**Project Report**

**CS6364: Artificial Intelligence**

Shreeprasad Sonar

NetID: SXS210482

Introduction

This project intends to build a computer program that plays variant of Nine Mens Morris game. Python programming language is utilized for implementation. The first step is to create two programs that can compute the optimum move for white in both the opening and midgame/endgame phases of the game. The second half involves the creation of two programs that use the Alpha-Beta pruning algorithm instead of Minimax, but produce the same estimate values as in Part I. The third step necessitates the use of two programs that compute Black's move rather than White's move. Ultimately, the fourth phase demands the development of an improved static estimate function to replace the original static estimation function.

Part I: MINIMAX

The objective of Part I of the project was to implement the MINIMAX algorithm to determine the best move for the game of Morris-Variant, in both the opening and midgame/endgame phases. The algorithm had to be implemented in two programs, namely MiniMaxOpening and MiniMaxGame, and the programs had to take as input two file names for input and output board positions, and the depth of the tree that needs to be searched.

Program 1: MiniMaxOpening

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWBWBWBxBWWWWxxxBWBxx  Positions evaluated by static estimation: 282  MINIMAX estimate: 1 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WxWWWBWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 68  MINIMAX estimate: 1 |
|  |  |
|  |  |

Program 2: MiniMaxGame

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWxWBWBxBWxWWxxxBxBBW  Positions evaluated by static estimation: 181  MINIMAX estimate: 992 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WBWWxxWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 121  MINIMAX estimate: 992 |
|  |  |
|  |  |

Part II: Alpha-Beta

In this part, we are required to implement the ALPHA-BETA pruning algorithm instead of the MINIMAX algorithm used in Part I. The goal is to write two programs that behave exactly the same as the programs of Part I but return the exact same estimate values while evaluating fewer nodes.

Program 1: ABOpening

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWBWBWBxBWWWWxxxBWBxx  Positions evaluated by static estimation: 94  MINIMAX estimate: 1 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WxWWWBWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 28  MINIMAX estimate: 1 |
|  |  |
|  |  |

Program 2: ABGame

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWxWBWBxBWxWWxxxBxBBW  Positions evaluated by static estimation: 108  MINIMAX estimate: 992 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WBWWxxWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 64  MINIMAX estimate: 992 |
|  |  |
|  |  |

In both the cases, alpha-beta is result for board position and estimate is same while evaluating less positions than MiniMax.

Part III: Play A Game For Black

In this part of the project, we were tasked with writing two programs, MiniMaxOpeningBlack and MiniMaxGameBlack, that are similar to the ones written in Part I, but with the difference that the computed move should be Black’s move instead of White’s move.

Program 1: MiniMaxOpeningBlack

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWBWBWBxBWxxWBxxBWBBx  Positions evaluated by static estimation: 282  MINIMAX estimate: 1 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WBWBWBWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 57  MINIMAX estimate: 0 |
|  |  |
|  |  |

Program 2: MiniMaxGameBlack

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWBWBWBxBWxWWxxxBWBxB  Positions evaluated by static estimation: 163  MINIMAX estimate: -1013 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WBWxWBWBxBxXWWxxBBWBWB  Positions evaluated by static estimation: 87  MINIMAX estimate: -1019 |
|  |  |
|  |  |

Part IV: Static Estimation

In this part of the project, we were asked to implement an improved static estimation function for the Morris-Variant game. The goal of this task is to create a more accurate estimation of the game state than the one provided in the handout. This new function will be used in the programs of Part I, replacing the old estimation function.

Program 1: MiniMaxOpeningImproved

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWBWBWBxBWxWWxxxxWBBW  Positions evaluated by static estimation: 282  MINIMAX estimate: 58 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WxWWWBWxxBBXWWxxBBWBWB  Positions evaluated by static estimation: 68  MINIMAX estimate: 48 |
|  |  |
|  |  |

Program 2: MiniMaxGameImproved

|  |  |
| --- | --- |
| **Test Case 1** | **Test Case 2** |
| Given Board : WBWBWBWBxBWxWWxxxBWBBx  Given Depth : 2  Board Position: WBWxWBWBxBWxWWxxxBxBBW  Positions evaluated by static estimation: 181  MINIMAX estimate: 58 | Given Board : WBWxWBWxxBBXWWxxBBWBWB  Given Depth : 2  Board Position: WBWWxBWxxBBXWWxxBBWBWx  Positions evaluated by static estimation: 121  MINIMAX estimate: 68 |
|  |  |
|  |  |

Here change in static estimation function is reflected on outputs. Outputs produced are different from Part I, while both the functions end up making a mill, black piece removed in improved version is more carefully analyzed so that there is possibility of white forming a mill in the next turn.

Here estimation function is more sophisticated than the one provided in the handout. By taking into account the different potential mills that can be formed on the board, the current mill count difference, and the number of white pieces minus black pieces, function is able to capture more nuances of the game state and provide a more accurate estimate of the relative strength of the two players.

By considering the potential mills on the board, the function is able to take into account the fact that some positions may be more advantageous than others, depending on how they interact with the rest of the board. By considering the current mill count difference, the function can take into account the fact that some positions may be closer to victory than others, and therefore more valuable. Finally, by considering the number of white pieces minus black pieces, function can consider the fact that some positions may be more imbalanced than others, and therefore more vulnerable to attack.