Study IT in .ee

Chess AI

"Beat me if you can"

Gamal El Koumy, Mahmoud Kamel, Nesma Mahmoud
Institute of computer science, University of Tartu

Idea:

Our main goal is to use advanced algorithms for enhancing the run time/ improve the AI performance in the chess game. Using AI to generate all possible moves (Search Space) from any chess board state, then looking for the best move as shown in figure 1 within a reasonable time-limit. The main idea can be shown through figure 8.

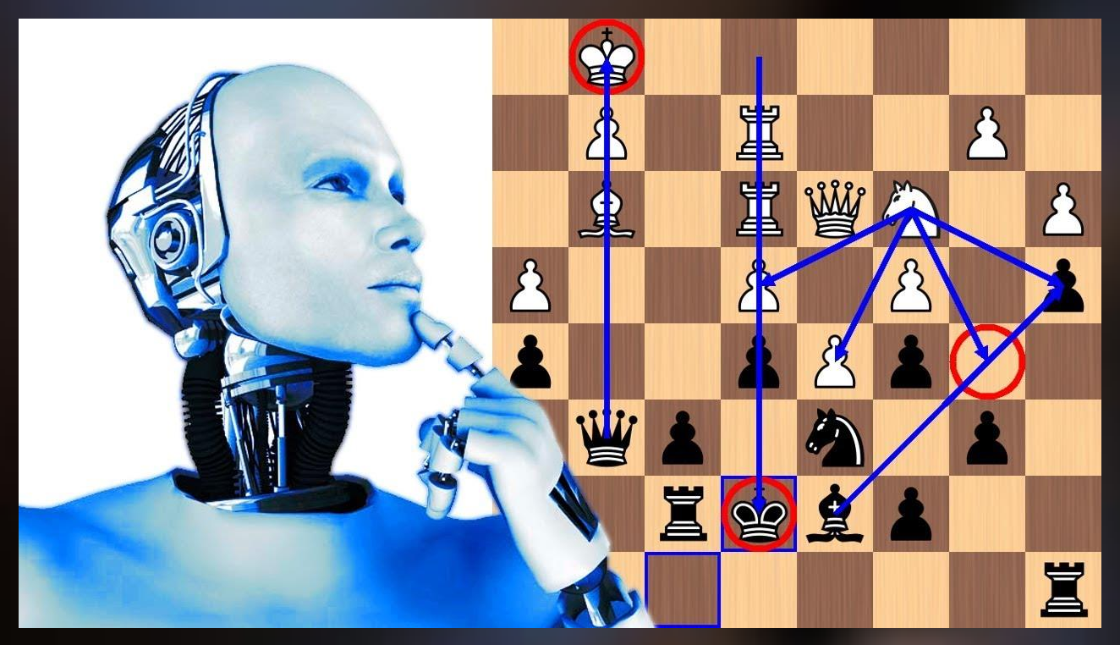

Figure 1: Chess AI

Methods:

1. NegaMax: a tree searching algorithm derived from minimax, which looks to the future to minimizes the maximum value that your opponent can take from you and decide how best to make your next move.

2. Alpha-beta pruning [1]: used to decrease the number of nodes that are evaluated by the negamax algorithm, see figure 2.

