Dresden University of Technology

Computational Logic ∴ Group

Dr. Hannes Strass

Dr. Sarah Gaggl

Algorithmic Game Theory

Summer Term 2026

Exercises 5

21/05/2026

Problem 1.
This task exercises the basic concepts of game playing, using Tic-Tac-Toe as an example. We define X_n as the number of rows, columns, or diagonals with exactly n Xs and no Os. Similarly, O_n is the number of rows, columns, or diagonals with just n Os. The utility function assigns $+1$ to any position with $X_3 = 1$ and -1 to any position with $O_3 = 1$. All other terminal positions have utility 0. For nonterminal positions, we use a linear evaluation function defined as

$$Eval(s) = 3 \cdot X_2(s) + X_1(s) - (3 \cdot O_2(s) + O_1(s))$$

(a) Approximately how many possible games of Tic-Tac-Toe are there?

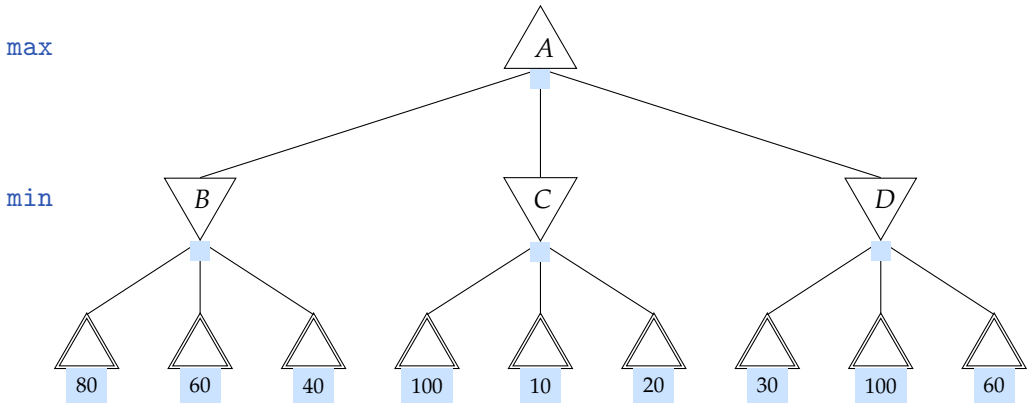
(b) Show the whole game tree starting from an empty board down to depth 2 (i.e., one X and one O on the board), taking symmetry into account.

(c) Mark on your tree the evaluations of all the positions at depth 2.

(d) Using the (heuristic) minimax algorithm, mark on your tree the backed-up values for the positions at depth 1 and 0, and use those values to choose the best starting move.

(e) Circle the nodes at depth 2 that would not be evaluated if alpha-beta pruning were applied, assuming the nodes are generated in the optimal order for alpha-beta pruning.

Problem 2.
Consider the following 2-player zero-sum game displaying the payoffs for the max player.



Do the following:

- Determine for each decision node of a player the best move by application of the minimax algorithm and indicate the minimax value of the game.
- Circle all nodes that would not be explored if the Alpha-Beta algorithm is applied to this game tree.

Problem 3.
Consider the two-player zero-sum game described in Figure 1 below.

(a) Draw the complete game tree, using the following conventions:

- Write each state as (s_A, s_B), where s_A and s_B denote the token locations.
- Mark each terminal state and write its game value beneath it.
- Mark loop states (states that already appear on the path to the root). Since their value is unclear, annotate each with a "?" beneath it.

(b) Now mark each node with its backed-up minimax value (also beneath the node). Explain how you handled the "?" values and why.

(c) Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

(d) This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins iff n is even.

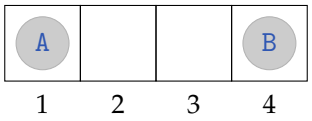


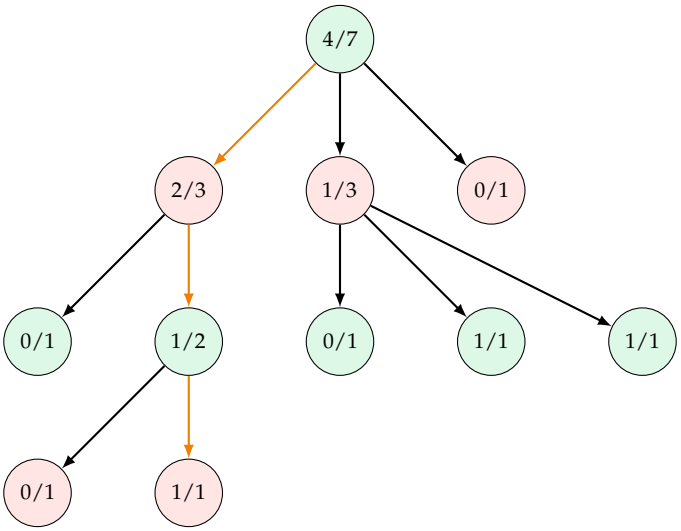
Figure 1: Player A moves first. The two players take turns moving, and each player must move their token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is $+1$; if player B reaches space 1 first, then the value of the game to A is -1.

Problem 4.
Consider the following game tree where the selection step has already been applied. Do the following:

(a) Follow the pre-selected path and graphically illustrate all subsequent steps involved in Monte Carlo Tree Search.

(b) How do the values along the selected path change for different simulation results?

(c) Decide whether the path selected in this example is also recommended by the UCT selection policy from the lecture (based on UCB1) where $c = \sqrt{2}$.


4/7
2/3
1/3
0/1
0/1
1/2
0/1
1/1
1/1
0/1
1/1