1CM240

TU/e

AI in Logistics
Assignment 1 (50% of course grade)

Preliminary information

- This assignment should be performed in the groups with whom you registered in Canvas.
- Before each question, the total amount of points is indicated. In total there are 100 points to be earned in this assignment. Your final score is the number of points divided by 10.
- This is a challenge-based course, so the assignment balances between semi-open questions that require you to proactively think and make well-motivated choices for formulating Markov decision processes, and programming exercises.
- Mathematical notation is extremely important. Without proper notation no points will be awarded to each of the questions.
- Questions 5 and 6 require programming in python. Please hand in the python code. You need to explain the code in enough detail either within the code via comments or by decomposing it as if read through a jupyter notebook.
- Please do not forget to fill in the mandatory peer-review forms. Without the peer-review forms no grade can be awarded to this course.

The assignment

1. **[10 points]** The video below describes how robots operate in an Amazon warehouse:

   `https://www.youtube.com/watch?v=dLJ6gM4AQI4`.

   Write down the MDP for a warehouse, that is operated by one KIVA robot. The robot should learn how to navigate through the warehouse given a list of orders to fulfill.

2. **[15 points]** We consider a single online store (i.e., webshop) that sells a number of different products. Customers can be distinguished in different types. Each customer type j has a particular utility for item i according to the utility equation:

   $$u_{ij} = v_{ij} + \epsilon_j,$$

   This implies that there is some fixed utility for each combination of customer type i and item j, and some uncertainty around it according to $\epsilon_j \sim N(0, \sigma_j)$ (i.e,. normally distributed with mean 0 and standard deviation σ_j). Note that the uncertainty only depends on the customer type.

   The interpretation of this utility model is simple. It basically implies that the utility for item i of customer type j is normally distributed with mean v_{ij} and some variance σ_j.

   The number of customers of each type j that visit the store follow a known poisson distribution with parameter λ_j. Each customer will always select the product with the highest **realized** utility. If that product is not available, the customer selects the product with the second highest

realized utility. If that product is not available, the customer selects the product with the third highest utility. If that item is not available, the customer is lost. Each time a customer enters the webshop, the owner of the webshop can potentially remove items from being on display. We assume that as soon as the customer arrives on the webshop we know its type but not its realized utilities. Each item has a different profit margin.

Each day within the time horizon, the store can order new items at a fixed time in the morning. All items are supplied by the same supplier. This supplier asks for a relatively large fixed cost if an order is placed (unspecific of the items), and a fixed cost for ordering the individual item (not a variable cost!!). The items are not directly available, it takes a particular lead time before they arrive in the store.

You realize directly that this is an amazing example of a stochastic sequential decision-making process that concerns both determining assortments and reordering items. Formulate an MDP that determines assortment in *real-time* and orders new items. You need to introduce mathematical notation and terms for all the assumptions outlined in the story above. Furthermore, you are challenged to introduce the relevant cost parameters in order to retrieve a sensible MDP model, so that the model can serve as a starting point for providing decision support. Motivate your choices.

3. **[10 points]** In a real-world application of the MDP formulated in (2) you often deal with a large amount of customer types and items. The MDP can then not easily be solved to optimality. One approach can be to solve model (2) heuristically via reinforcement learning. However, we do not want to go down that road (yet). Another approach is to reformulate the MDP from (2) to a simpler MDP. This MDP may not be a correct representation anymore of the real-world setting, but a simplified version. Follow this approach and reformulate your MDP from (2). To make your setting clear you can set the amount of customer types and demand parameters to explicit values. You need to motivate your choices and assumptions, and clearly explain why this will benefit the webshop and the decision-making process associated with the assortment and/or replenishment.

