Introduction

In this mini-project, your task is to model different problems in PDDL and solve them using a planner. For modelling the problem, two files must be provided: a domain file and a problem file. Once you have modelled the problem, you can solve it by using one of the state-of the art planners that already exist. You can either install Fast Downward (`http://www.fast-downward.org/`), or use the online editor (`http://editor.planning.domains/`).

Submission: Master solutions won't be provided for the mini-project. We have enabled a submission in Moodle (deadline May 2nd), where you can submit your `.pddl` files for feedback. Also, feel free to send any questions by email (`alto@cs.aau.dk`).

Exercise 1 : Hamiltonian Path Problem

A Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once.

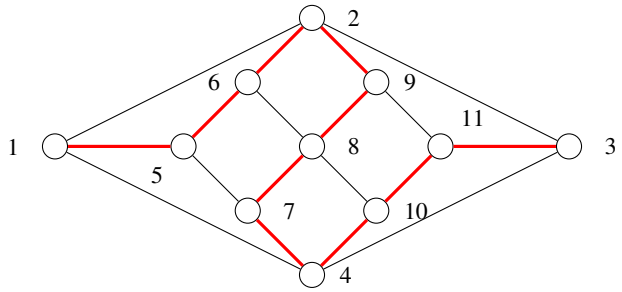


Figure 1: Hamiltionian Path Example

(i) Encode the problem of finding a Hamiltonian path in the graph of Figure 1 in PDDL. Note that may there be posible solutions of the problem, and the red edges are just one example.

(ii) Modify the encoding of (i) so that the path starts at vertex 1 and ends at vertex 3.

(iii) Consider now that each path has a different cost (see Figure 2). Encode the problem of finding the minimum-cost Hamiltonian path in PDDL. The costs in Figure 2 are just an example, it is also fine if you use different values for your encoding.

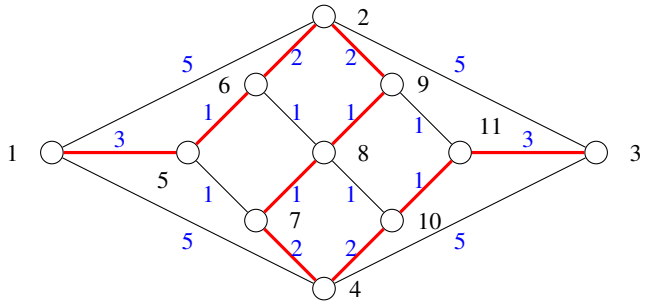
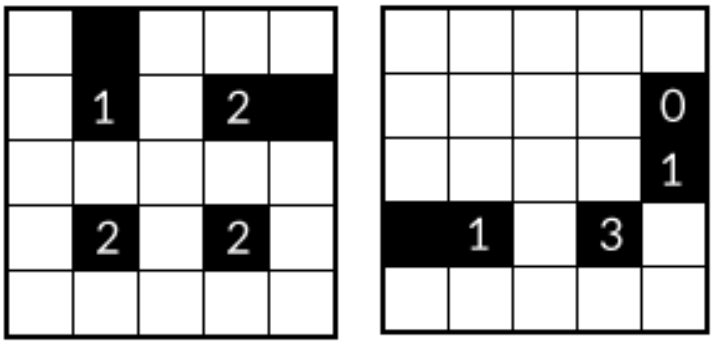


Figure 2: Hamiltionian path with costs for part (iii).

Exercise 2 : Light Up

In this exercise, your task it to model the *Light Up* puzzle[1] in PDDL. This game is based on a 2D grid consisting of black and white cells. White cells can be lit up by placing light bulbs on the grid. Placing a light bulb at some coordinate $\langle x, y \rangle$ lights up all white cells of the same column (x), and all white cells of the same row (y), such that no black cells lies in between $\langle x, y \rangle$ and the cell, i.e., light propagation is stopped by black cells. A light bulb may only be placed in a cell that is not lighted up by another light bulb, yet. Some black cells are associated with numbers. This number gives the number of light bulbs that must be placed directly (horizontally or vertically) adjacent to the black cell. For example, if the number is 0, no light bulb may be placed in a cell adjacent to the black cell. If 1, exactly one light bulb may (and must) be placed to a cell adjacent to the black cell. The goal is to light up all white cells while adhering to the aforementioned constraints.



`p01` `p02`

Figure 3: Two Light Up instances with grid size of 5×5. The upper left corner has coordinate $\langle 0, 0 \rangle$. The bottom right corner $\langle 4, 4 \rangle$.

(i) Write a PDDL domain file that can be used for any instance of the puzzle, i.e., for arbitrary grids. We already provide a `domain.pddl` template file. You are free to introduce new (or remove/modify the existing) types, predicates, and/or actions.

(ii) Encode the two Light Up instances in Figure 3 into two PDDL problem files. You may start from the provided `p01.pddl` and `p02.pddl` files.

Exercise 3 Warehouse:

Encode the problem of distributing products on a warehouse where there is a conveyor belt, as well as robots that can move around the factory.

[1] `https://en.wikipedia.org/wiki/Light_Up_(puzzle)`

Our task is to devise a plan to deliver all products, while minimizing the amount of time (turns required). The conveyor belt is a circular belt with 10 positions. The conveyor belt is circular, so that position10 is connected with position1 again. The belt is continously moving, so any packages put in the conveyor belt will change their position to the next position each turn.
There are three shelves where the objects are initially. The shelves are at positions 1, 5, 7. The task is to transport all packages (initially distributed between shelves 1 and 7), to the delivery shelf (in position 5). If the robot is at the same position as the shelf, it can move one package from the shelf to the belt (if there is no other package at the belt in that position), or viceversa. Each turn, each robot can do at most one action.

Hint: A way of modelling "turns" in PDDL is to have the actions "move-robot"/"move-package" to require that the robot has not moved yet; as well as an action "pass-time", which moves all packages in the conveyor belt and allows the robots to move again.

Note: This problem was inspired by the following assignment: `https://github.com/AI-Planning/modeling-in-pddl/blob/main/assignments/warehouse/robots.pdf`. Here, the modelling has been simplified to avoid requiring temporal planning.
Optionally, you could also use temporal planning encoding to solve a more realistic version of this problem.

Exercise 4 Free Modelling (Optional)

Model another problem of your choice![2]
It is likely that some sub-problem in your current (or perhaps some past) semester project, can be modelled and solved using a planner. Feel free to use this opportunity to try out how this technology can help and feel free to ask for advice and/or feedback on how to do that.
To find suitable problems of your interest, you may think of cases where the solution can be expressed as a sequence of steps. Perhaps you may also look at lists of NP/PSPACE problems to find some inspiration.

[2]We are often in need of new domains, so if you do this, please consider to submit your solutions and let me know if you'd be willing to make them publicly available for research purposes.