A Deliberative Agent for the Pickup and Delivery Problem

In this exercise, you will learn to use a deliberative agent to solve the Pickup and Delivery Problem. A deliberative agent does not simply react to percepts coming from the environment. It can build a plan that specifies the sequence of actions to be taken in order to reach a certain goal. A deliberative agent has goals (e.g. to deliver all tasks) and is fully aware of the world it is acting in.

Unlike the reactive agent, the deliberative agent knows the list of tasks that must be delivered. The deliberative agent can therefore construct a plan (a certain path through the network) that guarantees the optimal delivery of tasks.

Whenever a deliberative agent is called to execute a move, it applies the following internal planning process :

1: **if** (Current plan is not applicable anymore) **then**
2:     Compute optimal plan
3: **end if**
4: Execute the next action in the plan

State-based search algorithm

In this exercise you will use a state-based search algorithm for finding the optimal plan. The optimal plan minimizes the agent's overall cost for delivering a set of tasks. The cost is computed by multiplying the total distance traveled by the vehicle with the cost per kilometer of the vehicle. When you build a plan, you must consider the following constraints :

— The vehicle must deliver all available tasks.
— The vehicle can transport multiple tasks at the same time as long as the sum of the weights does not exceed the capacity of the vehicle.

You will first choose an adequate state representation of the world, that you will then test using a breadth-first search with cycle detection. You will then implement an A* heuristic search.

Several deliberative agents

A serious limitation of deliberative agents is their lack of speed when faced with uncertainty. Every time the situation is different from the one anticipated by the agent, the

plan must be recomputed. Computing a plan is computationally intensive and therefore, efficient planning algortihms should be used.

When more than one deliberative agent is present in the environment, the lack of coordination can lead to a very inefficient outcome.
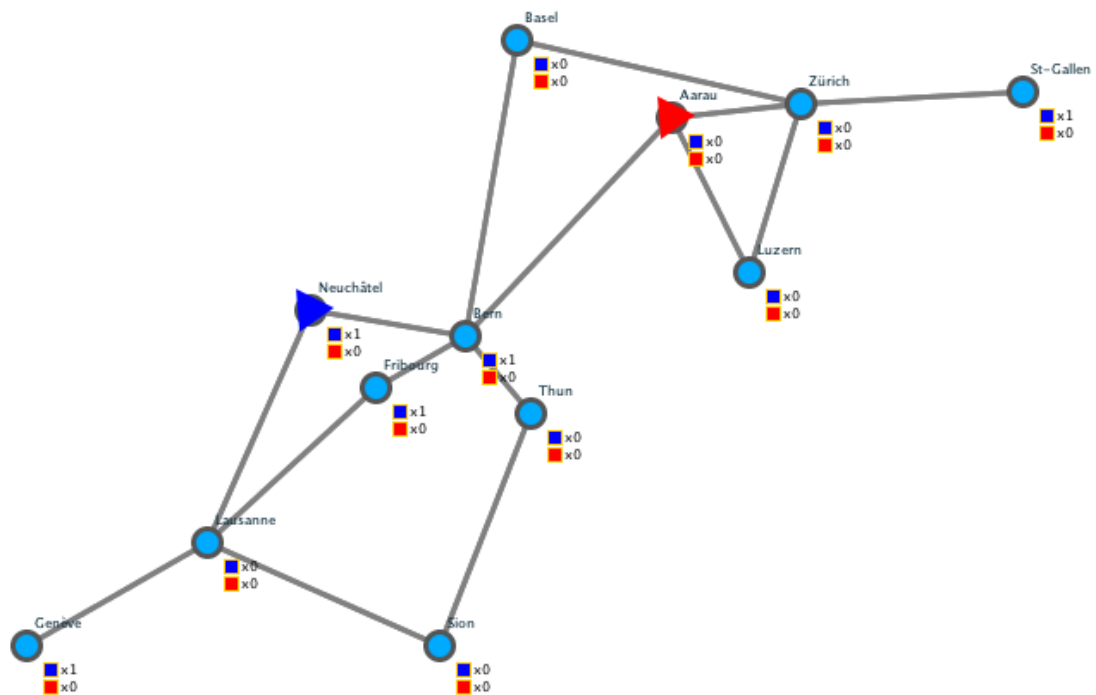

