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

Group №89: Wenuka Gunarathna, Sofia Dandjee

October 6, 2020

1 Problem Representation

1.1 Representation Description

The state we defined has three parameters namely, the current city, task availability of the current city and the destination city if a task is available. The action of the agent has two main parameters which are pick up and destination city.

State	Actions	
	No PickUp and Move to city x'	PickUp and Deliver to city y
City x with task to city y	- cost(x,x')	reward(x,y)-cost(x,y)
City x with no task	- cost(x,x')	Not applicable

Table 1: Reward table $R(s, a)$

Table 1 shows the reward calculation for each state and action. City x' is a neighbour city. Note that in table 1, the reward is defined as the per km reward and the cost is also defined per km. Since in the given example the reward per km is fixed, we can eliminate that from the equation. Further, if an agent picks up a task, the agent has to deliver it and it is enforced in the implementation. The probability transition table $T(s'|s, a)$ is defined as shown in Table 2. $Pr(x', y)$ is the probability that there is a task from city x to city y.

Action a and State s	State s'	
	City x' with task to city y	City x' without task
Task available with pick up	Pr(x',y)	Pr(no task)
Agent refused available task & moved to city x'	Pr(x',y)	Pr(no task)
No task available & agent moved to city x'	Pr(x',y)	Pr(no task)

Table 2: Transition table $T(s'|s, a)$

1.2 Implementation Details

We have implemented two classes to store states and actions as defined above. The *state* class was defined with an *originCity*, a Boolean *taskAvailability*, a *taskDestinationCity*. The *Act* class was defined with a *reward* (Double), a *destinationCity* and a Boolean to store *pickingUp* or not.

The *State* class also contains a list of *Act* actions that can be done from the state.

We have also created a list of states for every possible city in the topology with a task available to every other city, as well as with no task available (destination is null). Also, we created a list of actions,

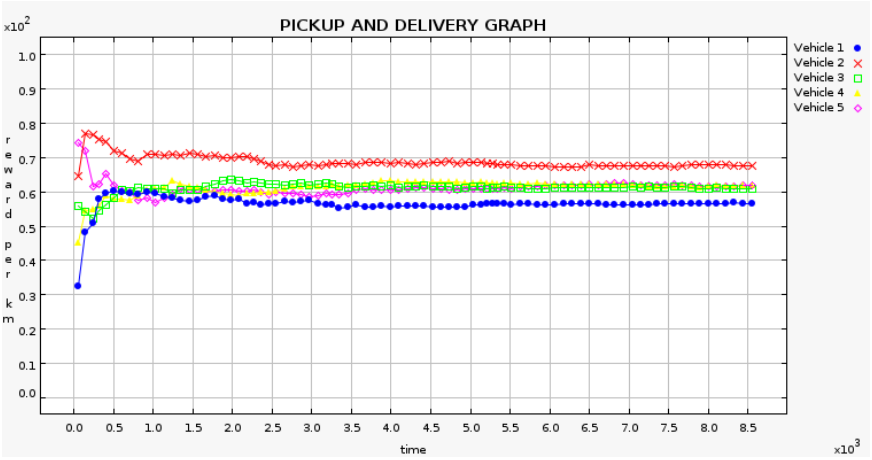

Figure 1: Reward per km for different discount rates

going to every possible city either to deliver a task or not. Instead of storing the V values for each action in a separate table, we store them in the *Act* itself.

Then, for every state, we add to the list of possible actions from the previously created one. If the state has a task available to destination y, we add a pickup action to city y and a non-pickup action to every neighbor city. If the state doesn't show a task available, we add a non pickup action to every neighbor city.

Finally, in the reinforcement learning algorithm we try to find the V value for each state which gives an equivalent matrix of reward value for each state. For this we loop through the list of states and through the list of its possible actions and for a given state, compute the reward according to the previously defined V value table and update V table from the maximum reward for each state. We do this iteration until the V value table converges and the difference is smaller than a defined number. For the experiment we have chosen 1 as the maximum difference allowed between two iterations and the convergence happened during 18 steps.

After having looped through all the possible actions for a given state, we store the best action's reward. During simulation, we decide what action to choose depending on the state, resulting in the highest per km reward.

