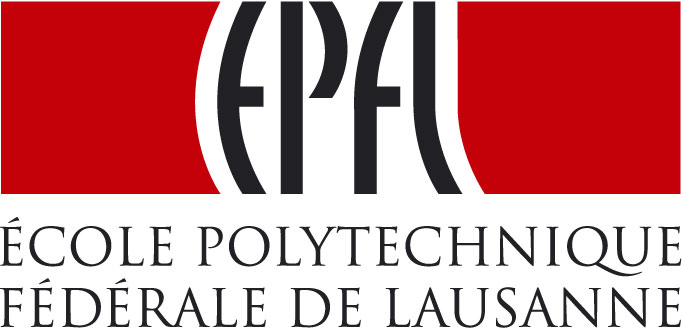
Assignment 2

Programming: Implementing a Reactive Agent





Intelligent Agents Course 2009/2010

Radoslaw Szymanek

Brammert Ottens

G10

Bello Ruiz, Javier

Pocard Du Cosquer De Kerviler, Tugdual

A Reactive Agent for the Pickup and Delivery Problem

In this exercise we have learnt how to use a reactive agent to solve the Pickup and Delivery Problem. We have implemented a RLA (reinforcement learning algorithm) in order to compute an optimal strategy off-line, which is going to be used by the agent to decide the actions that it should make to pickup and delivery tasks when travelling through the network.

At first we defined on paper the following state representation of the world:

State= {(currentCity, destinationCity)}

It means that a task is available to pick-up in the current city and to be delivered at the destination City. If destinationCity==null, it means that there is no task available in this city and the agent should move to another city.

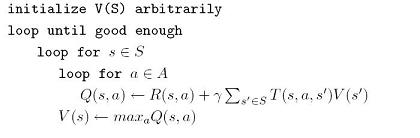
|State| = numberOfCities\*(numberOfCities +1)

Action= {move(anotherCity), pick-up}

The agent can decide between to move to another city or to pick-up a task if it is available in the city, together with the corresponding rewards, and the probability of the transition.

|Action| = numberOfCities+1

In the Markov Decision Processes the state transitions are deterministic. We build a transition table based on a state transition function: T(s, a, s’) = p(s’|s, a). The goal of this process is to find a strategy based on a reward function R(s, a), in which the average reward is maximized. To compute the best action that should be allocated in the vector V(S), with its corresponding accumulated value to take from a specific state we applied a reinforcement learning algorithm following the Markov Decision Processes to learn an optimal strategy:



We have the following variables:

* Table P(i,j) which indicates the probability that in city i there is a task to be transported to city j.
* Table AR(i,j) gives the average reward rij for a task to be transported from cityi to cityj.
* Table R(s,a) indicates the rewards for taking action and being in state s:

(getRewardPerKm(currentCity,destinationCity)–costPerKm)\*shortestDistanceBetween(currentCity,destinationCity)

In which getRewardPerKM is 0 if there is no task in the city or the action is move.

* Table T(s,a,s') which defines the probability to arrive in state s' given that you are in state s and that you take action a, or in other words: T(s,a,s') = p(s'|s,a).

If we can go from s to s’ with action a (condition1)

T(s,a,s’)= P (s’.currentCity, s’.destinationCity).

Else

T(s,a,s’)= 0

With (condition1) = (a.type==MOVE)

? (areNeighbors(s.currentCity,s’.currentCity)&&(a.destination==s’.currentCity))

: ((destination!=null)&&(s.destinationCity=s’.currentCity))

* The discount factor gamma should be a chosen value between 0 and 1, preferably near to 1 in order to ensure that the algorithm converges. We have chosen gamma= 0.8

We have also three xml configuration files: topology.xml, reactive.xml, tasks.xml. The topology configuration file specifies the routes, the tasks configuration file specifies probabilities/tasks and the reactive.xml specifies the framework setup, for example the classpath to behaviors, number of agents and their parameters. So we can choose the argument xml file: reactive-ex1.xml, reactive-ex2.xml and reactive-ex3.xml.

As it is written in the assignment we considered these assumptions,

* The vehicle starts from its home city. The vehicle can go freely through the network.
* The vehicle sees at most one task if is available when it arrives in a city.
* The vehicle can decide to deliver an available task or give it up and continue towards a different destination. Tasks that are not picked up disappear.
* There is a constant creation of new tasks.
* The vehicle can transport only one task. When a vehicle has accepted a task, it must deliver it on the shortest path.

When the agent receives the setup signal, the reinforcement algorithm computes the strategy. After learning, the agent begins to move through the network and when it receives inCity signal, it will choose the best action based on current state from V(S).

After running simulations of one, two and three agents using our optimally learned strategy V(S), these are the results, taking in account that in the long run agents will start to loop:

* One agent:
* Two agents:
* Three agents: