Karl WEHBE - 261085350

COMP424 - Final Project Report

I. Overall Motivation for Approach:

For playing Reversi Othello, I found that the strongest algorithm would be Alpha-Beta Pruning combined with Iterative Deepening and multiple heuristics to evaluate the board.

Alpha-Beta pruning is better suited for Reversi since it's deterministic, and scales well with heuristics. Reversi is also a deterministic and perfect information game, and α-β is a standard technique for these types of games which makes it the optimal one to use. In addition, the constraints present such as time limit and large board sizes make me lean more towards alpha-beta pruning.

I also implemented a Heuristic-Guided MCTS that influences which move is chosen for simulations, combining a heuristic (normalized) with UCT for the tree policy. However, with a 2-second limit, it struggled to run enough simulations to converge to optimal and meaningful decisions, especially on larger boards (10x10, 12x12). That, combined with the exponential growth of possible moves makes MCTS inaccurate especially in the early critical phases of the game.

I tested my Alpha-Beta Pruning agent against the MCTS agent and Alpha-Beta ended up winning most of the games further affirming my confidence in my choice of Alpha-Beta pruning.

To achieve strong performance, pruning was a necessity, and without it, the game tree would grow exponentially, making the search slower. Its efficiency depends strongly on move ordering and the heuristic, so I implemented a form of forward pruning with moves sorted based on their heuristic evaluation value. I had to come up with strong heuristics for the board evaluation in order to make efficient move ordering and maximize pruning as much as possible. The heuristic also helps me get the optimal play out of the algorithm, it plays an important role in the outcome of the algorithm. So I made sure to implement the best one I could by constantly adding new heuristics based on misplays I saw when simulating, general strategy, and constant testing between an agent with the new heuristic against an agent with the old heuristic and seeing if it improved the game results and logic.

II. Agent Design

The design of the agent is based on basic Alpha-Beta Pruning, with Iterative Deepening and enhanced by heuristics for the board evaluation.

The agent uses iterative deepening to explore moves by increasing the search depth until a set time limit is reached, helping avoid going over the maximum 2 second mark. At each depth, it runs the minimax algorithm to identify the best move, balancing between move quality and time constraints. A transposition table/cache stores results of previously evaluated board states to avoid repeated calculations, making more time for the other more important aspects.

Reversi has a big branching factor because of its many possible moves. The size of the board also heavily impacts the branching factor since the more possible moves we get for a node, the bigger branching factor becomes. So pruning is a necessity, and without it, the game tree would grow exponentially. Alpha-beta Pruning reduces the number of nodes/boards evaluated, focusing on the search on fewer branches, speeding up the search significantly.

Pruning efficiency is heavily dependent on move ordering as this affects what the α-β values are, so a form of forward pruning was implemented for the moves to get sorted based on their heuristic evaluation value. While all moves are initially considered, sorting them and evaluating high valued moves first can lead to more pruning, reducing the number of states we need to evaluate by skipping moves which will significantly improve performance. The move ordering is computed for both player and opponent nodes.

Strong board evaluation heuristics are used in order to make efficient move ordering and maximize pruning as much as possible all while also getting the best move out of the algorithm. Without a strong evaluation, the algorithm will find itself wasting time exploring weak branches, leading to slower performance and worse end results. All heuristics work together to give a score to the board, with some having a more significant weight than others based on their importance to the outcome of the game. For example, the value of a corner is more important than the value of the centerpieces. These weights can also shift based on the stages of the game (opening, mid, end).

The heuristic implemented guides the algorithm to prioritize moves and play in a strategic way by focusing on tactics like mobility, corner control, center control, and stability, making sure the algorithm is focused on securing crucial positions, gaining mobility and avoiding risky play and dangerous positions. The several heuristics implemented have a heavy impact on the outcome, some are general and some are very specific and detailed to a certain state of the game. For example :

- Example : If a wall (row or column) has the form X_XXX_ where X represents a cell captured by the opponent, playing a move in an adjacent cell to the corner (XOXXX_) will allow the player to get the opposite corner and all cells in between. (XOOOOO).

Over many games, I found that small changes and additions to heuristics led to big impacts on win rate and outcome of a game.

Overall, this algorithm aims to minimize its focus to fewer moves by decreasing overall search effort, all in the goal of returning the best possible move at each step.

III. Quantitative Performance

A) Depth :

The depth varies based on multiple factors such as board size, game phase and remaining time. Since we are using iterative deepening, the depth increases progressively until the limited time is reached. The depth can be deeper for certain branches for promising branches thanks to move ordering and pruning.

- 6x6 boards: Start: 4-5 / Mid: 4-5 / End: 4-10
- 8x8 boards: Start: 4-5 / Mid: 3-4 / End: 5-8
- 10x10 boards: Start: 3-5 / Mid: 2-4 / End: 4-8
- 12x12 boards: Start: 3-5 / Mid: 2-4 / End: 3-8

Depth varies between board sizes especially because of the number of possible moves we can have on larger boards. Depth is impacted mainly during the mid game for different sizes.

A good heuristic can improve the lookahead depth since a well-designed heuristic can more confidently prune branches that are unlikely to lead to better outcomes without fully exploring them. This allows more time to explore and go deeper into other more promising branches, increasing the lookahead depth.

B) Breadth :

The breadth or branching factor is also based on multiple factors such as board size, game phase and number of possible moves at a certain state of the game.

