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

Group №17 : Gervaise Lara, Khatib Karim

October 8, 2019

1 Problem Representation

1.1 Representation Description

A reactive agent should have a clear view of where to go in every state it is in. The **Table 1** below illustrates our well-defined states and actions. Take note that the lines of state and action are totally independent. Afterwards, the reward is based on the reward of the task minus the cost of travelling from the start city to the destination.

State	Task or no task	Current city	Destination city if there is a task
Action	Move to another city if no task is available	Ignore an available task and move to a neighboring city	Deliver the available task if any

Table 1: Defining criteria for states and actions

1.2 Implementation Details

The implementation of the above-mentioned model representation is illustrated in **Figure 1**. We start by finding all the states, and then list all the possible actions in every state while also taking into account the probability of every state.
After that, we execute the following reinforcement algorithm:

```
1: repeat
2:     for city ∈ all cities
3:         for state ∈ all states in city
4:             for action ∈ all actions in state
5:                 $Q(state, action) = R(state, action) + \gamma * \Sigma_{state' \in States} T(state, action, state') * V(state')$
6:             end for
7:         $V(state) = max_a Q(state, action)$
8:         end for
9:     end for
10: until good enough
```

where $R(state, action)$ is the reward for taking a given action in a given state, γ is a discount factor chosen to be between 0 and 1, $T(state, action, state')$ is the probability to get to state' by taking a given action from a given state and $V(state')$ is the value of state'.

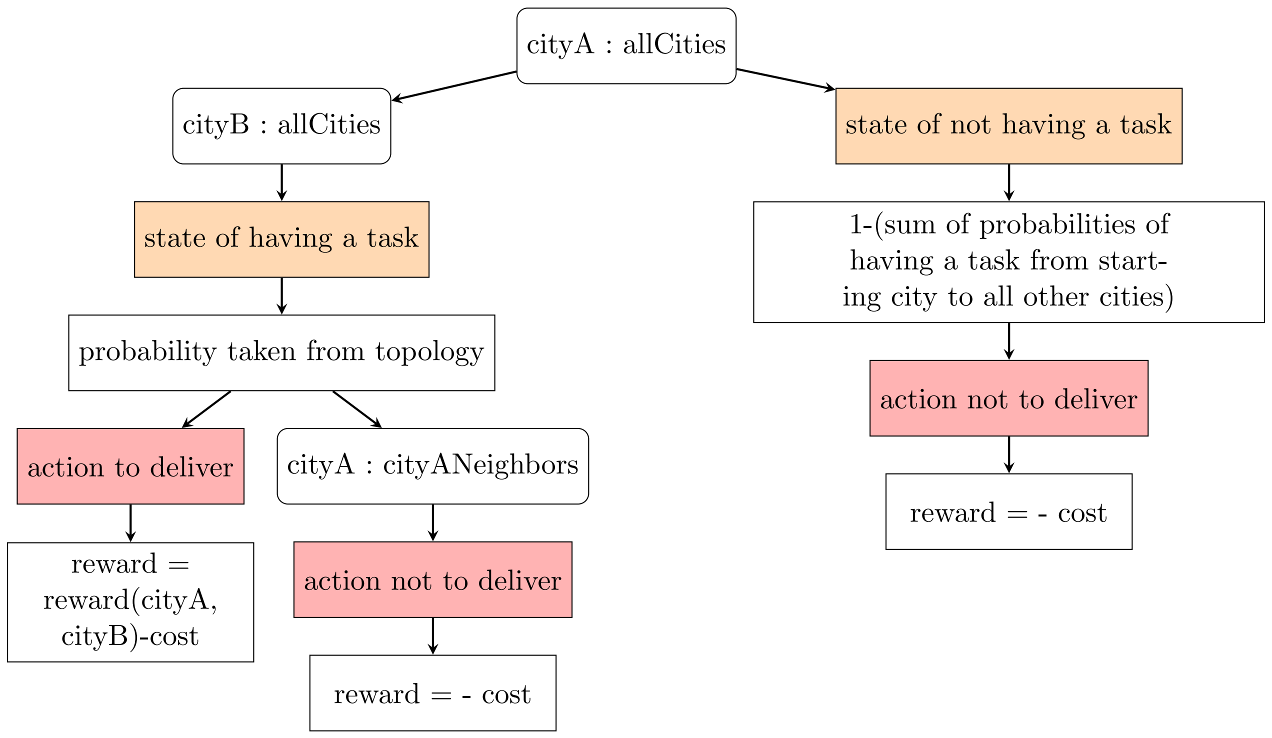
cityA : allCities
cityB : allCities
state of not having a task
state of having a task
1-(sum of probabilities of having a task from starting city to all other cities)
probability taken from topology
action not to deliver
action to deliver
cityA : cityANeighbors
reward = - cost
reward = reward(cityA, cityB)-cost
action not to deliver
reward = - cost

Figure 1: Implementation illustration

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

3 reactive agents in Switzerland with same speed (220), capacity (30) and cost/km (5) but different discount factors : blue=0.15, red=0.55 and green=0.95.

