A Reactive Agent for Pickup and Delivery

Junxiong Wang, Zixuan Liang 10/12/2015

State Representation & Action

We define states of the reactive agent as follow:

1. $isTask$: Whether there is a task in a certain state.
2. $taskFrom$: Current location of the agent. If $isTask$ is $true$, it is also departure city of the task.
3. $taskTo$: If isTask is $true$, it is the destination city of the task. If $isTask$ is $false$, it is $null$.

In a word, a state contains the following information: current location of the agent and whether there is an available task in current city, and if there is, the destination city of the task. Assume there are N cities in total. Then we have number of possible states

$$total\_states = states\_with\_task + states\_without\_task = {}^{N}P_2 + {}^{N}P_1 = N^2.$$

Possible actions for certain state is related to whether there is a task in such state.

1. If there is, possible actions include picking up the task and just moving to neighbor cities.
2. If there isn't, possible actions just include moving to neighbor cities.

Code Description

The code contains three classes:

1. $State.java$: It implements our state representation.
2. $AgentAction.java$: It implements agent action representation, including departure city of the action, destination city of the action and whether the agent picks up a task in this action.
3. $ReactiveTemplate.java$: It implements behavior of the reactive agent. First it constructs hash tables to store program data, including $\langle city, state\rangle$, $\langle state, possible\_action\rangle$,$\langle states, best\_action\rangle$, $\langle state, best\_value\rangle$, $\langle state, probability\rangle$ and $\langle action, reward\rangle$. Then it implements the state value iteration algorithm for this pickup and delivery problem until it converges, which means the values of the states remain unchanged between two iterations.

In $ReactiveTemplate.java$, the $setup$ method and $plan$ method implement the data structure and the value iteration algorithm respectively. The algorithms are shown in the next page as Algorithm 1 and Algorithm 2.

Algorithm 1 Construct Hashtables among Cities, States, Values, Actions and Rewards

Initialization: Reads current agent's discount factor. Reads all cities.
for $\forall A \in$ all cities **do**
  Initialization $taskProbability = 0$.
  for $\forall B \in$ all cities and $A! = B$ **do**
    Add $task : A \rightarrow B$ and its $probability$ to $\langle State, Probability \rangle$ table.
    Add $action : A \rightarrow Bor \rightarrow neighbors$ and its $reward$ to $\langle Action, Reward \rangle$ table.
    Add $probalibity$ from A to B to $taskProbability$.
    Add $state : task \quad A \rightarrow B$ and all $action$ above to $\langle State, Action \rangle$ table.
  end for
  Add $state : agent\_in\_A\_no\_task$ and $1 - taskProbability$ to $\langle State, Probability \rangle$ table.
  Add $action : A \rightarrow A's \quad neighbors$ and 0 to $\langle Action, Reward \rangle$ table.
  Add $state : A\_without\_task$ and $action : A \rightarrow A's \quad neighbors$ to $\langle State, Action \rangle$ table.
  Add A and the $state$ list above to $\langle City, State \rangle$ table.
end for

Algorithm 2 Compute States Value by Iteration

Initialization: Let $isConverge$ be $fasle$.
Let $\forall state \langle State, Value \rangle = 0 \quad \langle State, Best\_Action \rangle = null$
Loop until convergence
for $\forall state \in$ total states **do**
  for $\forall action \in action\_list$ **do**
    for $\forall next\_state \in state_list$ **do**
      Compute $this\_round\_value+ = discount\_factor * P(next\_state) * next\_state\_value$
    end for
    if $this\_round\_value > state\_value$ **then**
      $state\_value = this\_round\_value$
      update $\langle State, Value \rangle$ and $\langle State, Best\_Action \rangle$
    end if
  end for
end for

Simulation Result

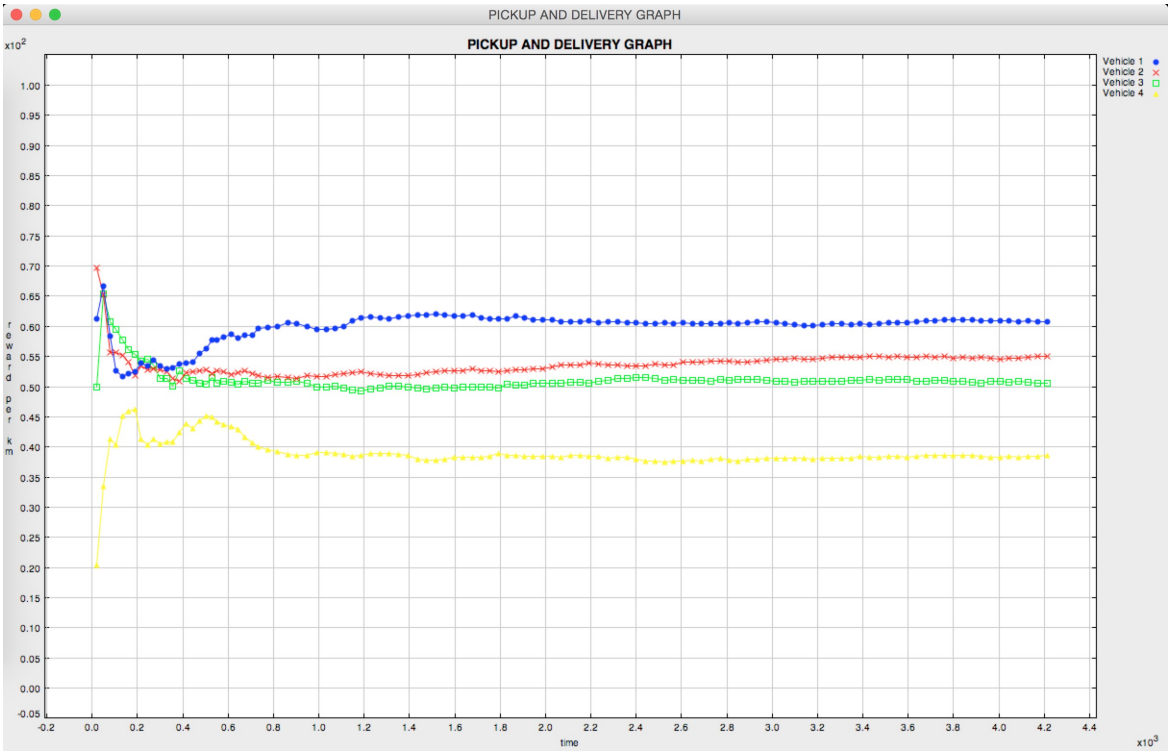

Figure 1. Average Reward of Three Reactive Agents and A Random Agent

Figure 1 shows simulation result, the average reward of three different reactive agents and a random agent over total travel distance. They have different discount factors (blue line: 0.85, red: 0.55, green: 0.10). The yellow line is the random agent that picks up available task randomly with probability of 0.95. As time goes by, we can see that the agent with a discount factor of 0.85 has the highest average reward, while the random agent has the lowest average reward. Also, the number of iterations increases as the discount factor gets bigger.

The result matches our prediction. Basically, when the discount factor becomes larger, the agent cares more about future state value. It will tend to ignore those tasks that have "unpromising future". In other words, it will refuse tasks that will lead to future states in which the probability of good tasks showing up is very low. Instead, it may choose a path that seems "worthless" at this moment, but it will have higher potential to discover good tasks along this path. And agents with lower discount factor tend to pick up available tasks "recklessly" and care less about future state value. Different discount factors lead to different actions and strategies, thus lead to different expected rewards.

Generally, from this exercise, we know that using value iteration algorithm to evaluate all possible states of the Markov Decision Processes we will get wiser strategies with better reward in the future. In this algorithm, the most important thing is to decide the discount factor which reflects how we value future state and it ideally should be near 1, though it will increase the number of iteration times.