Artificial Intelligence Fall 2023 - Project Proposal

Team Members

- Ahmad Suleiman - suleimad@clarkson.edu
- Nowreen Ahsan - Noahsan@clarkson.edu
- Katie Bonk - bonkka@clarkson.edu

Project Idea

Our project idea is about "*Solving Chess Checkmate Puzzle*". The problem falls under **Adversarial Search,** usually solved using a minimax algorithm.

Problem Challenges

Some of main challenges of chess problems are:

- **Large branching factor**: The number of possible move of the chess position is very high. The maximum number of moves per chess position can get upto **218** possible legal moves [1].
- **Evaluation:** Hard to choose which move leads to checkmate faster, because opponent might not choose that particular follow up move.

Problem Definition

The problem definition is as follows:

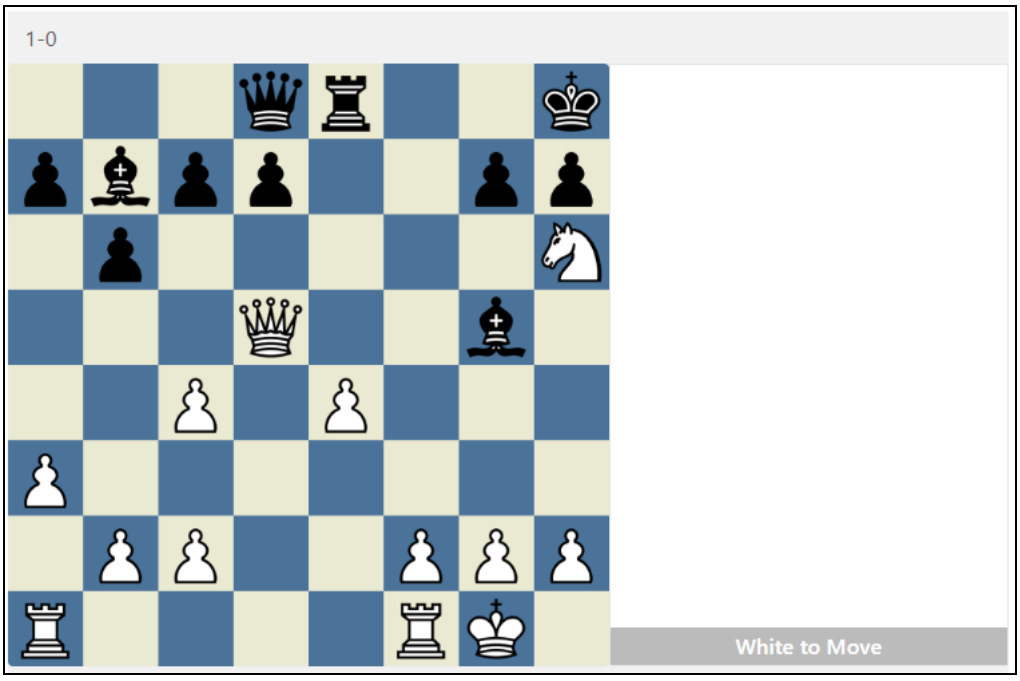
- **State:** A valid chess position with a forced checkmate in **n** moves and who to move next (i.e., either white or black). Also includes the number of **move circles** (i.e., both players add a move) made so far.

- **Initial State:** Initial valid chess position with a forced checkmate in **n** moves and who to move next (i.e., either white or black).
- **Successor**: A valid chess move (by pawn or piece) from a state to yield another state.
- **Terminal Test:** If there is a checkmate or the number of **move circles** is equal to **n**.
- **Utility function:** **+1** when a player from the initial state checksmates within **n** move circles, **0** when no player has a checkmate within n move circles, and **-1** if the player from the initial state got checkmated within **n** move circles.

