Foundations of Artificial Intelligence

T. Keller, F. Pommerening
S. Sievers
Spring Term 2023

University of Basel
Computer Science

Exercise Sheet 10

Due: May 14, 2023

Important: for submission, consult the rules at the end of the exercise. Non-adherence to the rules will lead to your submission not being corrected.

Exercise 10.1 (1.5+1.5 marks)

Consider the 8-Puzzle. We use (x, y) to denote the cell at coordinates (x, y), where x is the horizontal component and y the vertical component. Cell $(0, 0)$ is in the *upper* left corner (which allows us to specify the goal more naturally). Let $C = \{1, 2, 3\} \times \{1, 2, 3\}$ denote the set of all cells, and let $T = \{1, \ldots, 8\}$ denote the set of all tiles. We encode the 8-Puzzle as a STRIPS planning task $\Pi = \langle V, I, G, A \rangle$ with the following components:

- $V = \{\mathit{tile\text{-}at\text{-}cell}_{t,c} \mid t \in T, c \in C\} \cup \{\mathit{cell\text{-}empty}_c \mid c \in C\}$
  (We do not represent the blank explicitly.)
- I is an arbitrary *legal* state, where a state is legal if
  - each tile is at exactly one position and no two tiles are at the same position (for each $t \in T$, $\mathit{tile\text{-}at\text{-}cell}_{t,c}$ is true for exactly one $c \in C$ and $\mathit{tile\text{-}at\text{-}cell}_{t,c}$ is not true for two different t and the same c), and
  - there is exactly one empty position and a cell is empty iff no tile is at that cell ($\mathit{cell\text{-}empty}_c$ is true for only one $c \in C$ and $\mathit{tile\text{-}at\text{-}cell}_{t,c}$ has to be false for all $t \in T$ for that particular c).
- $G = \{\mathit{tile\text{-}at\text{-}cell}_{1,(1,1)}, \mathit{tile\text{-}at\text{-}cell}_{2,(2,1)}, \mathit{tile\text{-}at\text{-}cell}_{3,(3,1)}, \mathit{tile\text{-}at\text{-}cell}_{4,(1,2)}, \mathit{tile\text{-}at\text{-}cell}_{5,(2,2)}, \mathit{tile\text{-}at\text{-}cell}_{6,(3,2)}, \mathit{tile\text{-}at\text{-}cell}_{7,(1,3)}, \mathit{tile\text{-}at\text{-}cell}_{8,(2,3)}\}$
- $A = \{\mathit{move}_{t,c,c'} \mid t \in T, c \in C, c' \in \mathit{neighbors}(c)\}$ with

$$\mathit{neighbors}((x, y)) = \{(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)\} \cap C$$

and

$$\begin{aligned} \mathit{pre}(\mathit{move}_{t,c,c'}) &= \{\mathit{tile\text{-}at\text{-}cell}_{t,c}, \mathit{cell\text{-}empty}_{c'}\} \\ \mathit{add}(\mathit{move}_{t,c,c'}) &= \{\mathit{tile\text{-}at\text{-}cell}_{t,c'}, \mathit{cell\text{-}empty}_{c}\} \\ \mathit{del}(\mathit{move}_{t,c,c'}) &= \{\mathit{tile\text{-}at\text{-}cell}_{t,c}, \mathit{cell\text{-}empty}_{c'}\} \\ \mathit{cost}(\mathit{move}_{t,c,c'}) &= 1 \end{aligned}$$

for all $\mathit{move}_{t,c,c'} \in A$.

We define the *Manhattan distance* between two cells (i.e., pairs of coordinates) as $\mathrm{MD}((x, y), (x', y')) = |x - x'| + |y - y'|$, i.e., as the length of the shortest "grid path" between the two cells. The Manhattan distance heuristic $h^{\mathrm{MD}}(s)$ is the sum of Manhattan distances of each tile to its goal cell.

(a) Show that $h^+(s) \geq h^{\mathrm{MD}}(s)$ 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 $h^+(s) > h^{\mathrm{MD}}(s)$.

Exercise 10.2 (1+1 marks)

In this exercise, we consider the *Sokoban* problem. As a reminder, in the Sokoban problem, there is an agent who can move in a grid of which a subset of cells is blocked by walls. In the general form of the problem (see, e.g., `https://en.wikipedia.org/wiki/Sokoban`), there is a set of boxes, located at their initial cells in the grid, that must be pushed to goal cells in the grid. The agent can freely move to any cell not occupied by a wall or a box. The agent can only *push* boxes: if they stand next to a box and the cell behind the box from the point of view of the agent is unoccupied, they can push the box to that free cell, which also moves the agent to the cell the box was previously. The objective is to get the boxes to the goals with as few pushes as possible. The number of moves of the agent do not matter.

Consider the following scalable (for $n \in \mathbb{N}$) familiy 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. There are no walls/no occupied cells on the grid (but around it, not shown).

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			

Assume a STRIPS encoding that uses five variables for each cell of the grid. One variable encodes if the agent is at the cell or not, three variables, one for each box, encode if the box is at the cell or not, and the fifth variable encodes whether the cell is free or not. Additionally, there is a variable for each box encoding if it is at a/its goal positions, depending on the variant used (different in parts (a) and (b)). These variables have to be true in the goal. The cost of `push` actions is 1, that of `move` actions is 0. Let s_0 denote the initial state.

