Exercise 2: A Reactive Agent for the Pickup and Delivery Problem

Group 42: Niederhauser Thibault, Stefanini Niccolò

October 6, 2020

1 Problem Representation

1.1 Representation Description

A state is represented by the agent being in a specific city, which is supposed to be enough information to allow a complete coherent space representation for the Reactive Agent. Therefore, given a topology with N cities, there are N possible states and we note s_i the state corresponding to the city i.
The action space is composed by the 2 following actions: a_0 = "ignore" a task or a_1 = "pick-up" a task. In case there is no task available in a city, the only possible action is a_0. The action a_0 results in the agent moving to a neighbouring city, whereas if the action a_1 is taken, the agent picks up the available task and moves to the delivery city.
Given an agent in a specific state s_i (= in a specific city i), if it takes action a_1 = "pick-up", the probability to end up in state s'_j (= city j) is given by $p(i, j)$, which is the probability that there is a task for city j in the city i. While if the agent takes action a_0 = "ignore", the probability to end up in a non-neighbouring city is zero and the probability to end-up in a neighbouring city j is 1 divided by the number of neighbours (other policies can be applyed as greedy -¿ going to the best or a softmax of neighbors values). For the baseline, one can write:

$$T(a_0, s_i, s'_j) = \begin{cases} \frac{1}{N_{neigh}(i)} & \text{if city } j \text{ neighbour of city } i \\ 0 & else \end{cases} \tag{1}$$

$$T(a_1, s_i, s'_j) = p(i, j) \tag{2}$$

For a state s and an action a, the reward $R(s, a)$ is defined by the expected rewards minus the travel costs averaged over all the possible destination cities. The possible destination cities are all cities (except for the current one) in case of a_1 and all neighbouring cities in case of a_0:

$$R(s_i, a_o) = -\sum_{j \in Neigh(i)} \frac{1}{N_{neigh}(i)} C(i, j) \tag{3}$$

$$R(s_i, a_1) = \sum_{j \neq i} p(i, j)(E[r(i, j)] - C(i, j)) \tag{4}$$

where $E[r(i, j)]$ are the expected rewards for a task from city i to city j and $C(i, j)$ is the travel cost from city i to city j.

1.2 Implementation Details

During offline learning, the task probabilities $p(i, j)$ as well as the expected rewards $E[r(i, j)]$ are retrieved from the `TaskDistribution` class. The travel costs $C(i, j)$ are calculated by multiplying the distances between cities i and j by a cost per kilometer, which is considered to be known and constant for each vehicle. Those values are then plugged into equation 1, 2, 3 and 4, which allows to compute the Q-values. The offline phase is stopped when the relative change between the updated V-value and the former V-value gets smaller than 1% for all states.
During the online phase, the best actions are chosen based on the learnt V-values. If a task is available, the agent takes action a_1 = "pick-up" only if: $r_{task} + \alpha V(s_{dest}) > V^*(s')$,
where r_{task} is the reward of the task, α is a discount factor (set to 0.95 for the experiments), $V(s_{dest})$ is the V-value of the destination city and $V^*(s')$ is the maximal V-value of all the neighbouring cities.
If not satisfied or if no task is available, the agent moves to the neighbouring city with the highest V-value.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

We run a simulation with 5 different reactive agents with discounts factors (γ) 0.1, 0.75, 0.85, 0.95, 0.99 and the following settings:

Topology	Vehicles speed	Cost per km	Task distribution	Reward distribution	Reward policy
Netherlands	90	5	Uniform [0, 1]	Constant [100, 99999]	long-distances

