TD 12 – Refreshing

Exercice 1. *Alice needs help!*

Alice is writing her PhD thesis, using an unreliable old laptop. As she is afraid of losing work in case the laptop crashes, she regularly saves her work on an external disk. Alice decided to save her work on the external disk every three hours. Writing to external disk takes three minutes. During day 3, Alice's laptop crashed, she had to reboot, and lost 90 minutes of work. She thus decided to save to the external disk every half an hour instead. But after three additional days of work without problems, she realized that she did less work on days 4–6 than days 1–3 (despite the loss in the 3rd day).

1. Intuitively explain why she did more work on the first three days

Alice is puzzled now: what is the best frequency to save her work?
The technique of saving intermediate work is called *checkpointing*. Because Alice works for a constant amount of time between two checkpoints, her technique is called *periodic checkpointing*.
Computing environments are prone to faults. We can use exponential distributions to to quantify the rate or frequency at which these faults strike. The definition of $Exp(\lambda)$, the Exponential distribution law of parameter λ, goes as follows:

- The probability density function is $f(t) = \lambda e^{-\lambda t} dt$ for $t \geq 0$;
- The cumulative distribution function is $F(t) = 1 - e^{-\lambda t}$ for $t \geq 0$;
- The mean is $\mu = \frac{1}{\lambda}$.

Consider a process executing in a fault-prone environment. The time-steps at which fault strike are non-deterministic, meaning that they vary from one execution to another.

2. Using exponential distributions, create a model for the faults.

Based on the above modelization, a fault will strike every μ seconds. This is why μ is called the MTBF of the process, where MTBF stands for *Mean Time Between Faults*.

3. Show that the expected number of faults $Nfaults(T)$ that will strike during T seconds is such that

$$\lim_{T \to \infty} \frac{\mathsf{E}[Nfaults(T)]}{T} = \frac{1}{\mu} \tag{1}$$

Why are Exponential distribution laws so important? This is because of their *memoryless* property: if $X \sim Exp(\lambda)$, then $\mathsf{P}[X \geq t + s | X \geq s] = \mathsf{P}[X \geq t]$ for all $t, s \geq 0$. This means the delay until the next fault does not depend upon the time that has elapsed since the last fault i.e. the fault rate is constant.
We can now state the problem formally. Let $\textsc{Time}_{\text{base}}$ be the base time of the work that needs to be done, w/out overheads (checkpoints or faults). Assume Alice's laptop is subject to faults with a mean time between faults (MTBF) equal to μ.
The time to checkpoint is C seconds ($C = 180$ here). The period is $\textsc{Time}$ seconds; a checkpoint is taken after Alice has completed $\textsc{Time} - C$ seconds of work. When a fault occurs, all work after the last checkpoint is lost. After the fault, there is a *downtime* of D seconds to account for the temporary unavailability (for example Alice's laptop is restarted). Finally, in order to be able to resume the work, the content of the last checkpoint needs to be *recovered* which takes a time of R seconds. The sum of the time lost after the fault, of the downtime and of the recovery time is denoted T_{lost}. Let $\textsc{Time}_{\text{final}}(T)$ be the expectation of the total execution time of an application of size $\textsc{Time}_{\text{base}}$ with a checkpointing period of size T. The optimization problem is to find the period T minimizing $\textsc{Time}_{\text{final}}(T)$. However, for the sake of convenience, we rather aim at minimizing

$$\textsc{Waste}(T) = \frac{\textsc{Time}_{\text{final}}(T) - \textsc{Time}_{\text{base}}}{\textsc{Time}_{\text{final}}(T)}.$$

This objective is called the *waste* as it corresponds to the fraction of the execution time that does not contribute to the progress of the application (the time *wasted*).

4. Show that minimizing waste $\textsc{Waste}$ is equivalent to minimizing the total time $\textsc{Time}_{\text{final}}$

5. Calculate $\textsc{Time}_{\text{FF}}, \textsc{Waste}[FF]$, the execution time, and waste if Alice is lucky and no faults occur

Now let us consider the entire execution (with faults) of the application. Let $\textsc{Time}_{\text{final}}$ denote the expected execution time of the application in the presence of faults. This execution time can be divided into two parts: (i) the execution of chunks of work of size $T - C$ followed by their checkpoint; and (ii) the time lost due to the faults.

6. Write a formula for $\textsc{Time}_{\text{final}}$ using the above

In average, during a time $\textsc{Time}_{\text{final}}$, $Nfaults = \frac{\textsc{Time}_{\text{final}}}{\mu}$ faults happen. We need to estimate T_{lost}. A natural estimation for the moment when the fault strikes in the period is $\frac{T}{2}$. Intuitively, faults strike anywhere in the period, hence in average they strike in the middle of the period. We conclude that $T_{\text{lost}} = \frac{T}{2} + D + R$, because after each fault there is a downtime and a recovery. This leads to:

$$\textsc{Time}_{\text{final}} = \textsc{Time}_{\text{FF}} + \frac{\textsc{Time}_{\text{final}}}{\mu}\left(D + R + \frac{T}{2}\right).$$

Let $\textsc{Waste}$[fault] be the fraction of the total execution time that is lost because of faults:

$$\textsc{Waste}[fault] = \frac{\textsc{Time}_{\text{final}} - \textsc{Time}_{\text{FF}}}{\textsc{Time}_{\text{final}}} \Leftrightarrow (1 - \textsc{Waste}[fault])\,\textsc{Time}_{\text{final}} = \textsc{Time}_{\text{FF}}$$

We derive:

$$\textsc{Waste}[fault] = \frac{1}{\mu}\left(D + R + \frac{T}{2}\right). \tag{2}$$

Equations (**??**) and (2) show that each source of waste calls for a different period: a large period for $\textsc{Waste}[FF]$, as already discussed, but a small period for $\textsc{Waste}[fault]$, to decrease the amount of work to re-execute after each fault. We need a trade-off.

