Report Lab B

Charles Bowen-Rayner, Vassily Lombard

March 2024

1 Introduction

Breakthrough is a strategic board game that challenges players to beat their opponents by strategically advancing their pieces across the board. With each move, players must carefully consider the potential consequences and anticipate their opponent's actions to gain an advantage. In this project, our objective was to develop an agent able to play Breakthrough at a decent level. We aimed to create an AI player that could analyze the game environment, explore possible moves, and make strategic decisions to maximize its chances of success. With the implementation of the MiniMax algorithm and different utility functions such as Evasise and Conqueror, we were able to create an agent that navigates the complexities of Breakthrough game-play. Finally, our goal is to compare the efficiency of each utility function by making agents with different utility play against each other.

2 Models and Methods

For our implementation, we represented the Breakthrough game environment using a 2D array, where 'O' represents white pieces, 'X' represents black pieces, and empty spaces are dots ('.'). This representation allowed us to track piece positions and available moves.

Both agents employed the Minimax algorithm implemented with alpha-beta pruning to determine the best move available. The Minimax algorithm is an algorithm used in two-player games, aiming to minimize potential loss while maximizing potential gain. Alpha-beta pruning is an optimization technique that reduces the number of nodes evaluated by the Minimax algorithm, significantly improving computational efficiency.

In addition to the algorithm, we designed four distinct utility functions to evaluate how desirable a board state is:

- Evasive: Maximize the number of pieces on the board. It does not specifically look for eliminating opponent pieces. It is therefore evasive in the sense that it only cares about the number of piece the player has on the board.

- Conqueror: Minimize the number of opponent pieces on the board. As its name suggests it, conqueror's goal is to eliminate as many opponent pieces.
- Aggressor: Focuses on aggressive tactics. Emphasizing on capturing opponent pieces.
- Defensive: Prioritizes defensive moves to avoid capture and keep as many pieces.

Each utility function assigns a score to potential moves based on the current game state, allowing the AI to assess the strategic value of each action and select the most advantageous one. This approach ensures adaptability to diverse game scenarios, enhancing the overall gameplay experience.

3 Results

In this section we discuss the performance of each utility function. We examine final board states of different games where the AI use different utility functions. Additionally, we evaluate the success rates of our utility functions when they are matched against Evasive and Conqueror strategies. The images and the table are included below, pages 2-6.

After analyzing the results, we can conclude that Defensive and Aggressor are generally the most effective utility functions, regardless of the board size. They demonstrate strong performance against Conqueror, with an average winning rate exceeding 80 percent. However, their performance against Evasive is slightly lower, with a winning rate below 80 percent. Nonetheless, Defensive and Aggressor remain powerful AI agents capable of defeating players new to the game. Finally, when Defensive is pitted against Aggressor, we observed that Defensive emerged victorious most of the time, establishing it as the champion utility function in this project.

However, despite Defensive's dominance, it may have some weaknesses. It only considers the difference between the player's and opponent's number of pieces and does not take into account other aspects of the game, such as controlling the center of the board, which could provide a significant advantage.

	Evasive	Conqueror
Defensive	46%	40%
Aggressor	100%	100%

Table 1: Winning rate of each utility function when pitted against each other. (Approximation made over 50 games each)

```
.X....
X.XXXX
X...XX
O.OOOX
OO....
..X.OO
```

Figure 1: Evasive vs Evasive. Winner: X, Moves: 30, Pieces remaining: X=11 O=8

```
X.O...
XXX.X.
X.X..X
O.O..O
OOO..O
O.....
```

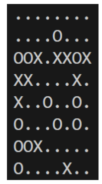
Figure 2: Evasive vs Evasive. Winner: O, Moves: 31, Pieces remaining: X=8 O=9

```
.X......
XX.X...X
.....XX.
........
.X....XX
...O.O..
......OO
.XO...O.
```

Figure 3: Conqueror vs Evasive. Winner: X (Evasive), Moves: 76, Pieces remaining: X=10 O=6

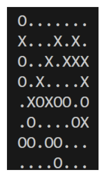
```
XXX...XX
X...XX.X
..O.....
........
X..OO...
..OX...O
.O....O.
O..O..X.
```

Figure 4: Conqueror vs Evasive. Winner: X (Evasive), Moves: 58, Pieces remaining: X=12 O=9



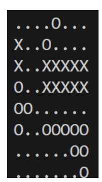
```
........
....O...
OOX.XXOX
XX....X.
X..O..O.
O...O.O.
OOX.....
O....X..
```

Figure 5: Defensive vs Evasive. Winner: X (Evasive), Moves: 84, Pieces remaining: X=10 O=10



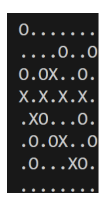
```
O.......
X...X.X.
O..X.XXX
O.X....X
.XOXOO.O
.O....OX
OO.OO...
....O...
```

Figure 6: Defensive vs Evasive. Winner: O (Defensive), Moves: 71, Pieces remaining: X=12 O=14



```
....O...
X..O....
X..XXXXX
O..XXXXX
OO......
O..OOOOO
......OO
.......O
```

Figure 7: Aggressor vs Evasive. Winner: O (Aggressor), Moves: 71, Pieces remaining: X=12 O=13



```
O.......
....O..O
O.OX..O.
X.X.X.X.
.XO...O.
.O.OX..O
.O...XO.
........
```

Figure 8: Aggressor vs Evasive. Winner: O (Aggressor), Moves: 95, Pieces remaining: X=8 O=13

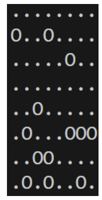


Figure 9: Defensive vs Conqueror. Winner: O (Defensive), Moves: 73, Pieces remaining: X=0 O=13

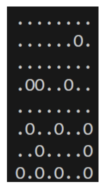


Figure 10: Defensive vs Conqueror. Winner: O (Defensive), Moves: 69, Pieces remaining: X=0 O=12

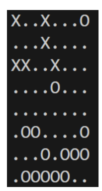


Figure 11: Aggressor vs Conqueror. Winner: O (Aggressor), Moves: 47, Pieces remaining: X=11 O=6

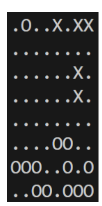


Figure 12: Aggressor vs Conqueror. Winner: O (Aggressor), Moves: 47, Pieces remaining: X=5 O=13

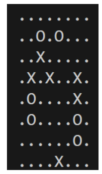


Figure 13: Defensive vs Aggressor. Winner: X (Defensive), Moves: 94, Pieces remaining: X=6 O=6

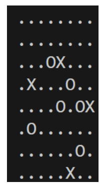


Figure 14: Defensive vs Aggressor. Winner: X (Defensive), Moves: 94, Pieces remaining: X=4 O=6

4 Conclusion

In summary, our project focused on creating an AI for Breakthrough, using the MiniMax algorithm alongside utility functions such as Evasive, Defensive, Aggressor, and Conqueror. Through comprehensive testing, we found that Defensive and Aggressor demonstrated superior performance compared to Evasive and Conqueror. Notably, Defensive emerged as the champion utility function, consistently outperforming Aggressor in our experiments.

However, it's essential to acknowledge that Defensive's success may be limited by its narrow focus on the difference between the player's and opponent's pieces, potentially overlooking strategic elements like controlling the center of the board.

Further steps may involve identifying alternative utility functions capable of outperforming human players in Breakthrough.