Excercise 3
Implementing a deliberative Agent

Group 5: Ignacio Aguado, Dario Martinez

October 25, 2016

1 Model Description

1.1 Intermediate States

The State Representation should consider all this parameters, given a new state on its change:

Parameter	**Description**
`currentCity`	The city where the agent is located in this moment.
`capacity`	Free storing space left in the vehicle.
`costPerKm`	Cost of each kilometer driven.
`totalDistance`	Distance driven by the vehicle since it started moving in the initial city.
`benefits`	Cost of the last movement. $Rewards - Losses$
`totalBenefits`	Current balance of benefits for all the movements done.
`toPickUpList`	List of Tasks ready to be picked up.
`toDeliverList`	List of Tasks the vehicle is carrying.
`parent`	Parent State.

1.2 Goal State

The Goal State has been defined as a Final State, in a Final State no more tasks have to be delivered or picked up. Therefore, State is goal if it has the `toPickUpList` and the `toDeliverList` empty.

1.3 Actions

There could 3 possible actions:

- *Move*: The vehicle moves to a neighbor city. The agent transitions to a new state in which `currentCity` is the city the vehicle has moved to, the `benefits` are the cost of the movement ($-distance * $ `costPerKm`) and they are added to the `totalBenefits`. The distance is added to `totalDistance`. The remaining parameters are the same as the Parent State.

- *Pickup*: The vehicle picks up an available Task. The agent transitions to a new state in which the task is deleted from the `toPickUpList`, added to the `toDeliverList` and the weight of the task is subtracted from `capacity`. The remaining parameters are the same as the Parent State.

- *Deliver*: The vehicle delivers a task in the pertinent city. The task is deleted from the `toDeliverList` and the weight of the task is added to `capacity`, the `benefits` are the reward and they are added to the `totalBenefits`. The rest of the parameters remain the same as the Parent State.

2 Implementation

2.1 BFS

The Breadth-First Search algorithm has been implemented using a LinkedList Queue to preserve the order of addition of each State added to the "To be visited" list. The algorithm must iterate through all the states located in a certain depth level and add their possible next states to the end of the queue. Once that depth level is completed (all possible States have been visited) the Queue will have stored the next level States.

For every State visited it checks if it's a Goal State. In case it is, it checks the rest of States in the current depth level looking for a better result (more benefits). Once it's finished, it will return the best Goal State found.

2.2 A*

The A* Algorithm has been implemented with a TreeMap with the f(s) function Double as key and the State as value. Thanks to the TreeMap all the elements are always sorted by the key, so every time `pollFirstEntry()` is called the first element taken and removed will be the one with lowest f(s) value (as A* requires).

For every State visited, all its possible next States are added to the TreeMap, computating its correspondent f(s) value. The main difference with the BFS algorithm is that it returns the first Goal State it finds. The optimality of that State is determined by the Heuristic chosen.

2.3 Heuristic Function

The function f(s) of the State s is calculated as:

$f(s) = g(s) + h(s)$ being $g(s) = g(parent) + cost(s)$ and $h(s)$ the Heuristic chosen.

The two Heuristic defined in the model are *Distance* and *Benefits* and are set as a parameter from the configuration agents.xml.

2.3.1 Distance

For the Distance Heuristic:

- $g(s)$: total distance driven by the Agent from the Initial State to the Current State.
- $h(s)$: minimum distance to be driven to reach a Goal State. Calculated as the maximum distance to be driven to achieve the pickup/delivery of the Task with more distance from the current City.

This Heuristic is not admissible because it isn't monotone (it doesn't get necessarily smaller as the Agent approach to the Goal State), as the minimum distance is often not the real distance needed to finish all the goals. Knowing this, the A* Algorithm using the Distance Heuristic is suboptimal.

2.3.2 Benefits

For the Benefits Heuristic:

- $g(s)$: total cost accumulated from the Initial State ($cost(totalDistance) - benefits(rewardsDelivered)$)
- $h(s)$: minimum cost needed to reach a Goal State minus the benefits remaining to be picked up / deliveredin the topology ($minimumDistance * costPerKm - rewardsPending$).

As happenned before, the Heuristic is not monotone, so it can't be considered admissible and the A* is suboptimal.

3 Results

All the experiments has been performed in a MacBook Pro 2,5 GHz Intel Core i7 with 16 GB DDR3 RAM 1600 MHz.

3.1 Experiment 1: BFS and A* Comparison

The aim of this experiment is to compare the BFS and the A* algorithm (with two different heuristic functions) in terms of optimality, number of iterations, efficiency...

3.1.1 Setting

Algorithm	Heuristic	Topology	N Vehicles	Capacity	Cost/km	N tasks
BFS	-	Switzerland	1	20	5	7
A*	Benefits	Switzerland	1	20	5	567
A*	Distance	Switzerland	1	20	5	567

3.1.2 Observations

The BFS algorithm does 6419231 iterations in 31 seconds, resulting in 30 actions with a total benefit of 322824. Therefore, the maximum number of tasks for which a plan could be built is 7. The experiment with 8 tasks is not viable (at least with the computers available) as the algorithm took a huge amount of time to execute.

The A* algorithm with the distance as heuristic function does 15946 iterations in 8 seconds, resulting in 2042 actions with a total benefit of 35601635. The maximum number of tasks for this experiment is 567. The A* algorithm with the benefit as heuristic function does 16282 iterations in 8 seconds, resulting in 2044 actions with a total benefit of 35590485. The maximum number of tasks for this experiment is 567.

The most efficient algorithm is the A*. This is because the BFS just expands the search by one step on every iteration iteration, which happens to have the effect of finding the smallest number of steps it takes to get to any given node from the root. The BFS is more limited as the number of states and therefore iterations is huge compared with the A*. On the other hand, the BFS will always return an optimal strategy, this is a plan that maximizes the total benefit.

3.2 Experiment 2: Multi-agent Experiments

The aim of these experiments are to test the behaviour of the agents when there are interferences between then. They should be able to reconsider their plan when a task is no longer available because another agents has already picked it up.

3.2.1 Setting

Algorithm	Heuristic	Topology	N Vehicles	Capacity	Cost/km	N tasks
BFS	-	Switzerland	3	20	5	6
A*	Benefits	Switzerland	3	20	5	6
A*	Distance	Switzerland	5	20	5	6

3.2.2 Observations

As it can be showed by the logs in the Java console and the graphs displayed by the app, the agents behave as intended. They recalculate the plan when a task is no longer available and all the tasks are delivered without problem.