AI Week 6 Notes

Here is your formatted version of the chapter:

Week 6: Adversarial Searches & Game Playing

Dr. Ammar Masood
Department of Cyber Security, Air University Islamabad
CS340 - Artificial Intelligence

Table of Contents

Introduction to Adversarial Searches

In many real-world problems, we often encounter environments where multiple agents or players make decisions that influence each other's outcomes. Adversarial search is a specialized form of search used in such competitive scenarios, particularly in two-player, zero-sum games like chess, tic-tac-toe, and checkers.

What is Adversarial Search?

Adversarial search is a decision-making process used in competitive environments where an agent must account for the actions of an opponent. Unlike traditional search problems (such as pathfinding), adversarial search involves two or more players who have conflicting goals.

Example: Chess

- You try to win the game by maximizing your position.
- Your opponent tries to do the same, minimizing your advantage.
- Thus, adversarial search requires strategies that anticipate and counteract the opponent's best possible moves.

Characteristics of Adversarial Searches

- **Two or More Agents:** Typically involves two players (MAX and MIN) competing.
- **Zero-Sum Property:** A gain for one player is a loss for the other.
- **Turn-Based or Simultaneous Moves:** Players alternate moves or act simultaneously.

Perfect vs. Imperfect Information

- **Perfect Information:** Players know the entire game state (e.g., chess).
- **Imperfect Information:** Some information is hidden (e.g., poker).

Deterministic vs. Stochastic

- **Deterministic:** No randomness (e.g., tic-tac-toe, chess).
- **Stochastic:** Random events influence the game (e.g., rolling dice in backgammon).

Games in AI

- **Multiagent Environment**
- **Cooperative vs. Competitive**

- **Competitive Environment:** Agents’ goals are in conflict.
- **Adversarial Search & Game Theory:**
  - A branch of economics that views the impact of agents on others as significant rather than competitive or cooperative.

Why Games?

- **Small, defined set of rules.**
- **Well-defined knowledge set.**
- **Easy to evaluate performance.**
- **Large search spaces (e.g., chess has a vast number of possible moves).**

Games as Search Problems

- **Each potential board or game position is a state.**
- **Each possible move is an operation to another state.**
- **The state space is huge!**
  - **Example: Chess**
    - Branching factor ≈ 35
    - Terminal state depth ≈ 50

Games vs. Traditional Search Problems

- **Unpredictable opponent:** Unlike normal searches, adversarial search has an opponent trying to counteract the moves.
- **Solution is a strategy:** The goal is not just to find a path but a plan that accounts for all possible opponent moves.
- **Time Limits:** Players must make decisions within a limited time, often requiring approximation.

Types of Games

	Deterministic	Chance-Based
Perfect Information	Chess, Checkers, Go, Othello	Backgammon, Monopoly
Imperfect Information	Bridge, Poker, Scrabble	-

AI Defeating Humans in Games Over Time

- **1997 – Deep Blue (Chess)**: IBM's Deep Blue defeats World Chess Champion Garry Kasparov.
- **2011 – IBM Watson (Jeopardy!)**: Watson beats Jeopardy! champions Ken Jennings and Brad Rutter.
- **2016 – AlphaGo (Go)**: DeepMind's AlphaGo defeats Go grandmaster Lee Sedol.
- **2017 – AlphaZero (Chess, Shogi, Go)**: AlphaZero trains itself and beats top chess engines.
- **2019 – OpenAI Five (Dota 2)**: OpenAI's team defeats professional Dota 2 players.
- **2022 – CICERO (Diplomacy)**: Meta's CICERO masters human negotiation in Diplomacy.

Two-Player Zero-Sum Games

- **Deterministic, two-player, turn-taking, perfect information, zero-sum games** (e.g., Chess, Go).
- **"Perfect information"** means fully observable.
- **"Zero-sum"** means what benefits one player hurts the other equally.

Optimal Decisions in Games

- **Initial State:** Board position & player to move.
- **Successor Function:** Returns legal moves and resulting states.
- **Terminal Test:** Checks if the game is over.
- **Utility Function:** Assigns numeric values to terminal states (+1, -1).

Game Trees

- **Root Node:** Initial state.
- **Branches:** Possible moves.
- **Depth:** Turns taken until a terminal state is reached.

Algorithms in Adversarial Search

Minimax Algorithm

- **A decision rule for minimizing the possible loss in a worst-case scenario.**
- **Assumes both players play optimally.**
- **Used in chess engines, tic-tac-toe solvers, etc.**

Minimax Search Strategy

- **MAX wants to find a winning sequence, but MIN counteracts each move.**
- **Optimal strategy found by evaluating the entire game tree.**
- **Computationally expensive, requiring efficient pruning techniques.**

Minimax Algorithm Properties

- **Complete?** Yes (if the tree is finite).
- **Optimal?** Yes (against an optimal opponent).
- **Time Complexity:** O(b^m)

- **Space Complexity:** O(bm) (depth-first search).

Alpha-Beta Pruning

- **Optimized version of Minimax.**
- **Prunes unnecessary branches to improve efficiency.**
- **Does not affect final decision but speeds up computation.**

Key Concepts

- **Alpha (α):** Best choice so far for MAX (highest value).
- **Beta (β):** Best choice so far for MIN (lowest value).
- **If a position is worse than a previous move, stop evaluating that branch.**

Alpha-Beta Pruning Benefits

- **Reduces the number of nodes evaluated.**
- **With perfect move ordering, reduces complexity to O(b^(m/2)), effectively doubling search depth.**
- **Example: Chess search can reach a depth of 80 instead of 40.**

Optimizing Minimax Search

- **Use Alpha-Beta Pruning.**
- **Evaluate most promising moves first.**
- **Avoid redundant paths using transposition tables.**

Cutting Off Search: Killer Moves

- **Replace terminal test with a cutoff test (stopping search early).**
- **Replace utility function with a heuristic evaluation function.**

This formatted version makes the content easier to read and review. Let me know if you need any further modifications! 🚀

2/2