FIGURE 1 – Deliberative agents on the swiss map

Let us consider a simple example in which one vehicle (V1) located in Neuchâtel has to transport three tasks :

— Task T1 from Lausanne to Neuchâtel

— Task T2 from Neuchâtel to Aarau

— Task T3 from Aarau to Lausanne

Vehicle V1's optimal plan is the following :

1. Pickup task T2

2. Move to Aarau and deliver task T2

3. Pickup task T3

4. Move to Lausanne and deliver task T3
5. Pickup task T1
6. Move to Neuchâtel and deliver task T1

While this plan is optimal for one vehicle, the presence of another vehicle can greatly influence the overall performance of the company. Let us assume that another vehicle (V2) located in Aarau sees the same tasks as above.

Vehicle V2's optimal plan is the following :

1. Pickup task T3
2. Move to Lausanne and deliver task T3
3. Pickup task T1
4. Move to Neuchâtel and deliver task T1
5. Pickup task T2
6. Move to Aarau and deliver task T2

When both vehicles start executing their plans, V1 will pickup T2 and move towards Aarau, while V2 will pickup T3 and move towards Lausanne. Once V1 tries to pickup its second task in Aarau it will realize that the environment has changed, consequently update its plan :

1. Move to Lausanne and pickup task T1
2. Move to Neuchâtel and deliver task T1

Similarly, when V2 reaches Neuchâtel it will realize that there are no more tasks to be delivered and stop executing its plan. Note that by the time V1 reaches Lausanne the task is gone and it has traveled the distance Aarau → Lausanne for nothing, reducing the total efficiency of the company. In the centralized exercise you will see how several agents can coordinate their actions in order to achieve a better outcome.

Your task

1. Choose a representation for the states, transitions and goals (final states) to be used in a state-based search algorithm that finds the **optimal** plan for delivering a set of tasks.
2. Implement the state-based breadth-first search and A* heuristic search algorithms. Choose one heuristic and explain why. Discuss the optimality of your new algorithm in relation to your heuristic.
3. Implement a deliberative agent which can use the above planning algorithms.
4. Compare the performances of the breadth-first search and the A* search algorithms for different problem sizes.
5. Run the simulation with 1, 2 and 3 deliberative agents and report the differences of the joint performance of the agents.

Implementation Hints

— You will need to implement the `DeliberativeBehavior` interface, which will use different search strategies to compute its plans. It should be possible to choose the search algorithm from the *agents.xml* file using either of the following tags :

```
<set algorithm="BFS" />
<set algorithm="A-star" />
```

Look at the starter code to see how to read properties from the configuration file. You can define your own properties if you wish (for example to choose a heuristic).

— When you create your `Plan`, make sure that you are in the correct city before you pickup or deliver a task. For this exercise, you will need to take into account the weight of the tasks and capacity of the truck. If the *LogistPlatform* finds a problem in your plan, the simulation will exit and a detailed error message is displayed.

— There is a seperate document that elaborates the relevant parts of the *LogistPlatform* for this exercise in more detail.

— The prepared package comes with various configuration files that allow you to test your solution. Feel free to change them and see how it affects the simulation of your program. When you hand-in your solution please make sure that you correctly set any other user-defined values (if you use them).

— The task distribution can be given a seed in order to make the task generation deterministic. This is invaluable when doing code debugging, but please test your program with other seeds, too.

Deliverables:

What to submit !
— A report in PDF answering the following questions :
  — Explain how you model states, transitions and final states. Motivate your choice.

  — Explain your heuristic for A*. Show that it preserves the optimality of A*.

  — Compare the performance of the breadth-first search and your A* implementation for one agent and several tasks.

  — What is the maximum number of tasks for which you can build a plan in less than one minute ?

  The report should be based on a latex template that is given on the exercise description moodle page.
— the name of your agent (in `agents.xml`) that represents your final solution should be `deliberative-main`

— rename the deliberative folder to sciper1-sciper2-ex2

— create a directory called doc within this folder and place the report in this folder—

create a zipfile sciper1-sciper2-ex3.zip containing everything in the sciper1-
sciper2-ex3 folder