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

Group №25: Goullet, Grondier

October 10, 2017

1 Problem Representation

1.1 Representation Description

A state is represented by our class RLState: it has an obligatory attribute describing the current city and two other optional describing if it has a pickup task and the destination of the pickup task. There are N^2 possible states: one for each of the N cities where no task is available, and $N-1$ per city for each of the possible task destinations.

The possible actions is a mapping from each state to each possible RLAction originating from the current city. There are $2N$ actions originating from each city, although some are invalid (see 1.2.2).

A reward is associated to each action as each action always has the same reward. We get the reward for deliveries from Task Distribution, and deduct the cost of traveling by knowing the distance between cities and the cost per kilometer of movement.

The probability transition table is generated using the probability of a pickup with each city as a destination appearing at the destination city.

1.2 Implementation Details

1.2.1 RLState

RLState is our class representing a state. A state has a mandatory *currentCity* attribute. It also has a boolean attribute $hasTask$ to describe if the city has a task ready to be picked up, and am associated destination city for the task *destinationCity* . When building our list of possible states, we ignore impossible actions, such as a task with destination the source city.

1.2.2 RLAction

RLAction is our super-class representing an action. An action can be a move (RLMove) or a pickup (RLPickup). We build an action list corresponding to each state. Again, we

ignore impossible actions or actions refused by logist, such as moving to a non-neighbouring city, the origin city, or delivering to the origin city.

1.2.3 The Reward Table

The reward table is a Map of the following form: $RLState \mapsto List(RLAction, Reward)$.

While iterating on our list of RLState, we create all RLAction possible from this state, then we associate with it the reward. We substract from the pickup reward the travel cost ($costPerKm * distance$), and simple moves are just a net loss of the travel cost.

1.2.4 The Probability Transition Table

The probability transition table does not need to be built before applying the algorithm: we can just deduce it from the probability table given by the Task Distribution in the setup.
This is because we only consider states whose *currentCity* attribute is the destination of the RLAction we are looking up the table for.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

All figures have seed 42 and using the settings reactive.xml, but we also tested with other seeds, with comparable results. We tested with the following discount values: 0, 0.5, 0.85, and 0.99.

