Foundations of Artificial Intelligence

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

University of Basel
Computer Science

Exercise Sheet 11

Due: May 21, 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 11.1 (1+1 marks)

Are the following statements about landmarks correct? Justify your answer.

(a) If L is a landmark and a is an action not occuring in L, then $L' = L \cup \{a\}$ is a landmark.

(b) Given a set of landmarks $\mathcal{L}$ which are pairwise disjoint (meaning we have $L_i \cap L_j = \emptyset$ for all $L_i \in \mathcal{L}$ and $L_j \in \mathcal{L}$ with $i \neq j$), the sum of the costs of all landmarks is an admissible heuristic.

Exercise 11.2 (1 mark)

Consider a task with actions $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$, cost function $cost = \{a_1 \mapsto 3, a_2 \mapsto 2, a_3 \mapsto 3, a_4 \mapsto 5, a_5 \mapsto 3, a_6 \mapsto 1\}$ and landmarks $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$ with $L_1 = \{a_2, a_5, a_6\}$, $L_2 = \{a_4\}$, $L_3 = \{a_1, a_2, a_3\}$, $L_4 = \{a_3, a_5\}\}$, and initial state I. Compute $h^{\mathrm{MHS}}(I)$.

Exercise 11.3 (3 marks)

Consider the delete-free STRIPS planning task $\Pi^+ = \langle V, I, G, A\rangle$, with variables $V = \{a, b, c, d, e, i, g\}$, initial state $I = \{i\}$, goal description $G = \{g\}$, and actions $A = \{a_1, \dots, a_6\}$, where

$$\begin{array}{lll}
pre(a_1) = \{i\} & add(a_1) = \{a, b\} & cost(a_1) = 2 \\
pre(a_2) = \{i\} & add(a_2) = \{c\} & cost(a_2) = 1 \\
pre(a_3) = \{a, b\} & add(a_3) = \{c, d\} & cost(a_3) = 3 \\
pre(a_4) = \{c\} & add(a_4) = \{b, e\} & cost(a_4) = 2 \\
pre(a_5) = \{c, e\} & add(a_5) = \{d\} & cost(a_5) = 4 \\
pre(a_6) = \{d, e\} & add(a_6) = \{g\} & cost(a_6) = 0.
\end{array}$$

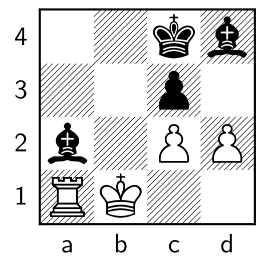
Compute $h^{\text{LM-cut}}(I)$ and provide all intermediate results in the same way they were given in the example of the lecture. Specifically, provide for each iteration (except the last):

- the justification graph with h^{max} annotations and marked goal zone
- the cut
- the cost of the cut
- the updated action costs

In cases where the precondition choice function is not deterministic, choose the precondition in alphabetical order.

Exercise 11.4 (2 marks)

Consider the following state in a (small-scale) chess game where white is to move:

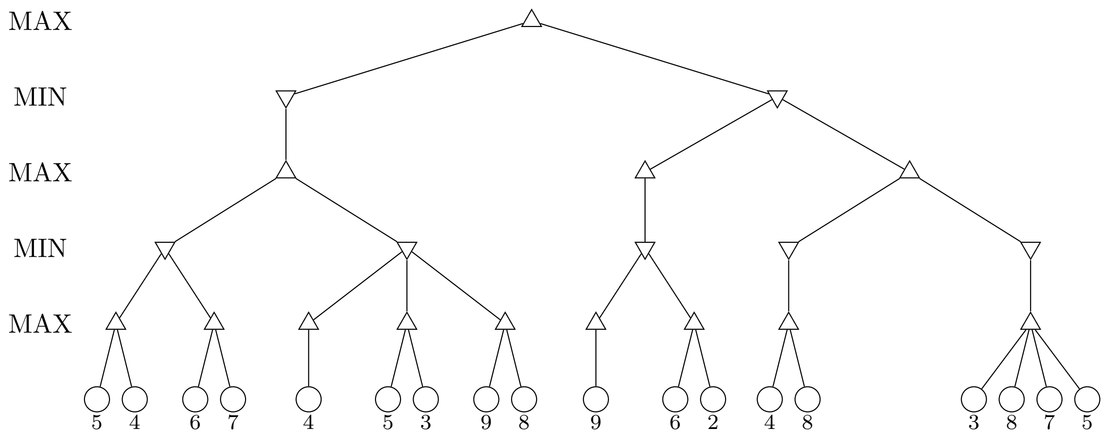


White started on row 1, so its pawns can only move upwards, and vice versa for black. Besides the movements rules for figures in chess (see, e.g., `https://en.wikipedia.org/wiki/Rules_of_chess`), it is important to know that players are not allowed to make a move that would put themselves into check. That is, in the above state, white cannot ignore the check position it is in due to the bishop on a2.

Draw the game tree of the chess game rooted at this state up to depth 2, with MAX being the white player and MIN the black player. To assign a utility to leaf nodes, use the evaluation function shown in the lecture wich assigns a value to each type of piece (pawn 1, knight 3, bishop 3, rook 5, queen 9) and computes the sum over all pieces on the board, using a positive sign for pieces of MAX and negative sign for those of MIN. Your nodes should contain a representation of the game state.

Exercise 11.5 (2 marks)

Consider the following game tree T:



Use the minimax algorithm to annotate T. According to the values computed by the minimax algorithm, what is the playout if both players play optimally? Mark it in the annotated tree.

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.