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

Group №: Liangwei Chen, Siyuan Li

8. Oktober 2019

1 Problem Representation

1.1 Representation Description

1.1.1 State Representation

$S = \{(A, B) | A \in CITY, B \in CITY \cup \{null\} - \{A\}\}$
Our state representation contains two variables, one is current city, the other is target city. If a task from A to B is available at the current city A, the target city would be the destination city B. Otherwise when there's no task in the current city A, then the target city is set to be null. Thus, if there are N cities in the map, we should have N^2 states on the states table.

1.1.2 Action Table

$A = Map : s- > PossibleAction(s) \forall s \in S$
$$PossibleAction(x) = \begin{cases} Neighbour(s.src), & \text{if } s.dst == null \\ Neighbour(s.src) \cup null, & \text{otherwise} \end{cases}$$
In general, we have two kinds of actions, one is picking up the task, the other one is going to the neighbour city. Specifically, if no task is available at current city, the valid actions are going to a neighbour city. Else, picking-up the task is also regarded as valid action. For each state, an action list is stored to indicate valid actions from it.

1.1.3 The Reward Table

$R : (s, a) - > \mathbb{R}$
$$R(s, a) = \begin{cases} td.reward(s.src, s.dst) - vehicle.costPerKm() * distance(s.src, s.dst), & \text{if } a == null \\ -vehicle.costPerKm() * distance(s.src, s.dst) \cup null, & \text{otherwise} \end{cases}$$
The reward is computed for each state-action pair. If there's a task in the current city, and the agent choose to pick up the task, it will receive a net reward computed by $profit - cost$ with $profit = valueOf(task)$ and $cost = vehicle.costPerKm() * distance(s.src, s.dst)$. If the agent choose to go to a neighbour city, it will only receive a cost by moving to the selected neighbour.

1.1.4 Probability Transition Table

$T(s, a, s') = \mathbb{P}(s'|s, a)$
$$T(s, a, s') = \begin{cases} \mathbb{P}(s'.dst|s'.src), & \text{if } (a = null \wedge s'.src = s.dst) \vee (a! = null \wedge s'.src = a) \\ 0, & \text{otherwise} \end{cases}$$
By taking action a from $s.src$, the vehicle arrives at a city C. Then the only possible states after (s, a)

are exactly the states s' with $s'.src = C$. As a result, the distribution of s' can be derived as $\mathbb{P}(s'.src = C, s'.dst = D|s, a) = \mathbb{P}(s'.dst = D|s'.src = C)$ which can be obtained from the td object.

1.2 Implementation Details

- The states S are implemented as a list of instances of class $State$ in which source and task destination city are stored.
- The actions A is implemented as a $HashMap < State, ArrayList < City >>$ whose formula are described above.
- The rewards R is implemented as a $HashMap < State\_Action, Double >$ where State_Action is a class with member variable state and action indicating the current state and action taken. The formula are given in above section.
- The transition probability T is implemented as a $HashMap < State\_Action, ArrayList < Pair < State, Double >>>$. For each (s, a), a list of possible next states with their probability are evaluated by formula given above.
- Value iteration:

getOptPlan (S, A, R, T)

for $s \in S$ **do**
    $V(s) \leftarrow$ random Doubles
while true **do**
    $V\_old \leftarrow V$
    for $s \in S, a \in A[s]$ **do**
        $Q(s, a) \leftarrow R(s, a) + discount\_facotor * \sum_{s'|T(s,a,s')>0} T(s, a, s') * V(s')$
    for $s \in S$ **do**
        $V(s) \leftarrow max_{a \in A[s]} Q(s, a)$
    if $\max_{s \in S} |V(s) - V\_old(s)| < threshold$ **then**
        break
for $s \in S$ **do**
    $\pi(s) = argmax_{a \in A[s]}(Q(s, a))$
return π

