Mathematical moves: How chess engines think

Introduction

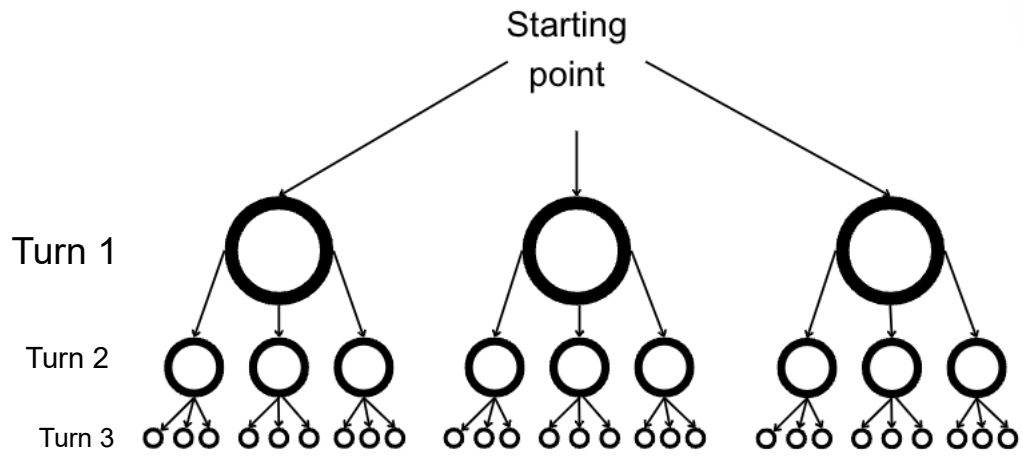
How many unique games of chess do you think are possible? Obviously quite a lot, but you might be surprised to find out that there are way more possible games of chess than there are atoms in the observable universe! In fact, the estimated 10^{80} atoms is trivially small compared to "The Shannon Number" of possible chess games, 10^{120}. While our modern computers are super-fast, if we asked them to calculate that many possibilities, our whole solar system would be long gone by the time it finished. Chess engines are what we call the computer programs that analyse chess games and calculate the best move.

11th May 1997. New York City. Gary Kasparov vs Deep Blue. This was the day that chess engines shocked the world. Kasparov, the unbeaten chess grandmaster represents humanity against the best chess computer of the time. After one win each, and three draws, Deep Blue won the final game after a stunning battle. This historic event marked the moment computers overtook humans at chess, and they have continued to rapidly evolve ever since. So, with the crazy number of possible outcomes, how did Deep Blue decided what were the best moves to clinch the win? Well, it's all thanks to the many mathematical algorithms and principles that they use to make calculations and decisions in the blink of an eye.

Exponential growth

On average, a player might have around 30 possible moves that are allowed. If the average chess match lasted about 80 turns, the possible number of unique games could be something like $30^{80} \approx 10^{120}$. Sound familiar? Yes, this is the very crude method that Claude Shannon used to estimate the number of possible chess games.

It is easy to visualise the increasing possibilities with a simplified "game tree" where we assume there are only three possible options at each stage.



Each circle would represent a different possible move. In theory, a perfect chess computer would be able to calculate to the bottom of the tree, look at all 10^{120} different outcomes, and decide based on which paths down the tree will lead to its victory. However, it has been calculated that the longest possible chess match could last as many as 11,800 moves[1], and with so many possible moves available on each turn, getting to the bottom of the tree is an impossible task. To get around this, the programs settle for building up the tree to the best of their abilities. The number of turns that it can calculate into the future is what we call "depth".

If our computer has a depth of, say, 10 turns, then it might not be likely that there is a checkmate within the horizon. It turns out that a checkmate isn't the only thing that the computer can look for; it is also looking for which moves will give it the best positional advantage. It does this by using what is called an evaluation function.

