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

Group №87: Ali El Abridi, Tariq Kalim

October 8, 2019

1 Problem Representation

1.1 Representation Description

1.1.1 State representation:

States are represented as the combination of the current city the agent is in and the available task that the agent sees in the current city.

1.1.2 Possible actions

When a task is available, the agent can choose to accept it or not. The agent can only accept a task from its current city.

If accepted, the agent will go directly to the destination city of the task without staying in any city in its path.

If refused, the agent has to choose which neighboring city to move to, the agent cannot stay in the same city.

1.1.3 Reward table

The reward table contains for each state and action the associated reward. The reward can be negative when the agent refuses a task. ($reward = payoff - cost$)

1.1.4 Probability transition table

The probability transition table $T(S'|S, a)$ holds the probability to end up in a state S' when the agent is in a state S and chooses an action a

1.2 Implementation Details

1.2.1 States

States are implemented as objects of the class `State` which is defined with two properties: `city` and `task` which are both objects of the class `City`. `city` is the current city of the agent and `task` is the delivery city of the available task (`null` when there is no available task)

1.2.2 Actions

Actions are implemented as objects of the class `City`, as they are the next city the agents will go to. The distinction between the acceptance and refusal of a task is done at the execution moment, if the `action` city is the same as the delivery city of a state (`task` property defined above), then the agent has accepted said task, otherwise meaning he refused

1.2.3 Reward table

The reward table is pre-computed and stored in a `HashMap`, where the key is an object of the class `StateAction`. `StateAction` is a class defined by a state and an action

1.2.4 Probability transition table

The probabilities are stored in a `HashMap` where the key is an object of the clas `ActionStateActionP`. `ActionStateActionP` is defined by two states (S and S') and an action. The probability can be 0, if for example the transition is illegal (i.e staying in the same city)

1.2.5 Reinforcement Learning Algorithm

For the RLA, we run the algorithm as long as $V(S)$ is changing. The algorithm basically loops for each city and each action computing the Q-value for that particular state and action (which depends on $V(s')$, s' being other states). The best Q-value for that state is used to update $V(S)$

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

We perform the experience with 6 different agents, each one with its one discount factor, the discount factors are as follow:
Vehicle 1: 0.85; Vehicle 2: 0.95; Vehicle 3: 0.7; Vehicle 4: 0.6; Vehicle 5: 0.5; Vehicle 6: 0.4

2.1.2 Observations

We can see on the that even though at the beginning the average rewards are different, all vehicles end close to each others. A discount factor of 0.85 has a slight advantage though. In conclusion, as long as $0 < discount factor < 1$, the agents will converge to nearly the same outcome (In terms of reward per Km).

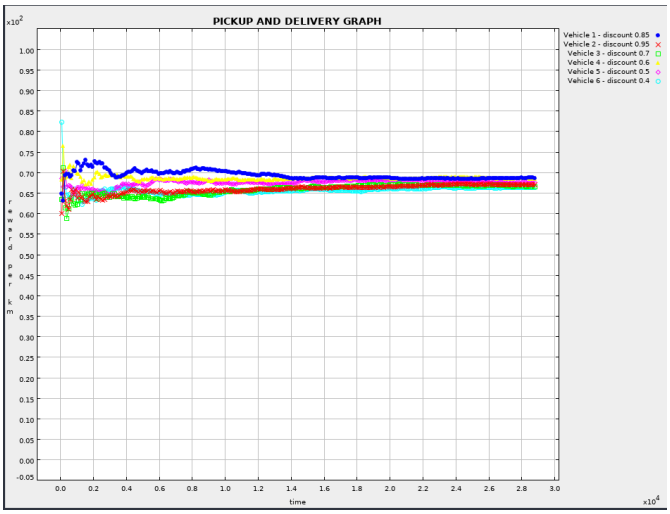

Figure 1: Reward per Km for agents with different discount factors

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

For this experiment, we use 3 agents with the same discount factor (0.85):

Trained agent : Agent trained through the offline reinforcement learning

Random agent : Agent that chooses to accept a task or not in a random manner, also chooses a neighboring city randomly

Dummy agent : agent that always accept tasks if they are available, and chooses randomly the neighboring city to go to when there is no available task

2.2.2 Observations

The trained agent largely outperforms its dummy counterparts, which is a good thing, meaning that the reinforcement learning is effective. As for the random and dummy agents, the dummy agents seems to have a slight edge, which could be explained by the fact that it never refuses a task.

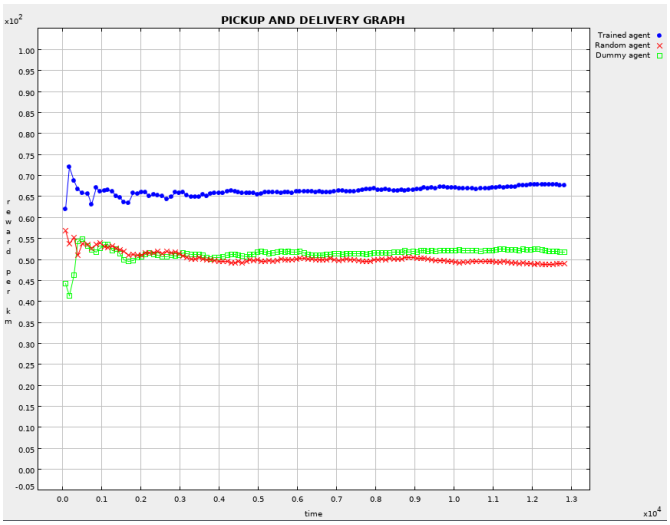

Figure 2: Reward per Km for different types of agents