Institute of Information Technology
University of Dhaka

Topic: Minichess

Artificial Intelligence (CSE-604)

Submitted by:

Mashiat Amin Farin - 1202

Mohammad Momenuzzaman - 1227

Monayem Sarker - 1228

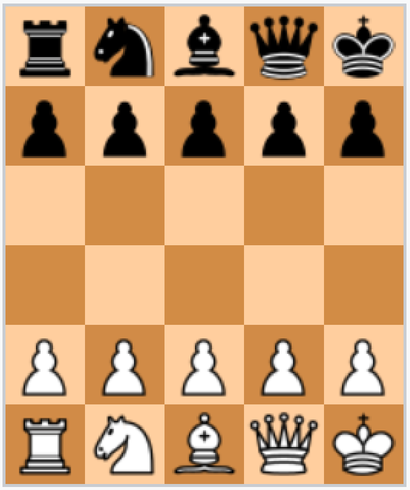
Submitted to:

Dr. Ahmedul Kabir
Associate Professor
Institute of Information Technology
University of Dhaka

Submission date: 11/09/23

Introduction

Minichess is a miniaturized and more accessible version of chess played on different sizes of boards, with the basic rules the same for all existing pieces. The version we have implemented is the following:


Chess Attack

Basic Features:

This implementation of minichess follows all normal moves of chess. We have implemented some features:

1. All basic movements of all pieces are implemented.
2. When the king is in check by the opponent, the game prevents any other move until the king is protected from check.
3. Pawn initial double move is also implemented
4. Pawn promotion works only to promote to a queen.
5. Turn indicator

However, castling has not yet been implemented.

AI:

➔ Base algorithm used: **Minimax**

➔ **Alpha beta pruning** is used to minimize the game tree.

➔ **Heuristic move ordering:** If the following move is able to eliminate a piece of higher power, with a piece of lower power, we consider that an advantage and traverse those moves first.

➔ **Evaluation Function:** While finding the best move, the evaluation function considers the

a. position of the pieces
b. number of pieces left
c. whether any move exposes the king.

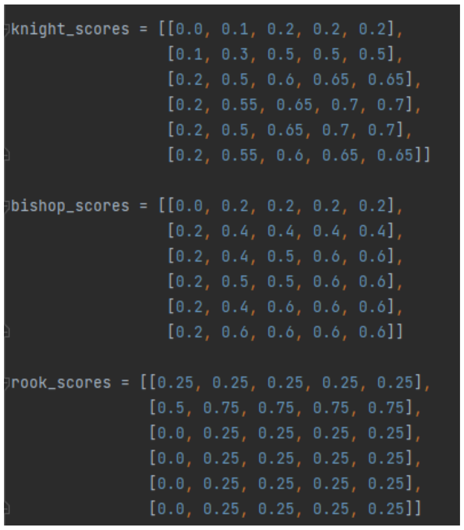
Appendix

To evaluate the board, we have chosen the following parameters:

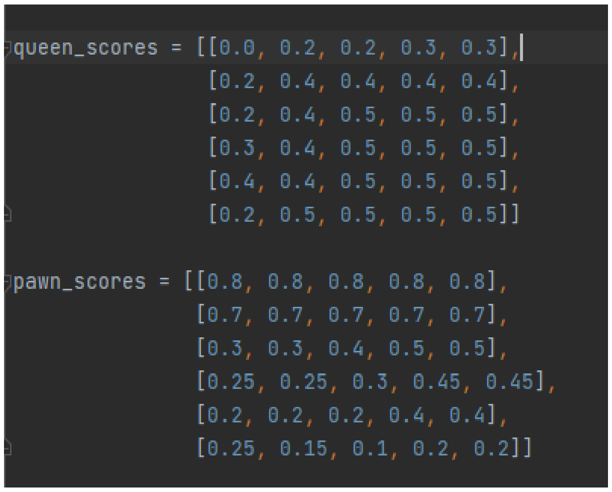
1. If you control the center, this is rather favorable.
2. If you have more pieces than your opponent, this is rather favorable.
3. If you lost your Queen, this is rather not favorable.
4. If you have a pawn that is close to being promoted this is favorable. [1]

We used two variations to consider the position of the pieces on the board.

1. **Reduction of 8x8 chess board:** Here the values are taken from a standard 8x8 chess board and reduced.

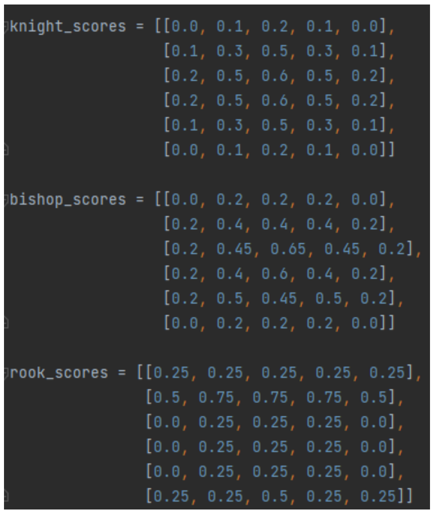


```
knight_scores = [[0.0, 0.1, 0.2, 0.2, 0.2],
                 [0.1, 0.3, 0.5, 0.5, 0.5],
                 [0.2, 0.5, 0.6, 0.65, 0.65],
                 [0.2, 0.55, 0.65, 0.7, 0.7],
                 [0.2, 0.5, 0.65, 0.7, 0.7],
                 [0.2, 0.55, 0.6, 0.65, 0.65]]

bishop_scores = [[0.0, 0.2, 0.2, 0.2, 0.2],
                 [0.2, 0.4, 0.4, 0.4, 0.4],
                 [0.2, 0.4, 0.5, 0.6, 0.6],
                 [0.2, 0.5, 0.5, 0.6, 0.6],
                 [0.2, 0.4, 0.6, 0.6, 0.6],
                 [0.2, 0.6, 0.6, 0.6, 0.6]]

rook_scores = [[0.25, 0.25, 0.25, 0.25, 0.25],
               [0.5, 0.75, 0.75, 0.75, 0.75],
               [0.0, 0.25, 0.25, 0.25, 0.25],
               [0.0, 0.25, 0.25, 0.25, 0.25],
               [0.0, 0.25, 0.25, 0.25, 0.25],
               [0.0, 0.25, 0.25, 0.25, 0.25]]
```

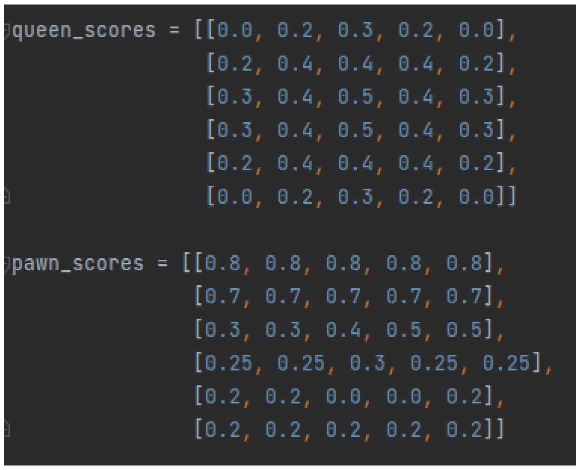


```
queen_scores = [[0.0, 0.2, 0.2, 0.3, 0.3],
                [0.2, 0.4, 0.4, 0.4, 0.4],
                [0.2, 0.4, 0.5, 0.5, 0.5],
                [0.3, 0.4, 0.5, 0.5, 0.5],
                [0.4, 0.4, 0.5, 0.5, 0.5],
                [0.2, 0.5, 0.5, 0.5, 0.5]]

pawn_scores = [[0.8, 0.8, 0.8, 0.8, 0.8],
               [0.7, 0.7, 0.7, 0.7, 0.7],
               [0.3, 0.3, 0.4, 0.5, 0.5],
               [0.25, 0.25, 0.3, 0.45, 0.45],
               [0.2, 0.2, 0.2, 0.4, 0.4],
               [0.25, 0.15, 0.1, 0.2, 0.2]]
```

2. **Manually refined 5x6 chess board:** Here, the values of the 8x8 board is analyzed and then placed manually on the board.



```
knight_scores = [[0.0, 0.1, 0.2, 0.1, 0.0],
                 [0.1, 0.3, 0.5, 0.3, 0.1],
                 [0.2, 0.5, 0.6, 0.5, 0.2],
                 [0.2, 0.5, 0.6, 0.5, 0.2],
                 [0.1, 0.3, 0.5, 0.3, 0.1],
                 [0.0, 0.1, 0.2, 0.1, 0.0]]

bishop_scores = [[0.0, 0.2, 0.2, 0.2, 0.0],
                 [0.2, 0.4, 0.4, 0.4, 0.2],
                 [0.2, 0.45, 0.65, 0.45, 0.2],
                 [0.2, 0.4, 0.6, 0.4, 0.2],
                 [0.2, 0.5, 0.45, 0.5, 0.2],
                 [0.0, 0.2, 0.2, 0.2, 0.0]]

rook_scores = [[0.25, 0.25, 0.25, 0.25, 0.25],
               [0.5, 0.75, 0.75, 0.75, 0.5],
               [0.0, 0.25, 0.25, 0.25, 0.0],
               [0.0, 0.25, 0.25, 0.25, 0.0],
               [0.0, 0.25, 0.25, 0.25, 0.0],
               [0.25, 0.25, 0.5, 0.25, 0.25]]
```



```
queen_scores = [[0.0, 0.2, 0.3, 0.2, 0.0],
                [0.2, 0.4, 0.4, 0.4, 0.2],
                [0.3, 0.4, 0.5, 0.4, 0.3],
                [0.3, 0.4, 0.5, 0.4, 0.3],
                [0.2, 0.4, 0.4, 0.4, 0.2],
                [0.0, 0.2, 0.3, 0.2, 0.0]]

pawn_scores = [[0.8, 0.8, 0.8, 0.8, 0.8],
               [0.7, 0.7, 0.7, 0.7, 0.7],
               [0.3, 0.3, 0.4, 0.5, 0.5],
               [0.25, 0.25, 0.3, 0.25, 0.25],
               [0.2, 0.2, 0.0, 0.0, 0.2],
               [0.2, 0.2, 0.2, 0.2, 0.2]]
```

In both cases, the AI performs well. But we felt that the reduction of the 8x8 board performs better. This could be due to the fact that

a. Original 8x8 board has gone through rigorous testing
b. The values are fine tuned and
c. Consider the position of the king and queen.

In the manually refined version, we may have missed some inner workings of chess and misplaced values.

References:
[1] engines - What is an accurate way to evaluate chess positions? - Chess Stack Exchange