7. Express $\textsc{Waste}$ in terms of $\textsc{Waste}[FF]$ and $\textsc{Waste}[fault]$.

8. Replace $\textsc{Waste}[FF], \textsc{Waste}[fault]$ in the above formula of $\textsc{Waste}$.

Let $u = C\left(1 - \frac{D+R}{\mu}\right)$, $v = \frac{D+R-C/2}{\mu}$, and $w = \frac{1}{2\mu}$.
Then, $\textsc{Waste} = \frac{u}{T} + v + wT$ is minimized for $T = \sqrt{\frac{u}{w}}$. The First-Order (FO) formula for the optimal period is thus:

$$T_{\text{FO}} = \sqrt{2(\mu - (D+R))C}. \tag{3}$$

Ultimately, with some extra mathematics we can see that the optimal checkpointing period is $T_{\text{FO}} = \sqrt{2\mu C} + o(\sqrt{\mu})$.
We now look into another method to calculate the optimal T_{FO}.
First we show how to compute the expected time $\mathsf{E}(\textsc{Time}(T - C, C, D, R, \lambda))$ to execute a work of duration $T - C$ followed by a checkpoint of duration C, given the values of C, D, and R, and a fault distribution $Exp(\lambda)$.
We should show the proposition below

Proposition 1.

$$\mathsf{E}[\textsc{Time}(T - C, C, D, R, \lambda)] = e^{\lambda R}\left(\frac{1}{\lambda} + D\right)(e^{\lambda T} - 1).$$

For simplification, we write $\textsc{Time}$ instead of $\textsc{Time}(T - C, C, D, R, \lambda)$

9. Write a recursive relation for $\textsc{Time}$

10. Using the previous question, write $\mathsf{E}[TIME]$

11. Calculate $\mathsf{E}[\textsc{Time}_{\text{LOST}}]$

12. Compute $\mathsf{E}[\textsc{Time}_{\text{REC}}]$. Note that there can be no fault during D but there can be during R (Hint: reuse the equation for)

13. Plug $\mathsf{E}[\textsc{Time}_{\text{REC}}]$ and $\mathsf{E}[\textsc{Time}_{\text{LOST}}]$ to the equation for $\mathsf{E}[\textsc{Time}]$

Now we know the expected time for a sole T. Splitting the work into k checkpoints, we have $T = \frac{\textsc{Time}_{\text{BASE}}}{k}$. With some maths and the above proposition we can find T_{FO}.

14. Intuitively explain why it makes sense to use the same T for checkpointing, and not different periods $T_1, T_2, \ldots$

Exercice 2. *Connexe*

Un graphe non-orienté $G = (V, E)$ sur n sommets est *connexe* si, pour tous sommets u et v, il existe un chemin de u à v. Autrement dit, le graphe n'est pas connexe si l'on peut partitionner V en (A, B) de telle sorte qu'il n'existe aucune arête entre A et B.

1. Prouver que si $p = (2+\varepsilon)\log n/n$ avec $\varepsilon > 0$, alors la probabilité qu'un graphe choisi aléatoirement dans $G_{n,p}$ soit connexe tend vers 1 quand n tend vers l'infini.

Exercice 3. *Théorème de Mycielski*

Recall that the *chromatic number* $\chi(G)$ is the smallest number of colors needed to color the vertices of G such that any two adjacent vertices have different colors. Clearly, graphs with large cliques have a high chromatic number, but the opposite is not true. The goal of this exercise is th prove Mycielski's theorem, which states that for any integer $k \geq 2$, there exists a graph G such that G contains no triangles and $\chi(G) \geq k$.

1. Fix $0 < \varepsilon < \frac{1}{3}$ and let G be a random graph on n vertices where each edge appears independently with probability $p = n^{\varepsilon-1}$. Show that when n tends to infinity, the probability that G has more than $n/2$ triangles tends to 0.

2. Let $\alpha(G)$ be the size of the largest *independent set* of G (A set of vertices X is *independent* if there is no edge between any two vertices of X in G). Show that $\chi(G) \geq n/\alpha(G)$.

3. Let $a = 3n^{1-\varepsilon}\ln n$. Show that when n tends to infinity,

$$\mathbb{P}(\alpha(G) < a) \to 1.$$

Deduce that there exists n and G of size n such that G has at most $n/2$ triangles and $\alpha(G) < a$.

4. Let G be such a graph. Let G' be a graph obtained from G by removing a minimum number of of vertices so that G' does not contain any triangle. Show that

$$\chi(G') > \frac{n^{\varepsilon}}{6\ln n}$$

and conclude the proof of Mycielski's Theorem.