Phase II Report: Adversarial Search on Connect-4

CSAI 301 – Fall 2025

Student Name: Omar Hazem Ahmed

1. Introduction

This report presents the design, modeling, and experimental evaluation of adversarial search algorithms applied to the game **Connect-4**. The objective of Phase II is to demonstrate the use of **Minimax** and **Alpha-Beta Pruning** in a competitive two-player environment and to compare their performance using node expansion and computation time metrics.

Connect-4 is a deterministic, perfect-information, zero-sum game, making it ideal for classical adversarial search.

2. Game Modeling

2.1 State Representation

The Connect-4 board is represented as a 6×7 matrix:

$$state \in \{-1, 0, +1\}^{6\times 7}$$

Where:

$$+1 = \text{Max (AI)}, \quad -1 = \text{Min (opponent)}, \quad 0 = \text{empty}$$

2.2 Initial State

An empty board with all values equal to 0.

2.3 Actions

A valid action is selecting a column c such that:

$$0 \le c < 7 \quad \text{and the column is not full}$$

2.4 Transition Model

A move inserts a piece into the lowest empty row of column c:

$$T(s, c, p) = \text{updated state after player } p \text{ plays column } c$$

2.5 Terminal Test

A state is terminal if:

- A player forms a horizontal, vertical, or diagonal four-in-a-row.
- The board is full (draw).

We define the terminal test as:

$$terminal(s) = (winner(s) \neq None) \vee isFull(s)$$

2.6 Utility Function

For terminal states:

$$U(s) = \begin{cases} +\infty & \text{if Max wins} \\ -\infty & \text{if Min wins} \\ 0 & \text{draw} \end{cases}$$

3. Evaluation Function

Because depth-limited search is used, we define a heuristic function based on 4-cell windows.

For each window W:

$$score(W) = \begin{cases} 1000 & \text{if Max has 4} \\ 10 & \text{if Max has 3 + empty} \\ 5 & \text{if Max has 2 + 2 empty} \\ -8 & \text{if Min has 3 + empty} \\ 0 & \text{otherwise} \end{cases}$$

Total evaluation:

$$h(s) = \sum_{W \in windows} score(W) + 3 \times (\text{center column advantage})$$

Center column weight improves strategic play.

4. Minimax Algorithm

The depth-limited Minimax function is defined as:

$$Minimax(s, d) = \begin{cases} h(s) & \text{if } d = 0 \text{ or } s \text{ is terminal} \\ \max\limits_{a \in Actions(s)} Minimax(T(s, a), d - 1) & \text{if Max} \\ \min\limits_{a \in Actions(s)} Minimax(T(s, a), d - 1) & \text{if Min} \end{cases}$$

Minimax guarantees optimal play but explores the entire space of the search tree at the given depth.

5. Alpha-Beta Pruning

Alpha-Beta is an optimized version of Minimax:

$$\alpha = \text{best value found for Max}$$

$$\beta = \text{best value found for Min}$$

If at any time:

$$\alpha \geq \beta$$

We prune the remaining branches.

Alpha-Beta returns the exact same optimal decision as Minimax but expands far fewer nodes.

6. Experimental Setup

We performed two categories of experiments:

1. **Single Board Evaluation** – Both algorithms evaluate the same mid-game state.
2. **Full Game Simulation** – AI plays against a Random Bot.

Depth was set to:

$$d = 4$$

Metrics collected:

- Nodes Expanded
- Execution Time (seconds)
- Best Move Selected

7. Results

7.1 Single Board Evaluation

Algorithm	Nodes Expanded	Time (s)	Best Move
Minimax	18,935	3.6296	3
Alpha-Beta	2,024	0.3624	3

Observation: Alpha-Beta is approximately **10 times faster** while producing the same move.

7.2 Full Game: Minimax vs Random

Metric	Value
Total Nodes Expanded	18,331
Total Time (s)	3.6629
Winner	Max (AI)

7.3 Full Game: Alpha-Beta vs Random

Metric	Value
Total Nodes Expanded	2,618
Total Time (s)	0.4684
Winner	Max (AI)

Alpha-Beta required:

$$\approx 86\% \text{ fewer nodes}$$

8. Discussion

The experiments confirm several theoretical properties of adversarial search:

- Minimax explores the full search tree, leading to high computational cost.
- Alpha-Beta pruning drastically reduces the number of explored nodes without affecting optimality.
- The evaluation function significantly impacts performance and decision quality.
- Against a Random Bot, both algorithms achieve a 100% win rate.

Alpha-Beta's pruning effectiveness increases as the game progresses and fewer moves remain.

9. Conclusion

Phase II successfully demonstrates adversarial search using Minimax and Alpha-Beta pruning. Key findings:

- Alpha-Beta pruning is dramatically more efficient.
- Both algorithms return identical optimal moves.
- The heuristic evaluation function is effective and crucial.
- Experimental data strongly matches theoretical expectations.

This completes the Phase II requirements of adversarial search modeling, implementation, experimentation, and evaluation.