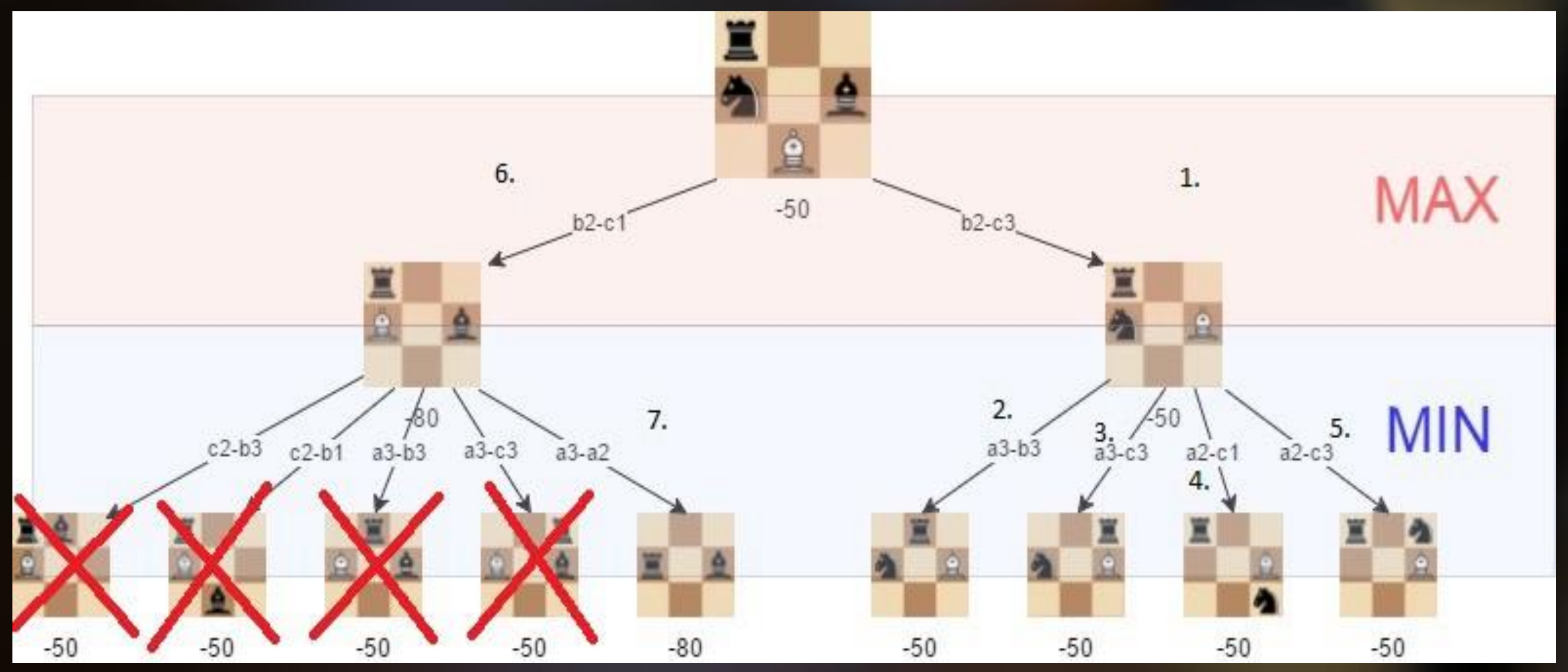

Figure 2: Alpha-beta pruning

Contribution [2][3]:

1. Evaluation function that uses the following heuristics (defined by chess players):
- A. Further developed pieces are better.
- B. Doubled Pawns are not good, see figure 3.
- C. Isolated Pawns are not good, see figure 4.

2. Iterative Deepening: It is the improvement for depth-limited search, Instead of specifying a fixed search depth, It starts working with depth 1, then depth 2, and continue incrementing the depth until the time allocated for the search is exhausted

3. Multi-Threading: We implement the search in the search tree to execute in parallel to reduce the total execution time.

4. We cache the evaluated steps, so we can make use of them in case of repetition to reduce the execution time.

5. We handle the end phase of the game in a special way and we use previously evaluated positions from online chess databases that contain the games of master players.