That means, the bigger the board, the more possible moves we could make, the larger the breadth. In addition, at each state of the game, the number of possible moves varies. At the start, we usually have a small number of moves that could increase rapidly. In the mid game phase, the number of possible moves reaches its peak. Finally, at the end game, the number of moves starts decreasing and we get a narrower breadth.

By using alpha-beta pruning, we can reduce that breadth. Worst-case runtime is still O(b**d), but using pruning and a move ordering, we can decrease that exponent. On average, time complexity is O(b**(3m/4)), which means on average, we pruned ¼ of the breadth on average for each node. (taken from slides)

A good heuristic can also heavily influence the breadth because it directly impacts the amount of branches pruned from the tree, reducing the overall number of nodes the algorithm needs to explore at each depth, allowing it to focus time on more critical paths.

C) Impact of board sizes :

Board sizes have a significant impact on the depth and breadth of the tree. Larger boards increase the number of potential moves and total number of moves played, which increases both depth and breadth of the tree. Therefore, within the same time limit, larger boards tend to have a smaller look ahead depth but a larger breadth of moves to consider. Efficiency can be improved by using move ordering based on heuristic evaluation and on larger boards, it has a bigger impact, as it helps control the branching factor.

D) Heuristics and other approaches

The heuristic functions implemented for the board evaluation are inspired by game strategies and continuous trial and error to make sure they are useful and help properly.

Inspiration from :

> https://www.coolmathgames.com/blog/how-to-play-reversi-basics-and-best-strategies#:~:text=Oftentimes%20if%20you%20can%20get,three%20squares%20surrounding%20the%20corner

The evaluation is based on many different heuristics, with some being very precise to a situation and others more general. Each heuristic has a different weight on the final value. Some with the highest impact on win rate :

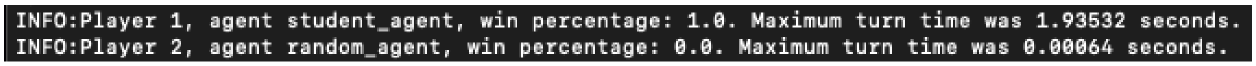
1. Corner Control : values corners taken by player and punishes corners taken by opponent
2. Inner Corners : values cell with potential of capturing a corner
3. Danger Zones : values cell if opponent controls them and punishes if player does
4. Wall for Corner : specific tactic that if found can guarantee a corner capture
5. Prevent Corner : specific tactic that values a board where an opponent won't be able to capture a corner and punishes boards where he could
6. Mobility & Potential Mobility : values boards that maximize the player's current possible moves and potential possible moves.

The heuristic helps the move ordering distinguish between high priority and low priority moves that were excluded from deeper search, reducing computational workload thanks to pruning and allowing the algorithm to focus on the highly evaluated boards.

Thus, a good heuristic can have a big impact on the amount pruned and on the game results. While it introduced a slight risk of missing optimal moves, the benefits outweigh the drawbacks, especially on larger boards.

E) Win Rate Prediction

The algorithm wins comfortably against the random agent, with multiple simulations returning a win rate of 100%.



```
INFO:Player 1, agent student_agent, win percentage: 1.0. Maximum turn time was 1.93532 seconds.
INFO:Player 2, agent random_agent, win percentage: 0.0. Maximum turn time was 0.00064 seconds.
```

Having played lots of games against my agent, I think that my level is close to the average human player, so if Dave has the same level he should expect to lose 85-95% of the time against it. Against my classmates' agent I hope to be in the top 5-20% or possibly win. However, having no information about how much effort others put into their project makes it hard to tell, however I am confident in my agent's abilities.

IV. Advantages and Disadvantages :

Reversi is a deterministic and perfect information game, and α-β is a standard technique for these types of games which makes it the optimal one to use. Having implemented both α-β pruning and MCTS, I found that Alpha-Beta pruning performed better in my case.

Given the constraints, particularly the 2-second limit and the challenges posed by larger boards, Alpha-Beta pruning allowed for more accurate decisions and better overall play compared to MCTS. While Alpha-Beta pruning has good precision, it's limited by small lookahead depths on larger boards. In contrast, MCTS can explore very deep searches but is limited by the time constraint which results in less reliable and accurate data.

Some potential failures and weaknesses of α-β pruning include:

- Possibly missing optimal play because of bad board evaluation and pruning.
- Algorithms reliant on a good heuristic especially for pruning.
- Incorrect fine-tuning of heuristic weights, leading to inaccurate assessments of board states.
- Incorrect use of heuristics that should only be used for certain boards sizes and not others.
- Missing strategies that reduce the efficiency of board evaluation.

V. Improvements :

Other than adding Machine Learning into the algorithm, some possible improvements could include :

1. Improved board evaluation by adding more sophisticated heuristics. This requires a deeper understanding of the game and its strategies. It could help create more pruning and thus get more look ahead depth.
2. Enhanced fine tuning for heuristics by optimizing the weights of existing heuristics to achieve more accurate scores.
3. Implementing an improved more detailed game phase specific board evaluation/heuristics.

4. Pattern matching to identify board configurations and play the optimal moves.
5. Getting an understanding of different openings and their counters similarly to chess.
6. Adaptive heuristic to the style of the opponent by creating heuristics that adjust/change based on the style of play of the opponent.
7. Implementing killer moves in the move ordering which can significantly improve pruning.