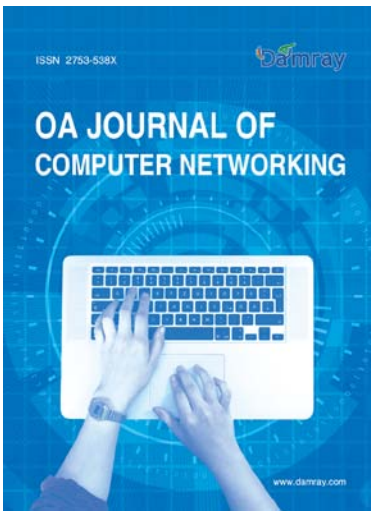
Damray | OA Journal of Computer Networking
2022, 1(2), 33-37
ISSN: 2753-538X

Combinatorial Games and Algorithmic Calculations Using Chess



https://oajcn.damray.com/

OPEN ACCESS

DOI: 10.26855/oajcn.2022.09.004
Received: September 28, 2022
Accepted: October 26, 2022
Published: November 25, 2022

Zhida Yin

Issaquah High School, Issaquah, 98027, WA, USA.

Abstract

Combinatorial games like chess and checkers are two-player strategy games like chess and checkers, containing simple moves that can result in many different variations. These variations have different strategic points to take note of, for example, the placement of the pieces. The abstract placement is the most important aspect of combinatorial games, as they mark the starting point for strategy. Games like chess have computer engines that calculate every single move and response to make a conclusion of the game's result. The computer engine uses unsupervised learning to analyze games and spot mistakes made by the players. It starts off with zero knowledge and over the course of millions of games played, it started to become the most powerful "player" in the history of chess, though many highly skilled chess players can easily beat a computer at the highest difficulty. This is a deep dive into the mind of a chess computer, the ways it thinks, processing, eliminating bad moves, and concluding the move it wants to make.

Keywords

Combinatorial games, Algorithmic calculations, Strategies

Strategies

There are many ways to approach the first move. The computer will think about strategic areas of the board like the center. At first there are only 20 possible moves for White (the player who goes first) and the same with Black (the player who goes after the first player). From one move alone from each side, there are already 400 distinct positions. The computer engine thinks about all of the moves and responses in a split of a second. It calculates the many different half-moves called a ply, using the formula P = 30^x, where x is the amount of half-moves and P is the number of distinct positions. In just 4 whole moves, there's already more than 600 billion possible positions, standalone an entire game (usually 30-50 moves), so how does a computer calculate over an almost infinite amount of moves and positions? The computer uses a "chess tree" to map out all good and bad moves. All bad moves lead to a game over, so the com-

puter maps out all the game-ending moves (there are only 2 game endings within the first 10 moves). Each move creates a branch, and off that branch, if the game will end in the next few moves, it becomes a “leaf”, and gives the definition of a variation in a chess position, for example, the Ruy Lopez position is a branch or the Center Game is also a branch. The entire tree is like a massive universe with a different move in every star inside a galaxy. That still doesn’t fully answer how a computer can calculate an absolute infinity amount of moves [1].