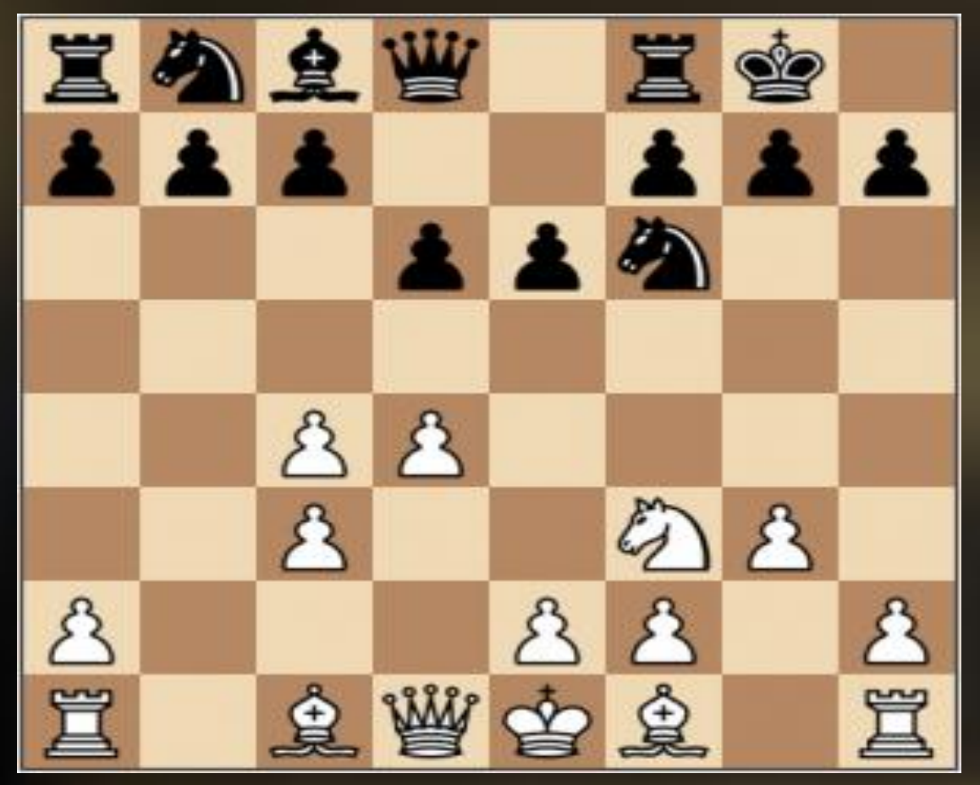

Figure **3:** doubled Pawns

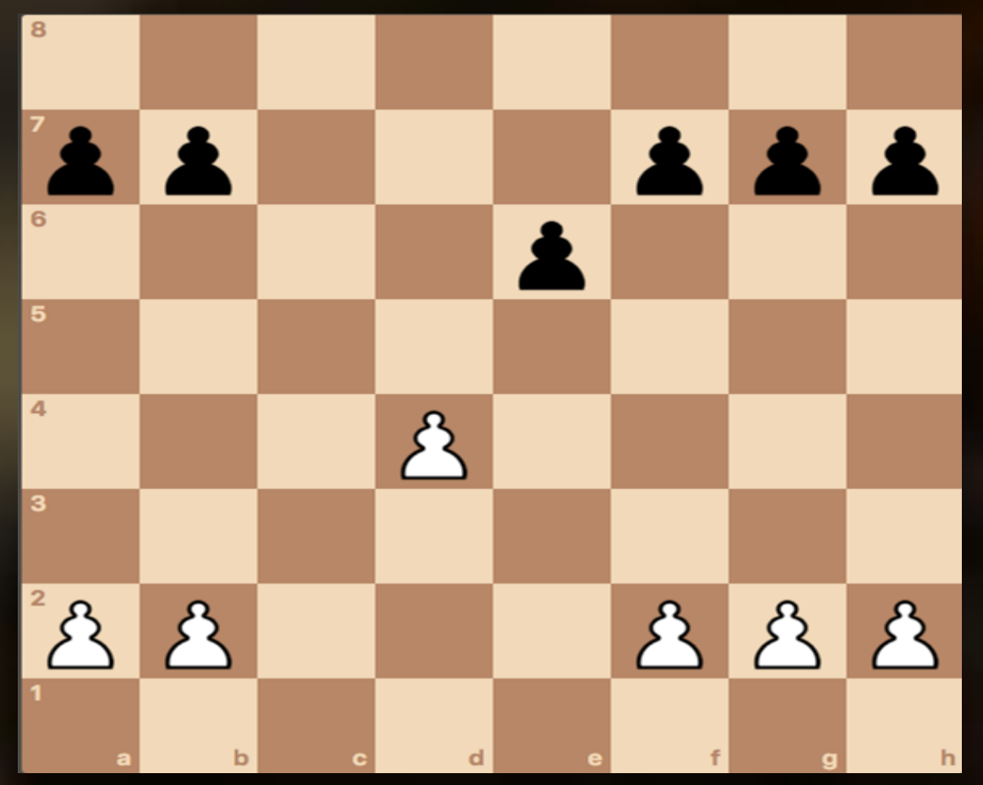

Figure **4:** Isolated Pawns

Results:

We developed a chess game based on AI where the user can play against AI. Figure 5 shows the start state of the board, where user can play with white and AI with black. User can select any piece and will have a guide to all possible positions that can move to as shown in figure 6.

