CCES – College of Computer Engineering and Sciences

Prince Sattam Bin Abdulaziz University

CS600: Advanced AI

Multiagent Search

Chapter 3 — Adversarial Games: Minimax, α–β, and MCTS

Dr. Khalil Chebil

Artificial Intelligence Course

Chapter Outline | What we will cover

Multiagent Environments
AND-OR Search Trees
Evaluation & Depth-Limited Search
Minimax
Alpha–Beta & MCTS
Choosing a Method & Summary

Learning Objectives

- Understand **multiagent** environments and strategy selection
- Model contingencies with **AND-OR** search trees
- Apply **minimax** and **alpha–beta** pruning to adversarial games
- Use **evaluation functions** and **Monte Carlo Tree Search**

Section 1

Multiagent Environments

From Plan Finding to Strategy Selection | Others react to your moves

In **single-agent** search, an action determines the next state. In **multiagent** search, other agents cooperate, compete, or act independently.

The central shift

A move does not define *one* future — it defines a **set of possible futures**. We seek a **strategy**, not a fixed plan.

Key challenges

- Others have different goals & information
- Effects depend on others' responses
- Needs contingency reasoning
- Search grows fast with each reply

Types of Multiagent Environments | Method depends on the relationship

Competitive

Conflicting objectives; one's gain is another's loss.

Chess, Go, Tic-Tac-Toe

$\Rightarrow$ adversarial search

Independent

Act without explicit coordination, yet still affect one another.

Self-driving cars in traffic

Cooperative

Share information or align objectives for a joint outcome.

Warehouse robots, rescue teams

Tic-Tac-Toe: after you place `X`, the opponent chooses `O` — your move opens many futures, not one.

Section 2

AND-OR Search Trees

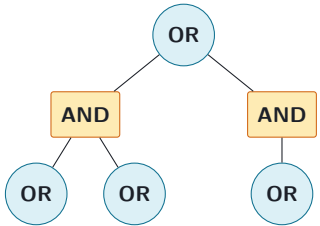
AND-OR Search Trees | Our choices vs. outcomes we must handle

- **OR node** — *our* turn: pick **one** good action
- **AND node** — outside our control: the plan must work for **all** relevant continuations

In adversarial games

OR = our move AND = opponent's possible replies.

(AND nodes may also model uncertain environment outcomes.)



One good choice at OR; ready for every branch at AND.

AO-Search & Many Agents | Guaranteed paths in an AND-OR tree

AO-search logic

- At an **OR** node: succeed if **some** action succeeds
- At an **AND** node: succeed only if **every** outcome succeeds

A plan is a *sub-tree* guaranteeing the goal.

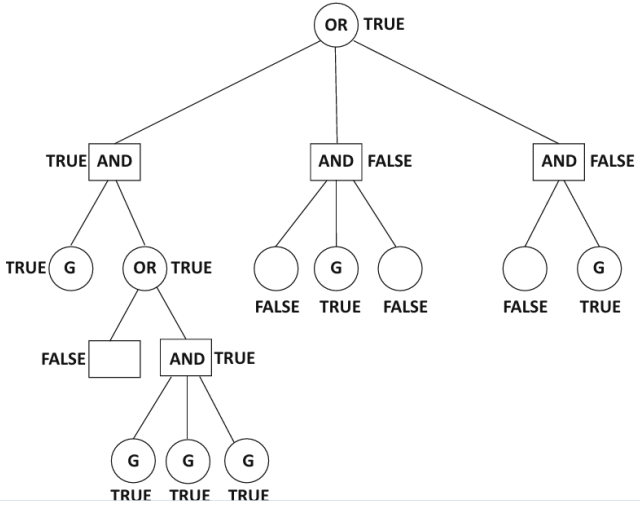
More than two agents

In a 3-agent game (A, B, C):

- OR nodes = agent A's choices
- AND nodes = combined choices of B and C

Combining actions blows up the branching factor.

The AND-OR Tree, Illustrated



Section 3

Evaluation & Depth-Limited Search

Evaluation Functions | "How promising is this state right now?"

Full trees are infeasible in chess or Go. An **evaluation function** estimates the quality of a **non-terminal** state.

Chess features

material advantage ▪ king safety ▪ center control ▪ piece activity

Powerful but imperfect

- Summarize rich positions with few features
- May miss deep tactical ideas
- Their quality drives decision quality

Terminology & Depth-Limited Search | Fix the vocabulary

Core terms

- **State / Action / Terminal state**
- **Utility**: larger is better for MAX
- **Loss / cost**: smaller is better
- **MAX / MIN**: the two adversarial roles

Depth-limited search

Cut off expansion at depth d, then apply the evaluation function at the frontier.

- Deeper search $\Rightarrow$ better decisions
- Too deep $\Rightarrow$ infeasible
- Good evaluation offsets shallow search

Strong engines combine **lookahead** with a good **evaluation function**.

Section 4

Minimax

For two-player, **zero-sum**, deterministic, perfect-information games.

