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

Group №70: Jelena Banjac, Stratos Triantafyllou

October 8, 2019

1 Problem Representation

1.1 Representation Description

Our implementation assumes as state a route from an origin city to a destination city. The next city the vehicle will visit next is modeled as an action. If a vehicle on a route is carrying out a delivery task, no destination city is defined - implemented with the method `hasTask()`.

This flag is used in the creation of the reward table. If the vehicle is carrying out a delivery task next, then the added reward R(s, a) is computed based on the income for the delivery from the destination city in s to the next city as defined in the action a, subtracting the distance between these two cities. If the vehicle is just moving to the next city without delivering a task, we just subtract the distance between the cities.

To compute transition probabilities, if the vehicle is carrying out a delivery task, we use the probability that the agent moves between the origin and the destination city. If the vehicle is just moving to a neighbor city, each neighbor has the same probability and the transition is random.

1.2 Implementation Details

Implementation of the state is in the `state.RouteState` with parameters `fromCity` and `toCity`. The initialization of all the states is done in the method `initializeStates`. In total we have N^2 states. The action is implemented in the class `action.RouteAction` with parameter `neighborCity`. The method `initializeActions` is initializing all the actions. In total we have N actions. Before Reinforcement Learning Algorithm (RLA), we fill in the Reward table - `R(s,a)`. Reward is calculated differently depending whther the agent has a tas or not. We also initialize Q-table - `Q(s,a)` together with the vector with state values and corresponding action to take in that state - `V`. When Q table is initialized, it does not use all the combinations of state and action. The constraints are: the agent cannot stay in the same city and if there is no delivery, the action needs to be a neighbor city. These tables are all located in the `tables.Tables`. We compute this vector in method `rla()` by value iteration. This value improves with each iteration, and our stopping criteria is until the difference between two successive iteration is smaller than ϵ=0.001.

2 Results

In this section, we will describe several results from the experiments with our reactive agent.

2.1 Experiment 1: Discount factor

In the first experiment, we want to understand how discount factor γ influences the results. The discount factor represents how much we discount the future, i.e. how much an agent is taking care of rewards from

the future states. If the γ=0, it means that we discard the future totally. Whereas, the higher the value of γ, the higher the effect of the future on the value of state value V(S).

2.1.1 Setting

This experiment was performed with the following settings: (1) discount factor for our reactive agent: γ ϵ [0.15, 0.50, 0.85, 0.95, 0.99]; (2) cost per km for all vehicles is 5 CHF; (3) capacity of all the vehicles is 30 KG; (4) speed of all the vehicles vehicle is 220 KM/H; (5) topology is Switzerland.

2.1.2 Observations

Number of iterations of RLA algorithm until good enough is: [6, 9, 11, 9, 7] for the discount factors γ ϵ [0.99, 0.95, 0.95, 0.85, 0.50, 0.15], respectively. We run several times (3 times) in order to get a confidence intervals of total profits and reward per kilometer results. Using this information we can find the best value of γ.

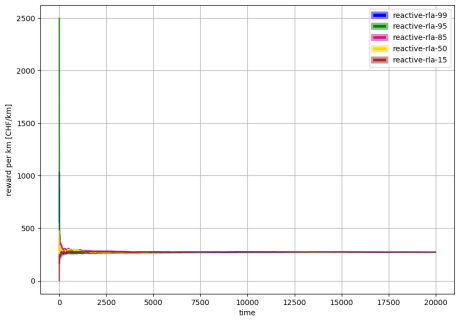

Figure 1: Reward per kilometer - full image

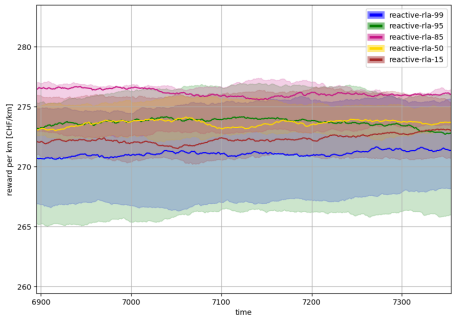

Figure 2: Reward per kilometer - zoomed

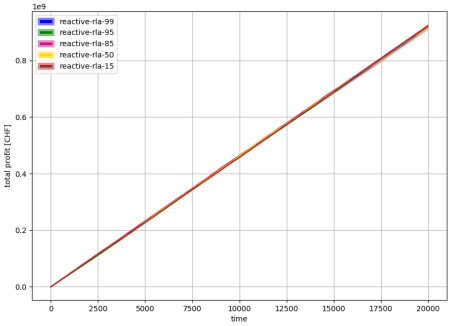

Figure 3: Total profit - full image

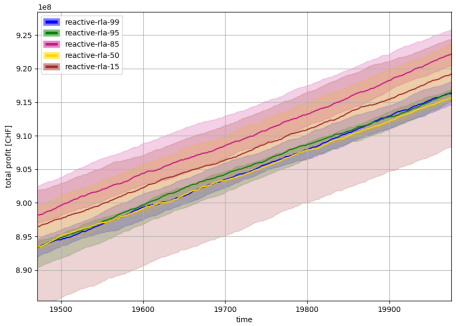

Figure 4: Total profit - zoomed

From the images above, we can see that with our settings the agent that performs the best is the one with the discount factor $\gamma = 0.85$. It means that it takes into an account the future pretty highly.

