Module 3

Problem Solving

ADVERSARIAL SEARCH
GAMES

- **competitive** environments, in which the agents' goals are in conflict, giving rise GAME to **adversarial search** problems—often known as **games**
- In AI, the most common games are of a rather specialized kind—what game theorists call deterministic, turn-taking, two-player, ZERO-SUM GAMES **zero-sum games** of **perfect information** (such as chess)
- For example, chess has an average branching factor of about 35,
- and games often go to 50 moves by each player, so the search tree has about 35^{100} nodes.
- Games, like the real world, therefore require the ability to make *some* decision even when calculating the *optimal* decision is infeasible.
- techniques for choosing a good move PRUNING when time is limited. **Pruning** allows us to ignore portions of the search tree that make no difference to the final choice, and heuristic **evaluation functions** allow us to approximate the true utility of a state without doing a complete search.

- first consider games with two players, whom we call MAX and MIN for reasons that will soon become obvious. MAX moves first, and then they take turns moving until the game is over. At the end of the game, points are awarded to the winning player and penalties are given to the loser.
- A game can be formally defined as a kind of search problem with the following elements
- **• S0: The initial state, which specifies how the game is set up at the start.**
- **• PLAYER(s): Defines which player has the move in a state.**
- **• ACTIONS(s): Returns the set of legal moves in a state.**
- **• RESULT(s, a): The transition model, which defines the result of a move.**
- **• TERMINAL-TEST(s): A terminal test, which is true when the game is over and false otherwise. States where the game has ended are called terminal states.**
- **• UTILITY(s, p): A utility function (also called an objective function or payoff function), defines the final numeric value for a game that ends in terminal state s for** a player p. Inchess, the outcome is a win, loss, or draw, with values +1, 0, or 1/2

- **zero-sum game** is (confusingly) defined as one where the total payoff to all players is the same for every instance of the game. Chess is zero-sum because every game has payoff of either 0 + 1, 1 + 0 or 1/2+ 1/2 .

• Game tree

- The initial state, ACTIONS function, and RESULT function define the **game tree** for the game—a tree where the nodes are game states and the edges are moves.

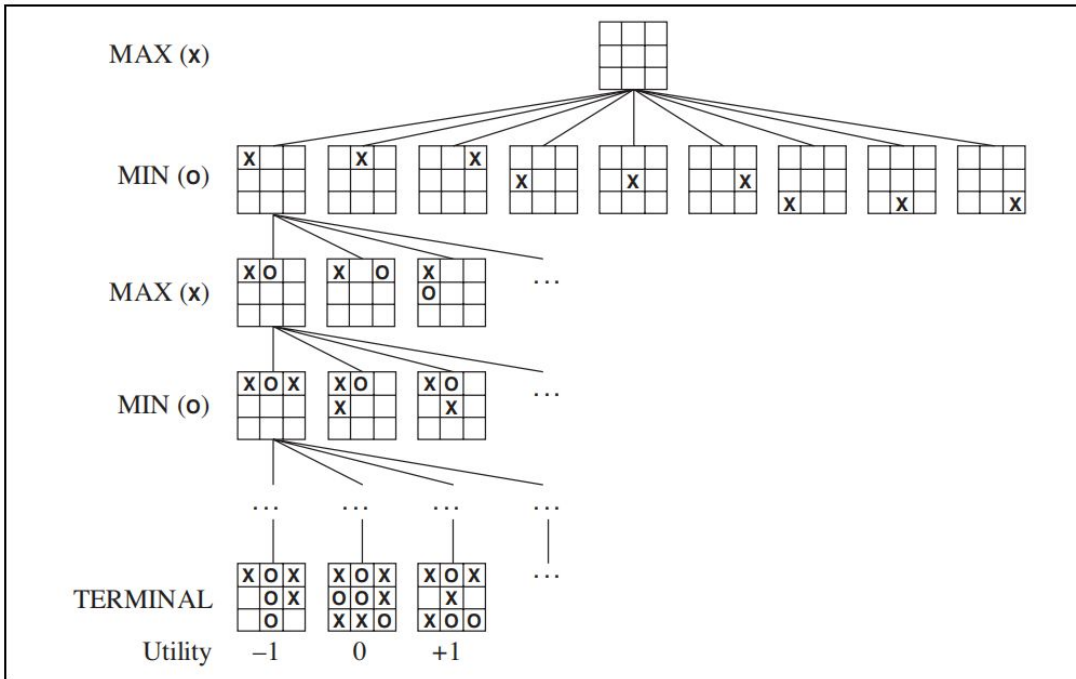


Figure 5.1 A (partial) game tree for the game of tic-tac-toe. The top node is the initial state, and MAX moves first, placing an X in an empty square. We show part of the tree, giving alternating moves by MIN (O) and MAX (X), until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.

- From the initial state, MAX has nine possible moves.
- Play alternates between MAX's placing an X and MIN's placing an O until we reach leaf nodes corresponding to terminal states such that one player has three in a row or all the squares are filled
- The number on each leaf node indicates the utility value of the terminal state from the point of view of MAX;
- high values are assumed to be good for MAX and bad for MIN
- **search tree** for a tree that is superimposed on the full game tree, and examines enough nodes to allow a player to determine what move to make.

OPTIMAL DECISIONS IN GAMES

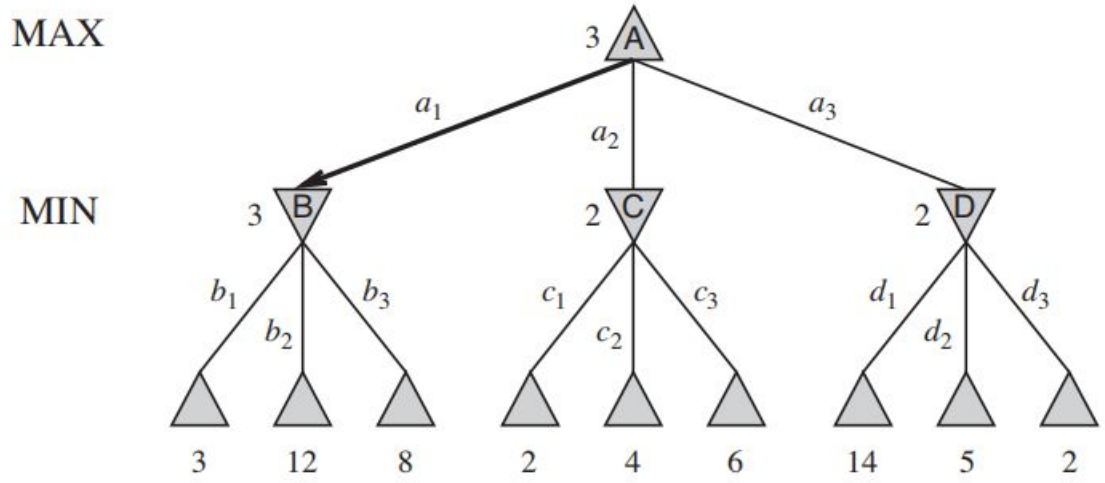


Figure 5.2 A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

- The minimax value of a node is of being useful in the corresponding state, *assuming that both players play optimally* from there to the end of the game.

the minimax value of a terminal state is just its utility(the state of being useful).

MAX prefers to move to a state of maximum value, whereas MIN prefers a state of minimum value.

- The terminal nodes on the bottom level get their utility values from the game's UTILITY function
- **minimax decision:** optimal choice for MAX at root that leads to the state with the highest minimax value

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

The minimax algorithm

- **minimax algorithm** computes the minimax decision from the current state.
- The recursion proceeds all the way down to the leaves of the tree, and then the minimax values are **backed up** through the tree as the recursion unwinds.
- minimax algorithm performs a complete depth-first exploration of the game tree
- It is recursive backtracking algorithm.
- maximum depth of the tree is m and there are b legal moves at each point,
- time complexity □O(b ^{m}).
- space complexity □ O(bm), for an algorithm that generates all actions at once, or O(m), for an algorithm that generates actions one at a time

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Figure 5.3 An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

Optimal decisions in multiplayer games

- replace the single value for each node with a *vector* of values
- in a three-player game with players A, B, and C, a vector vA, vB, vC is associated with each node

