Foundations of Artificial Intelligence

M. Helmert
J. Seipp
Spring Term 2019

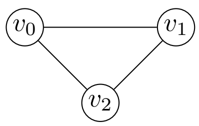
University of Basel
Computer Science

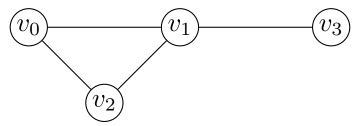
Exercise Sheet 10

Due: May 8, 2019

Exercise 10.1 (1+1.5+1.5+2 marks)

Consider the following variant of the *Seven bridges of Königsberg* problem: for a given set of bridges that cross a river, is there a tour that starts and ends at the same location and crosses each bridge exactly once? Formally, the problem can be defined as follows: given a graph $G = (V, E)$ with a set of vertices V and a set of edges $E \subseteq V \times V$ and an initial vertex $v_0 \in V$, is there a sequence of vertices from V such that i) all pairs of subsequent vertices are connected by an edge from E, ii) each edge in E occurs exactly once in the sequence, and iii) the first and last vertex of the sequence is v_0? For an illustration of the problem, consider the following examples:





The initial vertex is v_0 in both cases. For the graph on the left side, there is such a tour, e.g., $\langle v_0, v_1, v_2, v_0 \rangle$, while there is no such tour for the graph on the right side.

(a) You can find a PDDL description of the (original instance of the) *Seven bridges of Königsberg* problem on the website of the course. The domain description (variables and actions) is given in the file `bridges-once-domain.pddl`, and the problem description (objects, initial state and goal description) is given in the file `koenigsberg-problem.pddl`. Provide a graphical representation of the problem in the same way as the example above. Please do not forget to mark which location is the initial location.

(b) Obtain the domain-independent planning system *Fast Downward* by following the installation instructions that are given at

`http://www.fast-downward.org/ObtainingAndRunningFastDownward`.

Use *Fast Downward* with a configuration that performs greedy best-first search with the delete relaxation heuristic FF to solve the Seven bridges of Königsberg problem from the website. To do so, invoke the planner with

```
./fast-downward.py bridges-once-domain.pddl koenigsberg-problem.pddl
--search "eager_greedy([ff()])"
```

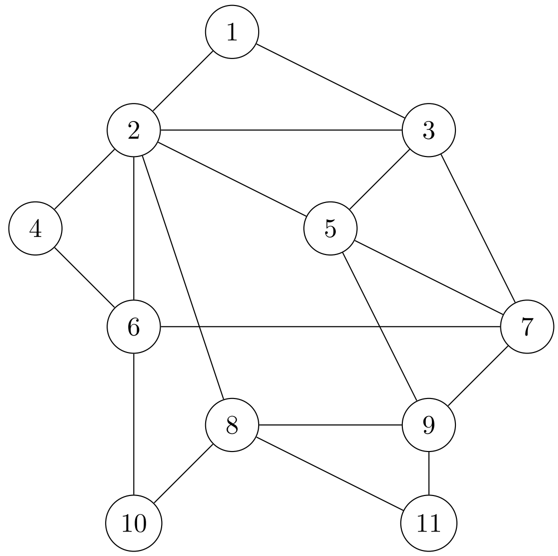
Provide the runtime and the number of expanded states. Discuss whether the solution is optimal. Is the problem solvable? If it is, provide the plan that was found.

(c) Modify the domain description (`bridges-once-domain.pddl`) such that it is possible to use the same bridge more than once. Solve the resulting problem with the same Fast Downward configuration that was used in part (b) of this exercise.

Provide the runtime and the number of expanded states. Discuss whether the solution is optimal. Is the problem solvable? If it is, provide the plan that was found.

(d) Formalize the following instance of the bridges domain in PDDL and solve it with the same Fast Downward configuration that was used in part (b) of this exercise where each bridge may be traversed only once.

Provide the runtime and the number of expanded states. Discuss whether the solution is optimal. Is the problem solvable? If it is, provide the plan that was found.



Exercise 10.2 (0.5+0.5+0.5+0.5 marks)

Consider the STRIPS formalization of blocks world (print-out version of slide set 34, pages 10–13). Consider the following task with blocks A, B and C, initial state $I = \{\mathit{on\text{-}table}_A, \mathit{on}_{B,A}, \mathit{on}_{C,B}, \mathit{clear}_C\}$ (left stack in the picture below) and the goal $G = \{\mathit{on\text{-}table}_A, \mathit{on}_{C,A}, \mathit{on}_{B,C}\}$ (right stack in the picture below).

(a) Calculate the perfect heuristic values $h^*(I)$ and $h^*(I')$ for the initial state I and the only successor state I' of I.

(b) Consider the STRIPS heuristic h^S (print-out version of slide set 35, page 4). Calculate the heuristic values $h^S(I)$ and $h^S(I')$.

(c) Calculate $h^+(I)$ and $h^+(I')$.

(d) Compare and discuss the results of exercise parts (a), (b) and (c).

Exercise 10.3 (2+2 bonus marks)

Hint: This is a *bonus* exercise, so its 4 marks are not considered when the amount of marks required for admission to the exam is computed. However, the marks you obtain in this exercise count towards your achieved total marks.

Consider the 8-Puzzle in a STRIPS encoding $\Pi = \langle V, I, G, A \rangle$ with the following components:

- $V = \{\text{tile-at-cell}_{t,c} \mid t \in \{1, \dots, 8\}, c \in \{(1,1), \dots, (3,3)\}\} \cup$
  $\{\text{cell-empty}_c \mid c \in \{(1,1), \dots, (3,3)\}\}$
- I is an arbitrary *legal* state, where a state is legal if each tile is at exactly one position, no two tiles are at the same position and there is exactly one empty position.
- $G = \{\text{tile-at-cell}_{1,(1,1)}, \dots, \text{tile-at-cell}_{8,(3,2)}\}$

- $A = \{move_{t,c,c'} \mid t \in \{1, \dots, 8\}, c, c' \in \{1,2,3\} \times \{1,2,3\}, c \text{ and } c' \text{ are neighbours}\}$

  All actions have cost 1 and are defined as follows:

  - $pre(move_{t,c,c'}) = \{\text{tile-at-cell}_{t,c}, \text{cell-empty}_{c'}\}$
  - $add(move_{t,c,c'}) = \{\text{tile-at-cell}_{t,c'}, \text{cell-empty}_{c}\}$
  - $del(move_{t,c,c'}) = \{\text{tile-at-cell}_{t,c}, \text{cell-empty}_{c'}\}$

(a) For this STRIPS encoding of the 8-Puzzle, show the claim of the lecture (Chapter 35, slide 18 in the printed version): $h^+(s) \geq h^{\text{MD}}(s)$ for all states s, i.e., h^+ dominates the Manhattan distance in the 8-Puzzle.

(b) Show that there exists a legal state s with $h^+(s) > h^{\text{MD}}(s)$.

Important: Solutions should be submitted in groups of two students. However, only one student should upload the solution. Please provide both student names on each file and each page you submit. We can only accept a single PDF or a ZIP file containing *.java or *.pddl files and a single PDF.