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

Group №1: Célia Benquet - 271518, Artur Jesslen - 270642

October 6, 2020

1 Problem Representation

1.1 Representation Description

To design a state representation, we created the class **State**. The first attribute corresponds to the city where the agent is located at the current time step. The second attribute contains the next visited city. This second attribute is nullable. In that case, it means that no task is available in the current city. On the opposite, if the attribute contains a city, a task is available and this city corresponds to the destination city.

In these scenarii, the possible actions are either to find the most rewarding neighboring city (when no task is available) or to decide, depending on the attractiveness of a task, to take care of a task or to go to a neighboring city.

The reward for a state consists in the reward for the given task (or 0 if the action is different from taking care of a task) minus the cost of the displacement to go to the destination city. For its part, the probability transition is the probability that, knowing the current location of the agent and the available task (current state), it takes a given action to arrive in another city with a new available task (another state).

State attributes	Actions
Current city	Move to a neighboring city (with or without available task)
Next city or *null*	Take care of an available task

Table 1: Range of the different states attributes and actions of an agent.

1.2 Implementation Details

The reinforcement learning process takes place offline. It is implemented in the **learnValue** method. The rewards were not stored in a proper table but rather computed through the **computeReward** method, which is invoked when needed in the value iteration (in **learnValue**) algorithm. Similarly, for the transition states, we computed it directly during the learning process, through the **computeTransitionProba** method. This last method implements the sum $\sum_{s' \in S} T(s, a, s') \times V(s')$, where $T(s, a, s')$ is just the provided table $p(i, j)$.

We stored the best Q-values and best state transitions in the **bestValueForState** and **bestStateForState** HashMap respectively. They are updated through the values iteration and used by the agent to decide on which action to take during the simulation, depending on the state.

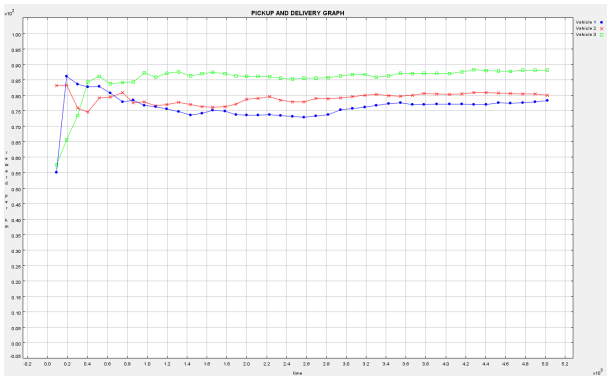

Figure 1: Evolution of the reward per km through time for 3 agents with different discount rate: 0.1, 0.5, 0.95 respectively for Vehicle 1, 2 and 3.

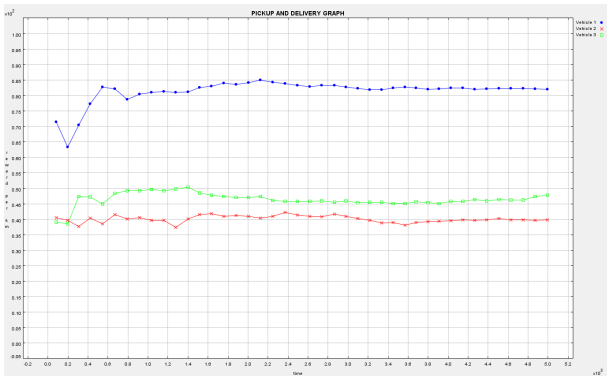

Figure 2: Evolution of the reward per km through time for 1 reactive agent (Vehicle 1) compared to 2 dummy agents of respectively 10 (Vehicle 2) and 10 (Vehicle 3) as cost per km travelled.

2 Results

2.1 Experiment 1: Discount factor

We implemented 3 reactive agents with 3 different discount factors: 0.1, 0.5 and 0.95 for vehicles 1 (Blue), 2 (Red) and 3 (Green), respectively. Other parameters were unchanged. Results are shown on Figure 1. One can see that the higher is the discount factor, the higher is the resultant reward. Hence, Vehicle 3 (Green) achieved a higher reward per km than the 2 others. As the discount factor allows to make the importance of future reward vary, an agent with a higher discount factor takes the future rewards more in consideration and is able to anticipate the optimal actions better.