4. **[20 points]**. We consider a single sourcing system. It concerns a single producer of steel end products in Germany. For this, the producer needs to import raw steel from Poland. The producer can only order 50 units at a cost of 1 per unit, or it can order 20 units at a cost of 1.20 per unit. The capacity of its inventory is 50. If the producer orders the product, it needs to wait 2 weeks before it arrives. The holding costs per unit are equal to 1, and the lost sales costs equal 20. The demand distribution per week equals:

Demand	10	30
Probability	0.25	0.75

The sequence of events is as follows: At the start of a period (a week), a new order can be placed. Afterward, demand for the week is observed and served. Then, the order ordered 2 weeks ago arrives, meaning the order placed at the start of the previous week.

Thus, the state of at the start of the period system is tuple $S_t = (I_t, O_t)$, describing the current inventory level I_t, the order made at the beginning of the the previous period O_t.

(a) Write down the state-value functions and the optimality equations (i.e., the Bellman equation). Note that it might be very useful to first formulate the MDP.

The producer has a simple policy. It orders 50 units as soon as its inventory position $(I_t + O_t)$ equals 0.

(b) The cost of the optimal policy can be obtained via the policy evaluation algorithm. Write out two iterations of the policy evaluation algorithm. Ensure the logic is clear.

(c) Assume that the policy evaluation algorithm converged after the two iterations. Proof that this policy is not optimal by applying the policy improvement algorithm.

(d) The above two steps can be seen as some form of policy iteration. Explain what would have been the procedure if value iteration was chosen as the method to solve this problem.

5. **[30 points - coding exercise]** You are the logistic manager of a large supermarket chain network in the Netherlands. As such you are responsible for the transport of goods to the supermarkets. You are in complete control of the trucks that deliver from the depots to the supermarkets. That means, you decide when a truck goes to supermarket, and the items and quantities that are on the truck. You have a smart employee, Mrs. Slim that already created a division of the complex network you normally have to handle and optimize. One division of the divided network contains three supermarkets, two in the south of Eindhoven and one close to the university in the center. The demand (in pallets) is distributed according to a Gamma distribution with

Index	Location	Street	Demand	
			Mean μ_i	Variance σ_i^2
1	South	Alsterweg	3	1
2	South	Bosranklaan	5	2
3	Center	Stationsplein	2	3

Demand is rounded up to full pallets. You need to plan the replenishment of the stores. When you replenish stores you have to pay a fixed cost and a route-dependent transportation cost. The sum of both costs are given in this table (in € per replenishment):

Route	(1)	(2)	(3)	(1,2)	(1,3)	(2,3)	(1,2,3)
Costs	40	40	55	60	75	75	500

If you fail to supply the right quantity to a store and the store stocks out, you have to pay a loss sales penalty of 19€ pallet. For every unit left at a store after sales of the day have ended, you need to pay a holding costs of 1€ per day and pallet. You assume that demand happens over the day and the replenishment happens at the start of the day. Demand realizes in units of pallets. You can assume demand is at most 10, and that at most 10 items can be in stock in each store. Upon replenishment, items are immediately put on shelves and are ready for sale. The specific details are given in the supplied python program.

(a) Formally write down the rewards, states and actions as part of the Markov decision process.

(b) Assume that every store i is replenished every day up to an inventory level of $I_i = \mu_i + 1.97\sigma_i$. Evaluate this policy using the policy evaluation algorithm. Explain the steps in your programming code. Assume the discount factor equals 0.8 and properly discretize the demand distribution and state space.

(c) Apply a single step of the policy improvement algorithm on the evaluated base-stock policy in (b). Show the steps in your programming code. Explain why solving this problem to optimality will become very challenging.

(d) Train tabular Q-Learning on this problem. Compare your rewards with the base-stock policy proposed in (b).

(e) Analyze the outcome of your Q-Learning algorithm. Give a brief intuition on the decisions proposed by the algorithm (or lay out the differences with your intuition).

6. **[15 points - coding exercise]** Take the problem described in (5) and solve it the best you can. You can make changes to the problem and investigate its impact on the solution.