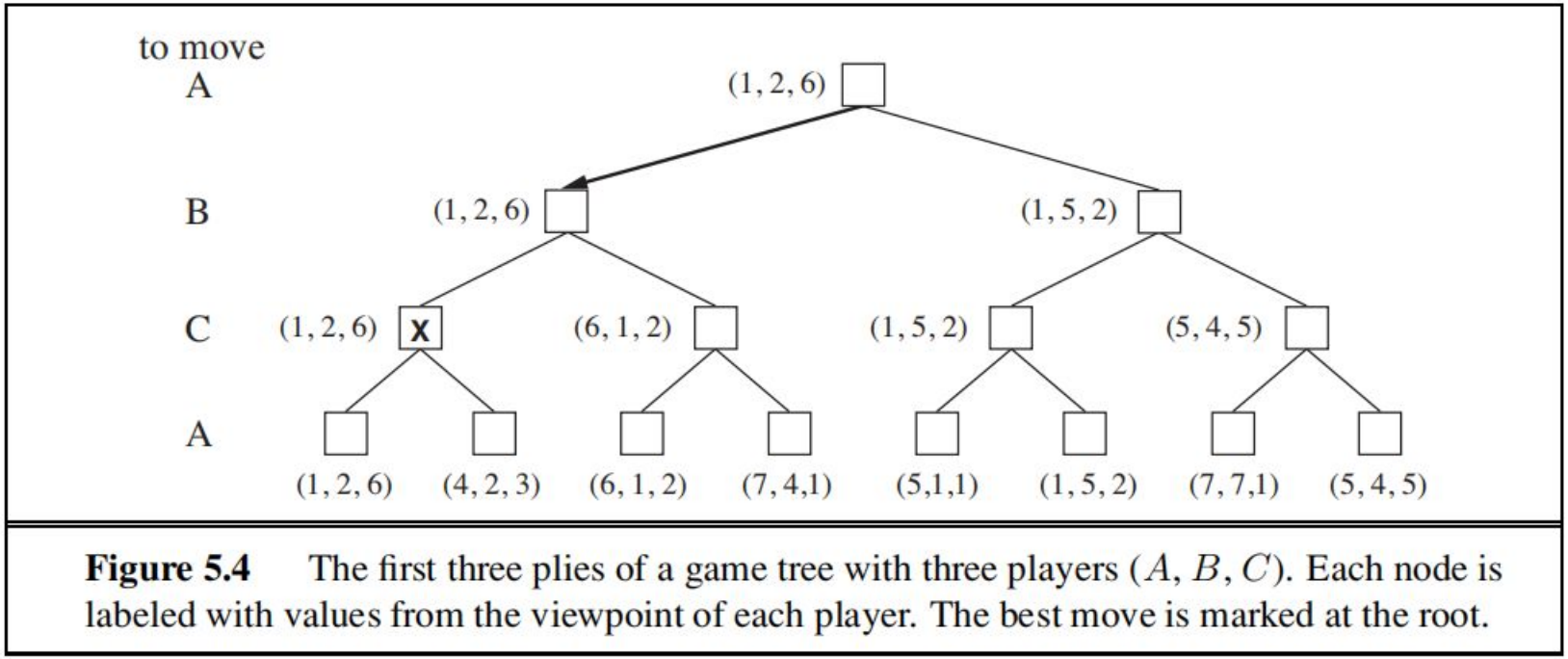


Figure 5.4 The first three plies of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

