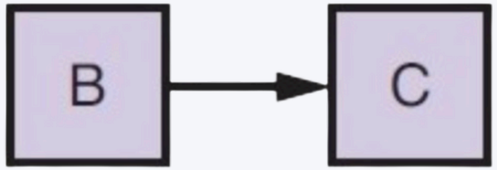
أكاديمية طويق
Tuwaiq Academy

Adversarial Search

ADVERSARIAL SEARCH



single agent controlling the environment

The agent looks to find a sequence of actions that leads to the goal | Goal-based search agent.

But what happens when other agents' interest conflict with you?

Game Playing **| agent tries to anticipate the unpredictable opponent’s next move**

Game-playing agents in AI are designed to be rational and will choose the move that leads to the best or high-quality gain for them.

Adversarial search is a special type of search where multiple agents influence the current problem state >> Competitive multi-agent environments.

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

ADVERSARIAL SEARCH

- The **solution in AI-based games** is not a fixed sequence of actions but a **strategy (policy).**
- We specify a move for every possible opponent reply.
- Game playing in AI assumes competitive (**unpredictable**) multi-agent environments.

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

ADVERSARIAL SEARCH

Assumptions:

- Environment is deterministic and fully observable (i.e., perfect information)
- Competitive (unpredictable) multi-agent environments.
- Two agents act alternately (taking turns).
- Zero-sum game: One player in the game tries to maximize a single value, while the other player tries to minimize it.

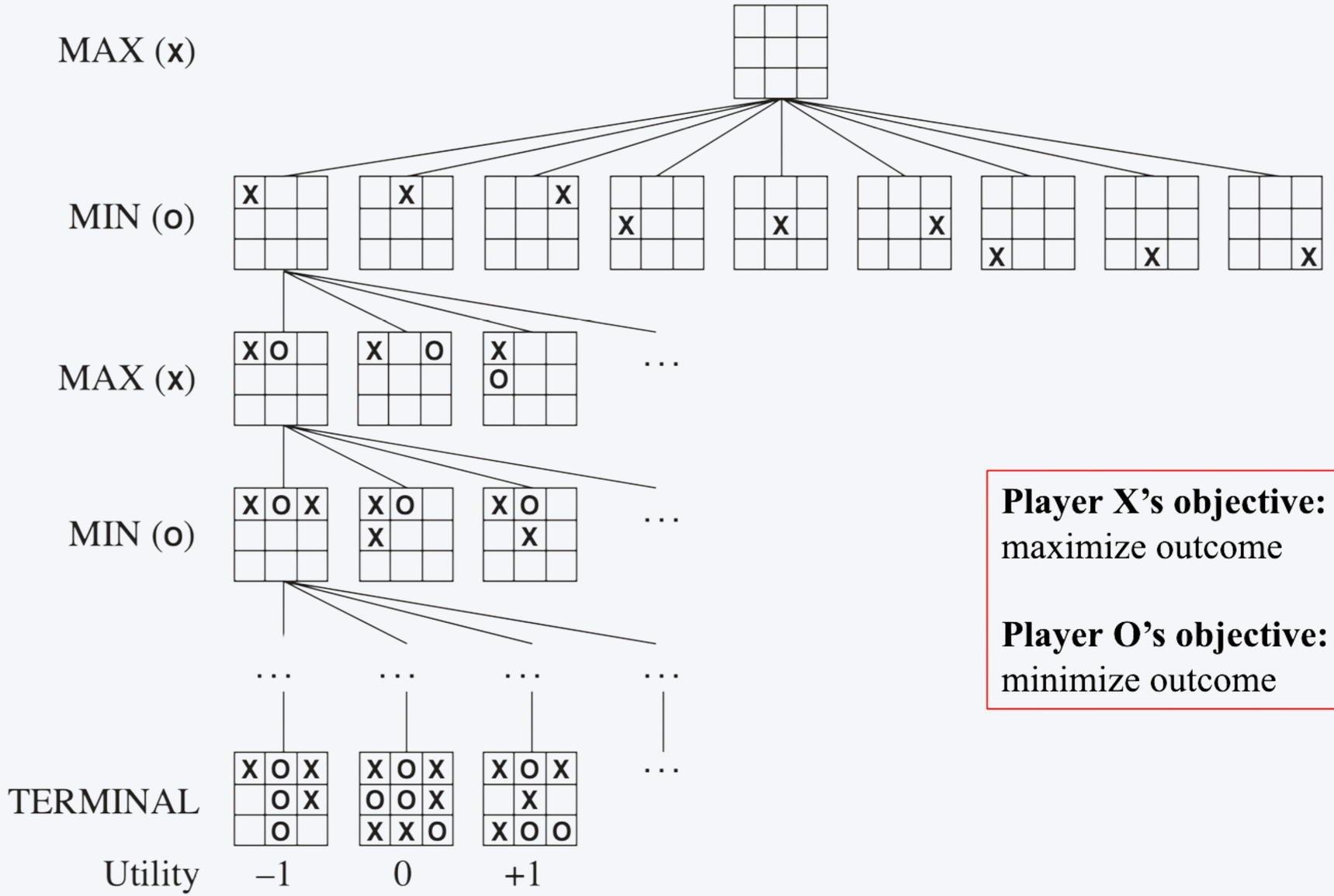
Game problem formulation:

- The initial state.
- **Operators/Actions:** legal moves a player can make.
- **Transition model:** defines the result of a legal move.
- **Goal (terminal test):** determines when the game is over.
- **Utility (payoff) function:** measures the outcome of the game and its desirability.