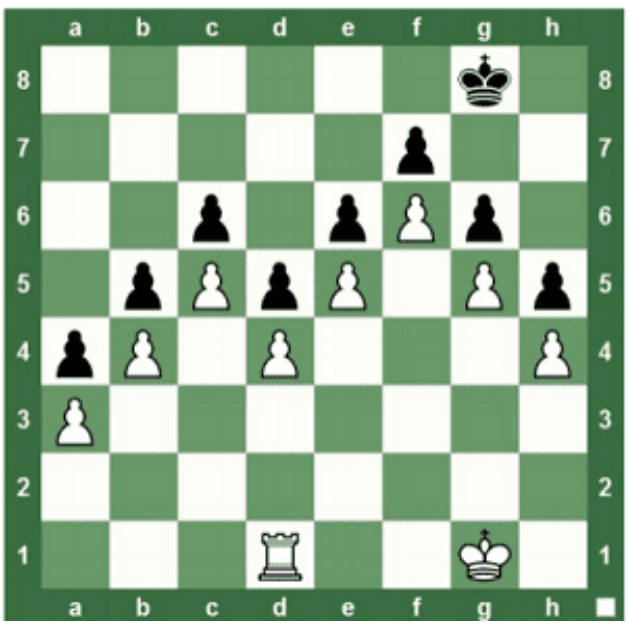
The computer sees the starting position, because the position will always be the same, the computer still does its usual evaluation habit. Using the formula b^ceil(n/2)+b^floor(n/2)-1 where b is the move per node and n is the depth, it can calculate the half-moves from the very beginning. The best and worst case scenarios are the times where the computer goes into the infinite universe and takes a visit each time, or narrowing down the best moves as better candidates.

The computer uses another method to see who has the advantage, this will narrow down the odds of thinking about every position and who is winning, plus, it can reduce time to create the chess engine. Grandmasters use simple algorithmic methods to rule out bad continuation moves with good, winning responses as candidate moves, but a computer can do it millions of times faster than a human can.

A computer uses different factors to use its powerful algorithm and evaluate which side is winning. It looks at the position, who has more pieces on the board, what responses each player will do if the player goes next will do. Then, the computer spits out a numerical value that shows how much winning potential a side has. If the move leads to a numerical drop (from +3 down to -1), the computer will no longer calculate the line from there as it leads to a dead end.

An engine cannot just go one way, it has to go the other way, which is why the computer will go to the end and evaluates backwards to see what gives the most gain in material and position. A person named Claude Shannon used this method to quickly and effectively calculate the result of a game, and is also used by every chess engine today. Because chess engines are so advanced today, the computers back in the 1970s and 80s are horrible at chess as they cannot calculate moves very well [2].

Chess engines are powerful, but there are ways to exploit the computer’s abilities to calculate a certain position, for example: this position.



White is up a full rook, which should give you a conclusion that white is winning, but due to the fact that the pawns are locked together, the chess engine gives up on calculating this position, it has no idea where and what to move, thus making this position not evaluable by a computer, but for a human brain, it is possible [1].

We know what a chess engine is, but do you know about what kind of chess engines are there? There are many types of chess engines, for example, the most famous engine: Stockfish. A chess engine like Stockfish evaluates chess position and gives out a score based on the material each side has and their strengths and weaknesses. The two most important things chess engines do is to evaluate and search for good moves. They look deeply into a position to see which side is better and what possible responses are the best.

There are other kinds of chess engines called neural networks which are more powerful and uses supervised learning

than unsupervised learning making them more powerful with the help of other chess theorists by giving it data. The neural networks need CPUs and GPUs to increase the speed of analysis [3].

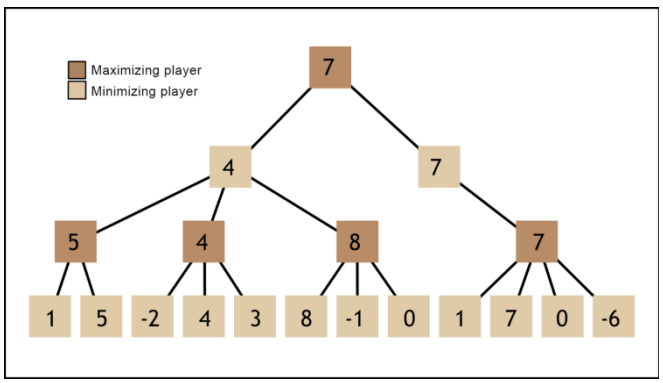
Stockfish is the most popular and famous out of all chess engines, but we know that all chess engines think differently, but ends up with the same or different results. The primary formula for chess engines are centipawns, or the advantage scoring. This is how computers conclude a position and which side is better. The reason why chess engines use centipawns are to easily show mistakes made by the players [4].

