Excercise 3
Implementing a deliberative Agent

Group №21: Dubravka Kutlešić, Nevena Drešević

October 20, 2020

1 Model Description

1.1 Intermediate States

The state of the agent is determined by the following:

- Current city - the city in which the agent is currently at
- Carried tasks - the tasks the agent has already picked up
- Available tasks - the tasks that are still left to be picked up in the world
- Vehicle - the vehicle that transports tasks of the agent

1.2 Goal State

The goal state is the state where the agent does not carry any task and there are no available tasks on the map (i.e. all tasks have been delivered).

1.3 Actions

The action of the agent is the transition of the agent from the city it is currently at, to the city it can either deliver task/tasks, pick up some task/tasks, or do both.

2 Implementation

Generating all possible states and their corresponding transition actions is implemented as a directed graph structure in the class *SearchAlgorithm*. The graph's nodes represent states and branches are possible actions from the current state to the next state. There exists a branch directed from $S1$ to $S2$ if $S2.currentCity$ is a destination city for at least one task in $S1.carriedTasks$ or a pickup city for at least one task in $S1.availableTasks$. When an agent arrives at the city in state $S2$ it will always deliver all carried tasks intended for that city. Since the agent can pick up multiple tasks at the new arriving city, all permutations of subsets of the new arriving city's pickup tasks that can fit into the vehicle are calculated and created as succeeding states.

The optimal traversal path for delivering all tasks is determined based on the type of search used, implemented by classes BFS and $AStar$. For $AStar$ heuristics is computed every time the graph is computed, while it is recomputed if the traversal plan changes (again with graph recomputation).

2.1 BFS

In the BFS algorithm, one hash set of states (visited) and one queue of states (queue) are used. Visited contains all states that have already been seen in the search. Queue preserves the order in which the states should be visited.

2.2 A*

In the A* algorithm, one priority queue of states (Q) and two hash sets of states (C and Qin) are used. In Q, states are stored in min heap depending on the value of $f = g + h$, where g is the shortest path from the root to the state found so far in the search and h is the heuristic value for the state. The state would be in the hash set C if processed in the algorithm as the state that currently has the lowest f value. Since the quick check if the state is presented in the min-heap Q is needed, the additional structure Qin is added. The state is in Qin iff it is in the Q.

2.3 Heuristic Function

Let $d(a, b)$ be the shortest distance between cities a and b. Let *cost* be the cost per kilometer for the vehicle. The cost for traveling from a to b via shortest path is $cost * d(a, b)$. Since the agent does not pickup or deliver tasks while moving from state s_a and its current city a to the state state s_b and its current city b, the cost for traveling from state s_a to s_b is $c(s_a, s_b) = cost * d(a, b)$.

Let the $t.s$ be the pickup city for the task t and $t.d$ be the delivery city for task t. For a given state s and its current city x, carried tasks CT and available tasks AT, the heuristic is calculated as follows : $h(s) = cost * \max\{\max_{t \in CT} d(x, t.d), \max_{t \in AT}(d(x, t.s) + d(t.s, t.d))\}$

2.3.1 Heuristic optimality

Let $h_s(t)$ be the travelling cost from the state s to the state where the agent delivers the task t in the optimal path. For each $t \in CT$, the travelling cost for the delivery from the state s in agent's optimal path cannot be lower than the lowest possible cost of the travelling from the city x to $t.d$, i.e., $h_s(t) \geq cost * d(x, t.d)$. Similarly, $\forall t \in AT$, the travelling cost for the delivery from the state s in agent's optimal path cannot be lower than the lowest possible cost of the travelling from the city x to $t.s$ and then from $t.s$ to $t.d$, i.e., $h_s(t) \geq cost*(d(x, t.s)+d(t.s, t.d))$. Let h_s^* be the travelling cost from the state s to the goal state in optimal path. Because the agent must deliver all tasks, it holds that $h_s^* \geq h_s(t), \forall t \in CT \cup AT$. Since the left-hand side of the inequality does not depend on t, h_s^* is greater than the maximum of all right-hand sides, i.e., $h_s^* \geq cost * \max\{\max_{t \in CT} d(x, t.d), \max_{t \in AT}(d(x, t.s) + d(t.s, t.d))\}$. Therefore, the heuristic is admissible and the A* will find the optimal path from the root state to the final state.

