Excercise 3
Implementing a deliberative Agent

Group №75: Thomas KIMBLE, Jules AFRESNE

October 22, 2019

1 Model Description

1.1 Intermediate States

Each State is represented by the following characteristics:

- *currentCity:* This is the city where the agent is currently found.
- *actionList:* This is a list of the actions previously taken to get to this state. Each action is added to this list to allow cost calculations and plan execution once a goal state is reached.
- *availableTasks:* This is a TaskSet containing the available tasks in a simulation. The agent has not picked up any of these tasks, they will have to be picked up and delivered.
- *transportedTasks:* This is a TaskSet containing the tasks to be delivered by the agent. These tasks have already been picked up and are waiting to be delivered.
- *currentWeightVehicle:* This is the weight that a vehicle is currently carrying. Thanks to it the free weight left in the vehicle can be computed.
- *gCost, hCost, fCost:* gCost is the cost function of the actions that lead up to the current state. For fCost and hCost see part 2.3 Heuristic Function.
- *isGoal:* This is a boolean that is false if we are in an intermediate state, and true if in a goal state.

1.2 Goal State

We are in a goal state once all tasks have been picked up and delivered. In our model this is the case when *availableTasks* and *transportedTasks* are both empty. If only *availableTasks* is empty we could still have tasks being transported, so we need to assure that *transportedTasks* is also empty. In this case the boolean *isGoal* is true.

1.3 Actions

We want to create a tree with the next possible states after the current one, and so on to figure out a plan. Therefore, from the current state we create a list of "children" states obtained by two possible actions: *Pick Up* and *Delivery*. The number of possible actions, and therefore the number of child states, is equal to the number of tasks being transported and remaining. Each possible action to get to a future state is added to the *actionList* of the said future state.

For the *Pick Up* action we scroll through the *availableTasks* TaskSet of the current state and first check if the task is too heavy. If so we do not proceed with this action. If not we proceed by picking up

the task or by going to a new city and picking it up. The vehicle weight is updated, we remove the new task from the *availableTasks* of the next state and add the new task to the *transportedTasks* of the next state.

For the *Delivery* action we scroll through the *transportedTasks* TaskSet of the current state and deliver a carried task to the current city or a new city. The vehicle weight is updated and we remove the new task from the *transportedTasks* TaskSet of the next state.

2 Implementation

2.1 BFS

First of all we create an initial state with all of the vehicle parameters and available tasks, given it an initial $gCost$ of zero. We declare four State Array Lists: an open list Q, the next layers open list $nextLayerQ$, a closed list C and a goal states list $goalState$. We implement the Breadth First Algorithm filling up these lists to find our Goal States in an optimal way. We sorted the $nextLayerQ$ list in order of increasing $gCost$ to optimise the algorithm.

To find the optimal solution, we compare the costs of all of the goal states and select the state with the minimal cost. This step requires more time and computational power, but it assures that the optimal solution is returned rather than the first found solution.

2.2 A*

As in the BFS algorithm we first create an initial state with all of the vehicle parameters and available tasks, given it an initial $gCost$ of zero. We declare two State Array Lists: and open list Q and a closed list C. We implement the A* algorithm with the heuristic function described in part 2.3 to assure that the optimal solution is found. We sorted the Q list in order of increasing $fCost$ to optimise the algorithm.

2.3 Heuristic Function

The heuristic function $h(n)$ corresponds to the estimated minimum cost to get from the current state to a goal state. Let us consider our model where we have three possible cases:

- One single possible action, a_1 = pick up a task. Here the goal state is the state after the next one. Therefore the heuristic function is equal to:

$$h(n) = (distanceToTask + distanceToDelivery) * costPerKm = h^*(n) \tag{1}$$

- One single possible action, a_2 = deliver a task. Here the next state is a goal state. Therefore the heuristic function is equal to:

$$h(n) = distanceToDelivery * costPerKm = h^*(n) \tag{2}$$

- k possible actions, $a_1, a_2, ..., a_k$, $k \in \mathbb{N}$ (pick up and/or delivery). Here the heuristic function:

$$h(n) = max(h(a_i)) \leq h^*(n), \quad i = 1, ..., k \tag{3}$$

In all three of these cases we assure that $h(n)$ under-estimates the true cost, but we try to get $h(n)$ as close as $h^*(n)$ to reduce the necessary number of iteration to find a goal state. This confirms that A* will always find the optimal solution.

3 Results

3.1 Experiment 1: BFS and A* Comparison

3.1.1 Setting

For Table 1 we observe a single deliberative agent using the BFS and A* algorithms in the **France topology** with four to eleven tasks. Each vehicle has a speed of 220, with a capacity of 30.

3.1.2 Observations

Table 1 shows us that for both algorithms the number of iterations and the execution time grow exponentially with each added task. However the A* algorithm is much faster and can perform with more tasks than the BFS. We also notice that the plan cost for each of the different number of tasks is the same for both algorithms. These observations confirm that A* is faster and uses less resources than BFS, however in both cases we find the optimal solution.

Number of Tasks		4	5	6	7	8	9	10	11
BFS	Iterations	490	2340	9997	37 808	141948	-	-	-
	Time [s]	0.017	0.038	0.399	3.355	48.317	-	-	-
	Cost	15 495	15 495	21 405	21 405	21 405	-	-	-
A*	Iterations	44	103	478	973	3 132	27 636	43 861	138 662
	Time [s]	0.007	0.012	0.035	0.057	0.192	7.75	23.548	286.713
	Cost	15 495	15 495	21 405	21 405	21 405	23 955	23 955	23 955

Table 1: Performance comparison between BFS and A* (number of iterations, execution time and cost)

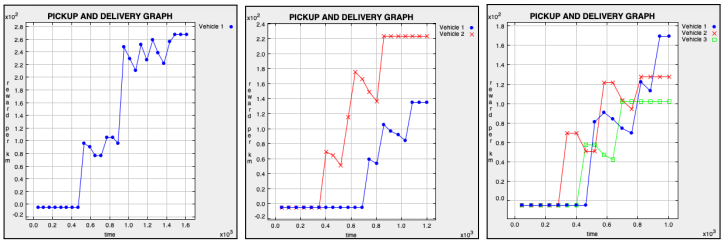
3.2 Experiment 2: Multi-agent Experiments

3.2.1 Setting

We observe 1, 2 and 3 deliberative agents using the A* in the **Switzerland topology** with 9 tasks. Each vehicle has a speed of 90, with a capacity of 30. We plot the Reward per km as a function of time.

3.2.2 Observations

If the current plan is no longer possible, we calculate a new plan and we notice that each new calculated plan is obtained faster because there will be less tasks available. We also notice that the more agents there are, the faster the tasks are completed, indeed $t_a = 14.8s$, $t_b = 10.8s$ and $t_c = 9.3s$. Nevertheless agents may lose their efficiency because each one computes its own plan without considering the other's plan. This depends a lot about the configuration especially the number of tasks to deliver and the initial position of vehicles. We calculate the average reward per second : $R_a^{avg} = 0.18$ per s, $R_b^{avg} = 0.19$ per s and $R_c^{avg} = 0.16$ per s. In our case, the model where there are two agents is the most effective.



(a) Single agent (b) Two agents (c) Three agents