Evaluation function

An evaluation function is used to determine which player has an advantage in the position. A very simple example could look like this:

$$E(x) = N_c - N_o$$

Where N_c is the number of pieces the computer has, and N_o being the number of pieces the opponent has. This is the foundation idea that is built on by taking to account may other different aspects of the position, assigning them a numerical value, and calculating the evaluation score of

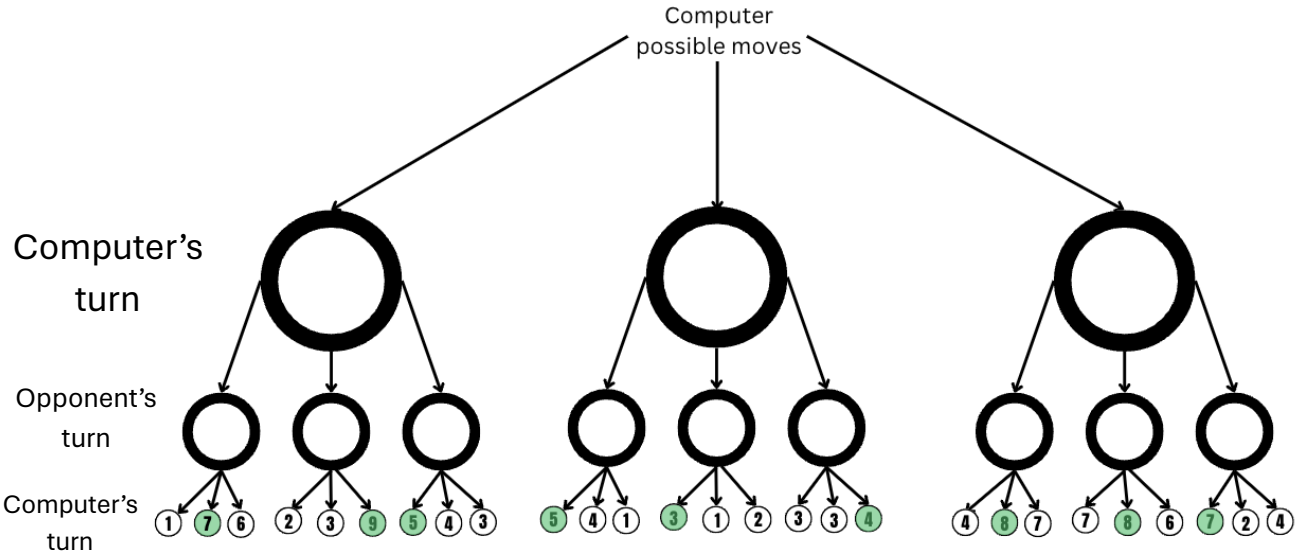
[1] It would not go on for ever because of the 50-move and the threefold repetition rule. The 50-move rule is that the game ends in a draw if no pieces have been taken and no pawns have moved in the last 50 moves. The threefold repetition rule states that a game ends in a draw if the same position on the board has been repeated three times. With this in mind, 11,800 is around the calculated maximum.

that position. For example, better pieces are weighted heavier than weaker pieces, with a queen assigned nine points while a pawn is worth one. It will also consider things such as king safety and pawn structure and will be given a weighting to determine how important it considers each factor.

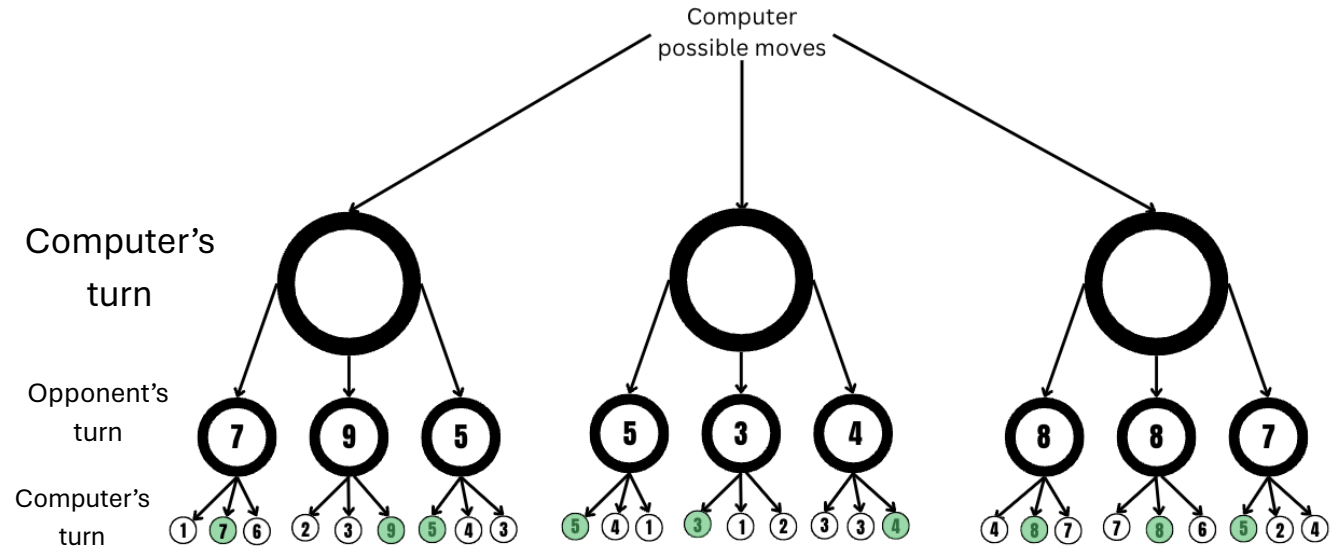
This is how the computer will assign a score to each position it has calculated on the game tree, and it will choose based on which position will lead to the greatest advantage.

