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

Group №70: Julien Perrenoud, Pierre-Antoine Desplaces

October 9, 2018

1 Problem Representation

1.1 Representation Description

States - We defined our states by answering the question "what variable changes at each decision point?" with two things: where the agent is and what he knows about the package. The later only includes the destination of the package as its value is already present in the rewards table later. Therefore,

$$\text{States} = \underbrace{\{\ \text{City}_1, \ldots,\ \text{City}_n\ \}}_{\text{Location}} \times \Big(\underbrace{\{\ \text{NoPackage}\ \} \cup \{\ \text{City}_1, \ldots,\ \text{City}_n\ \}}_{\text{Delivery}} \Big)$$

This representation does not explicitly exclude states where the destination and origin city of a package is the same. Instead, we expect a transition probability of 0 from the system if such states are not permitted.

Actions - This time, we tried to answer the question "what choice does the agent have asked to make a decision?". This is simpler, as our agent can only deliver the package or move to a neighboring city.

$$\text{Actions} = \{\ \text{Pickup}\ \} \cup \{\ \text{Move}(\text{City}_1), \ldots, \text{Move}(\text{City}_n)\ \}$$

In the problem description, only a subset of actions are available from each state (as the agent can only move to *neighboring* cities, and an agent cannot pick-up a non-existing task). However, it seemed more natural for us to define the action set independently from the current state in which the agent is in and rely on other mechanisms (such as the rewards table) to explicitly remove impossible actions for each state.

Transition Probability - The first thing to realize when filling the transitions table is that a lot of triplets (s, a, s') are actually not consistent according to the model. If the "Location" attribute of S' is different than the "Delivery" (or "Move") destination of s (or a), then the probability will always be 0. For all consistent cases, the probability of a transition will depend on the probability of getting a task for this city of. All in all, we can define our transitions as

$$Pr[\ s' \mid s, a\ ] = \begin{cases} p(s'.\text{Location}, s'.\text{Delivery}) & \text{if } a = \text{Pickup}, s.\text{Delivery} = s'.\text{Location} \\ p(s'.\text{Location}, s'.\text{Delivery}) & \text{if } a = \text{Move}(\text{City}_k), \text{City}_k = s'.\text{Location} \\ 0, & \text{otherwise} \end{cases}$$

where $p(s'.\text{Location}, s'.\text{Delivery})$ is the probability of receiving a package from s'.Location to s'.Delivery (or to not receive a package if $s'.\text{Delivery} = \text{NoPackage}$).

Rewards Table - As we try to optimize the profit per action of our agent, we need to fill our rewards table with the average profit of each action at each particular step. In order to calculate this, we simply use the expected reward of a particular task (if the action is "Pickup" and there is a task, otherwise we use 0) minus the cost associated with the selected travel.

Additionally, we use the rewards table to explicitly mark impossible actions we defined earlier. In order to do this and completely discourage our agent of undertaking such actions, we simply associated them with a reward of $-\infty$.

$$R(s,a) = \begin{cases} r\big(s.\text{Location}, s.\text{Delivery}\big) - c\big(s.\text{Location}, s.\text{Delivery}\big) & \text{if } a = \text{Pickup}, s.\text{Delivery} \neq \text{NoPackage} \\ -c\big(s.\text{Location}, \text{City}_k\big) & \text{if } a = \text{Move}(\text{City}_k), \text{City}_k \in N(s.\text{Location}) \\ -\infty & \text{otherwise} \end{cases}$$

where $r\big(\text{City}_i, \text{City}_j\big)$ is the *expected* reward of a task from City_i to City_j, $c\big(\text{City}_i, \text{City}_j\big)$ is the distance (in kilometers) between City_i and City_j multiplied by the cost per kilometer of the vehicle, and $N(\text{City}_i)$ is the set of neighbors of City_i.

1.2 Implementation Details

States and Actions - In our implementation, we first created `State` and `AgentAction` classes. In order to avoid redundant properties `State`'s `destinationCity` is `null` to indicate that there is no package (this is also consistent with Logist's `probability` functions). Similarly, `AgentAction`'s `moveCity` is `null` to indicate a "Pickup" action.

Reward and Probabilities - In order to implement

Skip Impossible Actions - In the basic version of `iterateQ` (our version of the MDP Learning algorithm), we iterated over a lot of impossible actions. In order to prevent this, we simply check and skip the iteration if the reward for the current state and action is $-\infty$. This allows us to skip approximately two thirds of iterations with no change in the final Q-Table.

