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

Group №27: Carolin Beer, Francesco Pase

October 9, 2018

1 Problem Representation

1.1 Representation Description

Let the network have a total of n cities. **State** The state s is a tuple $s = (s_1, s_2)$ of cities. They specify the current position of the agent and the destination city of a task, if existent. If no task is found in a city, the destination city is set to the current city, too.

Actions Possible actions a are "Move to city $i \in \{0, \dots, n-1\}$" (without a task) or "Pick up a task and deliver it". The action is denoted by an integer value. For moving to another city without task, the action is identified by the integer value of the destination city (i.e. $\{0, \dots, n-1\}$), picking up a task and delivering it is represented by a value of n. Therefore, $a \in \{0, \dots, n\}$.

Reward table The reward for $R(s, a)$ computes followingly:

$$R(s,a) = \begin{cases} -(\text{costs} * \text{distance}_{s_1,a}) & \text{For moving to another city without a task} \\ \text{reward}_{\text{task},s_2} - (\text{costs} * \text{distance}_{s_1,s_2}) & \text{For picking up and delivering a task} \end{cases}$$

where *costs* denote the costs per km and $\text{distance}_{i,j}$ the distance between to cities i and j in km. reward_task, s_2 represents the remuneration for the delivery of a task to city s_2.

Probability transition table The probability transition table computes as follows:

$$T((s,a,s') = \begin{cases} \mathbf{1}\{\text{s}_1 \text{ neighbor of } s'_1\} * \Pr\{\text{Task for city } i \text{ exists}\} & \text{For } s'_2 = i \in \{0,\dots,n-1\} - \{s'_1\}, s'_1 = a \neq n \\ & \text{or } a = n, s'_1 = s_2, s'_2 =\in \{0,\dots,n-1\} - \{s'_1\} \\ \mathbf{1}\{\text{s}_1 \text{ neighbor of } s'_1\} * \Pr\{\text{No task available}\} & \text{For } s'_2 = s'_1, s'_1 = a \neq n \\ & \text{or } a = n, s'_1 = s_2, s'_2 = s'_1 \end{cases}$$

1.2 Implementation Details

Discount factor The discount factor can be specified in the `agents.xml` file, if no value if present, it defaults to a value of 0.95.

Epsilon value For the Reinforcement Learning algorithm, we use an epsilon value of 0.00001 as minimum threshold (exclusive) that the accumulated reward for one state must increase in order to continue the search for better strategies.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

We use the configuration files `reactive.xml` with the topology `france.xml` and `reactive2.xml` with the topology `the_netherlands.xml` to test the effects of varying discount factors. We use the average reward (per action) after 1000 actions as a basis for our comparison.

In a next step, we run the simulation for 3 agents simultaneously to compare the results to the ones obtained for a single agent.

2.1.2 Observations

We now report the results of simulations by using different discount factor and different topology (France and The Netherlands).

While conducting the experiment, it becomes obvious, that a cut-off of 1000 introduces some randomness

Discount value	France	Netherlands
0.95	38790.129	44835.55
0.85	38790.129	44835.55
0.55	38929.079	44790.424
0.15	38929.079	44206.786
0	37146.1	43326.527

Table 1: Average reward per action for the RLA after 1000 actions with different discount values.

to the outcome as the average values still change heavily for this number of simulations. However, it is also difficult to conduct the experiment for significantly longer simulation periods as the runtimes become higher. However, higher discount values also lead to a higher effort to compute the strategy, i.e. the RLA algorithm goes through more iterations.
In theory, we would expect the average reward to become larger for higher discount values as it optimizes the reward in the long run. This trend is not entirely clear from the results, but should become apparent for longer simulation runs.

When running the experiment for 3 agents (each with 1 vehicle) simultaneously the results for a discount value of 0.85 in France are 39356, 38645, 37899 after 1000 actions each. This result itself doesn't allow for much insight, however, the plot in Figure 1 shows that all three vehicles converge towards the same rewards. This can be easily explained as all agents follow the same strategy. Furthermore, the allocation of tasks is regenerated for every step (i.e. old tasks expire), so that the different agents don't interfere with each other and the average rewards as well as the rewards per km are close to the values obtained for only a single agent.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

The dummy agent, as implemented in the template, chooses randomly whether to pick up a task or not, if existent in a city. If a task is present, the agent picks it up with a probability of the discount factor (i.e. by default 0.95 if not specified otherwise). If no task is available or the agent doesn't pick up the task, it will randomly move to one of the neighboring cities. Just like in Experiment 1, we use the average reward (per action) after 1000 actions as a basis for our comparison.

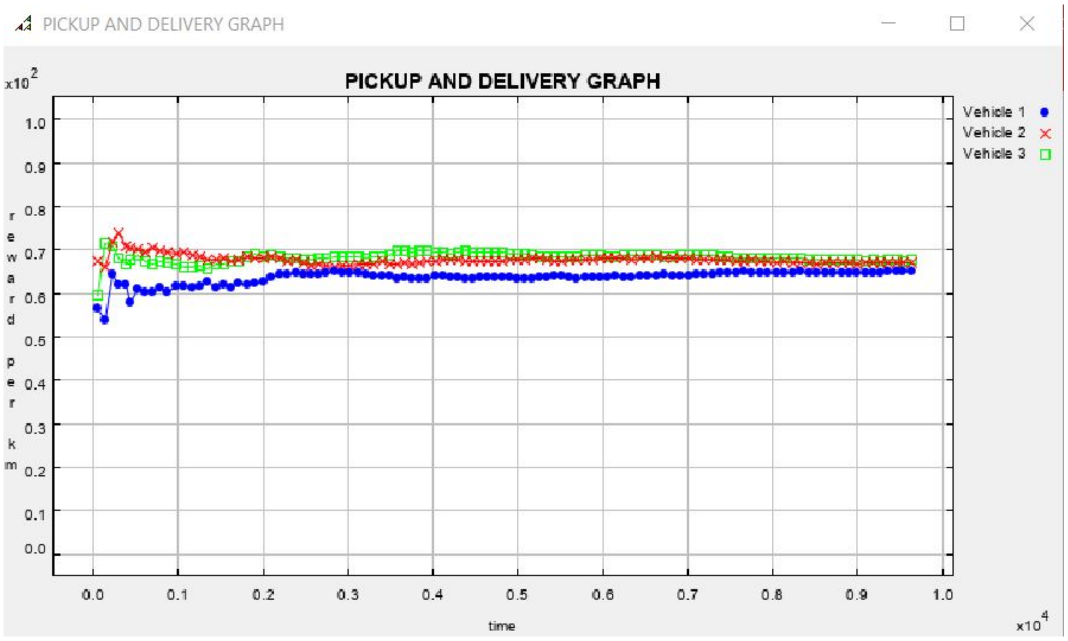

Figure 1: Plot for three vehicles with the RLA algorithm on the French map

2.2.2 Observations

The observations for the topology France and Netherlands can be found in Table 2. A high discount value seems to work better in this setup. This is the case, as for low discount values, the agent has a higher probability of just moving around without taking on a task, which incurs costs. This is avoided by setting a high discount value (for a value of 1, i.e. a Greedy strategy), the agent would take on every task that is available), so the average reward is significantly higher for this strategy. However, we can also see that even the Greedy strategy performs worse than the RLA which is to be expected as the Greedy algorithm doesn't leverage any additional information about the network topology.

Discount value	France	Netherlands
0.95	34083	41055
0.85	31051	35528
0	-2154	-379

Table 2: Average reward per action for the Random agent after 1000 actions with different discount values.