Game-Playing Agents for Othello

Changdao (Michael) He
Supervised by Professor Alice Gao
Department of Computer Science
UNIVERSITY OF TORONTO

Abstract

This project designs and implements intelligent agents for the board game Othello using two AI techniques:
- **Alpha-Beta Pruning (ABP) with heuristic function**
- **Monte-Carlo Tree Search (MCTS)**

Implemented in Python within a modular engine, the agents are tested against a random move agent and each other. Results highlight the trade-offs between deterministic search and simulation-based search in decision quality and strategic performance.

Othello

Othello is a two-player zero-sum game played on an **8×8 board** using **64 pieces** that are black on one side and white on the other. The goal is to finish the game with more pieces of your color than your opponent. The game begins with four pieces placed in the center (Figure 1).

Rules:
- **Black moves first.**
- Players place one piece per move. If any **legal move** exists, the player must make it; otherwise, they pass.
- A **legal move** must sandwich at least one opponent piece in a straight line (horizontal, vertical, or diagonal). Sandwiched opponent pieces are flipped to the player's color.
- The game ends when the **board is full** or when there is **no legal move** for either player. The player with the most pieces at the end wins; ties result in a draw.[4]

Othello is ideal for AI research due to its simple rules but deep strategic complexity.

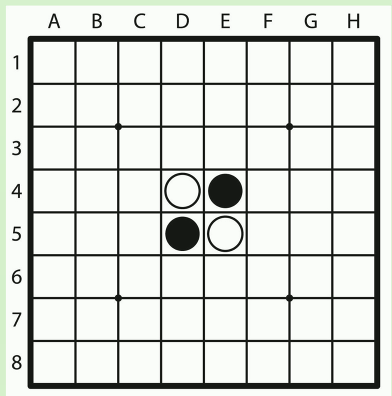

Figure 1
Initial Board

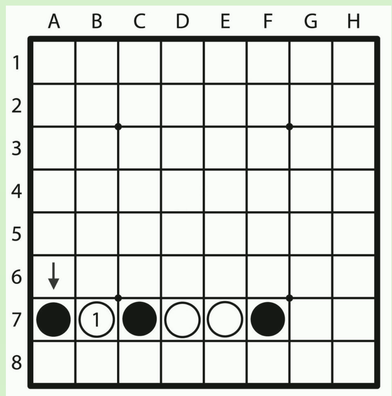

Figure 2
A black piece is placed on A7, it sandwiches and flips white piece 1 at B7 ONLY.

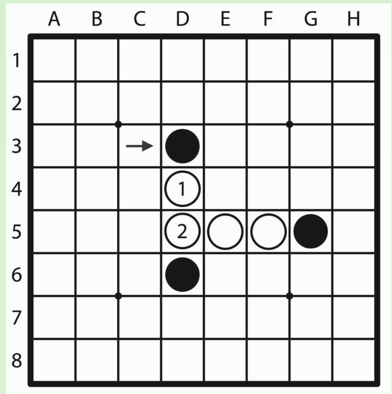

Figure 3
A black piece is placed on D3, white pieces 1, 2 at D4, D5 are flipped ONLY. Piece(s) may only be flipped as a direct result of a move.

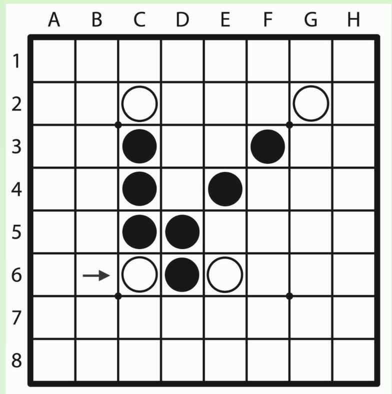

Figure 4
A white piece is placed on C6, and all black pieces are flipped.

Agents

Alpha-Beta Pruning (ABP) Agent
- Uses the **minimax search algorithm** with alpha-beta pruning to reduce the number of game states explored.
- Evaluates legal moves up to a fixed depth limit and chooses the move that maximizes the heuristic value.[2]

Heuristic Functions used for evaluation:
- **Parity Heuristic:** Difference in pieces owned
- **Mobility Heuristic:** Difference in legal moves
- **Corner Heuristic:** Difference in corners held
- **Stability Heuristic:** Difference in stability values
  - A piece is stable (+1) if it cannot ever be flipped.
  - A piece is unstable (-1) if it can be flipped immediately.
  - A piece is semi-stable (+0) otherwise.[3]

