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

Group №: 272257, 262609

October 6, 2020

1 Problem Representation

1.1 Representation Description

The problem studied in this report is the classical pickup and delivery problem. We try here specifically to implement one agent (which will be reffered to as *the truck* or the agent interchangeably from now on). We try to implement the truck as a reactive agent. To properly explain the comportment of the agent, we need to clearly state what are : *the states*, *the actions*, *the reward table*, and *the probability transition table*.

- **States**: The state represents the perception that the agent has of it's environment. Once the agent finishes an action (see definition of actions), it observes two things: the city it arrived in and the task (or absence thereof) it may accept. In our implementation, the states are represented in memory by two attributes: the city the agent is currently in and the city the task points to. Note that **the absence of task is represented by a contract from the current city to itself**.
- **Actions**: An action is defined as the movement of the agent from a city to another. Note that actions can describe the movement of the agent from any city to any other one **even if the two cities are not connected directly**.
- **Reward table**: The reward table is a $\#States \times \#Actions$ table in which each cell gives us the reward given to the agent if it is in a given state and chooses a given action. For two cities i and j, we are given the average reward of a contract from i to j, which will be denoted $r(i, j)$ and thus each cell will either contain zero, if the departure city and arrival city of the state and the action do not coincide, either contain $r(i, j)$ if they do.
- **Probability transition table**: The probability transition table is a $\#States \times \#Actions \times \#States$ table that gives us the probability of being in the state s' if the action a is performed in the state s. Note that if the city of departure of a is not the current city of s or the city of arrival of a is not the current city of s', then this probability is zero. Otherwise, we note that, since the city of the state of arrival is a deterministic variable when given a state and an action, we see that the probability is only determined by the probability of the presence of a contract from the city of s', i, to the city of arrival of the contract given in s', j. These probabilities are a given datum of the problem, and will be denoted $p(i, j)$.

1.2 Implementation Details

.

In this implementation, we implement the actions as the class *ActionMDP*, which comprises of two attributes: the first being *City startCity* that represents the city the truck is currently in and an other *City endCity* that represents the city will be in if the truck chooses the given action.

We implement the states as the class *StateMDP*, which comprises of two attributes: the first being *City city* that represents the city the agent is currently in and the second being *City task* that represents the arrival city of the current task.

We implement the reward table as a method of the *Reactive* class, *rewardFunction(StateMDP startState, ActionMDP action).* The output of this function for the a given state s and an action a is:

$$rewardFunction(s,a) = \begin{cases} 0 & \text{if } a.startCity \neq state.city, \\ r(s.city, a.endCity) & \text{otherwise} \end{cases}$$

where $r(s.city, a.endCity)$ is given above.

We implement the probability transition table as a method of the *Reactive* class, *transitionProbabilities(StateMDP startState, ActionMDP action, StateMDP endState)*, which for given states s, s' and an action a gives us the following output:

$$transitionProbabilities(s,a,s') = \begin{cases} 0 & \text{if } a.startCity \neq s.city \text{ or } a.endCity \neq s'.city, \\ p(s'.city, s'.task) & \text{otherwise} \end{cases}$$

where $p(s'.city, s'.task)$ is defined above. At a given state s, the action performed by the agent is given by the formula

$$\pi(s) = \text{argmax}_a \left\{ rewardFunction(s,a) + \gamma \sum_{s' \in S} transitionProbabilities(s,a,s')V(s') \right\}$$

where S is the set of states, γ is the discount factor, which is also a given datum of the proble, and $V(s)$ is the rewards that can be reached from the state s using an optimal choice of actions, this is implemented as the *policy* method of the *Reactive* class. Note that the values $V(s)$ are thus given recursively by

$$V(s) = \max_a \left\{ rewardFunction(s,a) + \gamma \sum_{s' \in S} transitionProbabilities(s,a,s')V(s') \right\}.$$

We use algorithm seen in class to approximate those values since this is actually a **Markov decision process** as the state transitions are not deterministic but we can analyze the current state with certainty:

Algorithm 1: Solving the MDP

```
initialization: Set $V(s)$ arbitrary and $\infty \to V_{prev}(s)$ .
while $\max_s |V(s) - V_{prev}(s)| > \frac{0.01*(1-\gamma)}{2\gamma}$ do
    for $s \in S$ do
        $V(s) \to V_{prev}(s)$
        for $a \in A$ do
            $Q(s,a) \leftarrow rewardFunction(s,a) + \gamma \sum_{s' \in S} transitionProbabilities(s,a,s')V(s')$
        end
        $V(s) \leftarrow \max_a Q(s,a)$
    end
end
```

where A is the set of actions. Note that this algorithms terminates since the set of rewards is finite and thus bounded. Also note that the number in the while condition is here to assure that when the algorithm terminates we have $\max_s |V(s) - V^*(s)| < \epsilon = 0.01$, where $V^*(s)$ is the *true value of the state* s, the latter being the value we want to approximate.

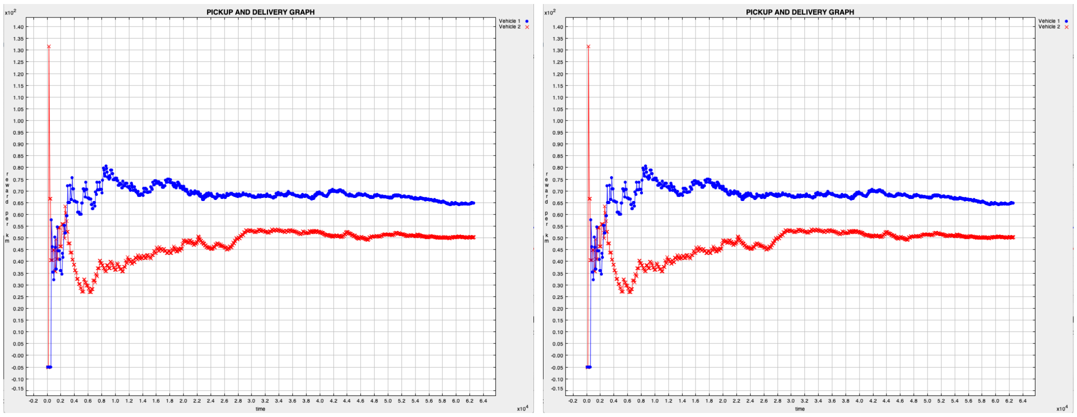

Figure 1: **Left :** evolution over time of the average reward per kilometer for a reactive MDP agent (in blue, with a discount factor of 0.85) and a dummy agent (that takes random decisions, in red). **Right :** Similar plot for the evolution of rabbit (red) and grass (green) population equilibriums as the grass growth rate changes.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

We compare one of our *MDP* reactive agents (with a discount factor $\gamma = 0.85$) to a randomly action agent shows a clear performance difference, the topology is a model of France's road network.

2.1.1 Observations

This is shown on Figure 1, left. But our algorithm maximizes the average reward per action not by kilometer, the experiment also clearly shows shows improvement by this metric : one a 250 actions sample we have :

$$\begin{array}{ll} \textit{A total profit of } 9926295 \textit{ (average / action : } 39705.180) & \textit{for the reactive agent} \\ \textit{A total profit of } 7987058 \textit{ (average / action : } 31948.232) & \textit{for the dummy agent} \end{array}$$

Our system is showing a $24,27\%$ improvement compared to the dummy base case.

2.1.2 Setting

2.2 Experiment 2: Discount factor

2.2.1 Setting

2.2.2 Observations