2.2 Experiment 2: Comparisons with dummy agents

In this experiment, we compare the results of our agent with two dummy agents: the random agent in the starter files and another dummy agent that has γ=0. We perform this experiment on different typologies.

2.2.1 Setting

The first dummy agent is a random agent does not implement the Reinforcement Learning Algorithm (RLA) and chooses the next path to move randomly. The second dummy agent is the agent that discards

the effect of the future states, therefore, it acts only to maximize an immediate reward. The third agent is our final agent that performs the best. Additional settings we have are: (1) cost per km for all vehicles is 5 CHF; (2) capacity of all the vehicles is 30 KG; (3) speed of all the vehicles vehicle is 220 KM/H

2.2.2 Observations

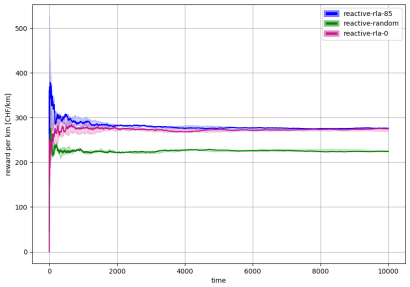

Figure 5: Switzerland topology - Reward per kilometer (full image)

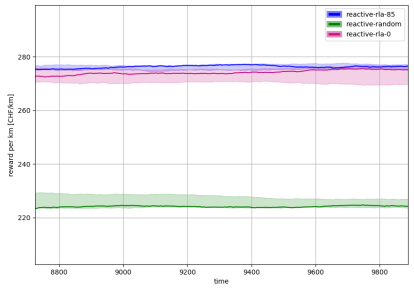

Figure 6: Switzerland - Reward per kilometer (zoomed)

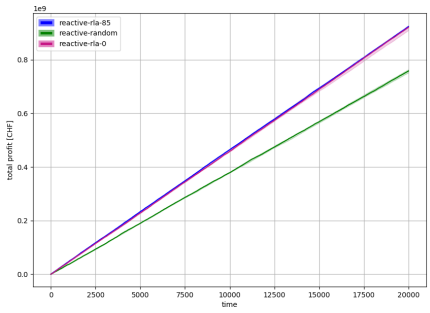

Figure 7: Switzerland - Total profit (full image)

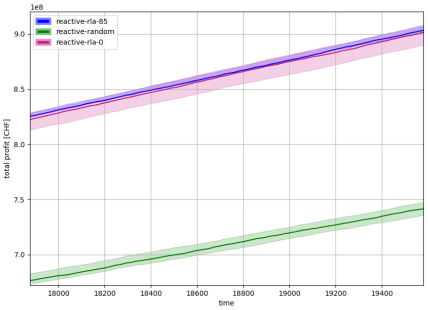

Figure 8: Switzerland - Total profit (zoomed)

From the images of the performance of agents in Swiss topology, we can see that random agent (green) performs worse than the ones that implement the RLA, see Fig.5, Fig.6, Fig.7, and Fig.8. Even the agent that is using the discount factor γ=0 is performing better than the random one (which is expected since it is maximizing an immediate reward and does not go randomly). The agent which we think was the best was the one with γ=0.85 (blue).

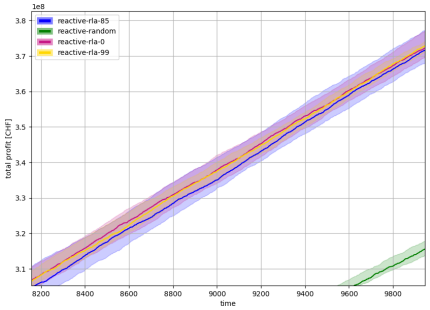

Figure 9: France - Total profit (zoomed)

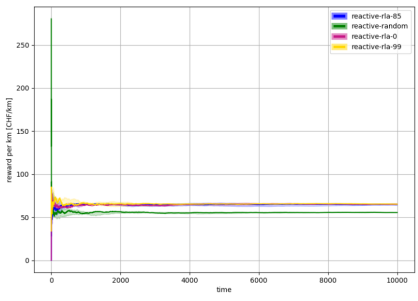

Figure 10: France - Reward per kilometer (full)

However, when we ran the agents in a different topology (France), we got that the random one still performs the worse, however, the agent with γ=0.95 performs better than the one with γ=0.85. And the agent that gets the immediate reward has performance with the result in between these agents, see Fig.9 and Fig.10. We did the same for the topology of England. The discount factor that performs the best there was γ=0.5. More information can be found in the notebooks.