2 Results

2.1 Experiment 1: Discount factor

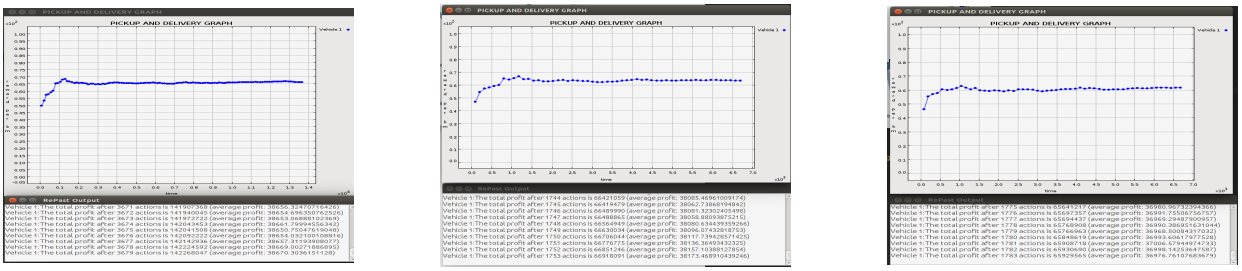
2.1.1 Setting

Reactive Agent 1	Reactive Agent 2	Reactive Agent 3
0.99	0.5	0

2.1.2 Observations

Discount Factor	0.99	0.50	0.0
Average reward	38670	38173	36976

Resulting figures show that the different discount factor does influence the final performance of the model. If we set the discount factor to 0, we observe the final performance (avg profit) of our model is



(a) 0.99 (b) 0.50 (c) 0

Abbildung 1: plots of average reward for different discount factors

around 36000. If we set the discount factor bigger such as 0.5, the average profit we get will increase to around 38000. The discount factor of 0.99 yields the best performance which is 39000. The results demonstrate that the agent who consider future rewards has the better strategy than the greedy agent who only consider current profit.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

vehicle 1	Reactive agent with 0.95 discount factor
vehicle 2	Dummy Agent with $P = 0.95$ to pick up a task
vehicle 3	Dummy Agent with $P = 0.5$ to pick up a task

2.2.2 Observations

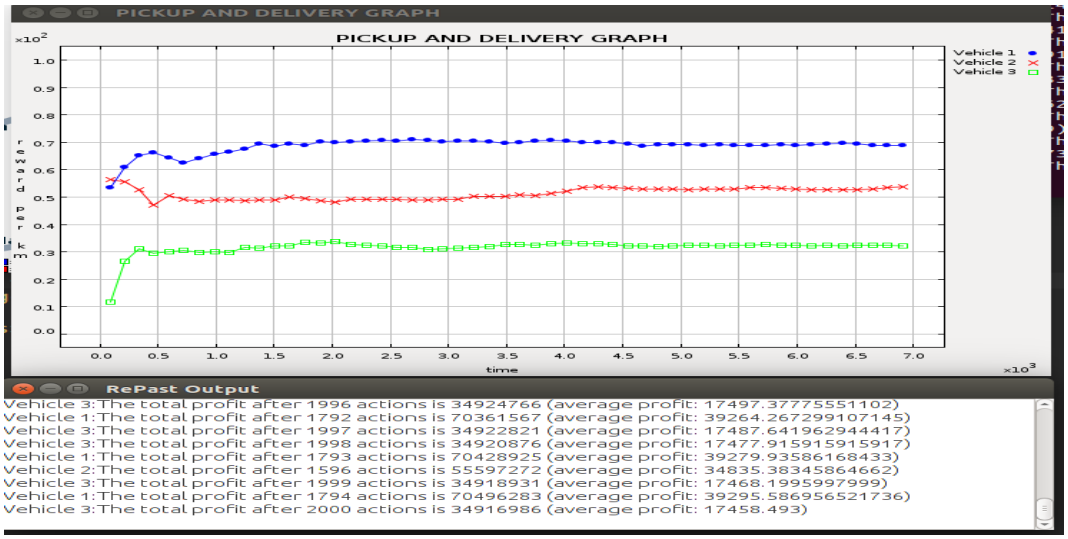


Abbildung 2: plot of reward of multi agents

⋮

Vehicle	vehicle 1	vehicle 2	vehicle 3
Average reward	39279	34835	17458

- The comparison between dummy and reactive agents' rewards indicates that, even though the reactive agent values future reward almost as much as immediate reward (0.95 discount factor), the reactive agent outperformed than dummy ones shortly after the beginning.