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

Group №34: Carl Hatoum, Raphaël Linsen

October 6, 2020

1 Problem Representation

1.1 Representation Description

- **State:** A state is represented by 2 variables: the current city of the vehicle, and the task available in the city, if any.
- **Actions:** There are two possible actions from one state to another:
  - Move: the vehicle moves from the current city to a neighbouring one.
  - Pickup: the vehicle picks up and delivers the available task, he then directly arrives in the destination city of the task.
- **Reward table:** The reward $R(s, a)$ is equal to the reward of the delivery ($r(i, j)$) in the case of a Pickup action, minus the cost due to travelling, which is equal to the distance travelled times the cost per km of the vehicle.
- **Transition table:** The probability of a transition $T(s, a, s')$ depends on the type of action a:
  - Move: the vehicles arrives in one of the states corresponding to the target city i of the Move action. The possible next states then correspond to every possible task in this city, or the no-task-available state. The probability is thus given by $p(i, j)$ if there is a task for city j, and by $1 - \sum_j p(i, j)$ if there is none.
  - Pickup: The same logic applies, except the target city i is now the destination of the task.

  For any other combination, the probability is 0.

1.2 Implementation Details

The following classes were created: State, MyTask, MyAction, MyMove *extends* MyAction, MyPickup *extends* MyAction. The policy is then a map from each possible State to a corresponding best MyAction. Otherwise these classes are solely used for the creation of the policy.

If a city has no task, the *task* variable of the State is simply *null*.

$R(s, a)$ and $T(s, a, s')$ are not stored in a table in the code, but are simply functions defined within the code.

For the the reinforcement learning algorithm, the $V(s)$ table is initialed with zeros, and $\varepsilon = 10^{-14}$ is used for the stopping criterion.

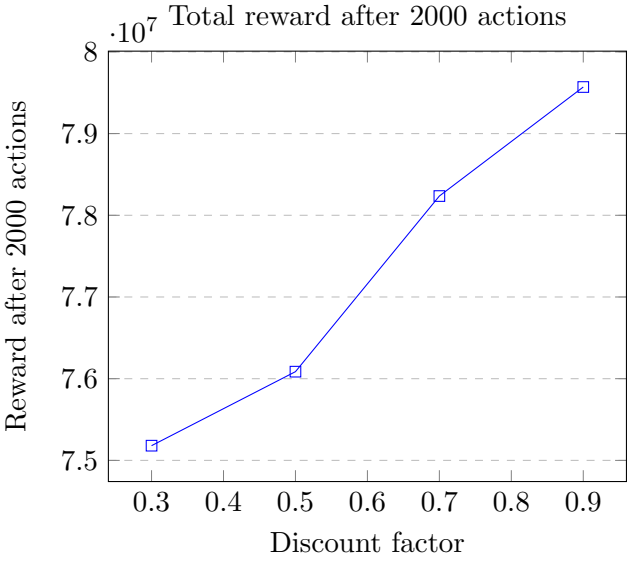
2 Results

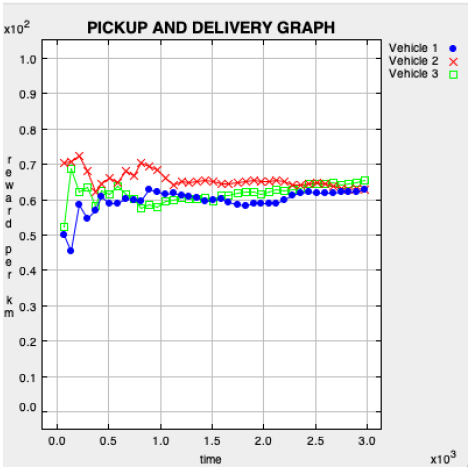
2.1 Experiment 1: Discount factor

2.1.1 Setting

We ran three agents with different discount factor, respectively 0.25, 0.55, and 0.85 on the same topology (in our case, France). The vehicles they control have the same cost per km as well. We also plotted the total reward after 2000 actions for 0.3, 0.5, 0.7 and 0.9 discount factors.

