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

Group №11: Marcel Dubach, Maxime Gardoni

October 6, 2020

1 Problem Representation

1.1 Representation Description

1.1.1 State representation

The state variable is represented by a pair of cities, the first one being the city where the vehicle currently is, and the second one is the destination where the task available in the present city needs to be delivered to. If there is no task available, the destination is set to *null*. The state where both cities are identical, which means there is a task to deliver to the current position, is excluded in our implementation. Therefore the number of states for a simulation with n cities equals n^2.

1.1.2 Action possible

The number of possible actions the agent may take depends on the state in which it is at that time. We differentiate between two sets of states, according to whether there is a task available at the current state or not.

In the case that there is no task available in the current state, which means if the current position of the vehicle is city k, then $s = (k, null)$. In this case the possible actions the agent may take are restricted to move to one of the neighbor cities $i \in \mathcal{N}_k$, where $\mathcal{N}_k$ denotes the set of neighbor cities of city k, which contains $n_{NB,k}$ cities. The choice of the neighbor city is randomized uniformly over all possible neighbor cities.

If there is a task available at the current city, the agent can decide to either accept to deliver the task (i.e. to *pickup*) or to refuse it. If the agent refuses the task, it must move to one of the neighbour cities from the city it is currently in. In the case that the refused task had the delivery destination of one of the neighboring cities, the action to move to that neighbor city without the task is removed from the possible list of actions, as it would be even less optimal to go to that city without a task than to deliver the available task at that city.
Note that the agent does need to do one of the possible actions and it cannot stay in the same state.

1.1.3 Reward table

The direct reward of an action a, performed by an agent at state $s = (i, j)$ is given as follows:

$$R(s = (i,j), a) = \begin{cases} p(i,j)r(i,j) - d(i,j) \cdot c(d), \text{ if } a\text{=”pickup”} \\ -d(i,k) \cdot c(d), \text{ if } a\text{=”move to } k\text{”} \end{cases}$$

where $d(i,j)$ is the distance between city i and j, and $c(d)$ is the cost per kilometer of travel. $r(i,j)$ is the reward of the task of transportation from city i to city j, and $p(i,j)$ denotes the probability of that task appearing.

Assuming that in a present state s there is no task available, the transition probabiliy to future states s' is given as follows:

$$T\{s'=(i,j)|(s=(k,null),a=\text{move to } i)\} = \begin{cases} \frac{1}{n_{NB,k}} p(i,j), \text{if the next state is (i,j)} \\ \frac{1}{n_{NB,k}}(1-\sum_m p(i,m)), \text{if the next state is (i, } null\text{)} \end{cases}$$

In the case that there is a task available at the present state $s=(k,l)$, where $l \neq null$, the transition table is a bit more complex: The state transition probability in this case looks as follows:

$$T\{s'=(i,j)|(s=(k,l),a)\} = \begin{cases} p(i,j), \text{ if a="pickup", } l=i \text{ and } j \neq null \\ 1-\sum p(i,j), \text{ if } a\text{="pickup" and } l=i \text{ and } j = null \\ \frac{1}{n_{NB,k}} p(i,j), \text{ if a="move to } i\text{", } i \neq l, \ j \neq null, \ j \notin \mathcal{N}_k \\ \frac{1}{n_{NB,k}-1} p(i,j), \text{ if a="move to } i\text{", } i \neq l, \ j \neq null, \ j \in \mathcal{N}_k \\ \frac{1}{n_{NB,k}}(1-\sum_m p(i,m)), \text{ if a="move to } i\text{", } i \neq l, \ j = null, \ j \notin \mathcal{N}_k \\ \frac{1}{n_{NB,k}-1}(1-\sum_m p(i,m)), \text{ if a="move to } i\text{", } i \neq l, \ j = null, \ j \in \mathcal{N}_k \end{cases}$$

Note that the action "move to k" in a state (i,k), which means that the vehicle would move to the task destination without performing the transportation task, is removed in our implementation. However, this is not mandatory, and the action can not be optimal for the reactive agent, since a "pickup" action would always be a better action, as long as the reward stays positive.

1.2 Implementation Details

We implemented a reinforcing learning algorithm with a stopping criterion on the maximum adaptation of the reward function $V(s)$. As soon as the adaptation of $V(s)$ for all states s is smaller than some constant ϵ the learning algorithm will consider the control policy as satisfying and will stop the optimization (in our implementation we set $\epsilon = 1e-3$.

