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

Group 42: Nicolas Jomeau, Vincent Gurtler

October 7, 2019

1 Problem Representation

1.1 Representation Description

- **States** are represented by a boolean and a pair of cities. The boolean is if there is a task in the current city. If the boolean is set to true, then the first city is where the vehicle is and also the pickup city of a task; the second city is then the delivery city. If set to false, both cities are the same: where the vehicle is.
- **Actions** are a destination city and if the vehicle will pick or not a task before going to the destination.
- The **reward table** is the reward value associated to a pair of state and action. If they are incompatible (ie two cities not neighbors), the reward is set to an extremely low value. If they are compatible and there is a task to deliver, the reward is the value of the task minus the fuel cost. Otherwise the "reward" is just the fuel cost.
- The **probability table** follows the same principle: if one state is unreachable from another state and action then its probability is set to 0. Otherwise it's equal to the TaskDistribution's probabilities.

1.2 Implementation Details

The implementation of an Action or a State follows their description. An Action has one boolean and one destination city. A State has one boolean and two cities. We added another class called StateAction which is a pair of a State and an Action to simplify the creation of the reward and transition table.

The setup function works as followed (for n cities):

1. It creates all the possible states ($S = n^2$), then all the possible actions ($A = 2n$) and finally all the possible pairs of state and action ($S*A = 2n^3$).
2.
   - It computes the reward function for every pair of state and action following some rules: If the vehicle is in a city with a task and the action requires to take it, then the total reward is the task value minus the fuel cost to go from the pickup city to the destination city.
   - Whether the city has a task or not, if the action doesn't require to take a task and the action destination is reachable from the vehicle current city (ie: in 1 move) then the total reward is minus the cost of fuel.
   - Any other situation will be heavily penalized by a huge negative reward to avoid the agent making an illegal Move or Pickup.

3. The reinforcement learning algorithm is the same as the one presented during the lecture. For each state, we take it's potential reward value (initially a large negative number) and we try to find the most rewarding action. This action is found by iterating other all possible states and adding to an accumulator the discounted value of the next state times the probability to be in this next state. If the accumulator is greater than the original potential reward at the end of the iteration then it replaces it and the associated action is saved as "the best action" for this state.
   We do this until no more best action is found meaning the algorithm has converged.

4. Finally, during the act() function, we create the current state from vehicle position and the available task (if there is one) and find the best action according to the policy we computed earlier. If the action requires to pick a task, then we do a Pickup action, otherwise a Move action.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

We will use the Netherlands topology as it greatly amplifies the effect of the discount factor with the city of Groningen having a lot of task to give but being far from the center of the network. We also tried on the vast topology of Switzerland as it's increased size will surely requires more "foresight" to maximize the profit. We tested the value 0.01 and 0.99

2.1.2 Observations

Running the test for approximately 20000 steps gave us on average 452$/km with a discount factor of 0.01 and 465$/km with 0.99 for the Netherlands (+3%) and 288 vs 306$/km for Switzerland (+6%).
This seems logical as the vehicle preferring short-term results (low factor) won't have any optimization on the future and thus won't be able to adapt to the whole network. With a low factor, only the immediate action is valued and we don't take into account the potential reward of the state we will go to meanwhile a high-discount-factor vehicle will be able to "avoid" high paying immediate reward in favor of a more advanced itinerary that is more rewarding in the long term.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

We will test the reactive agent against the random agent in the Netherlands and France with the default discount factor (0.85).

2.2.2 Observations

We observed an increase of the money made by kilometer driven of about 30% for both topologies between our reactive agent and the random agent. This difference is easily explained by the optimization learning to maximize profit.

Country	Random	Reactive	% diff
France	49.3	64.7	+31%
Netherlands	357.2	461.9	+23%
Switzerland	243.3	292.1	+20%

2.3 Experiment 3: Concurrent agents

2.3.1 Setting

We just wanted to see how 3 agents would behave with the same policy and how it would influence the total reward per kilometer. We used the reactive agent in the Netherlands.

2.3.2 Observations

Increasing the number of vehicles had not impact whatsoever on their average reward per kilometer. We discovered that the task generation is done when a vehicle arrives in a city and not on a "refresh each unit time" policy. Thus, if someone wants to maximize the profit of such a system, it is just needed to add more vehicles.