2.1.2 Observations





(a) Reward per km evolution over time for vehicle 1, 2, and 3, with respective discount factors 0.25, 0.55, and 0.85

The first chart shows the discount factor effect over the rewards after a certain number of steps, whereas the second one highlights the evolution of the average reward over time and its convergence.

At the beginning, the rewards fluctuate significantly, some vehicles seem to have a low or high reward, until they converge asymptotically to a certain value. When the discount factor becomes larger, the average reward increases over time.

The discount factor determines the importance of future rewards. A factor of 0 will make the agent only considering instantaneous rewards, whereas a factor greater than 0 will take in consideration future events. The higher the discount factor is, the more the agent gives importance to the future when it builds a policy.

For a high factor, at the beginning the agent will tend to ignore tasks that have a high reward over other ones. But in the long term, its actions will lead it in states where the rewards will be higher. Thus, the cumulative reward will be higher.

On the other hand, for a low discount factor, the agent considers more importantly short-term present rewards. They will seem high at the beginning, but these decisions won't necessarily lead to optimal states, where the average rewards are high. Consequently, the cumulative reward will be lower than the ones with a higher factor.

It should also be noted that the higher the discount, the longer the algorithm takes to converge for a given ε, until it does not for a factor of 1. Thus, there is a trade-off between high long-term rewards, and off-line learning time constraints. We should also take into account the average run time. If our agent has a short "lifespan" for example, we may not give that much importance to the future.

2.2 Experiment 2: Comparisons with dummy agents

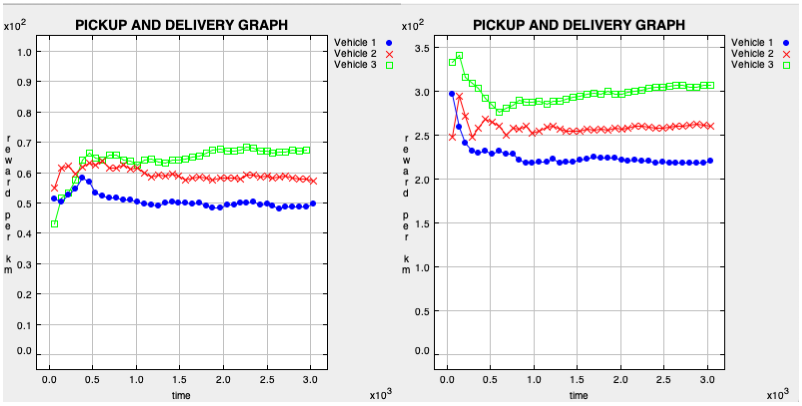
2.2.1 Setting

This experiment was run with three agents:

- Vehicle 1: The provided reactive-random agent who picks up a task with probability pPickup = 0.85 when available, otherwise, moves to a random city.
- Vehicle 2: The dummy agent always picks up a task if available, otherwise moves to the closest city.
- Vehicle 3: The reinforcement learning agent, with a discount factor of 0.85.

The other parameters remain the same as in experiment 1.

2.2.2 Observations



(a) (b)

Figure 2: Average reward for different topologies : France (a) and Switzerland (b)

We observe that the highest reward per km is obtained by the reinforcement learning agent, followed by the dummy agent. The random agent has the worst average reward, since it refuses the tasks in many cases. The performance of the dummy agent is relatively good, thanks to its decision to always pickup an available task, that on average, performs well.

The goal of the Value Iteration algorithm used here is to maximise the reward per action, without taking time into account. If the goal was to maximise reward per unit time or km, dividing the reward $R(s, a)$ by the distance would yield better results, and/or increase the discount of future rewards for longer actions, such as Pickup.

The advantage of the reinforcement learning agent resides in its ability to refuse some tasks in order to pickup others that are more profitable in upcoming states. We notice however that the dummy agent and reinforcement learning agent have more or less similar rewards in the beginning, until each one converges to different asymptotic rewards. One can then discuss the need of a reinforcement learning in some particular cases, for short run time for example, or when the policy learning process consumes a lot of resources.