Chess engine implementation utilizing evaluation algorithm

Development of computer hardware/software on the foundational play of chess

A Thesis Prospectus

In STS 4500

Presented to

The Faculty of the

School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science in Computer Science

By

Angelo Bechtold

October 27, 2022

Angelo Bechtold:

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Kent Wayland, Department of Engineering and Society

Briana Morrison, Dept. of CS

General Research Problem: Chess Implementations - Theory and Engine Development

Chess is a game where there are only 64 squares, alternating black and white, and two sets of 6 pieces, also black and white positioned on the back 2 horizontal ranks of each side of the board. With such simple rules, inherently it wouldn't suggest a very complicated game. Though we tend to accept the notion that chess is a game that requires a large amount of intelligence ("playing chess, not checkers"). However, this notion is false and the game largely requires a lot of exposure to it through pattern recognition, visualization, and deep knowledge of positions. As a result, computers are the perfect vessel onto which chess games can be analyzed. One can relay all necessary information about the game in a few lines of computer code, and then the software can analyze all possible positions* and spit out the best possible move. It's important to keep in mind that it wasn't always the case that a computer could parse and handle large amounts of information in a short timeframe. This is especially true with the data sets that are derived from the astronomically large number of chess positions after just 4-5 moves from any given position. It follows then that the number of CPU cores and raw performance a computer has, the quicker the software can evaluate a position and return its best move. It's easy to see why computer improvement and chess play have been tightly linked for the past 3 decades.

It is often the case that chess players are eager to exchange thoughts and ideas about their games after each game. This eagerness to exchange is mainly in the spirit of constant improvement for both players. However, with the introduction of strong chess engines, now players usually first resort to the engine to cross-reference their moves with the canonical "best" move from the engine. The player-player and player-engine relationships have developed vastly over the past few decades, especially at the professional level. Formally, this paper will introduce a technical research problem directly addressing a chess engine that was implemented using modern algorithms over summer 2022, and an STS research problem that directly addresses the relationship between the evolution of the theory/play of chess and the respective engines that evaluate those positions.

Implementation of Chess and an Algorithm-Based Chess Engine

How can an algorithm-based chess engine be implemented using modern evaluation functions?

Background

Computer-based chess is closely related to computer science, and subsequently artificial intelligence. This cohesion is only natural because computer science is simply the analysis of computers and their manipulation of data. Chess, while not so binary in the human mind, can be reduced to a 8x8 array for computers where an array is simply a list - think "pawn", "pawn", "king", etc. Computers can manipulate this chess array and perform a substantial amount of calculations on it faster and more accurately than even the top players when given large amounts

of time. For reference, the most prestigious and highly acclaimed title is the Grandmaster (GM) title; it requires a FIDE (one of the most popular governing bodies of chess) rating of 2500 and 3 norms. Only 0.3% of chess players in the world have achieved this title. With this in mind, the existing top chess engines have surpassed GM performance so substantially that the engine either always holds a draw or wins the match against GMs. This level of performance is derived through the algorithms that underlie these engines. Modern algorithms utilized in the engines consist primarily of minimax and alpha-beta pruning, with a bias towards implementations of a pseudo-GM thought process.

Over the summer, a colleague and I took part in an internship developing a smaller version of a production-grade chess engine. First, we defined the end goal for this project - what the game would include graphically and functionally. This included an application that would display a chess board, allow the player to play a whole game of chess - either against another player or the engine, and the engine itself that would hold a minimum rating of 1500 when compared to other stronger engines. This rating was chosen based on the distribution of ratings on chess.com on a skewed distribution curve. This project was completed using Python and the game library Pygame. The primary focus throughout the implementation of the engine was the minimax / alpha-beta pruning algorithm that underlies most engines.

Minimax, being the most basic and widely used algorithm in chess engines, is a strong foundation for the implementation of a chess engine. The algorithm is defined by the following characteristics: 1) able to backtrack through the current path of data it currently holds, 2) able to analyze/evaluate a certain game state (in this case, a certain chess position), and 3) finds the optimal move for each player (defined by the previously mentioned evaluation function) (Vuckovic et al. 2015). One main component of this engine that renders it viable is the initial pruning of completely unreasonable moves. For example, consider the following position.

