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

Group №90: Ruben Janssens, David Resin

October 8, 2019

1 Problem Representation

1.1 Representation Description

In our representation, we made the following decisions:

- **State representation** : There are n^2 possible states with n being the number of cities. Each state represents a task for a source-destination couple, with the absence of a task being represented with the source and the destination being the same. The source city always represents the city the agent is currently in.
- **Possible actions** : There are only 2 actions that can be taken : *TAKE* or *SKIP*.
- **Reward table** : The table is coded as a function that works in the following way :
  - For the *SKIP* action, the reward is negative and is the average cost to the travel to the neighbors.
  - For the *TAKE* action and no task available, the reward is negative infinity.
  - Otherwise, the reward is the reward of the given task (equivalent to a state in our representation) minus the cost of travel.
- **Probability transition table** : The table is coded as a function that works in the following way :
  - For the *SKIP* action :
    - If the two states cannot follow each other (cities are not neighbours), return 0.
    - Otherwise, return $1/n_{neighbors}$ times the odds of the next state happening for its source city (the task represented by that state being present in that city).
  - For the *TAKE* action:
    - If the current state has no task or the two states cannot follow each other, return 0.
    - Otherwise, return the odds of that next state happening for its source city.

The model could possibly be refined by differentiating the *SKIP* action into actions for moving to the different cities, as right now after a *SKIP* action the agent will move to a random neighbour.

1.2 Implementation Details

At the end of the reinforcement learning algorithm, the values of the best actions for all the states are saved in a `HashMap<MyState, Double> V`. The best action to take in every state, so the action that has the highest value for that state, is saved in a `HashMap<MyState, MyAction> Best`.

During the execution of the algorithm, we keep two maps for `V`: we have `currV` and `nextV`. At every iteration, `currV` represents the value of V before the iteration, `nextV` represents the value of V after the iteration, so the algorithm always writes in `nextV`.

After the iteration, the algorithm checks if the difference between `currV` and `nextV` is in no element bigger than a certain threshold, which is set at 0.0001 in our program. `V` is initialized with all elements being 0 in our implementation.

The transition probabilities and rewards are computed during the execution of the reinforcement learning algorithm: whenever a certain value is needed, it is computed at the spot. That means these two tables are not stored.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

The experiments were ran with the same configuration as in the `reactive.xml` file that was included in the assignment. Only the cost per km of the vehicles was changed to 20.

We ran the experiment with one, two and three agents. The agents are all `reactive-rla` agents. We let the experiment run until each agent has executed 6000 actions and then look at the average profit per action and the reward per km graph, generated by LogistPlatform.

2.1.2 Observations

We start with only one agent, and vary the discount factor from 0.1 to 0.9 in steps of 0.2.

In the graphs of the reward per km in Figure 1, we see that the lowest discount factor, 0.1, gives a slightly worse performance than 0.5, 0.7 and 0.9. The performance of discount factor 0.3 is similar to that of 0.1. 0.5, 0.7 and 0.9 have similar performances, with 0.5 having the best performance. It seems that a low discount factor results in a worse performance, with a possible optimum around 0.5. Only the graphs of discount factors 0.1, 0.5 and 0.9 have been included.

The average profit per action seems to vary only slightly with the differing discount factors, with a lower discount factor resulting in a higher average profit, and discount factors 0.7 and 0.9 showing almost no difference.

Discount factor	0.1	0.3	0.5	0.7	0.9
Average profit per action	27435	27240	27168	26994	26965

Then, two experiments were ran with two reactive-rla agents: one where the discount factor of both agents is 0.9, and one where one agent has a discount factor of 0.9 and the other a discount factor of 0.1.

The results of these experiments are much clearer than the ones with only one agent. This may be attributed to the randomness that comes with every simulation, and could be tested by running the experiment multiple times. In the graphs in Figure 2, we see that there is almost no difference between the two agents with the same discount factor, and that in the second experiment, the agent with the higher discount factor performs signifcantly better than the agent with discount factor 0.1. The average profits per action also reflect this.

Discount factor	Average profit per action	Discount factor	Average profit per action
0.9	26987	0.9	28222
0.9	27594	0.1	26824

Lastly, two experiments were ran with three reactive-rla agents: one where all three agents have a discount factor of 0.9, and one where one agent has a discount factor of 0.9, another 0.5, and another 0.1.

In this last experiment, the results are again not as distinctive as in the second experiment. In the reward per km graphs in Figure 3 (a) and (b), the agents with equal discount factors seem to perform similarly, with their relative performance varying over time. In the second experiment, the performance of the agents seems to increase with a higher discount factor, although the differences in performance are smaller than in the experiment with only two agents. In the average profit per action, differences are very small and no conclusion can be drawn from this.

Discount factor	Average profit per action	Discount factor	Average profit per action
0.9	27634	0.9	27927
0.9	27573	0.5	27682
0.9	28084	0.1	27765

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

We run the experiment with the same configuration as experiment 1. Now we have our agent competing against the random agent and an agent that systematically goes to the first city in the neighbor list and only takes the task if the destination of that task is that city (we call it ”short-sighted”).

2.2.2 Observations

We can see that random and short-sighted are both at the same level, with ours towering over them. It seems logical that short-sighted and random have similar results, since the tasks are still distributed randomly and therefore randomness is still present for the two dummy agents. Going for the first city every time doesn’t hold any advantage in itself.