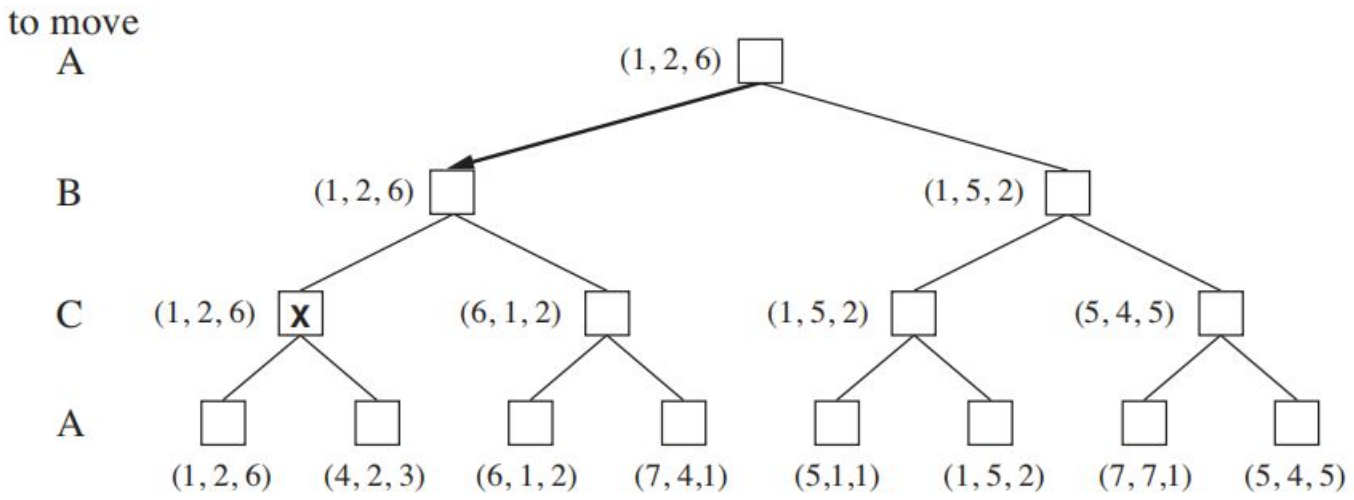


Figure 5.4 The first three plies of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

The backed-up value of a node n is always the utility vector of the successor state with the highest value for the player choosing at n

ALPHA BETA PRUNING

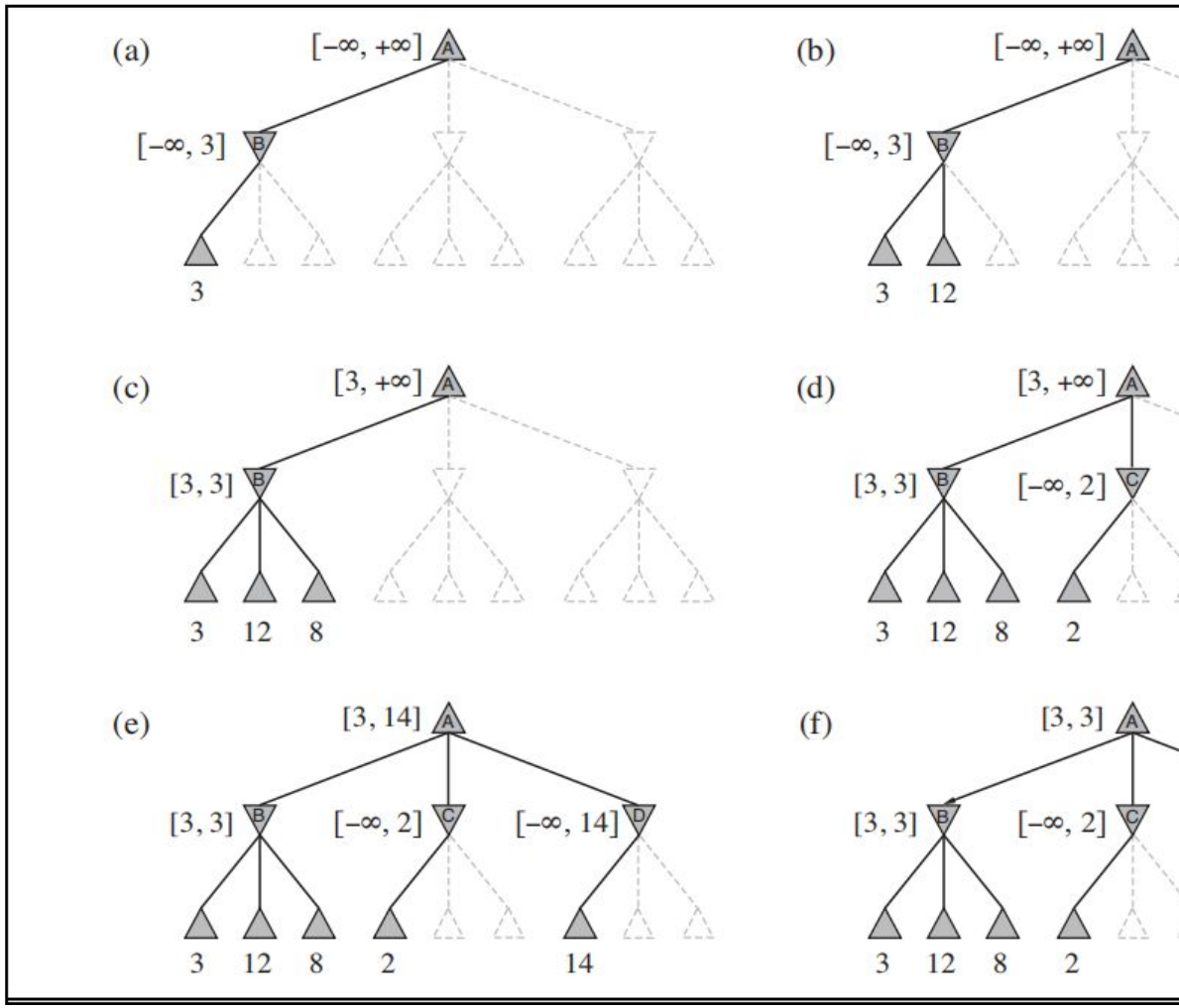
- The problem with minimax search is that the number of game states it has to examine is exponential in the depth of the tree
- To avoid this it is possible to compute the correct minimax decision without looking at every node in the game tree by **pruning**
- In alpha beta pruning algorithm it prunes away branches that cannot possibly influence the final decision
- Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves
- If Player has a better choice m either at the parent node of n or at any choice point further up, then n *will never be reached in actual play.* So once we have found out enough about n (by examining some of its descendants) to reach this conclusion, we can prune it.