Verification

When you are using a chess engine, you will see something like +0.46 depth 23, the depth is how many plies or half-moves the computer has calculated into a certain position. A depth of 23 means the engine read 23 plies into the future and concluded that a player has an average centipawn of 0.46 for White. Stockfish usually sees 14-50 plies per move [2].

We know about the centipawn now, but how does a computer see an inaccuracy, mistake, or blunder? An inaccuracy counts as a loss of approximately 40 centipawns (0.4 points of material), a mistake as a loss of about 90 centipawns (0.9 points of material), and a blunder loses more than 2 points of material [5].

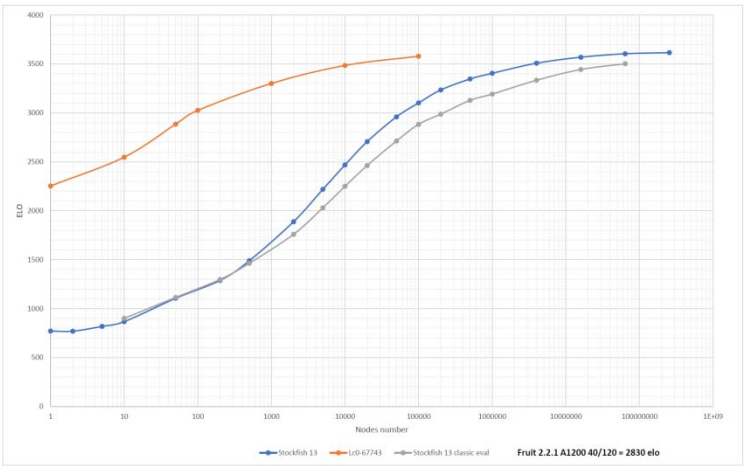
Stockfish is based on AlphaZero's best and most powerful engine: Lc0 (Leela Chess Zero), and is Stockfish's biggest rival to being the best engine in the world. Stockfish is number one on popularity, but Lc0 is the number one strongest. All these engines use hard-coded heuristics that's faster but harder search algorithm which is known as minimax search. Lc0 used this over months of rigorous training by Deepmind using large amounts of resources which made it the strongest and most likely the invincible chess engine [2] [3].

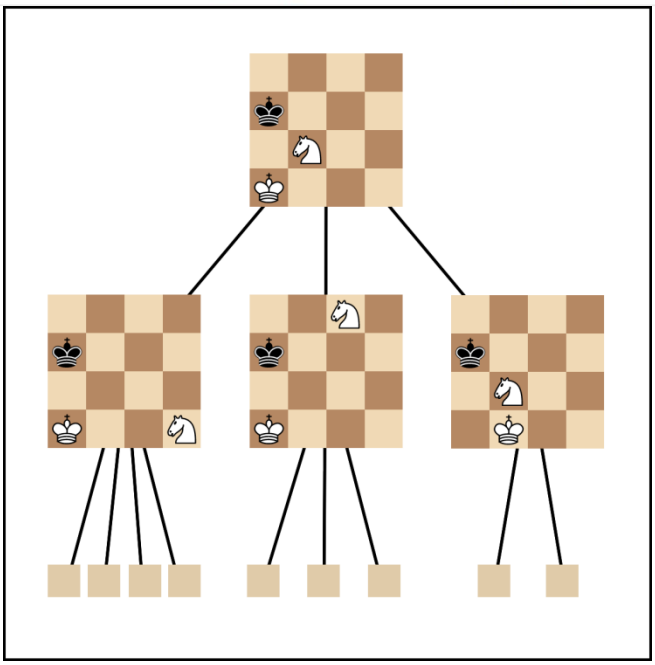


Chess engines don't play random moves, they find good moves, ones that maximize the odds of winning. The computer starts mindless, then by feeding it data, it understands the reasoning behind what moves and why they are good. They analyze the board to see the pieces on both sides, the positioning of said pieces, and tactical and strategic patterns. The way that minimax works is to split up moves for each possibility that it can be played.