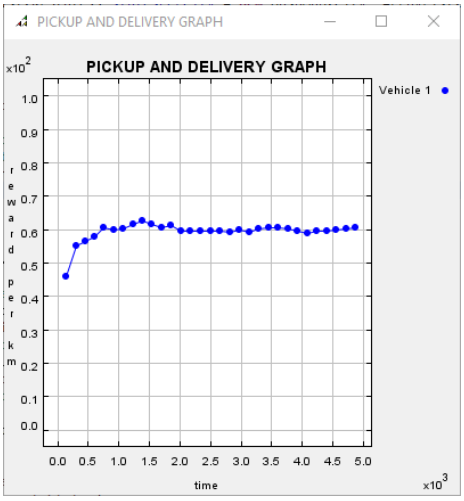

Figure 1: Discount = 0

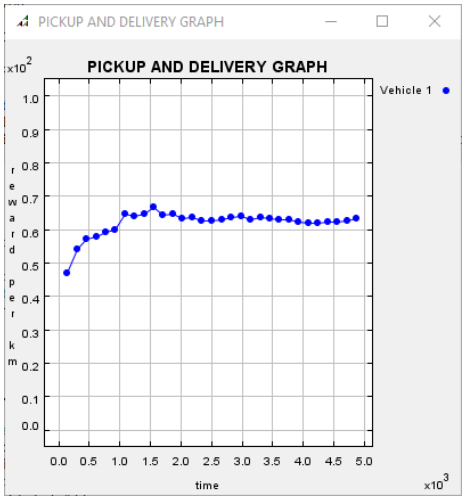

Figure 2: Discount = 0.5

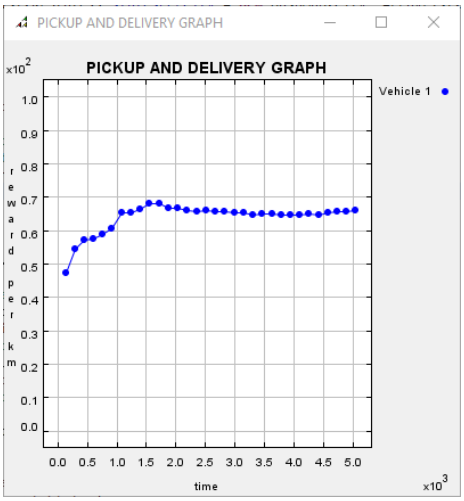

Figure 3: Discount = 0.85

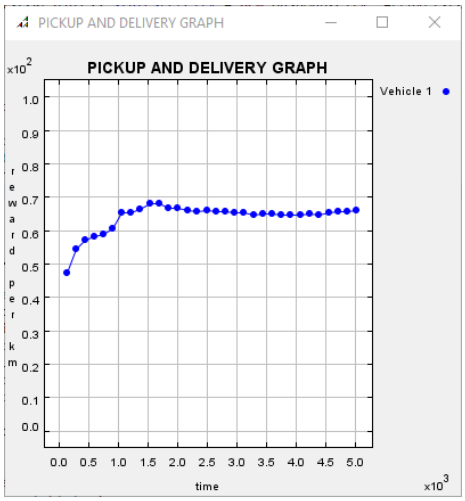

Figure 4: Discount = 0.99

2.1.2 Observations

We observe as the Discount factor gets closer to 1, we usually get better results. This is because when the discount factor gets closer to 1, the number of iterations to calculate the most optimal decision gets larger, and thus they are more precise. We also notice that there is a diminishing return after a certain point, since we have reached the "good enough" decision (In our example, 0.85 and 0.99). We have tested with 0.999, and it takes roughly

10 times longer (14000 iterations vs 1000) to get only slightly better results.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

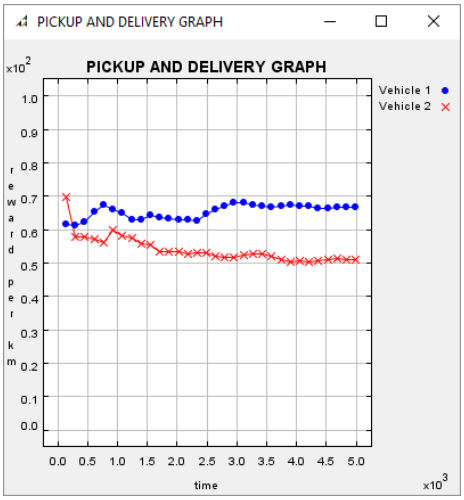

Figure 5: Seed = 42. Discount = 0.85. RLAgent is Vehicle 1, Dummy agent is Vehicle 2

2.2.2 Observations

As we can see, our agent performs significantly better than a totally random agent. We also notice that the graphs for the Reactive agent isn't the same as previously because the addition of another agent alters the pRNG results for our agent, since only one pRNG is used for spawning tasks for the whole simulation. However in the end, our agent still converges toward the same values.

2.3 Experiment 3: Three of our agents

2.3.1 Setting

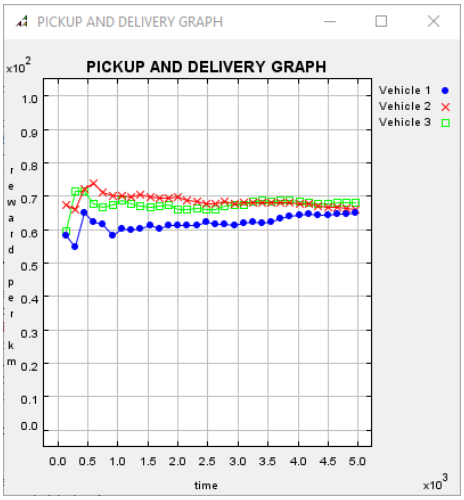

Figure 6: Seed = 42. Discount = 0.85

2.3.2 Observations

Even though each of the agents has different results at the beginning, they all end up converging toward the same value as found before.

2.4 Experiment 3: Netherlands

2.4.1 Setting

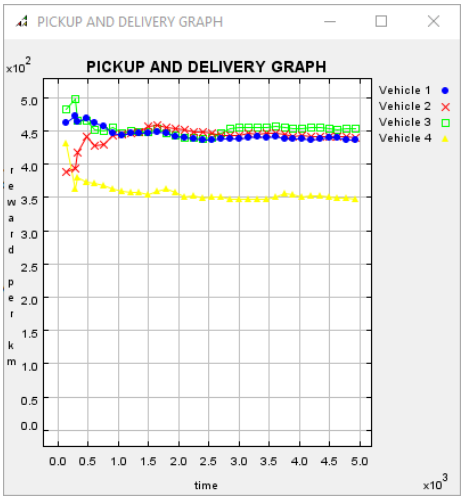

Figure 7: Seed = 42. Discount = 0.85. All are RLA agents but vehicle 4.

2.4.2 Observations

There is nothing much to say that wasn't found in the previous analysis. We would just like to note that closer cities with better connections lead to better profits, even for a random agent, as less money is spent on travel costs.