3 Results

3.1 Experiment 1: BFS and A* Comparison

In this experiment, the performances of BFS and A* are compared. The machine where the tests are run has 16 GB RAM and 2.6 GHz 6-core Intel i7 processor.

3.1.1 Setting

For the experiment, the Switzerland topology is used and the rngSeed is 81881. There is only one agent in the world. For the same setup, BFS and A* are compared, varying the number of initial tasks.

3.1.2 Observations

It can be observed that the computational time of the BFS algorithm is always better than A*. However, the BFS traverses through all possible states, while A* uses its heuristic to find an optimal path in significantly less traversed states. The maximum number of tasks for which the plan can be built in less than one minute is 11 for BFS and 10 for A*.

	Number of tasks	7	8	9	10	11	12
	Graph creation [s]	0.1	0.25	0.84	4.56	16.32	116.29
BFS	Traversal time [s]	0.02	0.09	0.37	1.42	3.86	14.82
	Total execution time [s]	0.13	0.36	1.22	6.0	20.20	131.13
A*	Traversl time [s]	0.01	0.25	1.71	16.77	137.18	Timed out
	Total execution time [s]	0.12	0.54	2.49	20.38	152.21	Timed out
	Traversed states	819	4 487	26 648	64 977	151 889	x
	Number of states	6 199	20 467	74 035	229 123	565 374	1 588 037

Table 1: Comparison of BFS and A* computational time and number of traversed states

3.2 Experiment 2: Multi-agent Experiments

This experiment examines the effects of having multiple deliberative agents in the world.

3.2.1 Setting

In the experiment, the Switzerland topology is used and there are 1, 2 or 3 agents in the world. The rngSeed is 81881.

3.2.2 Observations

The tests were run in different scenarios. The plots below show 1-3 BFS agents.

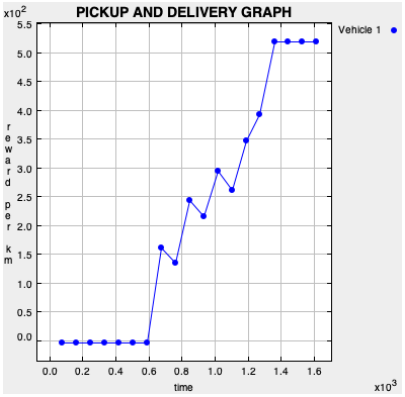

Figure 1: Reward during time: 1 agent and 7 tasks

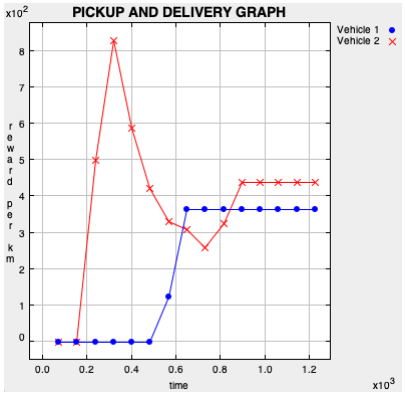

Figure 2: Reward during time: 2 agents and 7 tasks

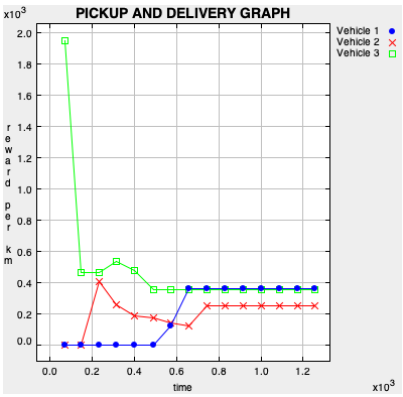

Figure 3: Reward during time: 3 agents and 7 tasks

The total cost when there is only one agent is 5250, when there are 2 agents 6700 (2750, 3950) while the cost with 3 agents is 9700 (costs of agents are 2750, 3450, and 3500). Agents are not working together, but executing actions that cause the inconsistency of plans and its recomputation. In conclusion, when there are multiple agents, they affect each other's plans, making the whole path of delivering tasks longer and less efficient.