- **MAX** maximizes the utility value
- **MIN** minimizes it
- Terminal states give the true payoff
- Others inherit the value of the best move for the player to act

Mutual recursion

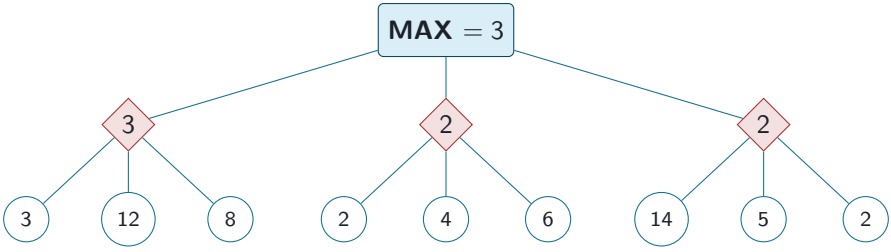
$$\texttt{max\_player(s)} = \max_a \texttt{min\_player(s')}$$

$$\texttt{min\_player(s)} = \min_a \texttt{max\_player(s')}$$

base case: depth $= 0$ or terminal $\Rightarrow$ `utility(s)`

A model of **perfect play** — the best *guaranteed* outcome.

Minimax Example | Read the tree from the leaves upward



MIN nodes: $\min(3,12,8)=3$, $\min(2,4,6)=2$, $\min(14,5,2)=2$ $\Rightarrow$ MAX: $\max(3,2,2) = \mathbf{3}$ (left branch).

Section 5

Alpha–Beta & MCTS

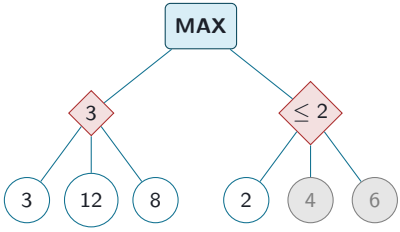
Alpha–Beta Pruning | Same answer as minimax, fewer nodes

Skip branches that cannot change the final choice.

- α — best value **MAX** can guarantee so far
- β — best value **MIN** can guarantee so far
- Prune the branch when $\alpha \geq \beta$

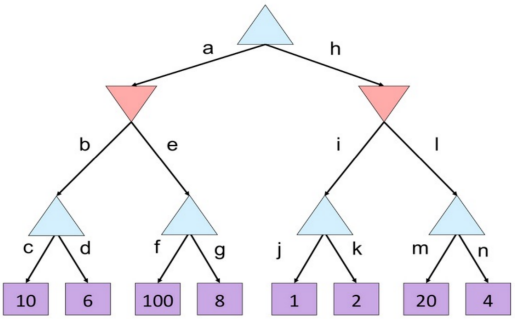
Move ordering matters

Examining strong moves early creates tighter bounds and prunes more.

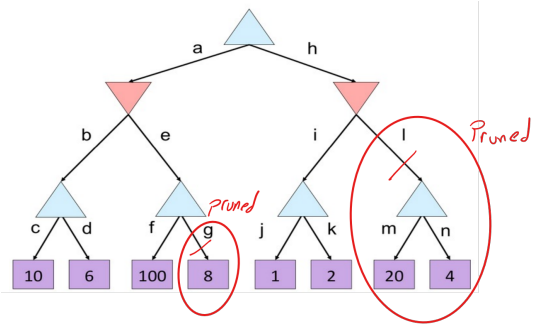


After leaf 2, the right MIN $\leq 2 < 3 = \alpha$ — prune the rest.

Alpha–Beta: Worked Example | Track α and β after each child



Where does pruning occur (left-to-right)?



Solution: a node already worse than an earlier alternative cannot improve the parent's decision.

Monte Carlo Tree Search (MCTS) | Estimate by simulation, not exhaustion

Ideal for **huge branching factors** or when a good evaluation function is hard to design.

Four phases

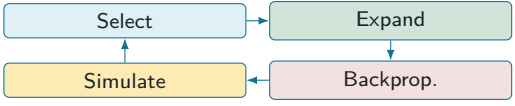
1. **Selection** — descend via UCT
2. **Expansion** — add a child at the frontier
3. **Simulation** — roll out to estimate quality
4. **Backpropagation** — update visits & rewards

Anytime: more simulations ⇒ better estimates.

UCT: exploit vs. explore

$$u_i = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$$

w_i reward, n_i visits to i, N_i parent visits, c exploration weight.



Section 6

Choosing a Method & Summary

Choosing the Right Search Method | No method dominates everywhere

Situation	**Method**	Reason
Contingent planning, uncertain outcomes	**AND-OR / AO-search**	Must cover multiple continuations
Two-player game, manageable depth	**Minimax**	Optimal under perfect play
Same setting, deeper trees	**Alpha–beta**	Minimax result, less search
Huge branching / weak heuristic	**MCTS**	Simulation focuses computation

Beyond board games: **autonomous driving** ▪ **cybersecurity** ▪ **robotics** ▪ **auctions**.

Chapter Summary | Key takeaways

Modeling

- Good decisions account for others' responses
- **AND-OR** trees capture contingencies
- Evaluation functions make big trees tractable

Algorithms

- **Minimax**: optimal in zero-sum games
- **Alpha–beta**: same answer, fewer nodes
- **MCTS**: simulation for huge trees

Looking ahead

Ch. 4: Propositional Logic and knowledge representation.

Thank you!

Questions & Discussion

Next: Chapter 4 — Propositional Logic