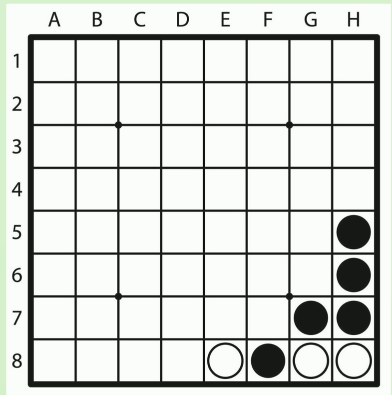

Figure 5

In Figure 5, for Black:
- Parity:

$$5 - 3 = 2$$

- Mobility:

$$1 - 3 = -2$$

- Corner:

$$0 - 1 = -1$$

- Stability :

$$(-1 \times 4 + 0) - (1 \times 2 - 1) = -5$$

Monte-Carlo Tree Search (MCTS) Agent
- Selects moves through numerous randomized simulations rather than exhaustive search.
- Builds a search tree rooted at the current board state, creating child nodes for each legal move. Then repeatedly performs:
  - **Selection:** Traverse from the root, selecting child nodes using the Upper Confidence Bound 1 (UCB1) formula, which balances exploration vs. exploitation.
  - **Expansion:** Add a new child node when a node with unvisited moves is reached.
  - **Simulation:** Play out a full game from the new node, either randomly or via a policy.
  - **Backpropagation:** Send results back up the tree, updating visits and win stats.[1]
- After all simulations, selects the move from the root's children with the highest UCB1 value.

Results

Overall, Stability proved the strongest heuristic in deterministic ABP play, while MCTS was most effective against Mobility but struggled against Corner and Stability.

Both agents were evaluated against a random move agent and showed **clear advantage** over the baseline (Figure 6).

Agent	Color	Opponent	Depth/Time Limit	Wins	Ties	Losses	Win Rate
ABP \w Parity	Black	Random	depth 5	93	1	6	93.5%
	White			87	1	12	87.5%
ABP \w Mobility	Black			87	0	13	87%
	White			86	1	13	86.5%
ABP \w Corner	Black			98	0	2	98%
	White			98	0	2	98%
ABP \w Stability	Black			99	0	1	99%
	White			98	0	2	98%
MCTS	Black		10 sec	84	5	11	86.5%
	White			89	2	9	90%

Figure 6

Each pair of ABP agents with different heuristics played two deterministic games (Figure 7). Results show a clear ranking: **Stability > Corner > Parity > Mobility**.
- Stability dominates all others.
- Corner outperforms Parity.
- Parity outperforms Mobility.
- Mobility beats Corner as Black and draws as White.

Agent	Parity	Mobility	Corner	Stability
Parity		43 - 21	30 - 34	3 - 60
Mobility	6 - 58		43 - 20	23 - 41
Corner	42 - 22	32 - 32		28 - 36
Stability	64 - 0	51 - 13	62 - 2	

Figure 7

For each heuristic function, 100 games were played, with the MCTS agent taking 50 as Black and 50 as White against the ABP agent (Figure 8). The ABP agent used **iterative deepening** (20s per move, max depth 10), evaluating the deepest reachable state with its heuristic.
- MCTS achieved its best results against the **Mobility Heuristic**.
- Performance was weakest against **Corner Heuristic**.

Agent	Color	Opponent	Time Limit	Depth Limit	Wins	Ties	Losses	Win Rate
MCTS	Black	ABP \w Parity	20 sec	depth 10	12	1	37	25%
	White				10	1	39	21%
	Black	ABP \w Mobility			23	3	24	49%
	White				22	4	24	48%
	Black	ABP \w Corner			5	1	44	11%
	White				3	1	46	7%
	Black	ABP \w Stability			10	1	39	21%
	White				5	1	44	11%

Figure 8

References & Acknowledgments

1. Hayward, R. B. (2018, November 23). Monte Carlo Tree Search. University of Alberta. https://webdocs.cs.ualberta.ca/~hayward/396/jem/mcts.html
2. Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed., Chapter 5.3, p. 167). Prentice Hall.
3. Sannidhanam, V., & Annamalai, M. (2015). An analysis of heuristics in Othello.
4. World Othello Federation. (n.d.). Official Othello rules (English). https://www.worldothello.org/about/about-othello/othello-rules/official-rules/english

I would also like to thank Daniel Cheng, Max Djaya, Justin Li, and Ryan Ning for their valuable suggestions and feedback.