COMP30024 Artificial Intelligence - Tutorial Problems (Part 2)

Questions based on exercises from Russell and Norvig (3rd edition) have the original question numbers shown in brackets. Many of these questions are designed to provoke discussion in tutorials, rather than having a simple, closed-form answer.

4. *Game Playing*

4.1 (RN5.9) - This problem exercises the basic concepts of game playing using Tic-Tac-Toe/Noughts-and-Crosses as an example.

a. Can you think of a good evaluation function for noughts-and-crosses?

b. We define $X(n)$ as the number of rows, columns, or diagonals with exactly n X's and zero O's. Similarly for $O(n)$. 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. We will use a linear evaluation function defined as:

   $Eval = 3X(2) + X(1) - (3O(2) + O(1))$

   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. You should have 3 positions at level 1 and 12 positions at level 2.

   Mark on your tree the evaluations of all positions at level 2.

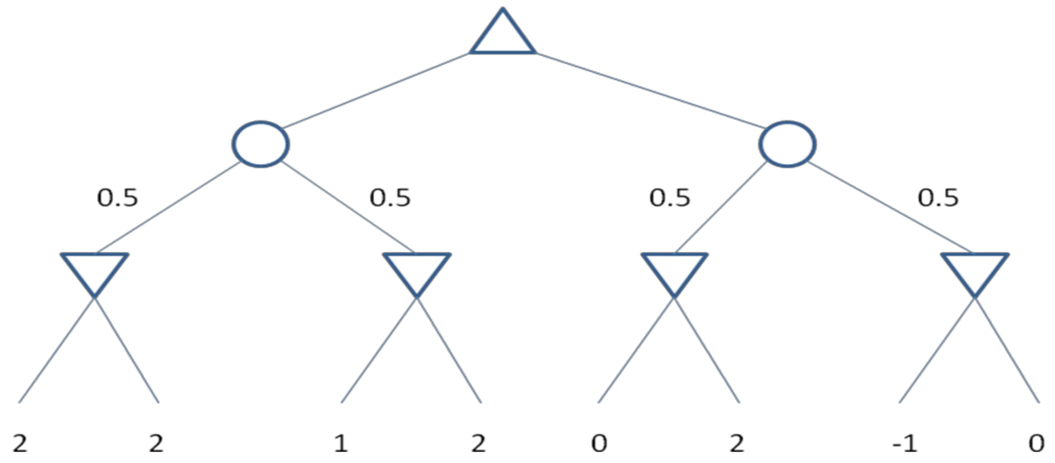
c. Mark on your tree all the values for the positions at levels 1 and 0, using the minimax algorithm, and use them to choose the best starting move.

d. Circle the nodes at level 2 that would *not* be evaluated if alpha-beta pruning were applied, assuming the nodes were generated *in the optimal order for alpha-beta pruning*.

4.2 (RN 1st ed) The Chinook checkers/draughts program makes extensive use of endgame databases, which provide exact values for every move with 6 pieces or less on the board. How might such databases be generated efficiently?

4.3 (RN 1st ed) The minimax algorithm returns the best move for MAX under the assumption that MIN plays optimally. What happens when MIN plays suboptimally?

4.4 (RN5.16) Consider the complete game tree for a trivial game, which includes chance nodes (shown as circles) where a fair coin is tossed. Assume that the leaf nodes are to be evaluated in left-to-right order, and that before a leaf node is evaluated, we know nothing about its value, i.e., the range of possible values at a node is $-\infty$ to $+\infty$.



a. Mark on the tree the expecti-minimax value of each node, and use an arrow to indicate which is the best move at the root.

b. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers.

5. *Learning for Game Playing*

5.1 Discuss how you would apply each of the following approaches for learning in games to Noughts-and-Crosses. What are the advantages and disadvantages of each approach for this game?

a. Book learning
b. Learning an ordering for moves to maximise alpha-beta pruning
c. Learning the weights of an evaluation function such as the function in Ex4.1

5.2 One of the pioneers of machine learning in AI was a British researcher called Donald Michie (he also worked with Alan Turing as a code-breaker during World War 2). He developed a technique called MENACE, which could learn to play a highly effective game of Noughts-and-Crosses from experience.

For each possible state of the game, MENACE maintains a set of counters $\{c_1, \ldots, c_9\}$, where $p_i = c_i / [\ \Sigma_{j=1,..,9}\ c_j\ ]$ corresponds to the probability of choosing move *i* in that state. Note that if a square *i* is occupied in a particular state, then the corresponding move *i* in that state is not legal, and the counter $c_i = 0$ in that state.

a. How many possible states are there in Noughts-and-Crosses (ignoring symmetry or reflections)? [For a challenge: If you eliminate redundant states due to symmetry and reflection, what is the minimum number of states you need to consider?]

In MENACE, during a game the computer remembers the sequence of states and moves taken in the game. Then at the end of the game, for each state in the game where the computer made a move, MENACE updates the counter c_i corresponding to the move *i* made in that state as follows:

- If the computer won the game, then increase the counter c_i in that state.
- If the computer lost the game, then decrease the counter c_i in that state.
- If the game was a draw, then leave the counter c_i in that state unchanged.

b. After a large number of games, what effect do you expect these updates to have?

c. Should you always update counters in the visited states by the same amount? Explain your answer.

d. What are the strengths or weaknesses of applying the MENACE learning technique to the game we are using in the project?