Exercise 3
Implementing a deliberative Agent

Group №9: Fengyu Cai, Wanhao Zhou

October 21, 2019

1 Model Description

1.1 Intermediate States

For each vehicle, we define the state representation as follows, and the first three properties uniquely define a state.

- `currentCity`. The city where the vehicle is currently in.
- `taskOnBoard`. Tasks the vehicle has already undertaken.
- `taskAvailable`. The remaining tasks to be undertaken (picked up).
- `spaceLeft`. The remaining space for the vehicle to pick up tasks.
- `cost`. I.e., $g(n)$, the cost paid so far (the distance travelled times the cost of the vehicle per km)
- `heuristic`. I.e., $h(n)$, the estimated cost to the final state.
- `totalCost`. I.e., $f(n)$, which equals to $g(n) + h(n)$.

1.2 Goal State

The goal state is achieved if and only if there is no more `taskOnBoard` and no more `taskAvailable`. In our searching algorithm, we would consider this as the final state. Typically, in A*, when we reach the goal state, the algorithm will directly return the plan.

1.3 Actions

We run the algorithm directly on the states described above to search for an optimal state transition (from the initial state to the goal state) which yields the lowest cost.

Between the neighbouring states, we can define the actions that lead from the previous state to the next state. Actions include:

- `Move(nextCity)`. This action is chosen when two neighbouring states have different `currentCity`.
- `Pickup(task)`. This is chosen when the `taskAvailable` fields are different, meaning that there are fewer tasks available, since one of them is already picked up.
- `Delivery(task)`. Similarly, this is chosen when the `taskOnBoard` fields are different.

2 Implementation

2.1 BFS

We implemented two versions of BFS algorithm, namely `BFS` and `BFS_FAST`. The only difference is in the cycle detection part. BFS runs with the `Queue`. While the queue is not empty, the algorithm continues to run. When

the first element of the queue is polled out, we would check if the state is final or not. If it is final we will record its final cost of the plan, and update the current best plan. The algorithm terminates when the queue is empty.

For the cycle detection, the slower BFS uses the technique similar as that of A*, meaning that we regard a state as redundant if there is already a copy in our `HashMap` and the current one has a cost greater than its previous copy. `BFS_FAST` will not consider any states that have been visited, irrespective of the cost. Indeed, `BFS_FAST` is the same as the algorithm given in the slides.

2.2 A*

A* runs with the heuristic function as an estimate of the cost till the final state. There are three major differences from BFS.

- Instead of using the `Queue`, which polls element out in the order that is added, the algorithm uses `PriorityQueue` and set the key as the `totalCost`. `PriorityQueue` maintains a Min-Heap that each time we poll out the first element, it will be the state that has the lowest `totalCost`.
- Like the slower version of BFS mentioned above, we maintain a `HashMap` that matches a state to its cost. We discard states that already have a copy in `HashMap` and the current one's cost is higher.
- Once we poll out a goal state from the heap, the algorithm terminates and returns the plan.

2.3 Heuristic Function

Let T_A denotes all tasks available of the current state, and T_B denotes all tasks already undertaken. Our estimation heuristic of the current state s is calculated via the following formula:

$$h(s) = \max_{t \in T_A \cup T_B} \text{CostToGoal}(t)$$

CostToGoal is calculated separately for T_A and T_B. If a task is not undertaken, we calculate the distance from current city to the pickup city plus the distance from the pickup city to delivery city, while for tasks on board, we only consider the distance from current city to the delivery city. The final distance times the cost per km gives the cost.

We can easily verify that $h(s)$ underestimates the actual cost to the final state. From the triangle inequality we know that $h(s)$ equals to the actual cost if and only if all tasks' pickup city and delivery city are on the same line from the current city, and $h(s)$ will be lower otherwise. Since $h(s)$ underestimates the cost, this gives us admissibility and optimality of the heuristic function.