Minimax takes advantage of the fact that chess is a zero-sum game. This is how chess engines maximize their victory advantages. It gives the engine different routes to take on the search tree to see which are the best and worst scores. It uses another method of minimax called null pruning, assuming that giving the opponent another move, if the opponent cannot make a comeback, the computer would conclude that the opponent has no good moves and is trapped in a zugzwang (meaning no good moves) and the computer reached another dead end. If the minimax algorithm is set free, it will have a 100% win rate, but it's very slow because there are almost an infinite amount of positions, which is why minimax uses the branching factor. It's fast, accurate, and can easily conclude the worst possible moves to save space for new moves [3]. After doing some analysis, the computer double-checks using the principal variation from an alpha-beta search (finding the best moves) and moves on if the move ends up being the best. The quickest conclusion a chess engine would make is max(a, b) = -max(a, b) [6].

After using the alpha-beta search, the computer goes into the final step of confirming the path: quiescence search, this further narrows down the possibilities of bad moves and looking at trapped pieces on the board. It concludes faster without checking some consequences, but it is can be overwritten using another alpha-beta search. A complex, but straight mathematical equation would be a function: f(x)=w1*f1(x)+w2*f2(x)+… up to w5*f5(x). With this function, the computer can easily evaluate the board based on the amount of pieces on it. f1(x)= number of white queens - black queens, f2(x)= number of white rooks - black rooks and so on. The w1 in the function shows the weight of the piece on the board, with w1 with 9 points, w2 with 5 points, w3 and w4 with 3 points, and w5 with 1 point [2].





Evaluating the board is only small part of the puzzle. The computer still needs to evaluate the game phase (how long the game lasted), if a player has both bishops still alive on the board, piece-square tables (piece positioning), how safe the king is, how many possible moves there are, the pawn structure, and the space that rooks can move [7].

Conclusion

Computers use many algorithms to go from a starting position to finishing a game in the best ways possible. Using many formulas and equations, computers uses the data that is given to them and make quick conclusions of the result of the game. The conclusions are made by narrowing a giant database with an almost infinite amount of positions and narrows it down to the best possible moves. They only move on if the move is good enough, while all bad moves are filtered out and ignored. Just note that not all positions can give a conclusion, the computer will "soft-lock" itself if the position has a limited amount of moves with many pieces cramped together.

References

[1] Hercules, Andrew. "How Does a Chess Engine Work? A Guide to How Computers Play Chess." *Hercules Chess*, 3 Mar. 2022, https://herculeschess.com/how-does-a-chess-engine-work/.

[2] Hercules, Andrew. "How to Beat the Computer in Chess? Win Everytime!" *Hercules Chess*, 17 Feb. 2021,

https://herculeschess.com/how-to-beat-the-computer-in-chess/.

[3] "Computer Chess Engines: A Quick Guide." *Chess.com*, Chess.com, 7 May 2019, https://www.chess.com/article/view/computer-chess-engines.

[4] "How to Use a Chess Engine? Your Quick & Effective Guide." *DecodeChess*, 16 Dec. 2021, https://decodechess.com/how-to-use-a-chess-engine-guide/.

[5] Walker, Levi. "The Anatomy of a Chess Ai." *Medium*, Medium, 21 Aug. 2020, https://medium.com/@SereneBiologist/the-anatomy-of-a-chess-ai-2087d0d565#:~:text=The%20strength%20of%20chess%20engines,as%20Monte%2DCarlo%20tree%20search.

[6] "Stockfish and LC0, Test at Different Number of Nodes." *MeloniMarcoit*, 29 Aug. 2021, https://www.melonimarco.it/en/2021/03/08/stockfish-and-lc0-test-at-different-number-of-nodes/.

[7] Cuenca, Pepe. "How Do Chess Engines Think?" *chess24.Com*, 24 May 2017, https://chess24.com/en/read/news/how-do-chess-engines-think.