(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 $h^*(s_0)$ and $h^+(s_0)$ as a function of n. Justify your answer, e.g., by describing optimal (relaxed) plans. Discuss the ratio of $h^+(s_0)$ to $h^*(s_0)$ for increasing n.

(b) Consider the general version of Sokoban where each box can be pushed to any empty goal location. Provide $h^+(s_0)$ and $h^*(s_0)$ as a function of n (you do not have to consider the case where $n < 7$). Justify your answer. 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 $V = \{a, b, c, d, e, f\}$, $I = \{a\}$, $G =$

$\{d, f\}$, and $A = \{a_1, a_2, a_3, a_4, a_5\}$ with $cost = \{a_1 \mapsto 2, a_2 \mapsto 4, a_3 \mapsto 1, a_4 \mapsto 1, a_5 \mapsto 1\}$ and

$$\begin{array}{lll}
pre(a_1) = \{a\} & add(a_1) = \{b, c\} & del(a_1) = \{\} \\
pre(a_2) = \{a\} & add(a_2) = \{c, d\} & del(a_2) = \{\} \\
pre(a_3) = \{b, c\} & add(a_3) = \{d\} & del(a_3) = \{c\} \\
pre(a_4) = \{c\} & add(a_4) = \{e\} & del(a_4) = \{c\} \\
pre(a_5) = \{e\} & add(a_5) = \{f\} & del(a_5) = \{a, b\}.
\end{array}$$

(a) Provide the relaxed planning graph for Π up to depth 4 (i.e., the resulting graph should have five variable layers and four action layers).

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

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

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

Exercise 10.4 (1+1+0.5 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. Initially, he is located at position T (top), the key is at position M (middle), and the treasure at position B (bottom). The task is defined in SAS$^+$ as follows. $\Pi = \langle V, I, G, A \rangle$, where

- $V = \{location, key, treasure\text{-}chest\}$ is the set of variables with $\mathrm{dom}(location) = \{T, M, B\}$, $\mathrm{dom}(key) = \{Y, N\}$, and $\mathrm{dom}(treasure\text{-}chest) = \{O, C\}$;
  (Y and N stand for yes and no; O and C for the treasure chest being open or closed.)
- $I = \{location \mapsto T, key \mapsto N, treasure\text{-}chest \mapsto C\}$ is the initial state;
- $G = \{treasure\text{-}chest \mapsto O\}$ is the goal;
- $T = \{move_{T,M}, move_{M,T}, move_{M,B}, move_{B,M}, pick\text{-}up, open\}$ is the set of actions with

$$\begin{array}{lll}
pre(move_{T,M}) = \{location \mapsto T\} & eff(move_{T,M}) = \{location \mapsto M\} & cost(move_{T,M}) = 3 \\
pre(move_{M,T}) = \{location \mapsto M\} & eff(move_{M,T}) = \{location \mapsto T\} & cost(move_{M,T}) = 3 \\
pre(move_{M,B}) = \{location \mapsto M\} & eff(move_{M,B}) = \{location \mapsto B\} & cost(move_{M,B}) = 3 \\
pre(move_{B,M}) = \{location \mapsto B\} & eff(move_{B,M}) = \{location \mapsto M\} & cost(move_{B,M}) = 3 \\
pre(pick\text{-}up) = \{location \mapsto M, & eff(pick\text{-}up) = \{key \mapsto Y\} & cost(pick\text{-}up) = 1 \\
\quad key \mapsto N\} & & \\
pre(open) = \{location \mapsto B, & eff(open) = \{treasure\text{-}chest \mapsto O\} & cost(open) = 2 \\
\quad key \mapsto Y, & & \\
\quad treasure\text{-}chest \mapsto C\} & &
\end{array}$$

(a) Draw the state space, i.e., all states, labeled transitions, and mark the initial state and the goal states. The state space consists of 12 states, some of which are not reachable from the initial state. Label each state $\{location \mapsto x, key \mapsto y, treasure\text{-}chest \mapsto z\}$ with xyz. You may abbreviate action names.

(b) Draw the abstract state space that is induced by the projection π_P on $P = \{location, treasure\text{-}chest\}$. Use the same way as in part (a), i.e., write xy for an abstract state $\{location \mapsto x, treasure\text{-}chest \mapsto y\}$.

(c) Use the abstraction from part (b) to derive a pattern database (PDB) heuristic h^P. To do so, provide the heuristic function h^P for all concrete states, i.e., the perfect heuristic value of the abstract state the concret state is mapped to.

Submission rules:

- Exercise sheets must be submitted in groups of two students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending .pdf) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Either use page numbers on all pages or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, only create those code textfiles required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded.
- For the submission: if the exercise sheet does not include programming exercises, simply upload the single PDF. If the exercise sheet includes programming exercises, upload a ZIP file (ending .zip, .tar.gz or .tgz; *not* .rar or anything else) containing the single PDF and the code textfile(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to ADAM, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.