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

Group №: 2 Cosmin-Ionut Rusu, Sorin-Sebastian Mircea

October 8, 2019

1 Problem Representation

In our problem, the agent has to decide between two actions:

1. Pick an available package and deliver it to its destination; or

2. Move to a neighbor.

We are interested in learning a mapping V from a state to an action such that, by following that mapping (policy), we expect to get higher reward. In other words, we find the policy that results in a maximal expected reward.

At each step, we have to make a decision with respect to the package that we receive. We can either pick up the package and deliver it, or we can move to a neighbor of the current city in the hope that we might find a more rewarding package to deliver there, skipping the current one. The greedy algorithm which picks the packages as they arrive suffers from *opportunity cost*, since taking a low-rewarded package now might lead to missing out on a higher reward package.

1.1 Representation Description

State

Following that intuition, we considered our state as being a tuple of the current city and the potential destination city of the current package: $(current\_city, dest\_city)$, with the special case when no package exists in our current city, in which case the value of our $dest\_city$ will be $NULL$.

Reward

$$R(s,a) = \begin{cases} r(s.current, s.dest) - distance(s.current, s.dest) & a = PICKUP \\ \\ -distance(s.current, s.neighbour) & a = MOVE(neighbour) \end{cases}$$

Where $r(current, dest)$ is the reward (in money) that we get by delivering that package.

We defined the reward to be the difference between the reward we got and the distance that we travel (profit). We give a negative reward when taking a move action such that closer neighbors (measured in distance) receive better (still negative) reward then farther away neighbors.

Transition probabilities

$$T(s,a,s') = \begin{cases} P(s'.current, s'.dest) & s.dest = s'.current \;\;\&\;\; a = MOVE(s.dest) \\ P(s'.current, s'.dest) & s.dest = s'.current \;\;\&\;\; a = PICKUP \\ 0 & otherwise \end{cases}$$

This formula not only make sense from an intuitive perspective, but it's also a probability in the mathematical sense. $T(s,a,s') = Pr(s'|s,a)$. For a fixed s and a, $\sum_{s' \in S} Pr(s'|s,a) = \sum_{s' \in S} P(s.dest, s'.dest) = 1$, since we know P is a probability measure from a known distribution.

1.2 Implementation Details

In order to nicely integrate the Q-learning algorithm and to make everything crystal clear, we've defined an abstract class $AgentMove$ which is then extended as *PickupAction* and $Move(city)$. Each action has it's own method *encode* that encodes these moves as an integer. A *decode* method is also implemented which then can decode the integer value and build back an $AgentMove$ class. We've followed a similar approach for the $State$ class.

To encode the actions to integer, we've used $N+1$ values in the range $[0, N]$ (where N is defined as the number of cities) as follows:

$$encode(a) = \begin{cases} neighbour.id & a = MOVE(neighbour) \\ N & a = PICKUP \end{cases}$$

Decode is trivially defined since *encode* is bijective.

To encode the states, we've used $N*(N+1)$ values since we know we can have $NULL$ values for the destination cities (i.e. no package).

$$encode(s) = \begin{cases} city.id * (N+1) & s = (city, NULL) \\ city.id * (N+1) + dest.id + 1 & s = (city, dest) \end{cases}$$

No decode was implemented for the states since we don't need those. We only need to decode the actions when we are given a state and we learned the mapping V to an encoded action.

2 Results

Unless otherwise stated, the settings for each experiment are the ones depicted in the Table 1.

Parameter	Default Value
Topology	$France$
Vehicle	1
Ticks	4000

Table 1: Default values for the parameters

2.1 Experiment 1: Discount factor

2.1.1 Setting

The purpose of this experiment is to understand how the discount factor influences the result. We experimented with values of 0.2, 0.5 and 0.99.

2.1.2 Observations

As the discount factor is increased we can observe that the average profit increases slightly. So, setting a higher value will lead to better profits.