2.2 Experiment 2: Comparisons with dummy agents

The reactive agent (Vehicle 1, Blue) is set with the default parameters. We implemented the random agents in order to assert the effect of the cost per travelled kilometers. Consequently, the first dummy (Vehicle 2, Red) has a cost per travelled km of 10 while the second one (Vehicle 3, Green) still has the default values of 5. Other parameters were kept unchanged with respect to the default values. Results are presented on Figure 2. We can see that our reactive agent is clearly performing better than the 2 dummy agents. It makes sense and validate the reinforcement learning that we implemented through the values iteration algorithm. Moreover, we observe that, as expected, the agent with a higher cost per travelled km (Vehicle 2, Red) has a lower reward per km. They both have a random behavior and pick up tasks with the same probability but Vehicle 2 (Red) pays more for each displacement. It is then logical that this last one doesn't perform as well as Vehicle 3 (Green).

2.3 Experiment 3: Reward Rate

2.3.1 Topology 1 - France

Using the reward rate parameter, it is possible to control the percentage of the initial reward attributed to each available task. Thus, the new reward is calculated as follow:

$$newReward = rewardRate \cdot initialReward$$

. For 4 different reactive agents, we set the reward rate to 1.0, 0.5, 0.1, 0.01 as illustrated with respectively Vehicles 1 (Blue), 2 (Red), 3 (Green) and 4 (Magenta) on Figure 3. Other parameters were kept unchanged

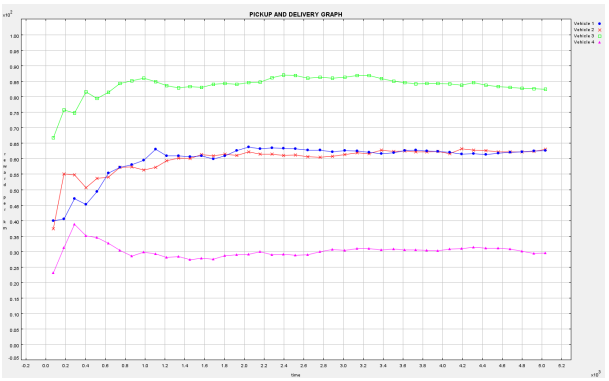

Figure 3: Evolution of the reward per km though time for 4 reactive agents (Vehicles 1-4). The values for the reward rate for Vehicle 1 to 4, represented in Blue, Red, Green and Magenta, are set to 1.0, 0.5, 0.1 and 0.01, respectively. The world topology corresponds to France.

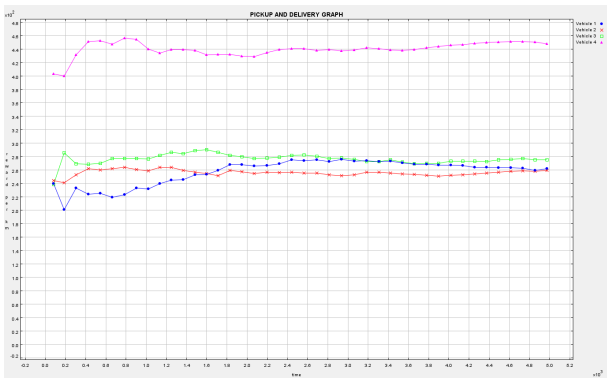

Figure 4: Evolution of the reward per km though time for 4 reactive agents (Vehicles 1-4). The values for the reward rate for Vehicle 1 to 4, represented in Blue, Red, Green and Magenta, are set to 1.0, 0.5, 0.1 and 0.01, respectively. The world topology corresponds to England.

to the default values.

The results show that Vehicle 3 (Green) performs very well while Vehicle 4 (Magenta) performs badly and Vehicle 1 and 2 (Blue and Red) have average performances. The great score (Vehicle 3, Green) shows that a good balance between the cost and the reward leads to better results. Indeed, decisions taken by Vehicles 1 (Blue) and 2 (Red) rely mostly on reward offered by available tasks and Vehicle 4's decisions depend more on the covered distance. Hence, most of the time, they either choose to take the task (Vehicles 1 and 2) or to deny it (Vehicle 4) leading to worse performances than Vehicle 3.

2.3.2 Topology 2 - England

All parameters for this experiment are similar to the parameters in section 2.3.1. The only new parameter is a world with a different topology. The first topology (France) was composed more or less of equidistant cities. Only Marseille and Monaco were closer to each other, which induced the bad performance of Vehicle 4 (Magenta) in the previous experiment. This different topology (England) is composed of closer cities in the center and further peripherally cities. The results found in the previous experiment are still valid except for Vehicle 4 (Magenta) (see Figure 4). Indeed, as this vehicle prefers to lower the covered distance rather than accepting new tasks, it refuses to pick up tasks that are far from the center, which maximizes its score.