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

Group №: 29: 298032, 293330

October 8, 2019

1 Problem Representation

1.1 Representation Description

We consider the **state** (from the viewpoint of a single agent) to be the tuple of `(current city, available task)`. Whereas `current city` is the current city of the a given agent (or his only vehicle to be precise), and `available task` is the available task, which can be nothing. Based on this information the agent makes his next move.

For each of these states there are two types of possible **actions**: a) pick up the package (of which there can be zero or more actions), and b) move to and adjacent city. Assuming we have in total n cities and city A has only one neighbour, city A has $n-1$ possible package-destinations and 1 moving-destinations, resulting in a total of $(n-1)+1=n$ actions.

The **reward table** needs to consider the state and the taken action, it is thus a function (in the mathematical sense) that maps to a value, the long-term expected benefit (which is the reward minus the cost of travelling). The **probability transition table** is computed on the fly.

1.2 Implementation Details

The **state** is implemented as a class. The case that no package was available upon arrival in the city is handled by a `null` reference in the destination city. Since each state knows its origin city, it can generate the actions that can be done starting from this city. The **actions** are implemented as a class with a `type` (internally as a `Enum`, no generics nor traits). This class is not much more than a POJO. The other tables such as *value table*, *Q-table* and **reward table** are implemented as `HashMap`. Linking a state to a reward, resp. a state to a action-reward tuple.

In order to simplify the calculations and enforce correctness every value given by the framework is converted to a `double`. The weights of each package was ignored, as the agent can only carry one package at a time and the weight seemed to be low enough to not have any influence on the fuel consumption and thus cost-of-travel.

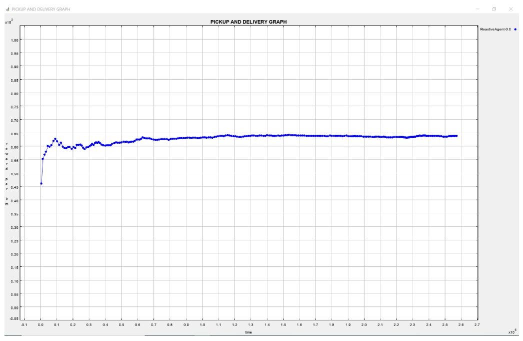
Upon arrival at a new city each agent considers its current state, looks up the recommended action, compares its benefits to the to-be-gained benefit from picking up the package (if any). If the to-be-gained benefit is larger or equals to the recommended action (which might be a move), the agent takes the sure gain and picks up the package.

2 Results

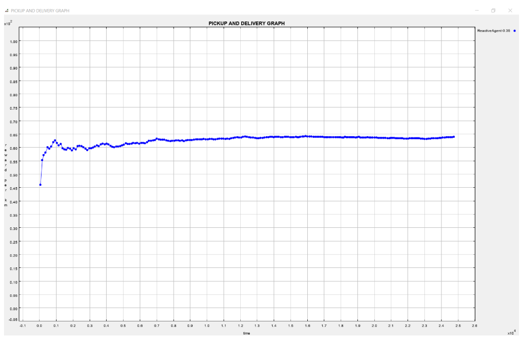
2.1 Experiment 1: Discount factor

2.1.1 Setting

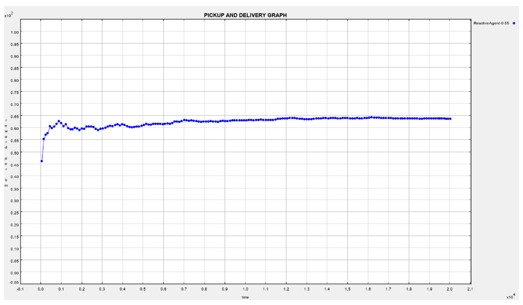
To experiment with the effect of the different discount factors, we have simulated by setting discount factors equals to 0, 0.35, 0.55, 0.95 respectively.



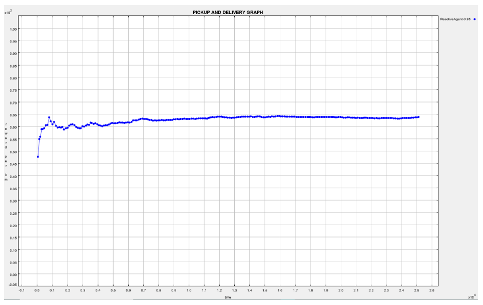
(a) discountFactor=0



(b) discountFactor=0.35



(c) discountFactor=0.55



(d) discountFactor=0.95

Figure 1: Average rewards of four Reactive Agents with different discount factor

2.1.2 Observations

Figure 1 shows the reward per kilometer of four reactive agents with different discount factors. The curve of average rewards could converge to a stable value in all situations, even when discount factor equals to 0 (as shown in (a)). However, the converged average rewards harvested by these four agents are almost the same. This result contradicts to our expectations. Basically, the agent will gain higher rewards when it was assigned with a bigger discount factor. This is because the agent will do more calculations to peek into the more promising future. It will learn to refuse some task temporarily, and then turn to the direction that promises it higher accumulated reward in the future, but instead turn to the direction that benefits it with immediate reward.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

To make a clear comparison between the Dummy Agent and Reactive Agent, we run the simulation of two agents (random agent and reactive agent) with the same discount factor (0.95).

2.2.2 Observations

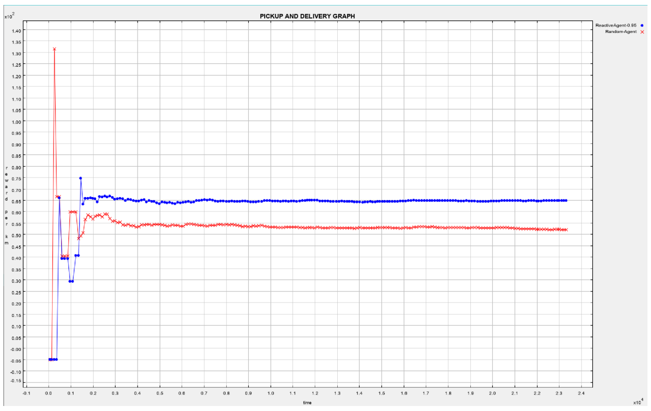


Figure 2: The performance comparison between Reactive Agent (Blue) and Random Agent (Red)

As shown in Figure 2, our reactive agent outperforms the random agent significantly. The average rewards harvested by our reactive agent still converges to 0.65, the random agent could only get 0.53 units of reward per kilometer.