(source: lichess.org)

The pawn highlighted on the 5th rank (read as row and starts at 1 on the bottom from the white side), and the 3rd file (read as column and starts at 1 from the left-most side from the white side), is currently attacking the black queen on d4. A GM would not consciously consider any other move other than moving the queen away from the attack of the pawn (*Levene, 2007)*; the queen has a much higher value than the pawn. The algorithm considers this and considers moves that drastically change the evaluation of either side. More formally, on each move, the computer will find the most "reasonable" moves for each side and then calculate the best one out of those choices. It is a waste of resources and it is very computationally expensive to evaluate *every* position from the position above - even 15-25 moves out. This is also known as the *horizon effect*, where due to the intensive load on computing, say reasonable moves on the 26th move-out, the engine stops short on 25 and picks the best move (*Vuckovic et al. 2015*). Using this algorithm, this portion of the paper will be a close following of the implementation of a chess game that mimics the functions of the standard game of chess with an engine that primarily uses the minimax, move-generation, alpha-beta pruning, and move-generation concepts. I decided to develop this project alongside a colleague over the summer where the goal was to make a project that utilized our knowledge of the game we love to play: chess. For most games, there is a component of luck or randomness embedded into the game just by the nature of the developer implementation. However, this is different in chess. In chess, every player has what is considered in game theory *perfect information*: "A class of game in which players move alternately and each player is completely informed of previous moves." (*Wolfram MathWorld)* This works nicely with computers as being able to define possible information, the inputs and outputs can be operated on directly and large amounts of information can be extracted from the game state itself. The internship over the summer follows my colleague and me taking the knowledge through computer science and directly applying the computer's ability to calculate large amounts

of information in a small amount of time. This was done through minimax and alpha-beta pruning algorithms applied to move generation and board evaluation concepts. Additionally, this portion will be noting how humans get better at chess through positional and tactical play. This is closely linked to the way the mini-max algorithm works, though not entirely conscious in the brain.

Applied Influence of Chess Engines on Chess Players and Modern Chess Theory

Over the course of the past few decades, the performance of the computer has dramatically improved. Moore's law states that roughly every two years, the number of transistors on microchips will double. As the number of transistors in modern processors continues to increase, comes the improvement of software that can take advantage of this newfound processing power. Chess is such an example where software requires brute force to come to a meaningful evaluation of the game state, where a game state is simply defined as the positioning of the chess pieces. (the default board is one state, and 1. e4 is another game state). For a chess position to be evaluated, there needs to be a set of instructions on how the state should be evaluated. This is where the introduction of chess engines comes into play. The introduction of chess engines has changed the way that chess professionals have improved not only how they think about the game, but also they changed their play by analyzing engine recommendations. These professionals that can perform at the top level have been utilizing engine lines for many years now, but this wasn't always the case.

Chess is a game that is old as time, dating back to India in the sixth century. Coming from sending moves by paper and waiting days or weeks for the next move to being able to instantly see the evaluation of a position, chess has evolved a long way. Ever since the rules were defined, players have been constantly trying to improve their understanding of the game. At first, it was common knowledge that controlling the center was important. Then, it became common knowledge that it's not entirely the case that controlling the center is the most important aspect - it's about king safety and piece coordination (*Hsu, 1999)*. There is this constant back and forth in the community of being able to evaluate new ideas and develop new theory for the game.

This development of chess theory allows for the introduction of the computer which also developed massively in recent times, as those same players that were improving were using computers to analyze their games. Due to the astronomical numbers that chess move calculation requires, the performance of the chess engine is directly correlated to the performance of the development of the processing of the computer. The player base of chess is developing in tandem with the processing power and evaluation functions of computers (Chess.com). Humans traditionally have improved their play from previous strong players, whether that be GMs or very strong coaches. With the rapid development in computing power however, the player base is able to implement new tactical and positional ideas from chess engines. Better engine evaluation

functions lead to better ideas, which lead to stronger human play. These ideas, unsurprisingly, are more intricate and complex than any grandmaster could come up with which in turn push the boundary for the players themselves. This relationship shows how the introduction of technology can vastly transform the landscape of a community through the way the game is played and how people represent the game in their minds. Currently, this relationship is not documented to the extent that the community of chess has evolved in recent years. Admittedly, as seen from Sindhu (2020) and Levene (2005), there are studies done on the use cases of engines and artificial intelligence in chess, but almost always these studies are strictly focused on one particular aspect of the game (the opening stage, middlegame, endgame) or specific skills that are utilized such as pattern recognition or memorization.