2 Results

2.1 Experiment 1: France

2.1.1 Setting

For this experiment, we used the provided "France" topology. We tested different discount factors, logged the results in a .csv file and plotted the values later using a python script. The name of the agents represent their respective discount factor, except for "random" which designates the random agent which was provided in the original template file.

2.1.2 Observations

Looking at the gained reward, all reactive agents clearly outperform the random agent. Surprisingly, when looking at Fig.1, a small discount factor seems better. When looking at the reward (Fig.2), it seems that discount = 0.15 or 0.95 share the best results. Note here that we did not optimize for the reward per kilometer, but only the amount of reward, irrespective of the distance covered.

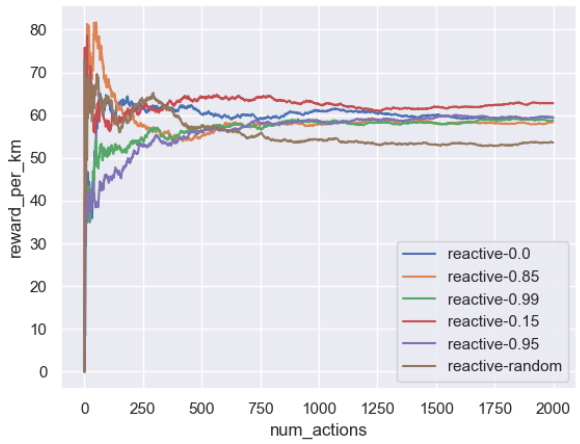

Figure 1: Reward per kilometer

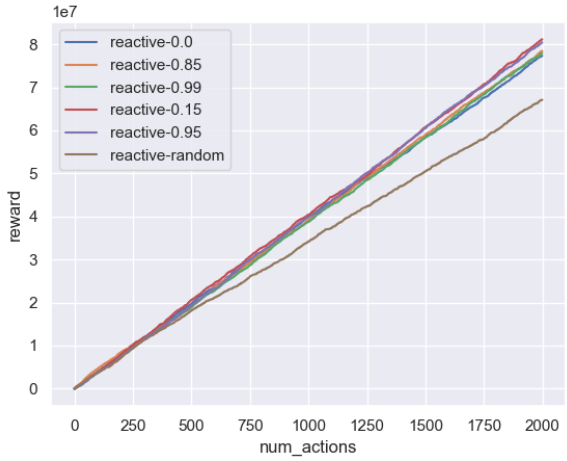

Figure 2: Reward

2.2 Experiment 2: Netherland

2.2.1 Setting

For this experiment, we used the provided "Netherland" topology and tested the same 6 different agents.

2.2.2 Observations

Results are similar to experiment 1. Again, the small discount factor seems to perform better, even we evaluated on 6000+ steps (not showed here due to space constraints). For detailed evaluation, the performance of the agents could be averaged over several runs to avoid stochastic effects. This is however beyond the scope of this report.

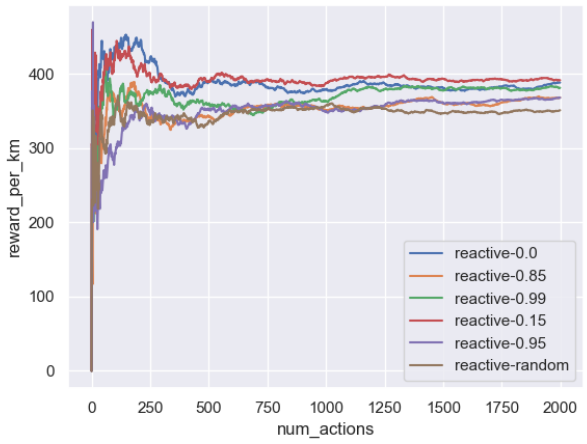

Figure 3: Reward per kilometer

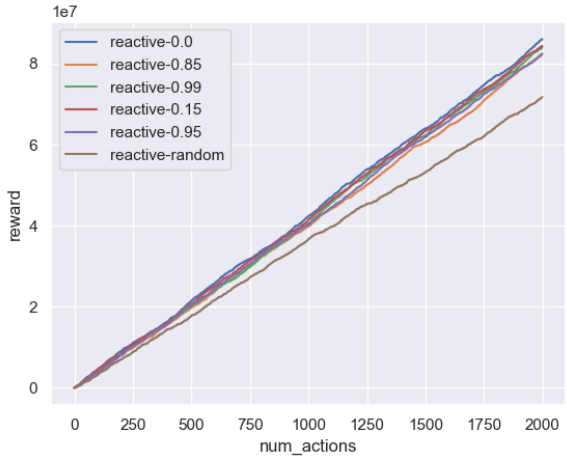

Figure 4: Reward