Foundations of Artificial Intelligence

M. Helmert
S. Eriksson
Spring Term 2021

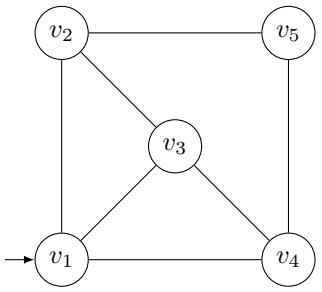
University of Basel
Computer Science

Exercise Sheet 10

Due: May 12, 2021

Exercise 10.1 (1+0.5+1 marks)

Given an undirected graph $G = \langle V, E \rangle$, a *Hamilton cycle starting in* v is a path that starts and ends in $v \in V$, and visits each vertex $v' \in V \setminus \{v\}$ exactly once (while visiting v exactly twice, once in the beginnning and once in the end). In the following graph the path $\langle v_1, v_3, v_2, v_5, v_4, v_1 \rangle$ is a Hamilton cycle starting in v_1.



(a) Download the archive `hamilton-cycle.tar.gz` from the course website or ADAM. The domain file `hamilton-cycle.pddl` describes the variables and actions of the hamilton cycle problem, while the file `hamilton-cycle-problem.pddl` describes a concrete problem instance. Provide a graphical representation of the problem in the same way as the example above. Please do not forget to mark which vertex is the start vertex.

(b) Obtain the domain-independent planning system *Fast Downward* from

`http://www.fast-downward.org/Releases/20.06`.

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

```
./fast-downward.py hamilton-cycle.pddl hamilton-cycle-problem.pddl
--search "eager_greedy([ff()])"
```

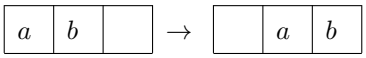
Provide the runtime and the number of expanded states. Is the problem solvable? If it is, provide the plan that was found.

(c) Modify the domain description (`hamilton-cycle.pddl`) such that it is possible to visit the same vertex more than once. Solve the resulting problem with the same Fast Downward configuration that was used in part (b).

Describe your modifications and provide the runtime and the number of expanded states. Is the problem solvable? If it is, provide the plan that was found.

Exercise 10.2 (1+1 marks)

In *Sokoban* an agent a in a grid world needs to move boxes to goal locations. He can only move a box b by moving next to it and *pushing* it to the next square, with the agent then standing where the box stood:



Consider the following scalable class of Sokoban problems where a represents the initial position of the agent, b_1, b_2 and b_3 the initial position of three boxes and g_1, g_2 and g_3 three goal positions for the boxes. The agent can move with cost 0, but each `push` action incurs a cost of 1.

1			g_2			
2						
...						
$n-2$						
$n-1$			b_2			
n		a	b_1		g_1	
$n+1$			b_3			
$n+2$						
...						
$2n-2$						
$2n-1$			g_3			

(a) Consider the variant of Sokoban where each box has a designated goal location, and assume that the goal location of box b_i is g_i for $i = 1, 2, 3$. Provide $h^+(s_0)$ as a function of n. Justify your answer by describing an optimal delete relaxed plan. Discuss the ratio of $h^+(s_0)$ and $h^*(s_0)$ for increasing n.

(b) Now consider the original version of Sokoban where each box can be pushed to any empty goal location. Provide $h^+(s_0)$ as a function of n (you do not have to consider the case where $n < 7$). Justify your answer by describing an optimal delete relaxed plan. Discuss the ratio of $h^+(s_0)$ and $h^*(s_0)$ for increasing n.