There is a plentiful amount of information available about how grandmasters (and average players) perform under certain circumstances about the engine, but not much information about how the engine influences the play of those players. The main ways in which this research will be conducted is through a combination of primary and secondary evidence from current chess GMs and literature from the past. It will be necessary to reference earlier dated conclusions about engines and their uses with more modern takes. Additionally, the player base itself will have to be analyzed through the available data collected from the largest chess websites. Chess.com, one of the largest chess websites to date has an "insights" functionality that keeps track of all of its users metadata. This data collection has 3 primary audiences: the chess player base, more specifically the professionals that play at tournaments, and the chess engines that underlie the two previous audiences. There is no doubt that chess is a well-studied and documented game, and that the improvement of chess engines is impressive, but this section intends to document the potentially not-so-obvious nuances of this relationship and how it has transformed chess forever.

Conclusion

It is through the lens of the computer that I wish to walk through the development of the chess engine. It is important to understand the underlying mechanics of the tool that has allowed an exponential growth in the understanding of chess. Equally as important, it is through the lens of the player that I wish to analyze the computer-player relationship that has allowed the development of the engine and playerbase to flourish. The positive feedback loop of the computer influencing the chess player, and the chess player developing the engine is a way in which we can see society and technology grow together.

Sources:

a. *Levene, M., & Bar-Ilan, J. (2007). Comparing typical opening move choices made by humans and chess engines. The Computer Journal, 50(5), 567–573. https://doi.org/10.1093/comjnl/bxm025*
b. *Feng-Hsiung Hsu. (1999). IBM's Deep Blue Chess Grandmaster Chips. IEEE Micro, 19(2), 70–81. https://doi.org/10.1109/40.755469*
c. *Rahul, A. R., & Srinivasaraghavan, G. (2015). Phoenix: A self-optimizing chess engine. 2015 International Conference on Computational Intelligence and Communication Networks (CICN). https://doi.org/10.1109/cicn.2015.134*
d. *Luqman, H. M., & Zaffar, M. (2016). Chess brain and autonomous chess playing robotic system. 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC). https://doi.org/10.1109/icarsc.2016.27*
e. *Rydzewski, A., & Czarnul, P. (2017). A distributed system for conducting chess games in parallel. Procedia Computer Science, 119, 22–29. https://doi.org/10.1016/j.procs.2017.11.156*
f. *Liaqat, A., Sindhu, M. A., & Siddiqui, G. F. (2020). Metamorphic testing of an artificially intelligent chess game. IEEE Access, 8, 174179–174190. https://doi.org/10.1109/access.2020.3024929*
g. *Barnes, D. J., & Hernandez-Castro, J. (2015). On the limits of engine analysis for cheating detection in chess. Computers & Security, 48, 58–73. https://doi.org/10.1016/j.cose.2014.10.002*
h. *Vázquez-Fernández, E., Coello, C. A., & Troncoso, F. D. (2013). An evolutionary algorithm with a history mechanism for tuning a chess evaluation function. Applied Soft Computing, 13(7), 3234–3247. https://doi.org/10.1016/j.asoc.2013.02.015*
i. *Levene, M., & Bar-Ilan, J. (2005). Comparing move choices of chess search engines. ICGA Journal, 28(2), 67–76. https://doi.org/10.3233/icg-2005-28202*
j. *Maharaj, S., Polson, N., & Turk, A. (2022). Chess AI: Competing paradigms for machine intelligence. Entropy, 24(4), 550. https://doi.org/10.3390/e24040550*
k. *Perfect information. from Wolfram MathWorld. (n.d.). Retrieved December 11, 2022, from https://mathworld.wolfram.com/PerfectInformation.html#:~:text=A%20class%20of%20game%20in,one%20or%20more%20optimal%20strategies.*
l. *Team, C. (2019, May 7). Computer Chess Engines: A quick guide. Chess.com. Retrieved December 11, 2022, from https://www.chess.com/article/view/computer-chess-engines*