\Longrightarrow$

m	1	2	3	4
ξ_m	10	10	10	10
λ_m	1.56	1.93	1.72	1.48
α_m	0.57	0.69	0.35	0.45
β_m	0.75	0.45	0.58	0.7
t_m^{PM}	1.52	1.26	1.96	1.54
t_m^{CM}	1.94	1.6	2.5	1.96
c_m^{PM}	2	1.65	1.11	1.04
c_m^{CM}	7.69	6.37	4.27	3.99
c_m^{DT}	7.84	9.81	8.73	8.31

Table 1: Meaning of the input file.

Figure 2: Example of an input file for $M = 4$

Travel time matrix:
The travel time matrix simply contains the entries θ_{ij} for all $i, j \in \mathcal{M}$. So,

```
0 1.74 1.36 1.83
1.74 0 1.61 1.55
1.36 1.61 0 1.64
1.83 1.55 1.64 0
```

$\Longrightarrow$

$$\Theta = \begin{bmatrix} 0 & 1.74 & 1.36 & 1.83 \\ 1.74 & 0 & 1.61 & 1.55 \\ 1.36 & 1.61 & 0 & 1.64 \\ 1.83 & 1.55 & 1.64 & 0 \end{bmatrix}$$

Initial state:
Regardless of the input file, you are expected to initialize the network as follows: (1) The FSE is placed at machine 1, and (2) all machines start in the healthy state, that is, $X_m(0) = 0$ for all $m \in \mathcal{M}$. Consequently, $t = 0$ will be the first decision epoch.

4 Assignment tasks

First and foremost, we are interested in the performance analysis of the network that you are given. In particular, we define the following output variables:

- ρ_m is the fraction of time machine m is operational (meaning that $\rho_m := \lim_{t\to\infty} \int_0^t \mathbb{1}\{X_m(s) < \xi\} \mathrm{d}s/t$ with $\mathbb{1}\{\cdot\}$ denoting an indicator function taking value 1 if the event is satisfied and value 0 otherwise).
- τ_m is the response time of the FSE to a failure at machine m, i.e., the time between a failure occurring at machine m and the *start of the corrective maintenance action* at machine m.
- $g_m(\pi)$ denotes the average cost criterion for a policy $\pi \in \{\text{Reactive, Greedy}\}$ per machine, i.e., $g_m(\pi) = \lim_{t\to\infty} C_m(t)/t$ where $C_m(t)$ is the cumulative maintenance cost paid for the m-th machine up until time t.
- $g(\pi)$ denotes the performance criterion for a policy π for the network, defined as $g(\pi) = \lim_{t\to\infty} C(t)/t$ where $C(t) = \sum_{m\in\mathcal{M}} C_m(t)$ denotes the total maintenance costs paid by the FSE up until time t.

Question 1:
Include a table with performance measures. We ask you to write a discrete-event simulation that computes the numerical results listed in Table 2, for the specific network that you obtained (in the table we assume 4 machines, but each student might have a different number of machines). Include a table like Table 2 in the appendix of your report, for each investigated policy. Interpret the results.

Machine	ρ_m	$\mathbb{E}[\tau_m]$	$\mathbb{V}[\tau_m]$	$g_m(\pi)$	$g(\pi)$
1					
2					
3					
4					

Table 2: Table of performance measures per policy π.

For the random variables τ_m, $m \in \mathcal{M}$, provide, in addition to their expectation and their variance in Table 1, the empirical distribution function. Can you fit a distribution to the simulated data?

Include a motivation of the number of runs and the simulation length in your report.

Hint: Implement the FSE actions, the machine degradation processes and the downtime penalties using *discrete event scheduling*.

Question 2:
In this Question, you are asked to implement a simple dependency between the degradation processes. When a machine breaks down, we assume that the load of this machine is distributed over the operational machines in the network uniformly. Recall that both splitting (thinning) and merging independent Poisson Processes results in independent Poisson processes. Thus, the arrival rate of an operational machine m at time t is given by

$$\lambda_m(t) = \lambda_m + \sum_{m' \in \mathcal{M}: X_{m'}(t) = \xi} \frac{1}{|\{m \in \mathcal{M} \mid X_m(t) < \xi\}|} \lambda_{m'}.$$

Implement this dependency and repeat Question 1.

Bonus Question:
For the base model, design a policy which outperforms the two policies proposed in Section 2.3 (in terms of the average cost $g(\pi)$). Clearly state your approach: How is the policy obtained? What are the advantages and disadvantages of this approach? Can you think of circumstances under which this policy will perform poorly? Investigate your policy using the introduced performance measures (e.g., repeat Question 1 for this policy).

Note that the quality of the design will be graded as well, e.g. a simple modification of the Reactive or Greedy policy will not earn you full bonus points. You are allowed to use **all** (distributional) information about the machines given to you in the input file and this assignment text, as well as real-time monitoring data (i.e, $\{X_m(t)\}_{m \in \mathcal{M}}$ is available to the FSE at each decision epoch).

Hint: A possible approach could be to design an interesting *set of policies* parametrized by one or multiple parameters. Use stochastic simulation to find the optimal policy in this set.