Minimax algorithm

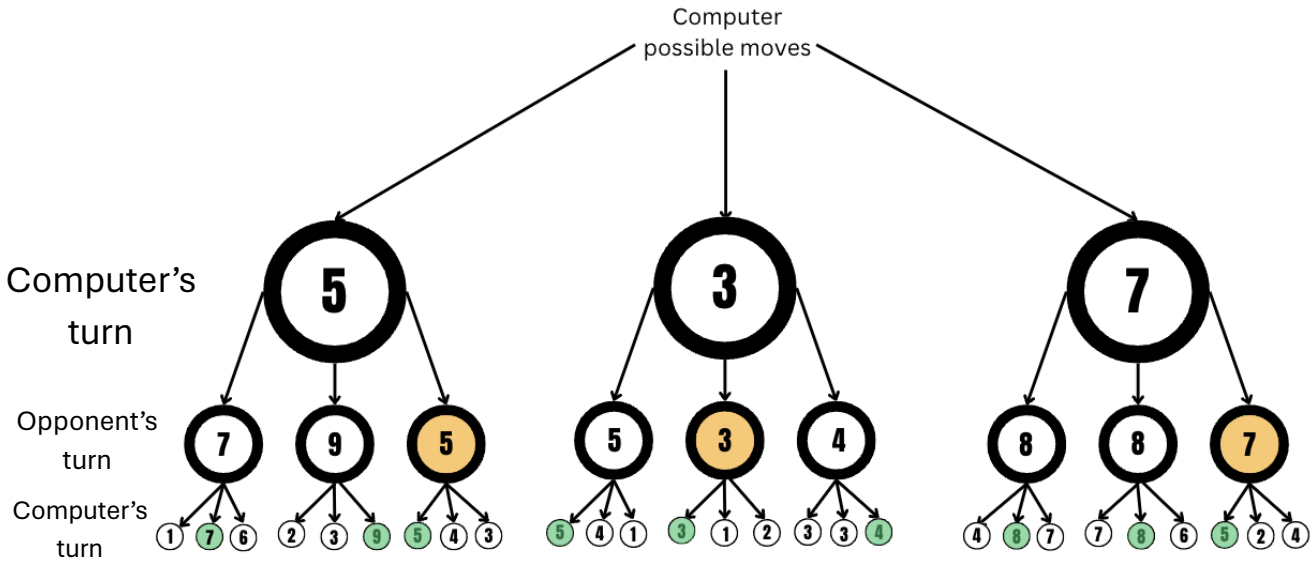
Because the computer doesn't know which move the opponent will make, it uses a mathematical "Minimax algorithm" to pick the best outcome. Consider the simple game tree below, where the computer has calculated the next 3 moves, and the evaluation score of the different outcomes. In this case, the higher the score, the better it is for the computer.



At the third stage, it is the computers turn again. So, the computer will want to have the best position then. As a result, it favours the options that have the maximum evaluation score and will assign that value to the stage above.



Now, looking at the second level, the outcome is decided by the opponent because it is their turn. The computer assumes that they will pick the move that is best for them, which be them selecting the minimum value, which will then get put in the stage above.