Exercise 10.3 (1+0.5+0.5+0.5 marks)

Consider the STRIPS planning task $\Pi = \langle V, I, G, A\rangle$ with $V = \{a, b, c, d, e\}$, $I = \{a\}$, $G = \{d, e\}$, and $A = \{a_1, a_2, a_3, a_4\}$ with $cost = \{a_1 \mapsto 3, a_2 \mapsto 1, a_3 \mapsto 8, a_4 \mapsto 2\}$ and

$$
\begin{array}{lll}
pre(a_1) = \{a\} & add(a_1) = \{b, c\} & del(a_1) = \{\} \\
pre(a_2) = \{b, c\} & add(a_2) = \{d\} & del(a_2) = \{\} \\
pre(a_3) = \{a\} & add(a_3) = \{d, e\} & del(a_3) = \{a\} \\
pre(a_4) = \{c\} & add(a_4) = \{e\} & del(a_4) = \{c\}.
\end{array}
$$

(a) Provide the relaxed planning graph for Π up to depth 2 (i.e., the resulting graph should have three variable layers and two action layers).

(b) Compute $h^{\max}(I)$. Provide the values for all nodes in the RPG.

(c) Compute $h^{\text{add}}(I)$. Provide the values for all nodes in the RPG.

(d) Compute $h^{\text{FF}}(I)$. Provide the marked RPG.

Exercise 10.4 (1+1+1 marks)

Consider a planning task where an agent aims to raise a treasure. To do so, the agent must collect a key and use it to open the chest that contains the treasure. Let the problem be formalized in the SAS^+ formalism as $\Pi = \langle V, \mathrm{dom}, I, G, A \rangle$, where

- $V = \{loc, key, treasure\}$ is the set of variables with $\mathrm{dom}(loc) = \{A, B, C\}$, $\mathrm{dom}(key) = \{\top, \bot\}$, and $\mathrm{dom}(treasure) = \{\top, \bot\}$;
- $I = \{loc \mapsto B, key \mapsto \bot, treasure \mapsto \bot\}$ is the initial state;
- $G = \{key \mapsto \top, treasure \mapsto \top\}$ is the goal description; and
- $A = \{move_{A,B}, move_{B,A}, move_{B,C}, move_{C,B}, take, open\}$ is the set of actions with

$pre(move_{A,B}) = \{loc \mapsto A\}$	$\mathit{eff}(move_{A,B}) = \{loc \mapsto B\}$	$cost(move_{A,B}) = 3$
$pre(move_{B,A}) = \{loc \mapsto B\}$	$\mathit{eff}(move_{B,A}) = \{loc \mapsto A\}$	$cost(move_{B,A}) = 3$
$pre(move_{B,C}) = \{loc \mapsto B\}$	$\mathit{eff}(move_{B,C}) = \{loc \mapsto C\}$	$cost(move_{B,C}) = 3$
$pre(move_{C,B}) = \{loc \mapsto C\}$	$\mathit{eff}(move_{C,B}) = \{loc \mapsto B\}$	$cost(move_{C,B}) = 3$
$pre(take) = \{key \mapsto \bot, loc \mapsto A\}$	$\mathit{eff}(take) = \{key \mapsto \top\}$	$cost(take) = 1$
$pre(open) = \{key \mapsto \top, loc \mapsto C\}$	$\mathit{eff}(open) = \{treasure \mapsto \top\}$	$cost(open) = 1$

(a) Provide the state space as a graph and mark the initial state and all goal states (it consists of 12 states, some of which are not reachable from the initial state). For each state, provide the values of all variables, e.g., in the form $B\bot\bot$ for the initial state and accordingly for other states.

(b) Compute the projection of Π to $P = \{loc, treasure\}$ (i.e., the variable *key* is ignored). Give the abstraction that is induced by P by providing the abstract state space in the same way as in (a).

(c) Use the abstraction from Exercise 11.2 (b) to derive a pattern database heuristic. Provide the database entries (i.e., the abstract distances for all states in the abstract state space) and use them to assign a heuristic value to each of the 12 concrete states.

Submission rules:
Upload a single PDF file (ending .pdf). If you want to submit handwritten parts, include their scans in the single PDF. Put the names of all group members on top of the first page. Use page numbers or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).