IASD 2023/24 Project Assignment #2: Autonomous shared vehicle transportation system

Rodrigo Ventura and Luis Custódio
Instituto Superior Técnico, University of Lisbon

(Version 1.2, October 6, 2023)

1 Introduction

The project for this assignment remains the same. The system receives as input a set of transportation requests from users, and schedules a fleet of autonomous vehicles to drive users from their pickup points to the desired destinations. Each vehicle has a pre-defined amount of seats. The goal is to maximize efficiency in passenger transportation.

2 Second Assignment and Deliverable

This assignment is a followup of the Assignment #1, where the problem was properly formalized. This second assignment aims at obtaining optimal solutions to the problem using **uninformed search methods.**

Associated with the IASD course book, there is a Python code repository with many Python files for the different chapters of the book[1]. In the repository you have the file `search.py` where you can find the definition of the class `Problem`, the superclass of the class FleetProblem, examples for other problems, and the functions that implement each one of the search algorithms mentioned in the IASD lectures. For this assignment, you only need to import search.py, which in turn imports utils.py. Both files are available in the repository. Copy both files to your work directory while developing and testing on your computer. Nothing else is needed.

The Python program to be delivered should be called `solution.py` and include (at least) a class with name `FleetProblem` containing (at least) the following methods (see Annex A for the class template):

`result(state,action)` returns the state that results from executing the given action in the given state.

`actions(state)` returns the list of actions that can be executed in the given state.

`goal_test(state)` returns `True` if the given state is a goal.

`path_cost(c, state1, action, state2)` returns the path cost of `state2`, expanded from `state1` (with path cost `c`) after applying action `action`.

`load(fh)` loads a problem from an opened file object fh (same as in Assignment #1)

`solve()` calls once (and only one) one uninformed search algorithm from `search.py` to solve the problem; the function should return a plan in the same format as defined in Assignment #1, i.e., a list of tuples representing actions.

In order to solve a problem one needs to:

- read the problem from a file object like you did for Assignment #1,
- decide and implement a representation for the state of the problem. You may change the representation used for Assignment #1 if you find a better one, and represent the initial state with that representation, and saved in the object FleetProblem (`self.initial`),
- (re)implement the methods `result`, `actions`, `goal_test`, `path_cost`, and `solve` as specified above,
- choose one uninformed search algorithm that the group considers best fits the problem at hand, and use it in method `solve`. See `search.py` for the set of uninformed search algorithms available.

[1]The link for the repository is `https://github.com/aimacode/aima-python`.

3 Evaluation

The deliverable for this assignment is shown through DEEC Moodle, with the submission of a single python file, called `solution.py`, implementing the modules mentioned above. Instructions for this platform are available on the course webpage. Finally, the grade is computed in the following way:

- 50% from the public tests;
- 50% from the private tests; and
- -10% from the code structure, quality and readability.

Deadline: **13-October-2023**.
Projects submitted after the deadline will not be considered for evaluation.

Closing Remarks on Ethics:

- Any kind of sharing code outside your group is considered plagiarism;
- Developing your code in any open software development tool is considered sharing code;
- You can use GitHub. Make sure you have private projects and remove them afterward;
- If you get caught in any plagiarism, either by copying the code/ideas or sharing them with others, you will not be graded; and
- The scripts and other supporting materials produced by the instructors cannot be made public!

A Class Template

```python
import search

class FleetProblem(search.Problem):

    def result(self, state, action):
        """Return the state that results from executing
        the given action in the given state."""
        pass

    def actions(self, state):
        """Return the actions that can be executed in
        the given state."""
        pass

    def goal_test(self, state):
        """Return True if the state is a goal."""
        pass

    def path_cost(self, c, state1, action, state2):
        """Return the cost of a solution path that arrives at state2 from
        state1 via action, assuming cost c to get up to state1."""

    def load(self, fh):
        """Loads a problem from the file object fh.
        You may initialize self.initial here."""
        pass

    def solve(self):
        """Calls the uninformed search algorithm
        chosen. Returns a solution using the specified format."""
        pass
```