2.1.2 Observations

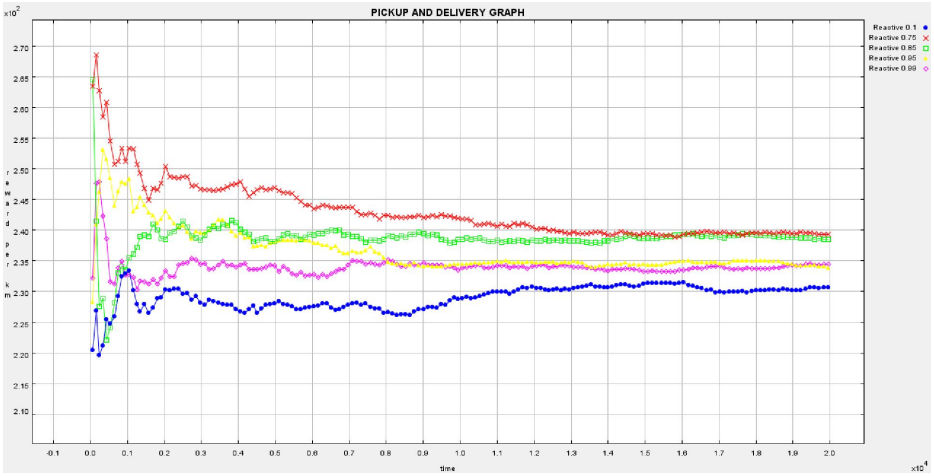

Figure 1: Reward per km

Discount factor	0.1	0.75	0.85	0.95	0.99
Average Profit ($\times 10^3$)	28.5	28.2	28.7	28.1	28.1

Table 1: Average profits

Table 2.2.1 shows that the choice of the discount factor affects the performance.The best average profit is achieved for $\gamma = 0.85$ and the worst for $\gamma = 0.1$, which means that taking future states into

account with a fairly high factor optimize the agent's behaviour. However there are still some uncertainty in future states and if the discount factor is too high, the profits drop slightly.
Our implementation tries to optimize the average profit, but Figure 1 shows that reward per kilometer is also better for agents that have a high profit. This can be explained by the fact that the optimization of the profit takes into account both the expected reward and the cost per km.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

The dummy agent is an agent that always accept an available task and move randomly to a neighbouring city if there is no task. We run 4 simulations on 4 different topologies: France, the Netherlands, Switzerland and England. For each topology, 2 intelligent agents with discount factors 0.75 and 0.85. We choose the following settings:

Vehicles speed	**Cost per km**	**Task distribution**	**Reward distribution**	**Reward policy**
90	5	Uniform [0, 1]	Constant [100, 99999]	short-distances

2.2.2 Observations

	Random	Dummy	RL agent 0.75	RL agent 0.85
Avg. Profit FR ($\times 10^3$)	31.1	37.1	38.6	38.9
Avg. Profit NL ($\times 10^3$)	39.0	42.1	44.7	44.8
Avg. Profit CH ($\times 10^3$)	37.8	43.7	45.8	46.1
Avg. Profit EN ($\times 10^3$)	37.3	43.3	45.0	45.1

Table 2: Average profits for different topologies

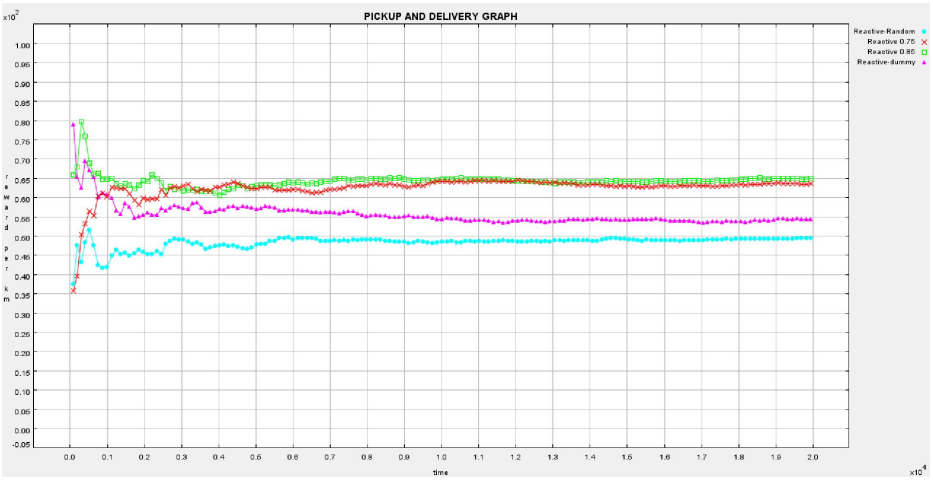

Figure 2: Reward per km, France

Table 2 and Figure 2 show that the RL agents perform better than the dummy and random agent. Hence, always accepting a task is not optimal. In some situations, ignoring a task and moving to a city with more profitable tasks is a better option and this behavior has been learnt by the RL agents. For this to work, it is important that the RL agent do not move randomly to a neighbouring city when a task is ignored or not available, bit that they move to the city with the best V-value.