2 Results

2.1 Experiment 1: Discount factor

2.1.1 Setting

For this first experiment, we run 4 agents using reinforcement learning at the same time, each with a different discount factor in [0.25, 0.5, 0.75, 0.95], for 3000 simulated time steps.

The results can be seen on Figure 1 and Table 1.

2.1.2 Observations

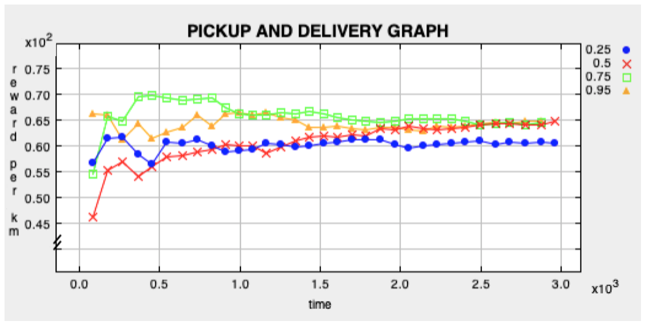
2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

In this second experiment, we run 3 different agents at the same time, again for 3000 simulated time steps.

Random : It accepts or denies every task with a 0.5 probability.

Greedy : It accepts every task it can and if there are none, it moves to the neighboring city with the maximum expectation $E = p_{city}(task) * E_R(city)$.



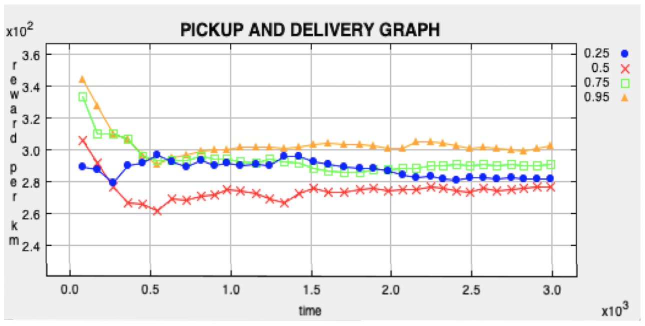
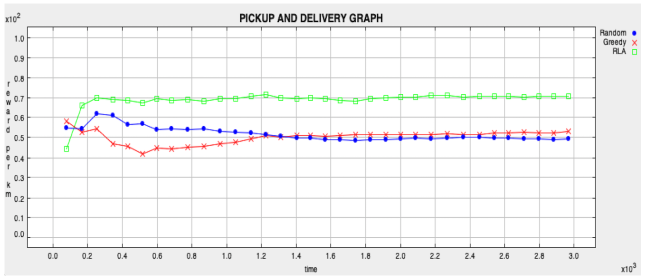


Figure 1: Reward per km over time for agents with different discount factors. On the left, in France and on the right, in Switzerland.



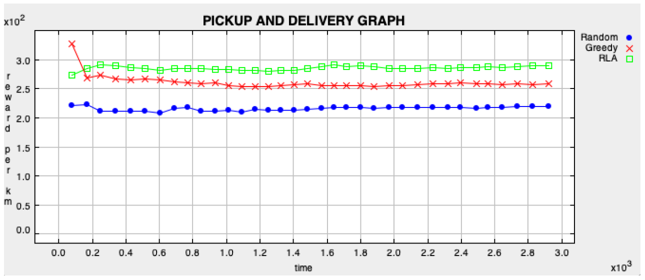


Figure 2: Reward per km over time for agents with different strategies. On the left, in France and on the right, in Switzerland.

Reinforcement Learning : The agent we implemented in part 1, with a discount factor of 0.95.

The results can be seen on Figure 2 and Table 2.

2.2.2 Observations

Discount \ Country	France	Switzerland
0.25	36339.81	47303.50
0.5	38854.97	46661.04
0.75	38705.52	47087.54
0.95	**39031.59**	**48065.10**

Table 1: Average reward per action after 3000 time steps for agents with different discount factors.

Agent \ Country	France	Switzerland
Random	31093.81	37188.36
Greedy	36944.03	43974.26
RLA	**39138.37**	**47759.41**

Table 2: Average reward per action after 3000 time steps for agents with different strategies.