$\alpha =$ the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX.

$\beta =$ the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN.

- Alpha–beta search updates the values of α and β as it goes along and prunes the remaining branches at a node (i.e., terminates the recursive call) as soon as the value of the current node is known to be worse than the current α or β value for MAX or MIN, respectively

The outcome is that we can identify the minimax decision without ever evaluating two of the leaf nodes



- Let the two unevaluated successors of node C have values x and y Then the value of the root node is given by

$$\begin{aligned}
\textsc{Minimax}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\
&= \max(3, \min(2,x,y), 2) \\
&= \max(3,z,2) \qquad \text{where } z = \min(2,x,y) \leq 2 \\
&= 3.
\end{aligned}$$

- In other words, the value of the root and hence the minimax decision are independent of the values of the pruned leaves x and y.

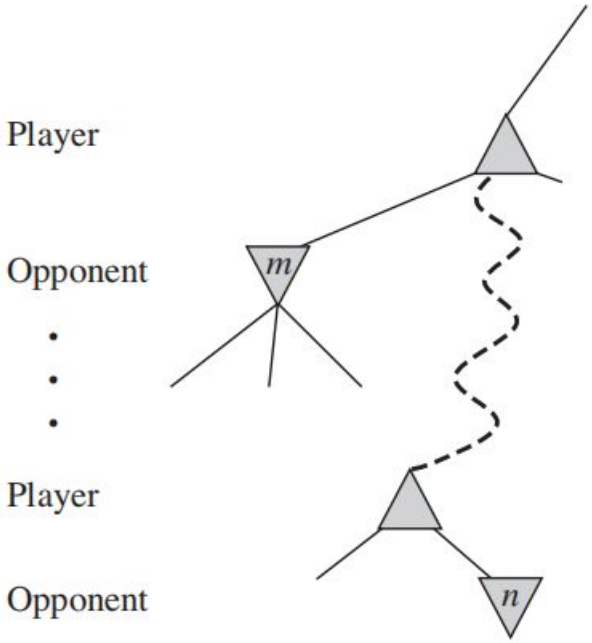


Figure 5.6 The general case for alpha–beta pruning. If m is better than n for Player, we will never get to n in play.

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Figure 5.7 The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

Move ordering

- The effectiveness of alpha–beta pruning is highly dependent on the order in which the states are examined.