This whole experiment was done with the hidden assumption that the environment does not change with time. If that is the case, we might have to introduce an online learning methodology where agent learns while traversing through the given topology.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

We compared reactive agents that had different discount factors : $\gamma = 0.0$, $\gamma = 0.5$, $\gamma = 0.85$, $\gamma = 0.95$, $\gamma = 0.99$ and $\gamma = 1.0$. Respectively it took 2, 6, 18, 51 and 247 iterations to find V values for the first five cases. For the $\gamma = 1.0$ case it did not converge. For cases like this we have introduced a maximum number of iterations allowed to converge the results. However it was not used here for this experiment.

2.1.2 Observations

The lower the discount factor is, the more the agent prefers immediate reward rather than postponing a pickup with the expectation of higher future rewards and vice versa. However we also can see in both the

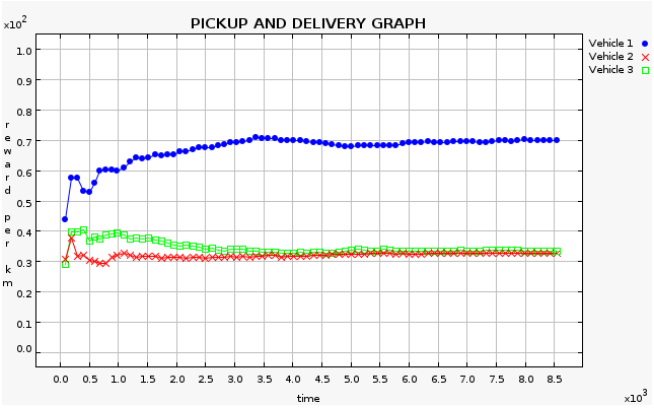

Figure 2: Reward per km comparison with reactive agent vs dummy agents for topology France

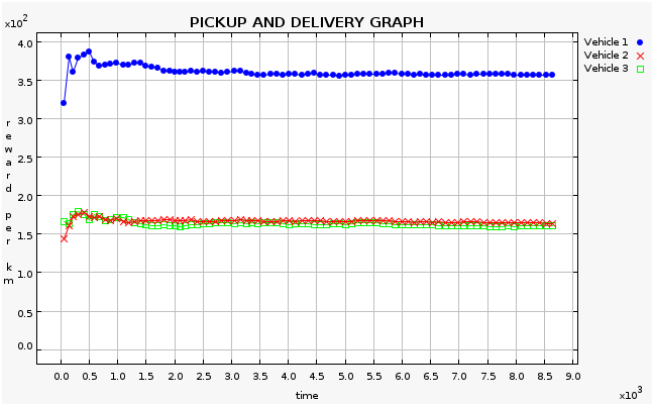

Figure 3: Reward per km comparison with reactive agent vs dummy agents for topology Switzerland

extremes the reward is lower than the $\gamma = 0.5$. This might be because now the agent is greedy enough to take what is there as well as vigilant about the next city, if it yields a higher return. Regarding the convergence issue at $\gamma = 1.0$, this is because agents V values will be equally affected by the other V values, resulting in a never ending optimization. The figure 1 shows these results respectively for each vehicle except for $\gamma = 1.0$

2.2 Experiment 2: Comparisons with dummy agents for France topology

2.2.1 Setting

We added a second reactive-random agent to the agents.xml file as well as our agent reactive-rla with a discount factor of 0.5. The experiment was done using the France topology.

2.2.2 Observations

The trained reactive agent performs well better than the two random agents. As the time passes, we can see each agents are converging to a constant reward per km. Results are shown in th Figure 2 where the reactive agent is in the blue vehicle and the two dummy agents are in green and red. The performance increase is roughly two fold in this case.

2.3 Experiment 3: Comparisons with dummy agents for Switzerland topology

2.3.1 Setting

We have done the same Experiment as Experiment 2, but with the topology for Switzerland.

2.3.2 Observations

Results were same as the Experiment 2 except the converging reward per km are different and are shown in figure 3. In this case also we can see a two fold performance increase in the reactive agent.

2.4 Overall Observation

As we can see, with the right discount factor, we can converge the results to find the optimal actions to choose for a given state. With the right optimization, we can achieve a higher per km profit and the profit per km always converges when the number of iterations are high.