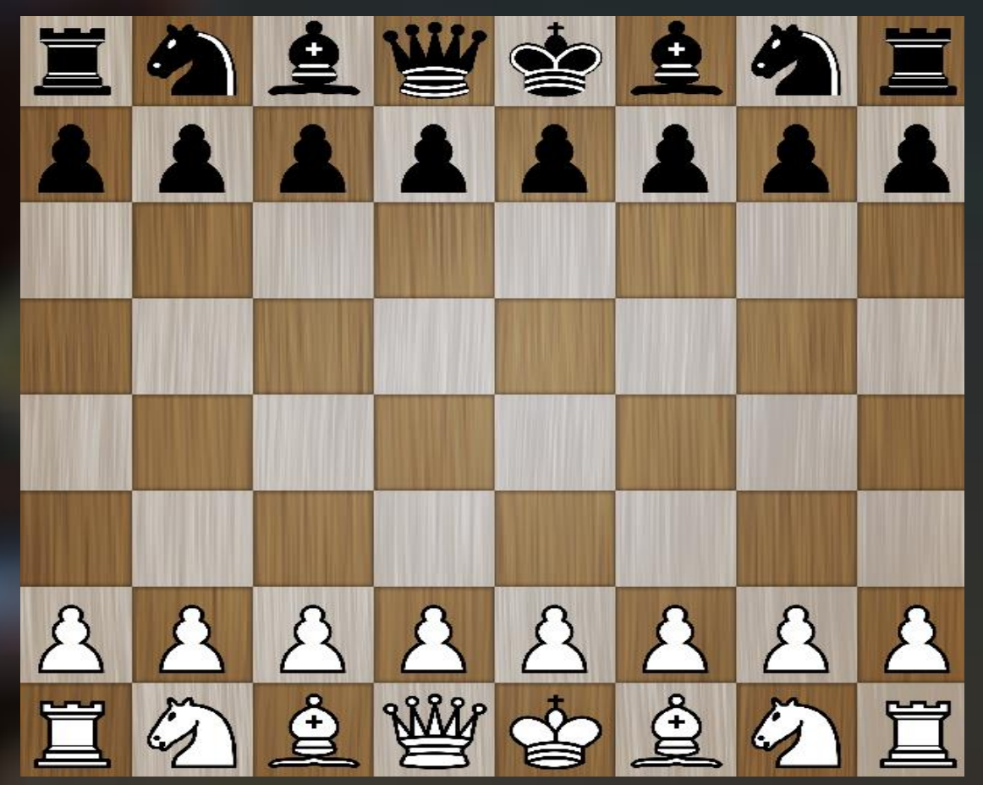

Figure 5: Start state of board game

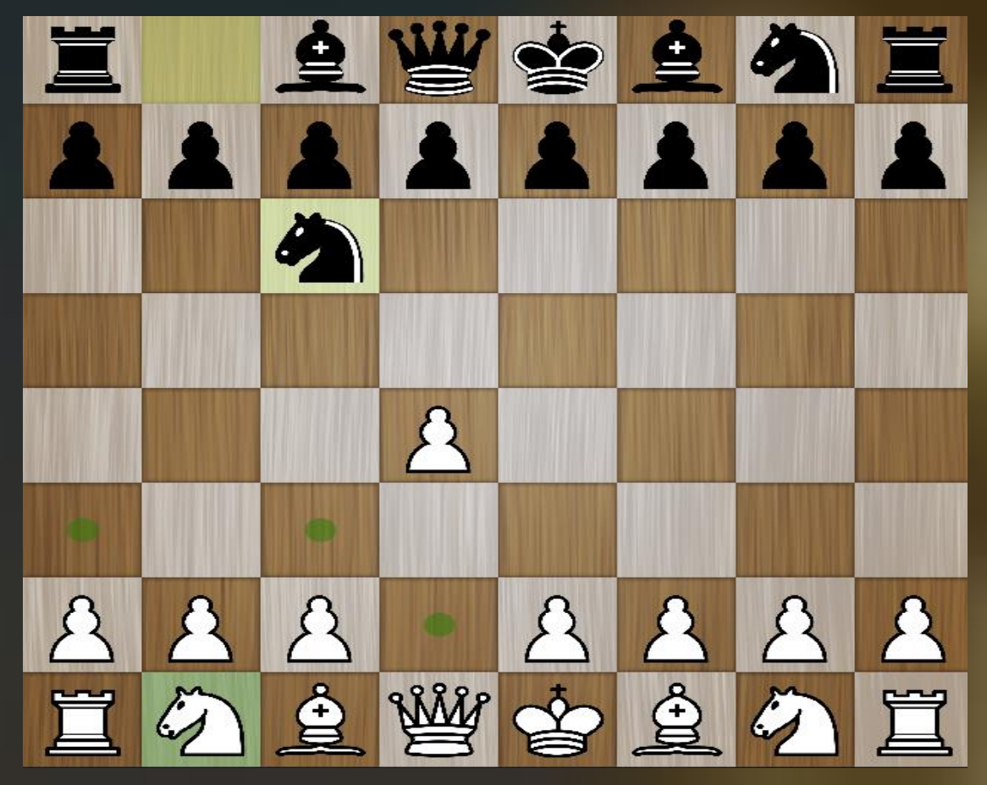

Figure 6: All possible moves for the knight white.

User can input the next move based on the click-click method, first click to choose a piece, and the second click to move the chosen piece. After each step program check for the end game condition as shown in figure 7 for checkmate condition.

