Informatics 2D Coursework 2: Planning with PDDL

Mattias Appelgren and Alex Lascarides

Deadline : **3pm, Thursday 26th March 2020**

Introduction

This assignment is about planning using the **Planning Domain Definition Language (PDDL)**. It consists of four parts:

- Part 1 - A written exercise in which you will formalize a planning problem using **PDDL** (worth 25%).
- Part 2 - Implementing and verifying the correctness of your written model using a PDDL planner (worth 20%).
- Part 3 - Extending the model to deal with additional complications in the environment (worth 40%)
- Part 4 - Theoretical extension considering planning in a wider context (worth 15%)

This assignment is marked out of 100, and it is worth 12.5% of your overall grade for Inf2D.

The files you need are available from the coursework archive:

`http://www.inf.ed.ac.uk/teaching/courses/inf2d/coursework/Inf2D-Coursework2.tar.gz`

Please be sure to read the README file carefully as it involves instructions for setting up and running the MetricFF planner, which will be used in parts 2 and 3.

Submission

Create a directory in which you keep the files you submit for this assignment. This directory should be called `Inf2d-ass2-s<matric>` where `<matric>` is your matriculation number (e.g 1234567).

In this directory, write your answers to part 1 and part 4 in a file you call `answers.txt`. For the implementation in parts 2 and 3, copy and edit the corresponding `*-template.pl` files, available in the coursework archive.

Submit your assignment by creating a new file from your directory `Inf2d-ass2-s<matric>`. You do this using the following command in a DICE machine:

```
tar -cvzf Inf2d-ass2-s<matric>.tar.gz Inf2d-ass2-s<matric>
```

You can check that this file stores the intended data with the following command, which lists all the files one will get when one extracts the original directory (and its files) from this file:

```
tar -t Inf2d-ass2-s<matric>.tar.gz
```

Also, you can check the size of the above file using the following command, to check that the file stores the intended data:

```
ls -l inf2d-ass2-s<matric>.tar.gz
```

Submit this file via LEARN by uploading the file using the interface on the LEARN website for the course Inf2D. If you have trouble please refer to this blogpost:

`https://blogs.ed.ac.uk/ilts/2019/09/27/assignment-hand-ins-for-learn-guidance-for-students/`

The deadline for submission is **3pm, Thursday 26th March 2020**.

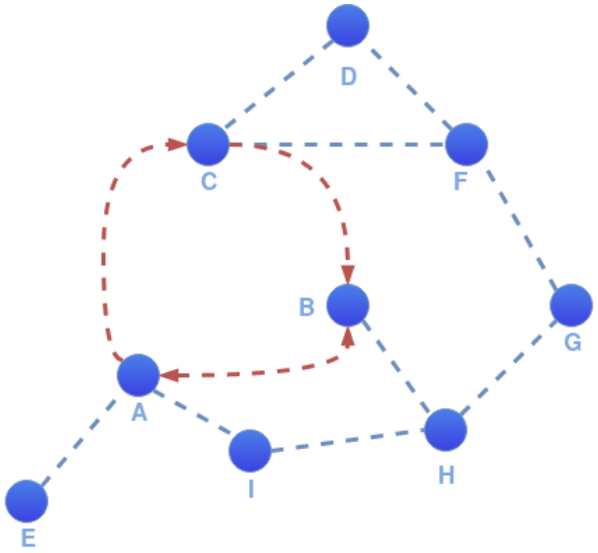

Figure 1: Travel Map

You can submit more than once up until the submission deadline. All submissions are timestamped automatically. Identically named files will overwrite earlier submitted versions, so we will mark the latest submission that comes in before the deadline.

If you submit anything before the deadline, you may not resubmit afterward. (This policy allows us to begin marking submissions immediately after the deadline, without having to worry that some may need to be re-marked).

If you do not submit anything before the deadline, you may submit *exactly once* after the deadline, and a late penalty will be applied to this submission, unless you have received an approved extension. Please be aware that late submissions may receive lower priority for marking, and marks may not be returned within the same timeframe as for on-time submissions.

For additional information about late penalties and extension requests, see the School web page below. Do **not** email any course staff directly about extension requests; you must follow the instructions on the web page.

`http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests`

Good Scholarly Practice: Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

`http://www.ed.ac.uk/schools-departments/academic-services/students/undergraduate/discipline/academic-misconduct`

and at:

`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

Part 1: Modelling The Domain (25%)

Problem Description

Your task is to model a travelling problem in the Planning Domain Definition Language (PDDL) and run a planner to find for a number of different scenarios. In this problem an agent wishes to travel and visit various locations on the map described in Figure 1. The agent can travel by car or by plane. Car routes can be traversed in either direction, and are marked with blue dashed lines on the map. To travel by car the agent must have a car in the same location as they are, which they can drive. Plane routes are directional, i.e. they may only be taken in a specific direction. They are marked with red arrows on the map. The agent can also visit a location. They can do so if they are in that location and have not visited the city before.

