AI Project 2 – Nine Men’s Morris AI

Justin Aegtema

July 2024

Table of Contents

[ReadMe 1](#_Toc171769050)

[Description of Attached Files 1](#_Toc171769051)

[MiniMax vs Alpha-Beta: 2 Savings Examples for both Opening and Midgame/Endgame 1](#_Toc171769052)

[Improved Static Estimation Function: 2 Examples for Both Opening and Midgame/Endgame 3](#_Toc171769053)

[Short Explanation of Why My Static Estimation Function is an Improvement 5](#_Toc171769054)

[Examples of MiniMaxOpeningBlack, MiniMaxGameBlack 6](#_Toc171769055)

[Explanation of ABOpenImpWhite, ABOpenImpBlack, ABGameImpWhite, ABGameImpBlack 6](#_Toc171769056)

[Conclusion 6](#_Toc171769057)

# ReadMe

To run any of the programs, simply type this into the command line:

python ProgramName.py board1.txt board2.txt 2

The program, all supplementary programs, and board1.txt must be in the same, current folder. Please reach out to me if you have any issue.

# Description of Attached Files

*1. Deliverables required for the assignment:* MiniMaxOpening.py, MiniMaxGame.py, ABOpening.py, ABGame.py, MiniMaxOpeningBlack.py, MiniMaxGameBlack.py, MiniMaxOpeningImproved.py, MiniMaxGameImproved.py

*2. Essential files that support the required deliverables:* building\_blocks.py, improved\_static\_est\_parts.py

*3. Non-required files* included because I improved upon assignment requirements and thought “why not include them?” : ABOpenImpWhite.py, ABOpenImpBlack.py, ABGameImpWhite.py, ABGameImpBlack.py

*4. Files that are essential to support non-required files:* ABOpeningImproved.py, ABGameImproved.py

# MiniMax vs Alpha-Beta: 2 Savings Examples for both Opening and Midgame/Endgame

**MiniMaxOpening vs ABOpening:**

*Savings shown below are 1) 73,440 to 542 positions evaluated, 2) 39,984 to 451 positions evaluated.*

1. Input board:

xxxxxxxxxxxxxxxxxx

MiniMaxOpening:

*python MiniMaxOpening.py board1.txt boardput.txt 4*

*Board Position: Wxxxxxxxxxxxxxxxxx*

*Positions evaluated by static estimation:* ***73440***

*MINIMAX estimate: 0*

ABOpening:

*python ABOpening.py board1.txt boardput.txt 4*

*Board Position: Wxxxxxxxxxxxxxxxxx*

*Positions evaluated by static estimation:* ***542***

*ALPHA-BETA estimate: 0*

2. Input board:

WBWBxxxxxxxxxxxxxx

MiniMaxOpening:

python MiniMaxOpening.py board4.txt boardput.txt 4

Board Position: WxWBWxxxxxxxxxxxxx

Positions evaluated by static estimation: ***39984***

MINIMAX estimate: 1

ABOpening:

python ABOpening.py board4.txt boardput.txt 4

Board Position: WxWBWxxxxxxxxxxxxx

Positions evaluated by static estimation: ***451***

ALPHA-BETA estimate: 1

**MiniMaxGame vs ABGame:**

*Savings shown below are 1) 170,304 to 11,529 positions evaluated, 2) 25,528 to 1515 positions evaluated.*

1. Input board:

WWBBWBWBWBWBWWxxxx

MiniMaxGame:

*python MiniMaxGame.py board10.txt boardput.txt 6*

*Board Position: WWBBWBWBWBWBWxWxxx*

*Positions evaluated by static estimation:* ***170304***

*MINIMAX estimate: 10000*

ABGame:

*python ABGame.py board10.txt boardput.txt 6*

*Board Position: WWBBWBWBWBWBWxWxxx*

*Positions evaluated by static estimation:* ***11529***

*ALPHA-BETA estimate: 10000*

2. Input board:

WWBBWBWBxxxxWWxxxx

MiniMaxGame:

*python MiniMaxGame.py board12.txt boardput.txt 5*

*Board Position: xWBBWBWBxxxxWWxWxx*

*Positions evaluated by static estimation:* ***25528***

*MINIMAX estimate: 2973*

ABGame:

*python ABGame.py board12.txt boardput.txt 5*

*Board Position: xWBBWBWBxxxxWWxWxx*

*Positions evaluated by static estimation:* ***1515***

*ALPHA-BETA estimate: 2973*

# Improved Static Estimation Function: 2 Examples for Both Opening and Midgame/Endgame

**MiniMaxOpening vs MiniMaxOpeningImproved:**

1. Input board:

xxxxxxxxxxxxxxxxxx

MiniMaxOpening:

*python MiniMaxOpening.py board1.txt boardput.txt 4*

*Board Position:* ***Wxxxxxxxxxxxxxxxxx***

*Positions evaluated by static estimation: 73440*

*MINIMAX estimate: 0*

MiniMaxOpeningImproved:

*python MiniMaxOpeningImproved.py board1.txt boardput.txt 4*

*Board Position:* ***xxxxxxxxxxxxxxWxxx***

*Positions evaluated by static estimation: 73440*

*MINIMAX-OPENING-IMPROVED estimate: 4.200000000000001*

***In the above case, MiniMaxOpeningImproved grabs a better opening position on the board – a position with more mill opportunities and more neighbors (increased move/mobility options).***

2. Input board:

WBxxxxBxxxxxxxWxxx

MiniMaxOpening:

*python MiniMaxOpening.py board5.txt boardput.txt 2*

*Board Position:* ***WBWxxxBxxxxxxxWxxx***

*Positions evaluated by static estimation: 182*

*MINIMAX estimate: 0*

MiniMaxOpeningImproved:

*python MiniMaxOpeningImproved.py board5.txt boardput.txt 2*

*Board Position:* ***WBxxxxBxxxxxWxWxxx***

*Positions evaluated by static estimation: 182*

*MINIMAX-OPENING-IMPROVED estimate: -72.0*

***In the above case, MiniMaxOpeningImproved places a piece in an area of the board with more mill opportunities and more neighbors (increased move/mobility options).***

**MiniMaxGame vs MiniMaxGameImproved:**

1. Input board:

WWWWxxxxxxxxxxBBBB

MiniMaxGame:

*python MiniMaxGame.py board15.txt boardput.txt 1*

*Board Position:* ***WxWWxxxxWxxxxxBBBB***

*Positions evaluated by static estimation: 5*

*MINIMAX estimate: -5*

MiniMaxGameImproved:

*python MiniMaxGameImproved.py board15.txt boardput.txt 1*

*Board Position:* ***WWxWWxxxxxxxxxBBBB***

*Positions evaluated by static estimation: 5*

*MINIMAX-MID/END-IMPROVED estimate: -21.579999999999995*

***In the above case, MiniMaxGameImproved takes a different route, preserving more opportunities to create Mills with future movements (unlike MiniMaxGame, all of MiniMaxGameImproved’s pieces are now missing just 1 more piece to make a mill).***

2. Input board:

WWWWWxxxxxxxxBBBBB

MiniMaxGame:

*python MiniMaxGame.py board16.txt boardput.txt 1*

*Board Position:* ***WWWWxWxxxxxxxxBBBB***

*Positions evaluated by static estimation: 7*

*MINIMAX estimate: 994*

MiniMaxGameImproved:

*python MiniMaxGameImproved.py board16.txt boardput.txt 1*

*Board Position:* ***WWWWxWxxxxxxxBxBBB***

*Positions evaluated by static estimation: 7*

*MINIMAX-MID/END-IMPROVED estimate: 926.21*

***In the above case, while both programs make a capture, MiniMaxGameImproved removes a more strategically valuable piece – one that would have provided black with more mill and mobility options.***