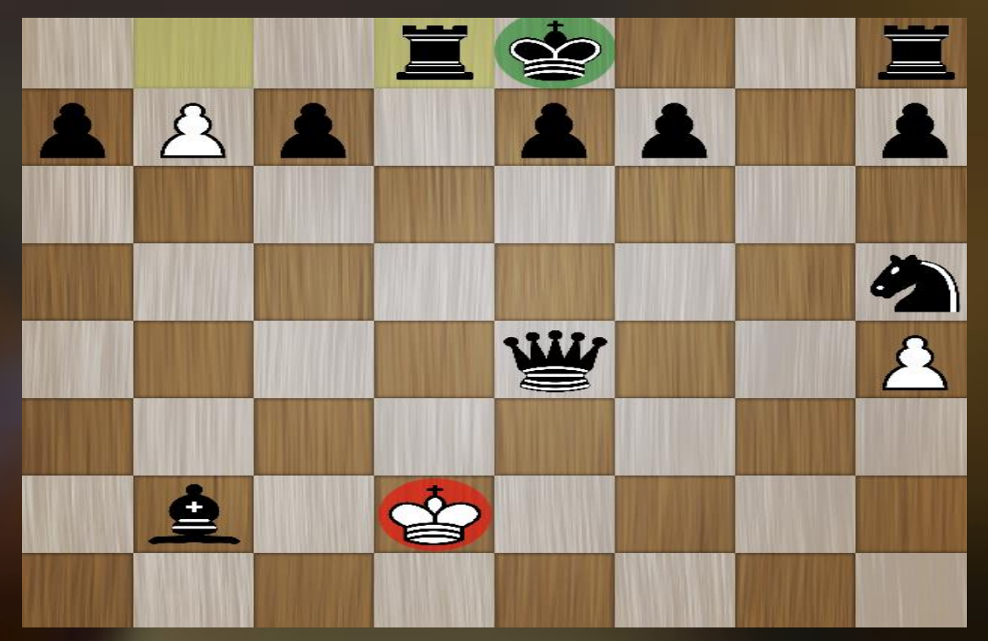

Figure 7: End game

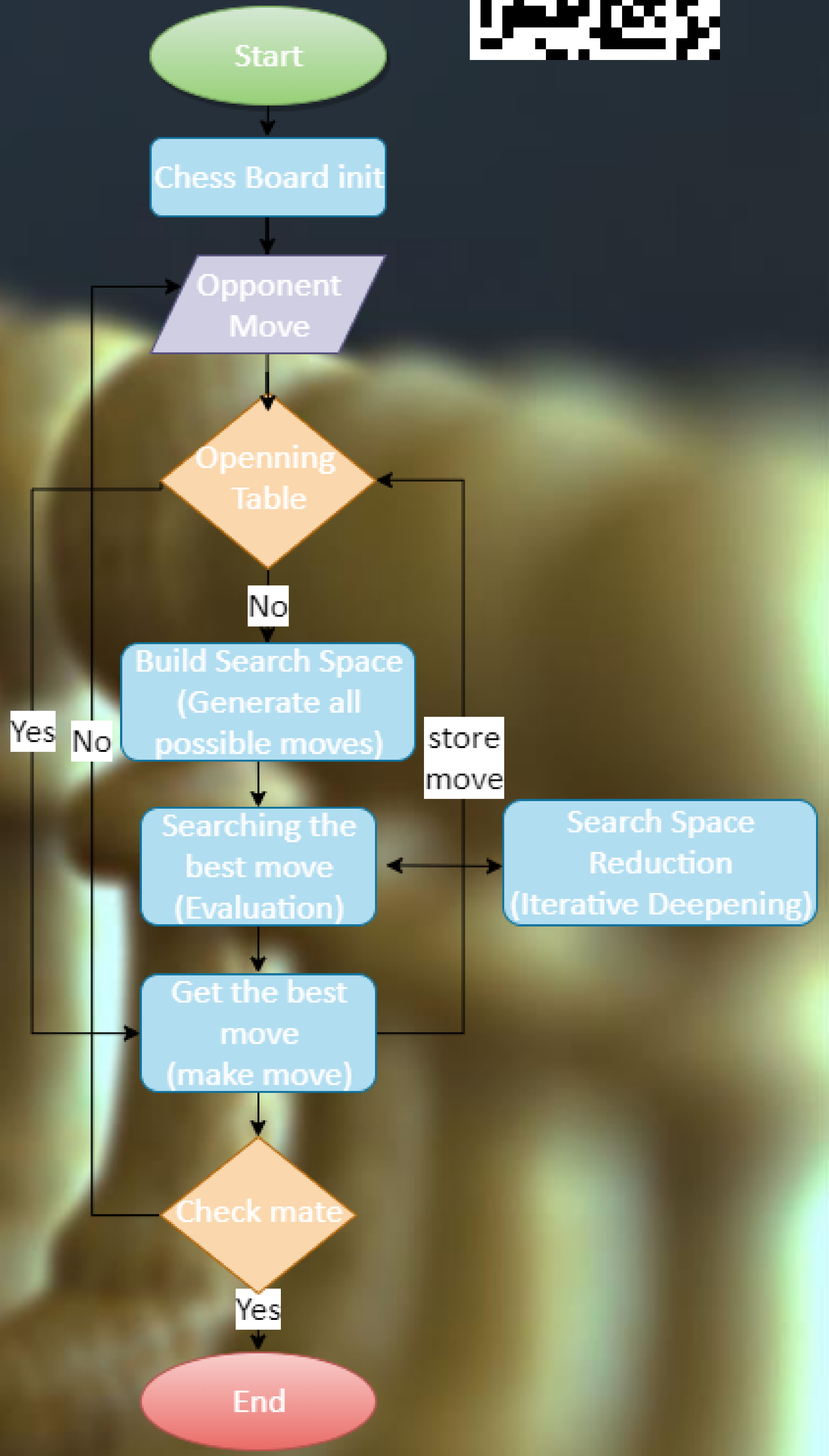
Start
Chess Board init
Opponent Move
Openning Table
No
Yes
No
Build Search Space (Generate all possible moves)
store move
Searching the best move (Evaluation)
Search Space Reduction (Iterative Deepening)
Get the best move (make move)
Check mate
Yes
End

Figure 8: Flow chart of the engine

Future work:

Applying further enhancement would make the engine better. We think that applying methods like Neural Network, Genetic Algorithms, and random selection in the evaluation function would have interesting results.

References:

[1] https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/
[2] https://www.chessprogramming.org/Main_Page
[3] Campbell, Murray, A. Joseph Hoane Jr, and Feng-hsiung Hsu. "Deep blue." Artificial intelligence 134.1-2 (2002): 57-83.