Game Tree

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

ADVERSARIAL SEARCH



Player X's objective:
maximize outcome

Player O's objective:
minimize outcome

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

HOW ADVERSARIAL SEARCH WORKS?

Game Tree Representation: The diagram represents all possible states of the game as a tree. The root is the current state of the game, and the branches are the potential moves available to each player.

Utility Values: At the terminal nodes (leaves), utility values represent the outcome of the game:

- +1 if Max wins.
- -1 if Min wins.
- 0 for a draw.

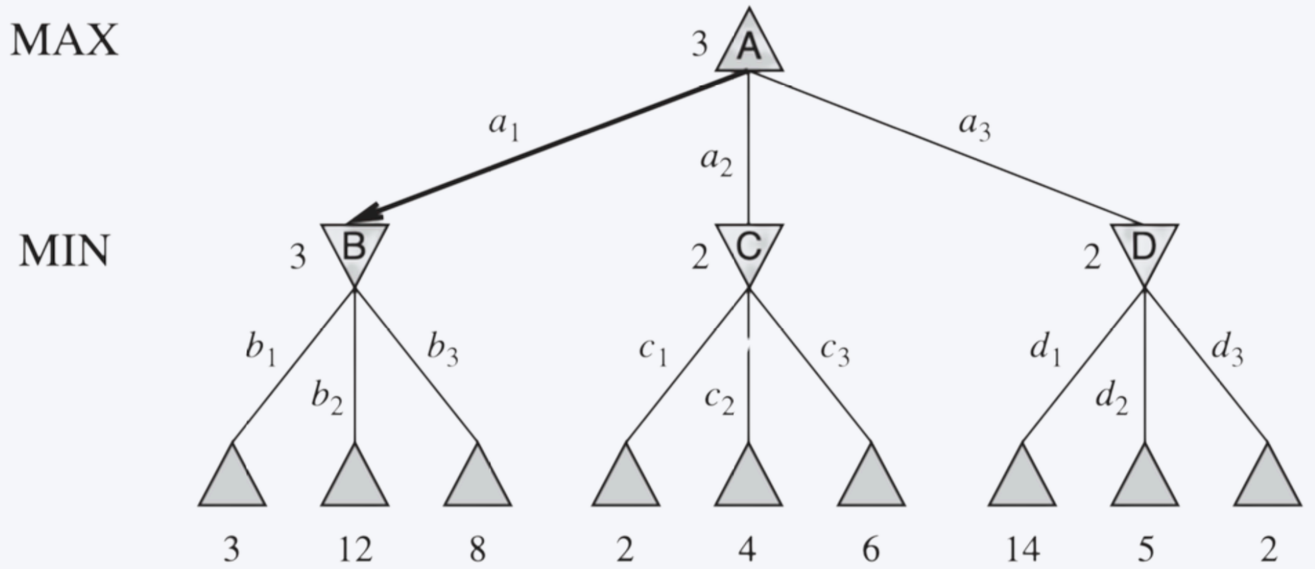
Backward Induction: The **Minimax algorithm** is applied to evaluate the **best move:**

1. Starting from the terminal nodes, the utility values propagate back up the tree.
2. Max selects the move with the highest value from the child nodes.
3. Min selects the move with the lowest value from the child nodes.
4. This alternation continues until the best move for Max is identified at the root.

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

MINIMAX ALGORITHM

1. **Generate the Game Tree:** Create a tree of all possible moves and outcomes from the current game state.
2. **Evaluate Terminal Nodes:** Assign utility values to terminal nodes (end states).
3. **Backpropagate Utilities:**
   - If it's Max's turn, choose the child node with the maximum utility value.
   - If it's Min's turn, choose the child node with the minimum utility value.
4. **Optimal Move Selection:** At the root node (current game state), Max selects the move that leads to the best utility value for him considering Min will respond optimally.



Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

WHY IS IT ADVERSARIAL?

The search is adversarial because the two players have directly opposing objectives:

- Max wants to maximize the utility for himself.
- Min wants to minimize Max's utility, effectively working against Max’s strategy.

This setup mirrors many competitive scenarios (e.g., chess, checkers, tic-tac-toe), where each player's decision depends not only on their strategy but also on predicting the opponent's moves.

Adversarial search is central to AI in games, allowing algorithms to simulate and counteract an opponent's moves effectively.

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

EVALUATING MINIMAX SEARCH:

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes
- **Time Complexity:** $O(b^m)$ (m is max depth of the tree).
- **Space Complexity:**
  - O(bm) when generates all actions at once
  - O(m) when generates actions one at a time.

The time complexity is a major problem in Minimax because it grows exponentially in the depth of the tree, effectively with real games.

Possible remedy: compute the correct minimax decision without looking at every node in the game tree à Pruning!

We shall consider a modification of Minimax technique called "**Alpha-Beta Pruning"**

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)

ALPHA-BETA PRUNING

A technique applied to the **standard Minimax tree** and it returns the same move as **minimax** but 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.

Main elements: Each node in the tree include two values, α and β.

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

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

Basic Principle: If a move is determined worse than another move already examined, then there is no need for further examination of the node, i.e., when $\alpha \geq \beta$.

Poole & Mackworth, 2010 (http://artint.info/slides/), and Russell & Norvig, 2005 (http://aima.cs.berkeley.edu/)