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

Group №9: Baris Sevilmis, Doga Tekin

October 3, 2020

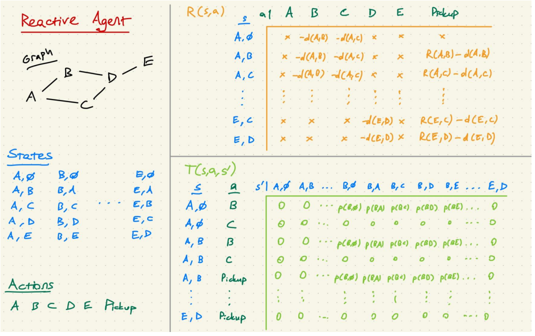
1 Problem Representation

1.1 Representation Description

• State Representation: States are represented by two cities, in which first city is the city agent is currently in and second one is the delivery city of the available task of the current city. In case of no existing task, second city is represented as empty/null as depicted in Figure 1.

• Action Space: Actions are represented as cities, more specifically the set of cities that are currently accessible from the current state. Additionally, there is an explicit pickup action, which corresponds to the pickup-deliver action.

• Reward Table: Reward table consists of a 2 dimensional space. In case of an invalid (s, a) combination, the entry is set to null. The reward of a move action is the negative of the distance (in km) between the two cities, multiplied by the cost per km of the moving vehicle. The reward of a pickup action is the reward for delivering the task minus the cost per km times the distance of the shortest path to the delivery city from the current city.



States

A, ∅	B, ∅		E, ∅
A, B	B, A		E, A
A, C	B, C	· · ·	E, B
A, D	B, D		E, C
A, E	B, E		E, D

Actions

A B C D E Pickup

R(s,a)

s \ a	A	B	C	D	E	Pickup
A, ∅	×	−d(A,B)	−d(A,C)	×	×	×
A, B	×	−d(A,B)	−d(A,C)	×	×	R(A,B) − d(A,B)
A, C	×	−d(A,B)	−d(A,C)	×	×	R(A,C) − d(A,C)
⋮	⋮	⋮	⋮	⋮	⋮	⋮
E, C	×	×	×	−d(E,D)	×	R(E,C) − d(E,C)
E, D	×	×	×	−d(E,D)	×	R(E,D) − d(E,D)

T(s,a,s')

s	a	s' | A,∅	A,B	…	B,∅	B,A	B,C	B,D	B,E	…	E,D
A,∅	B	0	0	…	p(B,∅)	p(BA)	p(BC)	p(BD)	p(BE)	…	0
A,∅	C	0	0	…	0	0	0	0	0	…	0
A, B	B	0	0	…	p(B,∅)	p(BA)	p(BC)	p(BD)	p(BE)	…	0
A, B	C	0	0	…	0	0	0	0	0	…	0
A, B	Pickup	0	0	…	p(B,∅)	p(BA)	p(BC)	p(BD)	p(BE)	…	0
⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮
E, D	Pickup	0	0	…	0	0	0	0	0	…	0

Figure 1: State Representation

• Transition Table: Transition table consists of a 3 dimensional space. As in reward table, first two dimensions correspond to current state and actions. Third dimension corresponds to the next states. In case of an invalid (s, a, s') combination, next state is set to null. For each possible next state $s' = (city_f, city_t)$, the corresponding probability is the probability of finding a task to $city_t$ at $city_f$.

1.2 Implementation Details

• Instead of using 2 dimensional or 3 dimensional matrices for reward and transition tables, HashMaps are used. Therefore, access to specific $R(s, a)$ or $T(s, a, s')$ are in $\mathcal{O}(1)$ time. In addition, invalid $R(s, a)$ or $T(s, a, s')$ combinations are not created, therefore space allocation are minimized. Lastly, in case of no task states, instead of creating a state of $(city_k, NULL)$, $(city_k, city_k)$ notation is used since there is no explicit state of $(city_k, city_k)$. This reduces complexity of code as well.

• Action space are represented by city ids instead of city names for convenience. Pickup action is depicted as -1, whereas city ids start from 0 and increase.

• Transitioning between states for move action and pickup action are different. State transition for a move and pickup action are the following:

$$MOVE \equiv \{(s, a) \to s' \mid s : (city_k, city_l), s' : (neighbor(city_k), city_m) \& k, l, m \in cities\}. \tag{1}$$

$$PICKUP \equiv \{(s, a) \to s' \mid s : (city_k, city_l), s' : (city_l, city_m) \& k, l, m \in cities\} \tag{2}$$

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

In this experiment, discount factor has been altered multiple times to witness the affect of discount on agent movement. Discount factor of 0.1, 0.5 and 0.99 has been tried.

2.1.2 Observations

As demonstrated in Figure 2, we observe that discount factor does not have a huge impact on the reward per km of the agents in this task. This implies that this task does not need to think too much about future rewards and therefore, a greedy agent that cares mostly about the immediate reward can still perform well. However, we still observe that agents with a higher discount performed slightly better than the ones with lower discount. It takes longer for Value Iteration to converge when the discount is higher, because it takes longer for each state value to consistently encode future rewards.

