Games and Utility

- In **adversarial search**, you need to choose actions in a situation where an adversary also gets to take actions that could impede your progress. A **zero-sum game** is one in which the sum of the players' scores is always zero (or more generally, the average of their scores is constant; in other words, there are no "win-win" or "lose-lose" outcomes).

- Each state of the game has some **utility**. Often, the utility depends on many deeper levels of the game tree and so cannot be computed exactly; in this case, it can be approximated using a **static evaluation** function.

- **Minimax** search is a simple method for choosing a move in a game where one player is trying to minimize the final score and the other player is trying to maximize it. It is an optimal strategy, assuming that both players are playing optimally.

- Minimax can be sped up (from $O(b^m)$ to $O(b^{m/2})$) using **alpha-beta pruning**, which avoids exploring nodes that could not result in a better outcome than has already been found.

- Because you often have limited resources, you can use **iterative deepening** (aka **progressive deepening**), searching to a certain depth at a time. The nodes at that depth can then be reordered to try to maximize the amount of pruning that alpha-beta can do in subsequent depths.

- Some games have such large search spaces that it is impractical to perform a full search. For such games, you can use **Monte Carlo tree search**, in which a move is chosen by randomly simulating many games starting from the current position, choosing the best ones and simulating those some more, etc, to estimate the best move.

- In **nondeterministic search**, you need to make plans when the outcomes of actions are uncertain. Instead of a sequence of actions, you need to specify a **policy**, a function π from states to actions.

- **Expectimax** search is a generalization of minimax that chooses moves based on Maximum Expected Utility (MEU). In the game tree, in addition to nodes where players make choices, there are chance nodes (representing, e.g., the roll of a die), and the value of a chance node is the expected value of the outcomes it could result in.

- In deterministic minimax, only the relative order of the utilities matters, so any monotonic transformation of the utilities will preserve the choices the players make. In expectimax, only affine transformations ($x \rightarrow ax + b$) will preserve the utilities, because the process of taking the expectation makes the cardinalities matter.

- A **preference** specifies which of two states is better; preferences are notated $A \succ B$ (A is better than B), $A \succeq B$ (A is at least as good as B), and $A \sim B$ (A and B are equally preferable). A **lottery** specifies probabilities for different states; for example, $L = [p, A; (1-p), B]$ means that lottery L has probability p of resulting in A and probability $(1-p)$ of resulting in B.
- Any "rational" set of preferences can be summarized as a utility function U from states to real numbers.

Exercises

1. Hexapawn is a deterministic two-player game invented by Martin Gardner. It is played on a rectangular board of variable size, for example on a 3 x 3 board or on a chessboard. The goal of each player is to advance one of their pawns to the opposite end of the board or to prevent the other player from moving. There is no en passant and no double first moves as in chess.

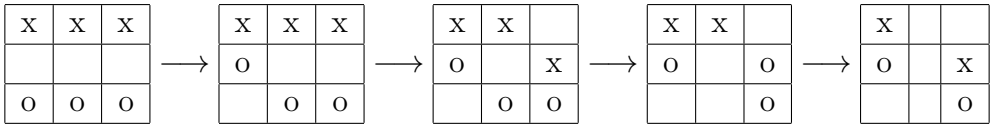


Figure 1: A sample game of hexapawn

Consider the following static evaluator for a board s of hexapawn:

$$\text{Eval}(s) = \begin{cases} 100 & \text{if the board is won by } o \\ -100 & \text{if the board is won by } x \\ \sum_{i \in O} r_i^2 - \sum_{j \in X} (4 - r_j)^2 & \text{otherwise} \end{cases}$$

where r_i denotes the row number of the ith piece. This construction makes the first player, o, the maximizing player.

(a) What is the static value of the following board position? Note the row numbering scheme, which we will be using throughout this problem.

3	x	x	x
2	o		
1		o	o

(b) What is the static value of the following board position?

x		x
o	o	
	o	

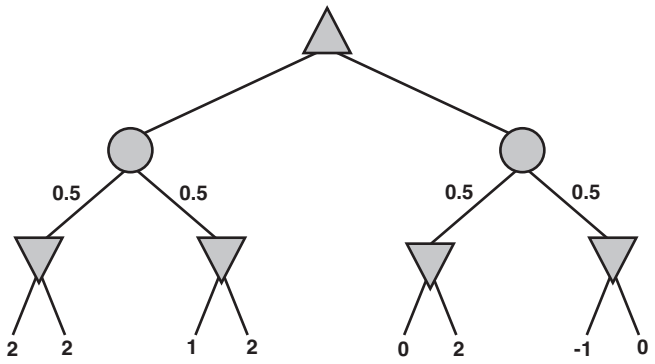
(c) You come across an abandoned game of hexapawn in the following state, with x to move:

x	x	x
		o
o	o	

Using the minimax algorithm to look 2 moves ahead, determine the best move for x. What is its static value?

2. (AIMA 5.7) Prove the following assertion: For every game tree, the utility obtained by `MAX` using minimax decisions against a suboptimal `MIN` will never be lower than the utility obtained playing against an optimal `MIN`.

3. (AIMA 5.16) This question considers pruning in a game with chance nodes. Figure 1 shows the complete game tree for a trivial game. 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 — the range of possible values is $-\infty$ to ∞.



   (a) Mark the value of all internal nodes, and indicate the best move at the root with an arrow.

   (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.

   (c) Suppose the leaf node values are known to lie between -2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?

   (d) Circle all the leaves that need not be evaluated under the assumption in (c).