# Short Explanation of Why My Static Estimation Function is an Improvement

***Description of static estimation function:***

For white:

(1000 \* (numWhitePieces - numBlackPieces)

+ 83 \* numWhitePotentialMills - 82 \* numBlackPotentialMills

+ 27 \* numWhiteThreeMillPos - 26.9 \* numBlackThreeMillPos - 26 \* numWhiteOneMillPos + 25.9 \* numBlackOneMillPositions

+ 4 \* numWhiteFourNeighborPositions - 3.9 \* numBlackFourNeighborPositions

- 0.24 \* numWhiteBlockedPieces + 0.25 \* numBlackBlockedPieces + 0.06 \* numWhiteMoves - 0.06 \* numBlackMoves)

Additionally, during midgame/endgame, either color being reduced to 2 pieces or zero moves will result in an appropriate signed score of + - 1 million.

The black static estimation is the same as that for white, \* -1.

***Short Explanation of Why My Static Estimation Function is an Improvement:***

Note that the main static estimation function is divided into 5 lines. The multipliers are generally designed so that ***each subsequent line, even if fully maximized in one extreme, cannot overpower any lines above it.*** Meanwhile, the numbers on the line are chosen such that the values will be treated roughly equally, except that the leftmost numbers, which are slightly higher, will break ties. This static estimation function is an improvement because it keeps sight of the overall important goal of removing pieces – placing by far the highest multiplier on that. But, it adds motivation for to look to maximize:

1. Potential mills (2 W’s and an x that together would form a mill),

2. Positions that can have 3 mills

3. Positions with 4 neighbors (more mobility/options)

4. Blocking opponents pieces

5. And also: looking to minimize 1 mill positions

With all these factors it seeks to maximize the good for oneself and minimize the good for the opponent. ***By prioritizing them in the right order, the algorithm never misses the excellent in pursuit of the mediocre.*** Thus it is able to consistently make smarter choices. ***The tie-break structure is designed with the hope that the opponent misses opportunities. For example, if the algorithm predicts both me and my opponent will have the opportunity to grab a 3 mill position, the tie is broken to tip the algorithm to have me grab the position now.*** If the opponent grabs it – fine, then it’s a tie. But ***the hope is that the opponent misses the opportunity, and it then the choice becomes much more valuable than a tie.***

# Examples of MiniMaxOpeningBlack, MiniMaxGameBlack

MiniMaxOpeningBlack:

Input board: xxxxxxxxxxxxxxxxxx

*python MiniMaxOpeningBlack.py board1.txt boardput.txt 4*

*Board Position: Bxxxxxxxxxxxxxxxxx*

*Positions evaluated by static estimation: 73440*

*MINIMAX-BLACK estimate: 0*

MiniMaxGameBlack:

Input board: BBWWBWBWxxxxBBxxxx

*python MiniMaxGameBlack.py board12b.txt boardput.txt 4*

*Board Position: xBWWBWBWxxxxBBxBxx*

*Positions evaluated by static estimation: 2238*

*MINIMAX-BLACK estimate: 2971*

# Explanation of ABOpenImpWhite, ABOpenImpBlack, ABGameImpWhite, ABGameImpBlack

These non-required files apply the improved static estimation function from MiniMaxOpeningImproved/MiniMaxGameImproved using the Alpha-Beta pruning algorithm. I made them for the tournament and figured why not include them.

# Conclusion

This assignment was a good opportunity to explore the important concepts of MiniMax, Alpha-Beta pruning, static estimation, optimization, game states, processing speed, benefits of increased exploration space, and more. I enjoyed the flexible nature of Part IV, and I felt the game was a good scale for a school project – not overwhelmingly huge, but also not trivially simple. It was very satisfying to improve the static estimation and create a program that could compete more effectively at the game.