UNIVERSITY OF MINNESOTA
Department of Computer Science & Engineering

CSCI 5511 **Introduction to Artificial Intelligence** **Fall 2025**

Adversarial Search: RandOthello

Apurv Kushwaha
kushw022@umn.edu
October 24, 2025

Overview

This report describes my implementation of the **RandOthello Adversarial Search** assignment. The goal was to design agents that play Othello on an 8×8 board with two randomly blocked cells. Four agents were implemented:

1. **RandomPlayer** – baseline that chooses random legal moves.
2. **MinimaxPlayer** – searches to a fixed depth using the minimax algorithm.
3. **AlphaBetaPlayer** – uses alpha–beta pruning with iterative deepening for efficiency.
4. **AdvancedPlayer** – combines alpha–beta search with improved move ordering and caching.

Evaluation Function

A single heuristic function was used across all agents:

$$Eval(s) = W_{pos} + 3(M_{me} - M_{opp}) - 2(F_{me} - F_{opp})$$

where:

- W_{pos} is the sum of positional weights (corners +20, edges small penalty, near corners negative),
- M_{me}, M_{opp} are the number of legal moves (mobility),
- F_{me}, F_{opp} are the number of frontier discs (pieces adjacent to empty squares).

This evaluation prefers stable discs, good mobility, and discourages risky frontier positions. The co-efficients (+3 and -2) were chosen by testing and observation. Mobility (number of legal moves) usually changes a lot each turn, while frontier discs change more slowly. Terminal states return large ± 100000 scores for win/loss.

Algorithmic Improvements

Alpha–Beta Pruning

The AlphaBeta Player implements the standard minimax algorithm with α–β bounds to prune branches that cannot influence the final decision. Additionally, iterative deepening allows the player to reuse results from shallower searches until a time limit of 1.5 seconds is reached.

Transposition Table (Caching)

A global dictionary stores previously evaluated boards using a key composed of board configuration, depth, and player color:

$$key = (color, depth, flatten(board))$$

This caching avoids recomputing identical subtrees, accelerating midgame searches.

Improved Move Ordering (Advanced Player)

The **AdvancedPlayer** enhances alpha–beta by reordering legal moves:

1. Corner moves are considered first (most valuable positions).
2. Remaining moves are sorted by the opponent's next mobility:fewer opponent options are explored first.

This ordering increases pruning efficiency and generally produces stronger play without changing the underlying algorithm.

Command and Execution

The game script automatically runs two matches as required:

```
# Automatic matches
python3 randothellogame .py

# Runs:
# 1. AlphaBeta (Black) vs Random (White)
# 2. Random (Black) vs AlphaBeta (White)
```

Each match prints the final board, counts of black and white discs, and declares the winner.

Results

Across multiple runs, Alpha–Beta and Advanced players consistently outperformed the Random baseline. Typical outcomes were around:

AlphaBeta/Advanced wins in 90–95% of games

with endgame differences of 15–25 discs. Both advanced search agents played strong corner-oriented strategies and avoided risky frontier placements.

Match	**Winner**	**Final Score (B–W)**
AlphaBeta (B) vs Random (W)	Black	41–21
Random (B) vs AlphaBeta (W)	White	18–44

Table 1: Representative game outcomes showing Alpha–Beta dominance.

Discussion

The transition from Minimax to Alpha–Beta pruning reduced computation time dramatically while yielding identical results. Introducing the transposition table and improved move ordering further reduced search depth by eliminating redundant evaluations and increasing pruning effectiveness. The weighted heuristic encouraged stable and corner-based play, resulting in consistently strong positions against Random opponents.

Conclusion

This project demonstrated the practical power of adversarial search algorithms. Starting from a simple random baseline, each incremental improvement (heuristic evaluation, alpha–beta pruning, iterative deepening, caching, and move ordering) made the agent faster and more strategic. The final **AdvancedPlayer** exhibits efficient search behavior and realistic Othello strategies, successfully integrating all discussed techniques.