Part I. MiniMaxOpening.py and ABOpening.py were applied to the initial board

CASE (1): xWxxxBxxWxxWBxxxBxBxx and depth 5.

For MiniMaxOpening.py, it visited 675540 leave nodes and take ~4.58 seconds;

For ABOpening.py, it visited 31702 leave nodes and take ~0.25 seconds;

Both produced the same static estimation 1.

CASE (2): xWxxxBBxWxxWBxxxBxxxx and depth 5.

For MiniMaxOpening.py, it visited 622474 leave nodes and take ~4.47 seconds;

For ABOpening.py, it visited 26719 leave nodes and take ~0.20 seconds;

Both produced the same static estimation 1.

In both cases, the alpha-beta pruning visited less nodes and required less running time.

Part II. MiniMaxGame.py and ABGame.py were applied to the initial board

CASE (1): xWxxxBxxWxxWBxxxBxBxx and depth 4.

For MiniMaxGame.py, it visited 526621 leave nodes and take ~12.89 seconds;

For ABGame.py, it visited 6218 leave nodes and take ~0.22 seconds;

Both produced the same static estimation -9.

CASE (2): xWxxxBBxWxxWBxxxBxxxx and depth 4.

For MiniMaxGame.py, it visited 549676 leave nodes and take ~13.40 seconds;

For ABGame.py, it visited 7423 leave nodes and take ~0.23 seconds;

Both produced the same static estimation -10.

In both cases, the alpha-beta pruning visited less nodes and required less running time.

Part III. Improved Static Estimation:

We proposed another function NearMill(board) to judge whether a board contains two same colored chess pieces which could potentially close a mill when adding another piece of the same color in the next move. As an example, if both positions 0 and 2 are white (position 4 is empty), then this board layout is preferable to White because if we add another white at position 4, it will close a mill. Thus, for both Opening and Midgame, we count how many such NearMill cases for White, and how many such NearMill cases for Black. We then add 10 \* (NearMillWhite – NearMillBlack) to the original static estimations.

Part IV. MiniMaxOpening.py and MiniMaxOpeningImproved.py were applied to the initial board

CASE (1): xWxxxBxxWxxWBxxxBxBxx and depth 5.

For MiniMaxOpening.py, it produced the move xWxxxBxxWxxWxxxxBxBxW;

For MiniMaxOpeningImproved.py, it produced xWxxxBxxWWxWBxxxBxBxx;

CASE (2): xWxxxBBxWxxWBxxxBxxxx and depth 5.

For MiniMaxOpening.py, it produced the move xWxxxBBxWxxWxxxxBxxxW;

For MiniMaxOpeningImproved.py, it produced xWxxxBBxWxWWBxxxBxxxx;

In both cases, the produced moves are different.

Part V. MiniMaxGame.py and MiniMaxGameImproved.py were applied to the initial board

CASE (1): xWxxxBxxWxxWBxxxBxBxx and depth 4.

For MiniMaxGame.py, it produced the move xWxxxBxxWxxxBxxxBxBxW;

For MiniMaxGameImproved.py, it produced xWxxxBxxxxxWxxxxBxBxW;

CASE (2): xWxxxBBxWxxWBxxxBxxxx and depth 4.

For MiniMaxGame.py, it produced the move xWxxxBBxWxxxBxxxBxxxW;

For MiniMaxGameImproved.py, it produced xWxxxBBxxxxWxxxxBxxxW;

In both cases, the produced moves are different.