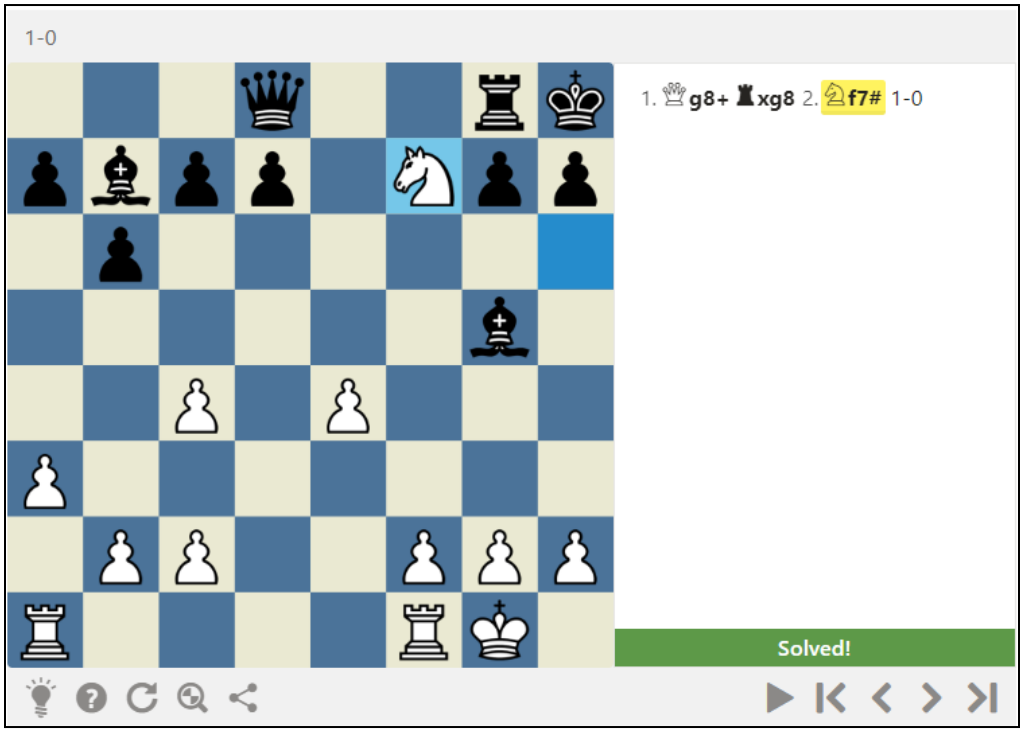
Problem Example

Chess positions are usually represented in the form of Forsyth–Edwards Notation (FEN) [2]. Below is an example of a **checkmate in a 2-moves puzzle**.

FEN: 3qr2k/pbpp2pp/1p5N/3Q2b1/2P1P3/P7/1PP2PPP/R4RK1 w - - 0 1



The solution to the puzzle is shown below:



Search Algorithms to Use

To solve this problem, we will attempt to use the following adversarial search algorithms:

- **Search 1:** The standard Minimax Algorithm
- **Search 2:** Minimax Algorithm with Alpha-beta Pruning
- **Search 3:** Minimax Algorithm with Alpha-beta Pruning and Heuristic Evaluation Function. We plan to try 3 different heuristic and compare their performance. Some of the heuristic we currently have in mind include:
    - Prioriortizing moves that are check
    - Prioritizing moves that limit the opponents king's available moves
- **Search 4:** Using a Local search approach. Though this method is fast, but it does not always find the correct solution to the puzzle. We plan to explore its accuracy rate.

We will analyze the performance of the above algorithms based on:

- Solution Accuracy
- The number of states expanded

Dataset

To validate and analyze the search algorithm, we downloaded a large dataset of checkmate puzzles from [3], [4] and [5]. For the scope of this class, we will only analyze and report puzzle checkmate in 2, 3, and 4 moves. The dataset of the checkmate puzzles we downloaded are all in FEN format.

Reference

[1] Chess: https://www.chessprogramming.org/Chess_Position#cite_note-4

[2] FEN https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation

[3] Checkmate in 2 moves: https://wtharvey.com/m8n2.txt

[4] Checkmate in 3 moves: https://wtharvey.com/m8n3.txt

[5] Checkmate in 4 moves: https://wtharvey.com/m8n4.txt