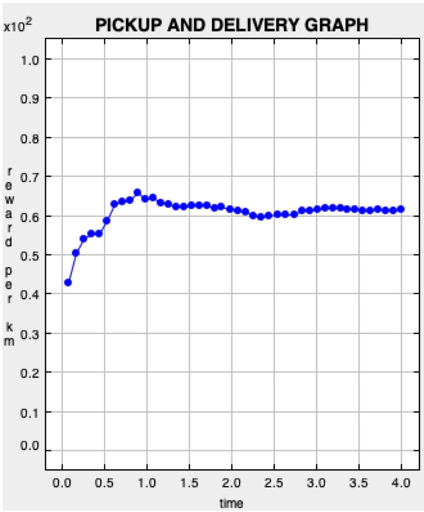

Figure 1: gamma 0.2; average profit 38013

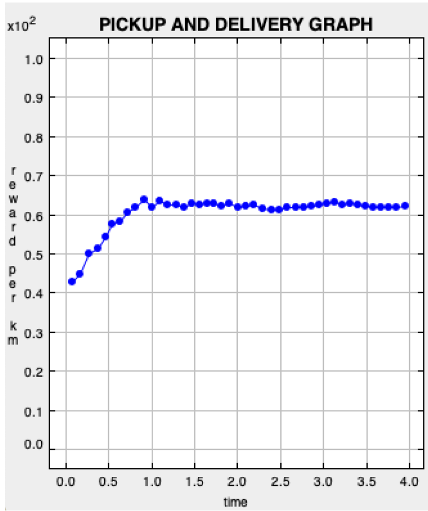

Figure 2: gamma 0.5; average profit 38052

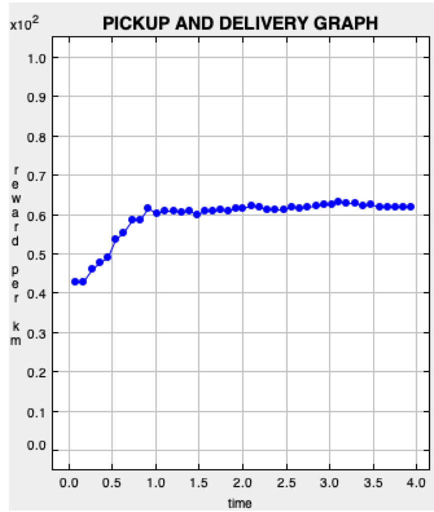

Figure 3: gamma 0.99; average profit 38145

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

In this experiment, we are comparing the results of our agent with two dummy agents: the random agent and a greedy agent which always chooses to pick up a package if there is one.

2.2.2 Observations

The random agent represents the baseline of the experiment. After learning the best policy, our agent was able to converge at 38145 average profit (**24%** improvement). An interesting aspect is defining a policy that will always pick the available package. This policy is getting closer to the average profit of our agent, but achieves **worse** reward per kilometer. Intuitively, this means that our agent learned to deny a low-rewarded package now for a higher-rewarded package in the future.

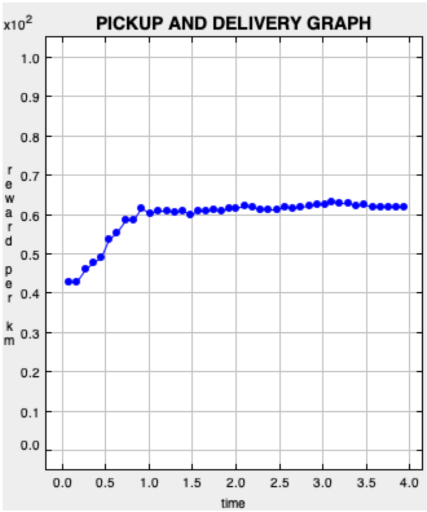

Figure 4: our agent (gamma is 0.99): average profit 38145

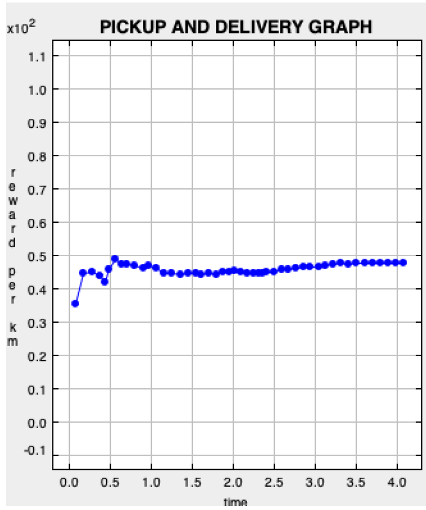

Figure 5: random agent: average profit 30760; pick 85% of cases

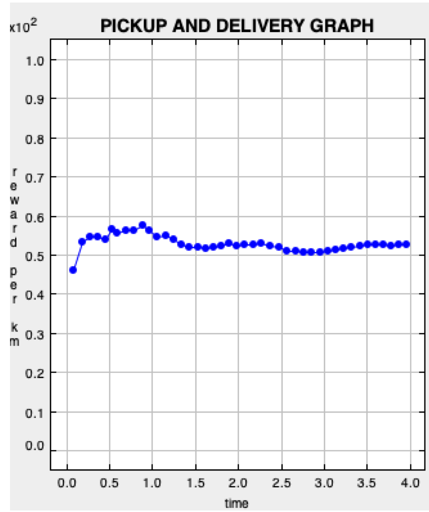

Figure 6: agent that always picks up; average profit 36560