2.3 Experiment 3: Price per kilometer & Discount factor

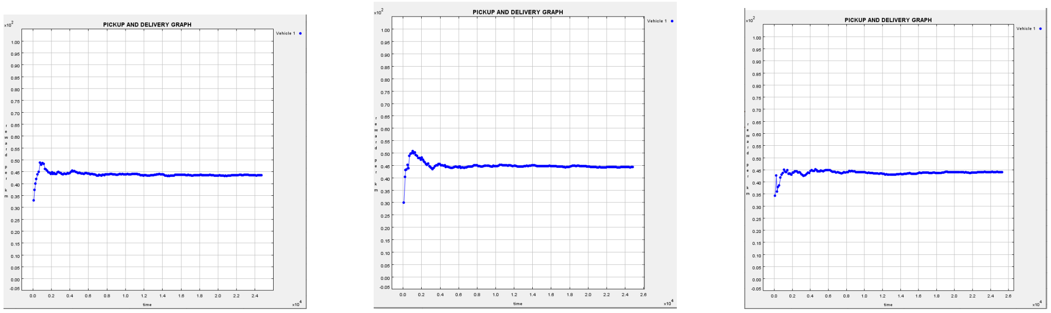
2.3.1 Setting

Again the same parameters as before except this time we varied two things: the price per kilometer and the discount factor. We had 5 vehicles with 5 different discount factors (.1, .3, .5, .7 and .9) and ran the experiment 3 times with prices 1, 5 and 50 per kilometer.

2.3.2 Observations

We can make three observations out of our results :

- The discount factor doesn’t seem to have a big influence on the reward per kilometer, as the lines are practically stuck to each other for any graph.
- However, the price per kilometer seems to have a big influence, but the lowest values obviously yield the best results. This is totally expected as the price per kilometer can be strictly defined as a ”low = good, high = bad” scale.
- With a higher price per kilometer, especially for the price being at 50, the distance between the performances of agents with different discount factors seems larger than with a lower price per kilometer.

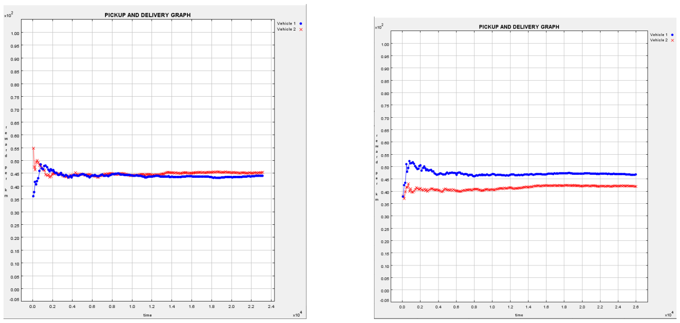


(a) Discount factor 0.1

(b) Discount factor 0.5

(c) Discount factor 0.9

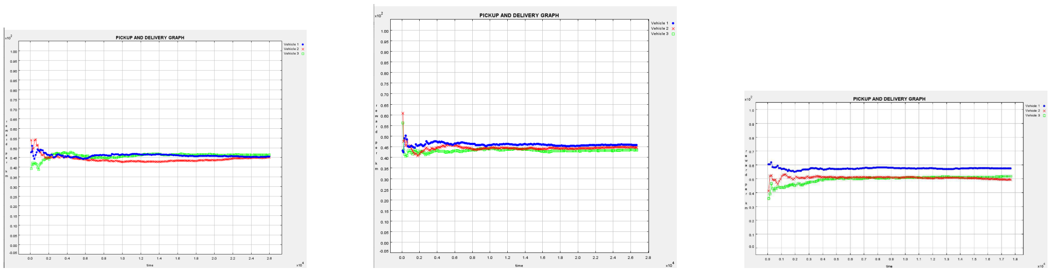
Figure 1: Graphs of the reward per km for single reactive-rla agents with a varying discount factor.



(a) Discount factors 0.9 (blue) and 0.9 (red)

(b) Discount factors 0.9 (blue) and 0.1 (red)

Figure 2: Graphs of the reward per km for two reactive-rla agents with varying discount factors.

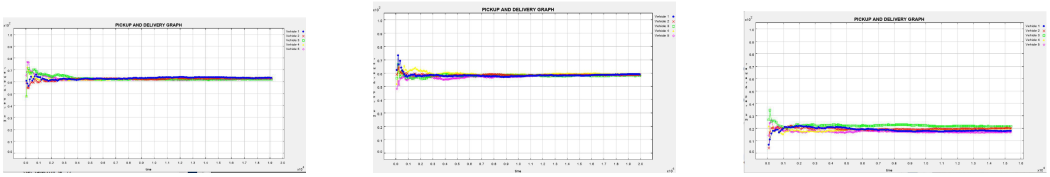


(a) Discount factors 0.9 (blue), 0.9 (red) and 0.9 (green)

(b) Discount factors 0.9 (blue), 0.5 (red) and 0.1 (green).

(c) Experiment 2 (reactive-rla agent blue, random red, short-sighted green)

Figure 3: Graphs of the reward per km for the experiments in experiment 1 with three agents, and experiment 2.



(a) 1 per kilometer

(b) 5 per kilometer

(c) 50 per kilometer

Figure 4: Graphs of the reward per km for 5 reactive-rla agents with varying prices per kilometer (experiment 3)