5 Deliverable

Write a program that can simulate the asset network with one FSE as described in this assignment. The program should be able to import the provided file to obtain the model parameters. It should be elegantly programmed: more points are awarded for well-structured code. Another important desirable feature of you code is its flexibility: how easy is it to make changes to simulate extensions of the model? How easy is it to add other policies or modify the existing ones?

Write a report, preferably in LaTeX, and hand in a PDF detailing your implementation and findings. The report must contain at least the following:

- A brief motivation of the problem and a short description of the stochastic model.
- A detailed description of your implementation. Use pseudo-code to most effectively communicate your implementation within the report.
- Well-designed and clear plots that provide insight in the relevant performance measures. The more scientifically interesting your plots are, the better!
- A brief discussion and conclusion based on your plots. The interpretation of your results is just as important as the results themselves!
- Exact copy of your code in the Appendix. Code in the Appendix does not count towards a page limit.
- A table like Table 2 containing the performance measures obtained by your discrete-event simulation.

5.1 Knockout criteria

Your report and code **must** meet the following requirements in order to be marked:

1. An electronic copy must be submitted in Canvas (PDF), on time.
2. It includes a title, author name, student number, and bibliography when citing.
3. There is a hard page limit of **10 A4 pages**, single column. This excludes code in the Appendix and the bibliography.
   Focus on what is important: it is about quality of content, not quantity.
4. Margins should be at least 1cm, and font size at least 10 pt.
5. The overall presentation is clean / neat / orderly.
6. Your texts are legible, in English, and contain few spelling mistakes.

5.2 Grading

The assignment will count towards 50% of the final grade of the course 2DI66. For Questions 1 and 2, you can earn up to 100 points. By doing the Bonus Question, you earn up to 10 additional points. The grade for this Assignment is determined as $\min\{(T_1 + T_2)/100, 10\}$, where T_1, T_2 are the number of points scored for Question 1&2 and the Bonus Question, respectively. The passing grade for this Assignment is 5.0.

We will be judging for effort and you may report difficulties or even failure. The reporting however, has to be detailed and genuine. Your implementations / attempts have to be critically reported. We expect you to be creative in the way you present your results. Merely doing the bare minimum of what we ask may not be sufficient to obtain full points.

5.3 More details

Urgent questions can be sent to s.kapodistria@tue.nl and/or p.verleijsdonk@tue.nl. Upload your report in PDF format before the deadline specified in Canvas. This is a hard deadline! Please include a zip file containing your source code. You can use the (object-oriented) programming language of your choice. However, the use of external libraries (such as SimPy) is **not** allowed!

Appendix: A

The MDP model:
An MDP is a model for sequential decision making under uncertainty, taking into account both the short-term outcomes of current decisions and opportunities for making decisions in the future. While the notion of an MDP may appear quite simple, it encompasses a wide range of applications and has generated a rich mathematical theory. In an MDP model one, can distinguish the following seven characteristics.

1. **The state space:**
   At any time point at which a decision has to be made, the state of the system is observed by the decision-maker. The set of possible states is called the state space and will be denoted by S. The state space may be finite, denumerable, compact or even more general.

2. **The action sets:**
   When the decision maker observes that the system is in state i, he (we will refer to the decision-maker as 'he') chooses an action from a certain action set that may depend on the observed state: the action set in state i is denoted by $A(i)$. Similarly to the state space, the action sets may be finite, denumerable, compact or more general.

3. **The decision time points:**
   The time intervals between the decision points may be constant or random. In the first case, the model is said to be a *Markov decision process*; when the times between consecutive decision points are random the model is called a *semi-Markov decision process.*

4. **The immediate rewards (or costs):**
   Given the state of the system and the chosen action, an immediate reward (or cost) is earned (there is no essential difference between rewards and costs, namely: maximizing rewards is equivalent to minimizing costs). These rewards may in general depend on the decision time point, the observed state and the chosen action, but not on the history of the process. The immediate reward at decision time point t for an action a in state i will be denoted by $r_i^t(a)$; if the reward is independent of the time t, we will write $r_i(a)$ instead of $r_i^t(a)$.

5. **The transition probabilities**
   Given the state of the system and the chosen action, the state at the next decision time point is determined by a transition law. These transitions only depend on the decision time point t, the observed state i and the chosen action a and not on the history of the process. This property is called the *Markov property.* If the transitions really depend on the decision time point, the problem is said to be *non-stationary.* If the state at time t is i and action a is chosen, we denote the probability that at the next time point the system is in state j by $p_{ij}^t(a)$. If the transitions are independent of the time points, the problem is called *stationary*, and the transition probabilities are denoted by $p_{ij}(a)$.

6. **The planning horizon:**
   The process has a planning horizon, which is the result of the time points at which the system has to be controlled. This horizon may be finite, infinite or of random length.

7. **The optimality criterion:**
   The objective of a Markov decision problem (or a semi-Markov decision problem) is to determine a policy, i.e. a decision rule for each decision time point and each history (including the present state) of the process, that optimizes the performance of the system. The performance is measured by a utility function. This utility function assigns to each policy a value, given the starting state of the process.