3 Results

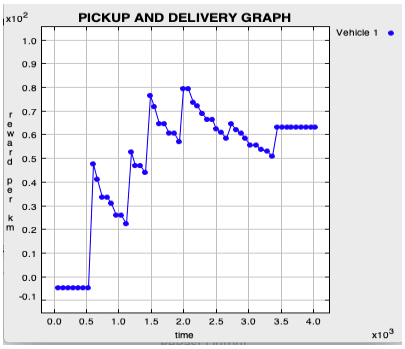
3.1 Experiment 1: BFS and A* Comparison

3.1.1 Setting

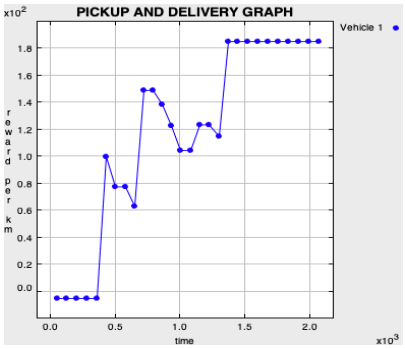
We set the seed as 23456. For different number of tasks, `BFS`, `BFS_FAST` and `ASTAR` yields the same optimal plan (w.r.t the cost of the whole plan, i.e. the total traveling distance), given in Table 1. Also

# Tasks	`BFS`	`BFS_FAST`	`ASTAR`	Total Distance
6	3.484s	0.941s	0.114s	1380
7	13.124s	3.521s	0.384s	1610
8	124.124s	28.036s	1.764s	1710
9	N/A	N/A	4.527s	1720
10	N/A	N/A	41.282s	1820

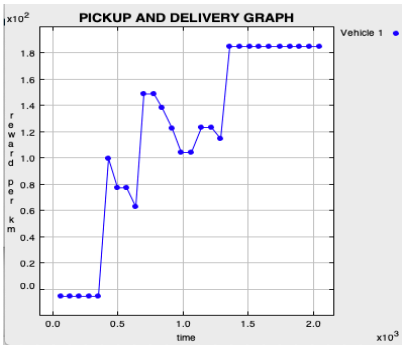
Table 1: Runtime comparison of different algorithms



(a) Reward per km for random plan



(b) Reward per km for BFS plan



(c) Reward per km for ASTAR plan

Figure 1: Experiment among different plan

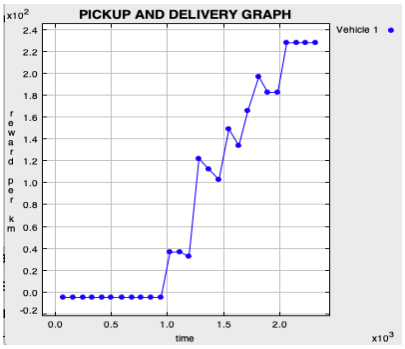
3.1.2 Observations

From Table 1, we could find out the plan generated by `BFS`, `BFS_FAST`, and `ASTAR` are the same, and execution time increases exponentially for all of them. `ASTAR` is much faster than `BFS`. Also, from the Figure 1, both `BFS` and `ASTAR` greatly improve the delivery gain when the number of task is 6 compared with the naive plan.

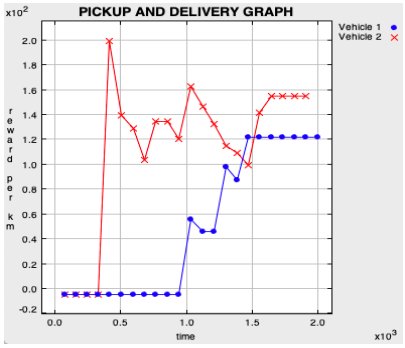
3.2 Experiment 2: Multi-agent Experiments

3.2.1 Setting

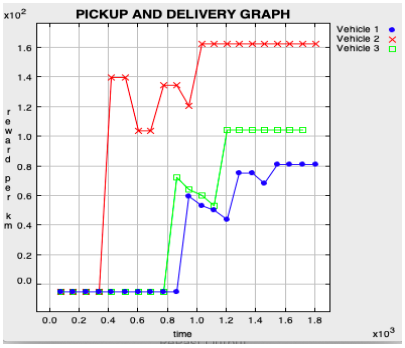
When the agent observe the change of the environment (pickup or delivery tasks in its plan), it will automatically re-generate the plan based on the current circumstance. Here, we set the total number of task is 8, and the default planning algorithm is `ASTAR`.



(a) Reward per km for agent-1



(b) Reward per km for agent-1 and agent-2



(c) Reward per km for agent-1, agent-2, and agent-3

Figure 2: Experiment for different number of agents

3.2.2 Observations

From the experiments above, we find that with the increasing number of agents involved, the total time spent will be decreased, and the sum of gain per km will be higher.