2.1.2 Observations

Figure 2 shows the simulation results. As time goes by, we can see that the agent with a discount factor of 0.95 (green) has the highest average reward/km while the agent with the lowest discount factor (blue) has the lowest reward/km. We should also take into account that the number of iterations in the Markov Decision Process algorithm increases with the discount rate. It takes more time to converge.

Finally, when the discount factor is larger, the agent cares more about future. Thus, it will refuse tasks that will lead to future states in which the probability of good tasks showing up is very low. Instead, it may choose a path where tasks seems less promising at this moment, but it will have higher potential to discover well rewarding tasks along this path. And agents with lower discount factor tend to pick up more available tasks and care less about future state value. Different discount factors lead to different actions and strategies, thus lead to different expected rewards. Here the best strategy is to have a high discount factor (close to 1).

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

2 dummy agents in Switzerland. The yellow one is the random agent given in the starter file and the magenta is a dummy agent with a speed of 220, a capacity of and cost/km (like the random agent) and

a discount factor of 0.35.

2.2.2 Observations

On the same plot (**Figure 2**) we also have the results of this experiment. We can see that even the best dummy agent (the yellow one) that picks up available task randomly with probability of 0.85 has a reward/km lower than the worst reactive agent. This verify that our reinforcement learning algorithm performs well.

Also, obviously, the modification of the pick up probability has a big effect on a dummy agent : the higher it is, the more tasks will be picked up and the more reward will the agent earn.

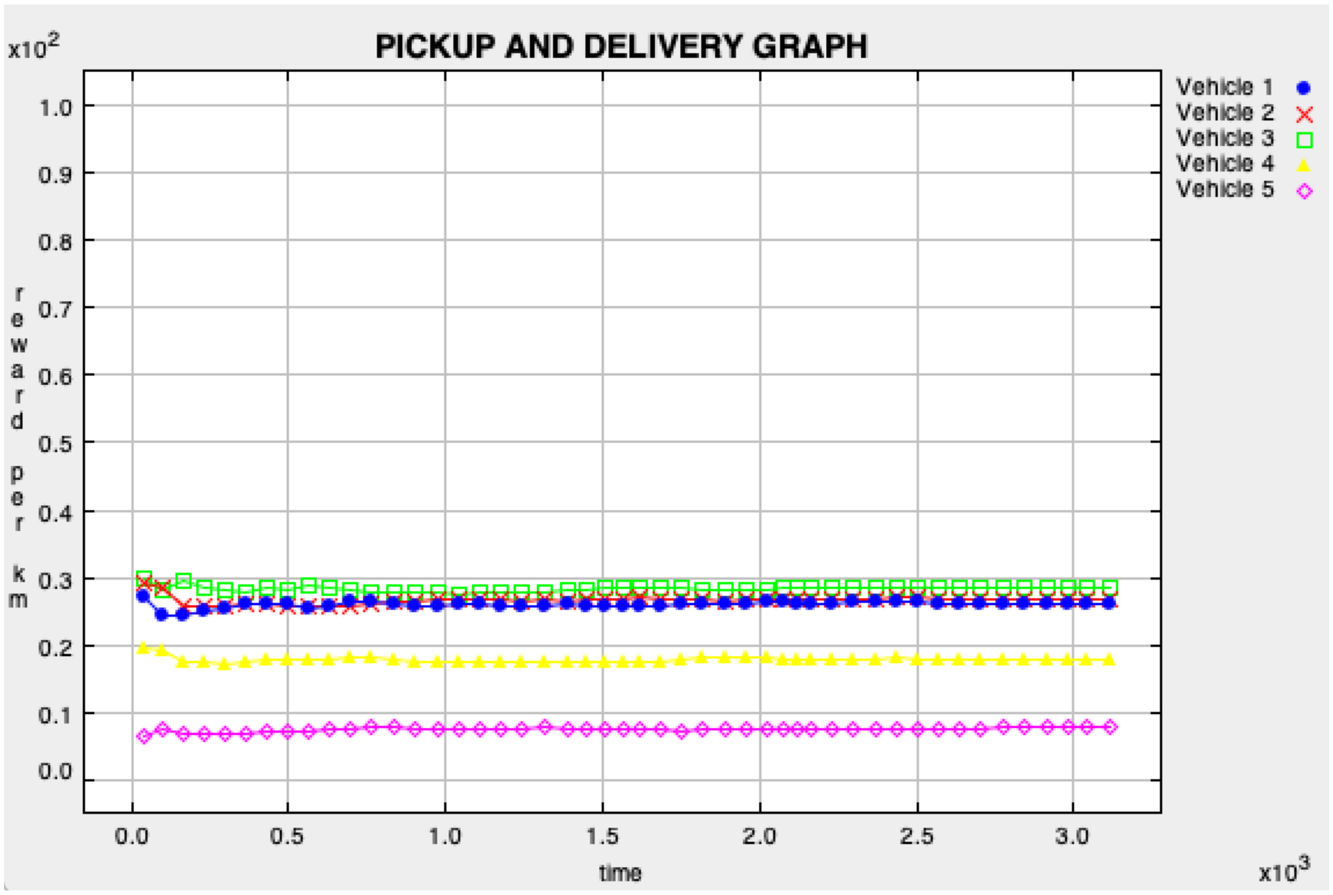

Figure 2: Reward/km of the different agents