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

Group №21: Dubravka Kutlešić, Nevena Drešević

October 6, 2020

1 Problem Representation

1.1 Representation Description

State representation: The state of the agent is determined by the pair $(currentCity, taskCity)$:

- Current city - the city in which the agent is currently at
- Task city - the destination city of the task in the current city

If there is no task in the current city, the task city has a value of *null*.

Actions: There are two types of actions:

- "accept the task"
- "go to city x", where $x \in Cities$

Considering current state, the action "accept the task" is possible if *state.taskCity* is not *null*. The action "go to city x" is possible if *state.currentCity* and x are direct neighbors.

Reward table: The immediate reward that the agent gets if it decides to refuse the packet in the city i and move to neighbor city j is the cost of traveling from city i to city j. The cost is calculated as the product of the vehicle's cost per kilometer and the distance between cities i and j. If there are no packets in the city i, the agent must move to the neighbor city and the reward is defined as in the previous case.

If the agent in the city i decides to accept the packet for the city j, the immediate reward is the reward for the delivery minus expenses for traveling to city j. Precisely, in $states \times action$ space reward is defined as follows:

$$R(state = (i, j), action) = \left\{ \begin{array}{l} -costPerKm * d(i, k),\ if\ action = "go\ to\ city\ k",\ i\ and\ k\ are\ neighbors \\ r(i, j) - costPerKm * d(i, j),\ if\ action = "accept\ the\ task" \\ not\ defined,\ otherwise \end{array} \right]$$

Probability of the transition: For each state $s = (currentCity, taskCity)$ and its allowed action a the possible next state is s' where:

- $s' = \{(x, y) : x = taskCity \wedge (y \in Cities \vee y = null)\}$, if a = "accept the task"
- $s' = \{(x, y) : x \in Neighbors(currentCity) \wedge (y \in Cities \vee y = null)\}$, if a = "go to city x"

The probability of transition to possible state s' from the state s by taking allowed action a is defined as:

$$T(s, a, s' = (x, y)) = \left\{ \begin{array}{l} Prob(no\ packets\ for\ delivery\ in\ the\ city\ x),\ if\ y = null \\ Prob(in\ city\ x\ there\ is\ a\ packet\ for\ city\ y),\ if\ y \in Cities \end{array} \right.$$

1.2 Implementation Details

The proposed representation is implemented by the following structures:

```
    private Set<State> states;
    private Set<Integer> actions;
    private Map<State, Map<Integer, Double>> R;
    private Map<State, Map<Integer, Map<State, Double>>> T;
    private Map<State, Integer> best; // best action for given state
    private Map<State, Double> V; // value of the state
    private Map<State, Map<Integer, Double>> Q; // value of the action in the state
```

If the transition between *initialState* and *nextState* is possible performing action, the transition probability is calculated as follows:

```
    T.get(initialState).get(action).put(nextState,
distribution.probability(nextState.getCurrentCity(), nextState.getTaskCity()));
```

This calculation includes the case when *nextState.taskCity* is *null* e.g. there are no packets in *nextState.currentCity*, because in that case probability method with argument *null* calculates the probability that there are no packets for delivery in *nextState.currentCity*.

The reinforcement learning algorithm (RLA) iteration stops if the current and previous values for $V(s)$ differ no more than ϵ for every state s. The constant ϵ is given as the configuration parameter. The maximum error comparing to the true V is $\frac{2\epsilon\gamma}{1-\gamma}$ for every state.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

The first experiment is used to examine the effect of the discount factor. The default probability distributions are used with France's topology. Also, the total profit is measured after 1000 actions and ϵ for convergence condition is equal to 0.0001. The values in column *Iterations* represent the number of iterations of RLA before its convergence.

2.1.2 Observations

Discount factor	Iterations	Error	Total profit
0.25	9	0.0000666	38 282 117
0.50	17	0.0002000	38 929 079
0.85	63	0.0011333	38 790 129
0.90	97	0.0018000	38 790 129
0.99	989	0.0197999	38 790 129

Table 1: Observations for experiment 1: Varying discount factor

The discount factor affects the execution of RLA in the sense of the bigger it is the more iterations are needed for training the agent. Also, the discount factor increases the profit to a certain extent, it is shown that for *discountFactor* ≥ 0.85 the gained profit remains the same. Therefore, to optimize the RLA execution and profit it is good to set the discount factor to 0.85.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

This experiment is used for comparison of the performance of reactive agents (RLA) with random and greedy agents. The random agent (RND) is the agent given in the starter files. The greedy agent (GRE) is an agent that takes the packet in a city if the reward for the delivery ($r(i, j)$ value) is greater than a given configuration parameter reward threshold. For the reactive agent, we used discount factor 0.99 and epsilon 0.001. For the random agent, we used probability for pick-up 0.85. Finally, the greedy agent used a threshold of 5000 (close to the mean of default reward distribution). Task distributions for the experiment are default task distributions given in the starter files. The starting cities for all agents are the same.

2.2.2 Observations

England			France			Switzerland			Netherlands		
RLA	RND	GRE	RLA	RND	GRE	RLA	RND	GRE	RLA	RND	GRE
90.4	73.8	85.9	76.6	61.0	73.9	95.0	73.3	88.0	95.1	77.7	91.2

Table 2: Observations for experiment 2: Total profit for different agents

The table represents the total profit in millions after 2000 actions. Comparing to the random agent, the reactive agent's profit is $22 - 30\%$ greater. However, compared to the greedy agent, the difference is $3.5 - 8\%$.

2.3 Experiment 3: Multiple agents

2.3.1 Setting

The final experiment is used to examine the effects of having multiple reactive agents in the simulation. In the first part of the experiment, all agents have the same discount factor (0.99), while in the second part discount factor is varied. All distribution configuration parameters are set as in the starter files and all vehicles have the same cost-per-km and speed. The number of observed actions is 2000 and ϵ is 0.0001.

2.3.2 Observations

In the table below, the total rewards in millions are shown. Every row represents a different simulation with a corresponding number of agents.

The left table represents the experiments in which the discount factor is the same for all agents. The differences in reward are not significant. It can be concluded that the multiple agent environment does not affect much since there is a constant supply of tasks.

The right table represents the experiments in which the discount factor is varied. The differences are slightly more prominent, and different discount factors impact the agent's profit in multiple agent environments.

Agent 1	Agent 2	Agent 3
76.6	-	-
77.6	78.2	-
74.8	78.4	77.6

Agent 1 ($\gamma = 0.5$)	Agent 2 ($\gamma = 0.85$)	Agent 3 ($\gamma = 0.9$)
-	77.1	-
78.0	-	76.2
78.8	75.8	77.2

Table 3: Observations for experiment 3: Multiple agents