Task 1.1: Describing The World State (5%)

Begin by creating a model which will describe the states for this planning domain. Your model should be able to describe:

1. the map, i.e. which locations are connected by road or by air
2. the location of the agent and the car
3. which cities the agent has visited

Name all predicates which you will use to describe the world as well as their purpose.

Finally, write down an initial state where the agent and the car are in **E**, and where the goal is for the agent to visit **D** and end up back in **E**.

Ensure you include all predicates which would be required to describe the map and to find a plan using the actions you create in Task 1.2.

Task 1.2: Actions (10%)

Define the agent's actions. The agent should be able to *Drive* the car from one location to another if they are connected by road and the car is in the Agent's location.

The agent can *Fly* by plane between two locations connected via air, but only in the direction of the air route.

The agent can *Visit* a location if it is in that location and has not visited it before (simply entering a location does not mean the agent visits it, this should be a deliberate decision the agent makes, allowing it to let its hair down and go take some selfies).

Task 1.3: Backwards state space search (10%)

In this question you must perform the backwards state space search algorithm as described in the lectures.

The starting state is that the agent and the car are in **E**. The goal is for the agent to visit **B**. Spell out the working of the algorithm by specifying at each step which are the *relevant* actions and what the current *goal state* is. In the first step state why the relevant actions are selected as relevant.

State clearly which action was selected at each step. For the purpose of this task you may assume the search always picks the correct action when there is a choice, so you do not need to simulate dead ends.

Finally, state clearly why the algorithm terminates when it does and state the plan which was selected.

Part 2: Implementation (20%)

Implement your domain model in PDDL using the example files provided as a starting point. Make sure to read the README file included with the coursework which provides instructions for setting up and running the FF planner.

Test your model by creating problem files for the following problems:

Task 2.1 (12%)

1. Agent and car begin in **E**
2. Visit **C**

Task 2.2 (3%)

1. Agent and car begin in **E**
2. Visit **D**
3. Agent ends in **E**

Task 2.3: Inverse Problem (5%)

Create a new problem file where you select your own goal. Create the goal in such a way that the agent enters every location (but does not necessarily visit all of them). Try to minimise the number of locations which the agent actually visits.

Part 3: Extension (40%)

In this part you will extend the domain to support additional actions and considerations.

Task 3.1 Buses (10%)

Add the ability for the agent to take a bus between any two locations which are connected by road.

Test your new domain on a problem where the agent and car begin in **E**. The goal should be that the agent visits **D** and the car and agent both end in **E**.

Task 3.2 Cost (15%)

Up until now all actions are considered to be of equal value. However, in reality different actions may be associated with different costs or value. In this task, travelling will have a cost associated with it and the agent will have a budget to spend. The agent should not be able to take an action if it cannot pay the cost and the total cost of the plan should not exceed the budget.

The cost of driving is 1, the cost of taking the bus is 5 and the cost of taking a plane is 10.

Create three problem files with the same goal but different budgets. The agent and car should start in E. The goal is to visit **B** and **D**. The budgets are **15**, **30**, and **40**. Call these files `problem-3.2-15.pddl`, `problem-3.2-30.pddl`, and `problem-3.2-40.pddl` respectively.

Task 3.3 Hire Cars (15%)

In this task the agent has totalled its car but still wishes to travel. Instead of buying a new one, it has elected to rely on hire cars. The agent can hire a car from one of three different car companies. Once hired a car can be driven as normal. However, the agent must always return any cars it hires to the same company from which it was hired. It costs 2 to hire a car and driving still costs 1.

You may assume there is a fixed number of cars which can be hired. There are three car rentals: **V**olvo in **E**, **T**oyota in **C**, and **R**enault in **A**.

Create a problem file where the agent starts in **E** and the goal is to visit **D** and return to **E**. The agent has a budget of **40**. Remember that all cars must be returned for the plan to be correct.

Part 4: Theoretical Extension (15%)

In this question we take a step back from PDDL implementation to think about planning more broadly.

The FF planner uses plan length as the main metric for how to select plans. However, in many cases there will be many different competing considerations which must be taken into account. For example, when

travelling there is the actual price, there is the length of the journey, the ease or comfort of the journey (e.g. number of changes), and the environmental impact of the chosen methods. Different people will value these competing considerations to different degrees. Imagine you need to create a planner which takes into account a traveller's preferences. How might you do this effectively? What planning tools and paradigm might you utilise?

Write your answer in your `answers.txt` file.

Keep your answers to 100 words or less.