Fog of War Chess Proposal

Alexandre Feren
COS470: Introduction to AI

March 12, 2021

1 Summary:

This project will be an individual implementation of a game-playing AI for the chess variant Fog of War Chess. There are a few key differences between this game and chess: the goal is to capture the opposing king rather than to put it in checkmate, and you only have information about the board where your pieces can 'see'. That is to say that you have full information about all squares where pieces can capture, and information about whether a piece exists where a pawn could otherwise move to. Ideally, I would like to make an AI that is strong enough to compete with myself, and I may test it against others if I can find some people who would be interested in testing it out. I will be maintaining this project on *GitHub*. I suspect the main challenge of implementation for this project will be a combination of accurate estimation of opposing piece locations, and risk estimation based on where they may be. I have spent some time playing this chess variant on *chess.com* and have enjoyed the game.

2 Description

Overview what will be contained in the project, what resources will be used, and how I plan on implementing it. The project will be maintained on my *GitHub Account*

2.1 Motivation

As mentioned, I have played a good few games of Fog of War Chess, but as far as I can tell, there are no available chess engines that can play the game or evaluate whether a move was the correct choice. In looking around, I only found one *possible AI implementation*, and it seems to be more of an informal competition than anything else. Given this, I decided that this made for a good opportunity for me to start working on my own implementation, and an excuse for me to spend some time playing around with the game as well.

2.2 Background

As there aren't many resources for Fog of War Chess AI implementation, I believe I am mostly going to need to reference various strategy guides and opinions of high-level players of the game. Here are some resources directly related to Fog of War Chess

Strategy Guide
Programming Contest
High Level Strategy Guide

After some more consideration since my pre-proposal, I have realized that machine learning may even be a useful technique (though not necessarily one that I will use) to understand where the enemy pieces are located. Based on this, I think it may also be worthwhile to look at some of the strategies from *this video* Where there are a bunch of very bad chess AIs made to battle against each other, and the related papers.

References to Related Papers

Finally, I believe it may be worthwhile to look into board representation and move generation for chess AIs, which are well documented on a chess programming wiki.

Chess Programming Wiki

there are several techniques noted on this wiki that I plan to use, such as piece-square tables.

2.3 Proposed Approach

Methods that I will be using, or hope to use if time permits me to do so. Low Priority techniques will mostly be optimizations and attempted improvements to the High Priority Techniques that I will be using, and will only be touched if I get a program that I am relatively confident I have gotten to work well within my limits on the High Priority Techniques.

2.3.1 High Priority Techniques

Max Expected Utility, and related ideas (Ch. 16): Maximum expected utility would be one way of approaching the problem, in that the system will aim to maximize what it thinks will happen, and if the strategy guides I have seen are to be believed, this places a high value on defense and known information. Related to this, though, are the ideas of maximum average utility, and minimum possible loss, which I may also have to explore to test the efficacy of each one. Additionally, I will need to use some variation of the representation noted in chapter 16.5.1, where there are representations for chance, decision, and utility, though it will have to be adapted to be multi-step. It also notes something that I had earlier: that there is value in information. In my case, the 'questions' would be moves that are designed to get more information about the state of the board, while also being mostly rational in the context of a regular game of chess.

Evaluation: based on the belief set that is arrived at and some other meta-information about the game, an evaluation is needed. There is perfect information at least about what pieces are still on the board, so this will likely be a major part of the evaluation, as well as other parts, such as piece structure (self, and expected for the opponent), piece-value squares, and perhaps some other heuristics such as information gathering. These heuristics will have to be discovered and tweaked as I progress with the project, and may try to programmatically generate them with some sort of machine learning, likely a genetic algorithm since they are generally easier to implement than neural networks from my understanding. This will be paired with some sort of decision making algorithm, whether it is ExpectiMinimax, Markov-Chain, or otherwise.

Constraint Satisfaction: This is the primary technique that I plan on using to estimate the state of the board, and I will make a graph of possible moves based on what could have

been played by the opponent with the knowledge that is available about the game state, and then testing for any contradictions. The final choice on which states to consider will likely be the top n for possible states, where n the size of a subset of the possible moves, and then using these in the search function, which will likely be some variation of expectiminimax. This will allow me to resolve the state to a smaller set of possible states and hopefull improve the performance of the search.

2.3.2 Low Priority Techniques

Machine Learning: Notably, I would be using some sort reinforcement learning, and it would only be for the understanding of the board state. Q-learning seems like a possible good option for this, and I would be giving it access to the moves that can be seen (so mostly just moves from the side that is playing), and ask it to learn what state the board is in. As noted in the introduction, I will likely only be using this method if I get some other setup for board state estimation to work to some degree. The quality function that I would be using would simply be whether the board state was correct, and perhaps some credit for some correct piece placements as well, so getting 10 of 16 opposing pieces would get some credit, though not full credit.

2.4 Schedule

- March 18^{th}: Complete Game Representation and Move Generation
- March 25^{th}: Game State Estimator Design and Implementation Start
- April 1^{st}: Game State Estimator Implementation Finished
- April 8^{th}: Utility Function and Move Probability Estimation
- April 15^{th}: Basic Game Playing AI
- April 22^{nd}: Optimized Utility Function Parameters
- April 29^{th}: Final Submission

If these deadlines are met ahead of schedule, I will move on to attempting to implement a better board state estimator with Markov Chains, Q-Learning, or some other relevant technique.

2.5 Deliverables

The final deliverable of this project will be a program that is capable of playing Fog of War Chess, and relatively accurately guessing the board state. The stated goal is the ability for this AI to be able to beat me consistently, but I am very much uncertain about the feasibility of this goal for now. If I end up using machine learning, it will likely be able to accurately guess what high-level players will have played in an uncertain position.

3 Equipment

No equipment is needed for this project.

4 Budget

No equipment is needed for this project.

5 Biographical Sketch

I do not have any notable experience in AI creation for game playing, and the closest program that I have made to date was simply a Connect-4 AI that used a quite bad implementation of minimax. This will be a learning experience all the way through, and I expect it to be a challenge for me to meet all of my stated goals.