From this, the computer will now select the maximum for its turn, which would option that leads to 7. By alternating between selecting the maximum for its turns and the minimum for the opponents turns, it can assume that the opponent will play the best moves for them, and therefore not aim for an outcome that relies on the opponent making a mistake.

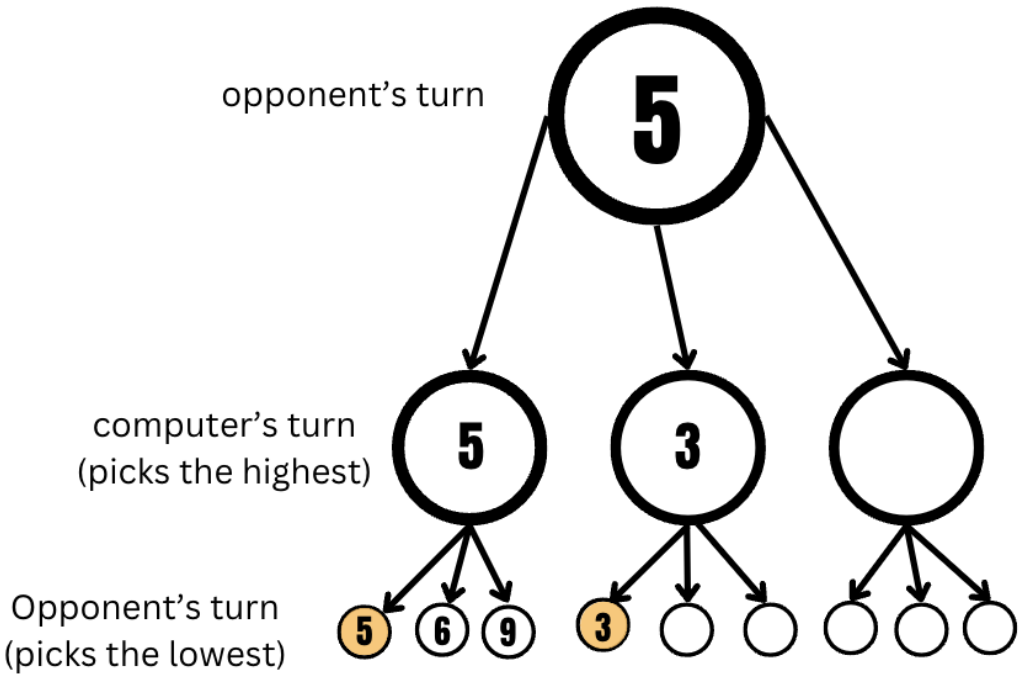
Although this can now actually get the computer to make a move within the lifetime of our planet, you may have noticed that brute-forcing through all

of these possibilities is still not very efficient, and you'd be correct. That's why minimax is accompanied by a technique called alpha-beta pruning.

Alpha-beta pruning

This technique can help the algorithm run twice as fast and require a lot less memory. It works by cutting down on unnecessary calculations by using logic to 'prune' branches of the game tree.

In the situation below, the opponent has played a move that leaves the evaluation at 5, and the computer is calculating what move to play. It has calculated the branch on the left, finding a 5, 6, and 9. Knowing that the opponent will pick the minimum, the computer puts a 5 in the stage above for this branch, and starts calculating the middle branch.



It finds a 3. Therefore, it knows for sure that the stage above it will be 3 or lower, as the opponent is picking the minimum values, so it can fill it in.

Since the computer has already found a branch with 5, and already knows that the second branch will be ≤ 3, it doesn't need to waste time calculating the other possibilities on the middle branch and can skip to the right branch instead, saving time and allowing the more promising branches to be explored deeper.

Conclusion

Back in 1997, Deep Blue was a unique, purpose-built supercomputer, and had a typical depth of 6-8. Nowadays, we can load up chess programs like stockfish on our laptops or phones with a depth of 40+. Minimax and alpha beta pruning are just some examples of the mathematical algorithms that have helped develop engines and make them more efficient. Chess engines have changed the game forever but have also had much wider influences too. They represent one of the earliest applications of AI, and the algorithms behind them have inspired new approaches for optimising intelligent decision-making programs.