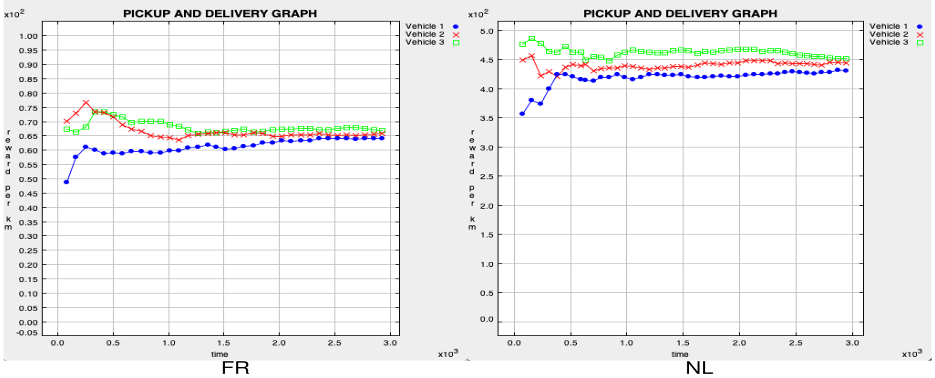


Figure 2: Discount values: $Vehicle_1 = 0.1$, $Vehicle_2 = 0.5$, $Vehicle_3 = 0.99$.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

In this experiment, reactive agents performance against random and dummy agent are considered in various topologies. Random agent accepts tasks with an 85% probability, otherwise randomly moves. Dummy agent has been implemented as random agent, however in case of an existing task, dummy agent always picks up the task.

2.2.2 Observations

As depicted in Figure 3, reactive agent performs best, dummy agent second and random agent third. First thing to notice is that the dummy agent performs better than random agent in both topologies, meaning that pickup action dominates over random choice over pickup action and move actions. Reactive agent dominates both other agents over performance, as a consequence of the value iteration algorithm. Value for the states, and the optimal action selection for each state in comparison to dummy agent indicates only prioritizing pickup is still not optimal enough.

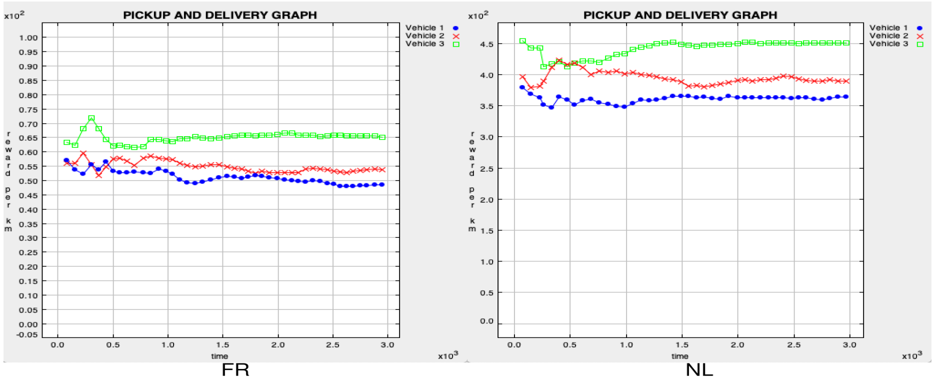


Figure 3: Agents: $Vehicle_1 = Random$, $Vehicle_2 = Dummy$, $Vehicle_3 = Reactive$.

2.3 Experiment 3: Multi-Agent System

2.3.1 Setting

In this experiment, multi-agent systems have been observed on France and Netherlands topologies such that different agent amounts have been tried on both. Respectively 1, 2, and 3 agents were tested.

2.3.2 Observations

Figure 4 refers to activating two or three instances of the same agent in the environment does not significantly affect any agent's performance. We see that they converge roughly to the same reward average depending on the topology. Perhaps if there were many more agents or if tasks were generated much more slowly, then there would be a shortage of tasks and the rewards would decrease. We also see that even though we are running agents with the same strategy, they might end up with slightly different performance values due to different starting cities and the randomness of task generation.

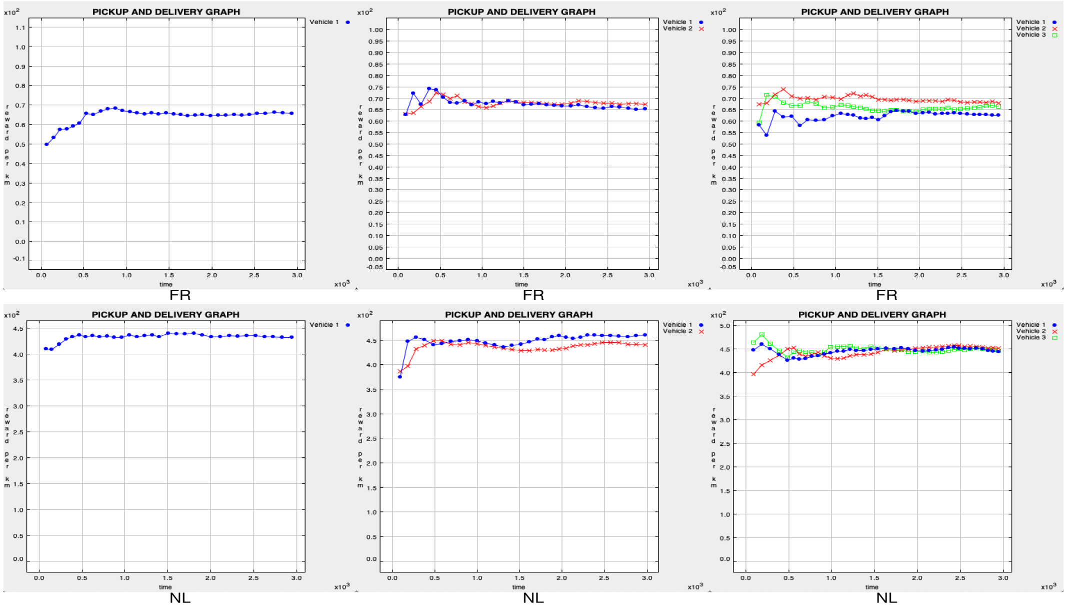


Figure 4: Number of agents: From left to right, 1, 2, 3.