Assignment 2

Romain Bazin

In this report, I'll explain the guiding lines of my reasoning to model and solve the problem of the grocery store scheduling.

State of the problem

In my model, the state of the problem is the number of boxes of mangoes in store after receiving the order from the previous hour but before the customers can buy them, and the hour of the day.

This implies the state is directly impacted by the number of boxes sold during the previous hour, which evolves randomly following two possible distribution laws.

state = (X,t) ∈ $[0, M]$x$[0, 24]$

Number of states

The number of boxes in store is capped by the parameter M and the time is considered in between 0 and 24. Meaning, there are 24*M possible states.

Range of actions

In this model, I considered one type of possible action : ordering a boxes of mangoes.
$a \in [0, M]$ because ordering more than the maximum capacity of the store would have no point.

Transition model

The transition model is defined by $P(X_{t+1} \mid X_t, a_t) = P(X_{t+1} = X_t + a_t - K_t \mid X_t, a_t)$
Where K_t is the number of boxes sold during the hour t
So
$P(X_{t+1} = X_t + a_t - K_t \mid X_t, a_t) = P(K_t = X_t - X_{t+1} + a \mid X_t, a_t) = P(K_t = k), k \in [0, X]$

Or $P(K_t = k) = \sum_{n=0}^{N} P(K_t = k|C_t = n)P(C_t = n)$ where C_t is the number of customers in store at time t. This random variable follows a normal or poisson law as stated in the assignment.

The second term, $P(K_t = k|C_t = n)$ is equal to the probability of a binomial law :

$$Q(k) = \frac{n!}{k!(n-k)!} P^k (1-P)^{n-k}$$

Reward function

First, I consider as a reward how many boxes were sold compared to the stock of the hour, and the anticipated cost of the order arriving next hour :
R(X, K, a) = -E1 (if a > 0) -E2*a -E3(X-K) (if X > K) + S*min(K, X) - C(K-X) (if K > X)

- Fixed cost of ordering boxes
- Proportional cost of ordering boxes
- Cost of maintaining boxes in good shape if there are remaining
- Revenue on boxes sold
- Cost of unsatisfied customers

This value depends on K, the number of boxes sold, which means it is a random variable.

As such, I computed the expected value of the reward :

$$ER(X,\ a) = \sum_{k=0}^{M} R(X,\ K,\ a)P(K = k) = \sum_{k=0}^{M} R(X,\ K,\ a) \sum_{n=0}^{N} P(K = k|C = n)P(C = n)$$

Utility function

The final utility function, thanks to Bellman's equation, is :

$$U(X_t) = max_{a\in[0,\,M]}(ER(X_t, a) + \gamma * \sum_{X_{t+1}} P(X_{t+1}|X_t, a)U(X_{t+1}))$$

Where γ is the discount factor. And the other terms have to be replaced with what we defined earlier.

Comments on the program

The program is functional but has the major flaw that it's very long to run.
Computing the utility of one state takes the value_iteration function 1 second. To run one full iteration of the value iteration algorithm then takes 24*M seconds (which easily amounts to multiple minutes).

This could have been largely improved by parallelizing the computation of the utility of the different states from the same hour (which are independent), but I didn't have time to implement it. Also, maybe some more optimized and well thought modelization could increase the speed of the algorithm.

This issue made me unable to clearly see if my implementation could converge, but for some small values it seems it does.

For the same reason, I couldn't try multiple combinations of parameters to see the complexity of the algorithm. My only guess is that It would be something like $O(S * A)$ where $S = 24 * M$ is the size of the state space, and $A = M$ is the size of the action space.