# Exercise 3
# Implementing a deliberative Agent

Group №90: Ruben Janssens, David Resin

October 22, 2019

## 1 Model Description

### 1.1 Intermediate States

A state consists of the following three components:

- **Current City:** the city the vehicle is currently in.
- **City Map:** a structure which maps all the cities in the country to a list of all tasks that are waiting in that city to be picked up.
- **Carried Tasks:** a structure which maps all the cities in the country to a list of all tasks, which have to be delivered to that city, that the vehicle is currently carrying.

### 1.2 Goal State

A goal state is a state in which both the City Map and the Carried Tasks fields are empty, i.e. there are no more tasks left to be picked up, and the vehicle is not carrying any tasks.

### 1.3 Actions

The vehicle can take the following three basic actions:

- **Pickup:** pick up as many tasks as possible in this current city, as long as the capacity of the vehicle is not surpassed. Only possible if there is at least one task in the city that does not have a larger weight than the remaining capacity of the vehicle.
- **Deliver:** deliver all tasks that the vehicle is currently carrying that have the current city as their destination. Only possible if the vehicle is currently carrying at least one task with the current city as its destination.
- **Move:** move to a specific city. The vehicle can only move to cities that are neighbouring the current city.

## 2 Implementation

### 2.1 BFS

This algorithm uses four data structures to keep track of its (intermediate) results:

- `Q`: the queue of all states that the algorithm still has to visit. Every state is represented as a pair, containing the state itself and the plan, which contains all actions that were taken to get to this state.
- `C`: a list of all states the algorithm has already visited.
- `bestPlans`: a map, which maps every state the algorithm has already seen, to the cheapest plan the algorithm has already found to reach that state.
- `goalStates`: a list of all goal states the algorithm has already encountered.

The algorithm starts by constructing the initial state, by filling the current city with the city the vehicle is currently in, populating the City Map with the tasks that were given to the algorithm, and populating the Carried Tasks with any tasks the vehicle was carrying, in case the algorithm is executed after the agent got stuck on an action. The initial state is then added to `Q`, paired with the empty plan.

For every state that is taken from `Q`, the algorithm goes through three steps. First, it checks whether this state is currently contained in `C`, i.e. whether it already visited this state. If so, the algorithm checks whether the plan that was paired with this state in `Q` (which we call `newPlan`) is cheaper than the current plan that this state is mapped to in `bestPlans` and updates `bestPlans` if needed. Then, the next state is pulled from the queue. If the algorithm had not yet visited this state, the state is mapped to `newPlan` in `bestPlans` and is added to `C`, and the algorithm continues.

Next, the algorithm checks if this state is a goal state. If so, the state is added to `goalStates` and the next state is taken from `Q`. Otherwise, all successor states from the state are calculated and added to `Q`, paired with the plans which are constructed as the plan which is mapped to the current state in `bestPlans`, to which the action that leads to that successor state is appended.

When `Q` is empty, the algorithm iterates over all states in `goalStates` and chooses the state with the lowest cost plan mapped to it in `bestPlans`. That plan is then returned.

### 2.2 A*

Due to time constraints, we unfortunately didn't manage to implement A*. Sorry for this.

### 2.3 Heuristic Function

## 3 Results

### 3.1 Experiment 1: BFS and A* Comparison

For some reason the system refuses to handle big number of tasks, and we unfortunately could not figure out why in time.

#### 3.1.1 Setting

#### 3.1.2 Observations

### 3.2 Experiment 2: Multi-agent Experiments

#### 3.2.1 Setting

The experiment is run on the Switzerland topology, with 6 tasks, distributed with the RNG seed 23456. Every vehicle has the same characteristics, which are the same as in the original configuration. The experiment is being done with the BFS algorithm.

### 3.2.2 Observations

When running the experiment with one agent, the total cost of the plan is 1380.

When running the experiment with two agents, the first agent will incur a cost of 1310, and the second agent 960, for a total of 2270.

When running the experiment with three agents, the first agent will incur a cost of 690, the second agent 960 and the third 790, for a total of 2440.

We see that the total cost of all agents combines only rises with the number of agents, while the reward stays the same (the task distribution is not different). This was to be expected, as there is no coordination whatsoever between the agents and thus the agents will actually hinder eachother.