INFO8006 Introduction to Artificial Intelligence

Exercises 2: Games and adversarial search

Learning outcomes

At the end of this exercise session you should be able to:

- Define formally the search problem associated to a game (IPATTU[1])
- Define and apply the Minimax algorithm
- Define and apply $\alpha - \beta$ pruning for Minimax
- Define H-Minimax, Expectiminimax, Monte-Carlo Tree Search

Exercise 1: Tic-Tac-Toe (AIMA, Ex 5.9)

Tic-tac-toe is a paper-and-pencil game for two players, X and O, who take turns marking the cells of a 3×3 *grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.*
We define X_n as the number of rows, columns, or diagonals with exactly n X's and no O's. Similarly, O_n is the number of rows, columns, or diagonals with just n O's. 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 $Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$.

1. Define the search problem associated with the Tic-tac-toe game.
2. Approximately how many possible games states of Tic-tac-toe are there?
3. Show the whole game tree starting from an empty grid down to depth 2 (i.e., one X and one O on the board), taking symmetry into account.
4. Mark on your tree the evaluations of all the positions at depth 2.
5. Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0, and use those values to choose the best starting move.
6. 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.

Exercise 2: 21 misère game (January 2019)

The game "21" is played as a misère game with any number of players who take turns saying a number. The first player says "1" and each player in turn increases the number by 1, 2, or 3, but may not exceed 21; the player who says "21" or a larger number loses.

1. Define the search problem associated with the 2-player version of the "21" game.
2. For this subquestion and the following, consider the game of "5" (still in its 2-player version) which has the same rule except that you should not say 5 or more. Show the whole game tree.
3. (a) Using the minimax algorithm, mark on your tree the backed-up values, and use those values to choose the best starting move
   (b) Assume alpha–beta pruning were applied in optimal order. Draw the game tree containing only the nodes that would be evaluated.

[1] Initial state - Player function - Actions function - Transition model - Terminal test - Utility function

Exercise 3: Quiz

1. In a fully observable, turn-taking, zero-sum game between two perfectly rational players, it does not help the first player to know what strategy the second player is using (that is, what move the second player will make, given the first player's move).
2. What is a quiescent position?
3. In MCTS, what is encouraged by each term of the sum in the formula $\frac{Q(n',p)}{N(n')} + c\sqrt{\frac{2\log N(n)}{N(n'}}$?
4. Which heuristic is correct (i.e. resulting in a rational agent) (see image below)?

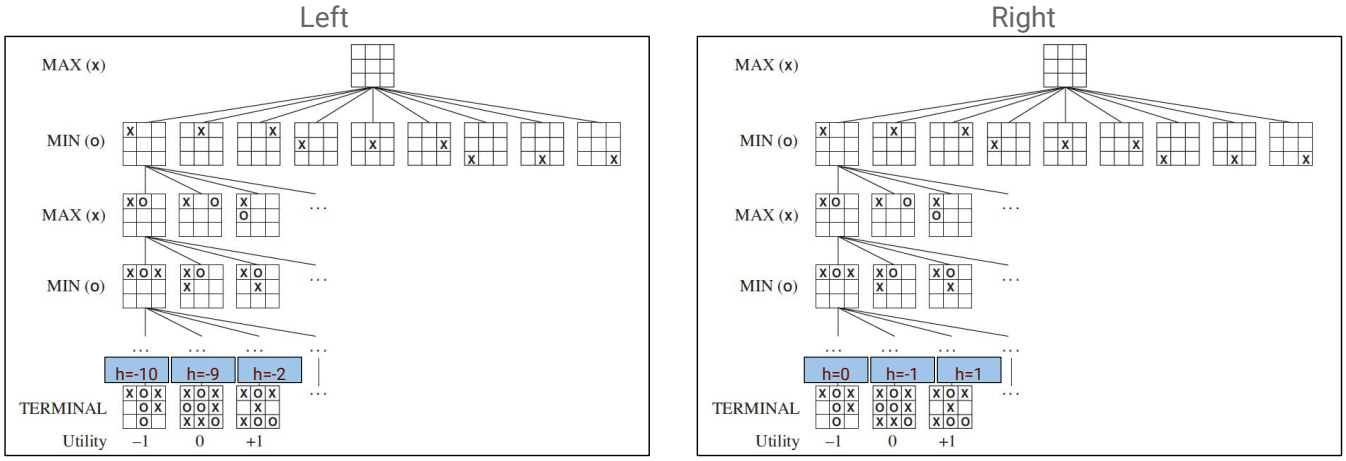


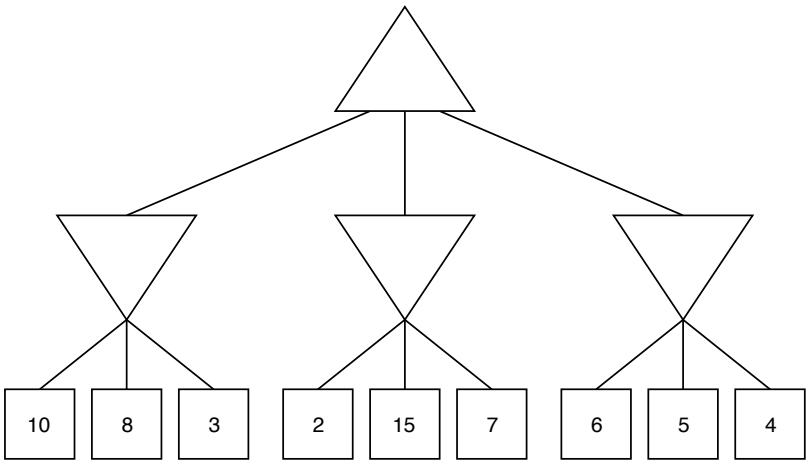
Figure 1: Two possible heuristics

Exercise 4: Chess and transposition table ⋆ (AIMA, Ex 5.15)

Suppose you have a chess program that can evaluate 16 million nodes per second. Decide on a compact representation of a game state for storage in a transposition table.

1. About how many entries can you fit in a 4-gigabyte in-memory table?
2. Will that be enough for the three minutes of search allocated for one move?
3. How many table lookups can you do in the time it would take to do one evaluation? Suppose that you have a 3,2GHz machine and that it takes 20 operations to do one lookup on the transposition table.

Exercise 5: Minimax ⋆ (Berkeley CS188 Fall 2019)



1. Consider the zero-sum game tree shown above. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, fill in the minimax value of each node.

2. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.

3. Again, consider the same zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random. Fill in the expectimax value of each node. The game tree is redrawn below for your convenience.

4. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not.