Excercise 3
Implementing a deliberative Agent

Group №3 : Yannick Grimault, Vincent Petri

October 25, 2016

1 Model Description

1.1 Intermediate States

Our States describe the current city the vehicle is in, and 2 TaskSets, respectively for available and carried tasks (we initially had a third TaskSet that was supposed to be used for the heuristic function, but we found another function that didn't use it, so we dropped it).

In our class State, we also implemented a method that returns a list of substates of this state. To compute them, we first compute the set of unique state where we delivered a carried task at the current city, as it has no cost and should always be done immediately. Then we compute all states where we moved to another city, or picked up a task available in this city as long as its weight doesn't exceed the maximum capacity of the vehicle.

1.2 Goal State

A State is a goal if and only if there are no more available tasks, and none are being carried. Thus, it needs to have availableTasks.isEmpty() and carriedTasks.isEmpty().

1.3 Actions

The different actions are move to another City (update currentCity), pickup a task (move it from availableTasks to carriedTasks), and deliver a task (remove it from carriedTasks).

2 Implementation

2.1 BFS

The BFS algorithm is implemented as described in the course, with the remark that we pick the first final goal we meet, ie the one with the least amount of Actions. We could have stored several final Plans and compare them to get the best one, but we deemed it not necessary. Other remarks can be found as commentaries in the code.

2.2 A*

For the A* algorithm, we also took it from the course, and works very similarly to the BFS algorithm, with the exception of the heuristic function.

2.3 Heuristic Function

Because the final state describes a state where all tasks are delivered, the cost of the plan is entirely defined by the distance travelled. Moreover, if we get 2 plans to get to the same state, the difference between the 2 plans is once again defined by the distance travelled. That's why we picked this function as our heuristic function, without taking into account the total reward of the tasks already delivered.

3 Results

3.1 Experiment 1: BFS and A* Comparison

3.1.1 Setting

We use the same topology for both algorithm. We increased one by one the number of tasks until more than 1 minute is required to compute the plan.

3.1.2 Observations

Let's compare first the computation time between both algorithms. A* needs 20 seconds to compute a plan for 6 tasks and 2 minutes for 7 tasks. BFS takes 15 seconds to compute a plan for 7 tasks and 1min 50 for 8 tasks. From this observation we can deduce that A* algorithm is slower to compute a plan leading to a final state.

We can also compare the reward for the same number of tasks. For 7 tasks, using the same seed:

- A*: $TotalReward = 330974$ and $TotalProfit = 321674$
- BFS: $TotalReward = 330974$ and $TotalProfit = 322424$

3.2 Experiment 2: Multi-agent Experiments

3.2.1 Setting

We used the same topology again. We choosed to pick 6 tasks with two agents.

3.2.2 Observations

With multiple agents the plan sometimes get incorrect. The agent try to pick a non existing task. Once this happen the agent has to recompute a plan. It causes the final plan to be non optimal.