Stockfish

- A study of the underlying AI techniques -

Authors:
H. Bjørgan, L. Folkestad, M. Klev

Professor:
J. Salceda

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Universitat Politécninca de Cataluya
Facultat d'Informàtica

Contents

1. *Introduction*

The following document reports the findings of our dive into the technicalities of Stockfish, an open source chess engine. We study the different techniques of artificial intelligence used in the engine and the impact these have on its commercial use. Furthermore we study the impact Stockfish, and its daring ventures into the realm of artificial intelligence, has had on the chess community and players over the last decade.

The backdrop for our choice of topic is this fall's World Chess Championship event between Magnus Carlsen (Norway) and Fabiano Caruana (USA). In Norway chess as a sport has gotten a lot of attention since Magnus Carlsen became the World Chess Champion back in 2013.[8] With the rising interest in chess followed a proportionately increasing interest in creating the strongest chess enigine.

Figure 1.1: Stockfish' logo

2. *Product description*

This chapter will give you a brief introduction of chess and computer chess. Although the end goal of both fields is the same - to create an agent, human or not, that excels at the game of chess - the approach to master each field is vastly different. Furthermore this chapter will introduce Stockfish, one of the strongest chess engines over the last decade.

2.1 The game of chess

Figure 2.1: A chess board at starting position. Source: [4]

As illustrated by the image in figure 2.1, chess is a board game played between two players.

2.1.1 The history of chess

The first credible references of chess date back to 600 A.D, making chess at least 1600 years old. From India and Persia the game made its way to Spain in the 9th century, and by the 11th century the game was widespread in Europe. Fast forward to the 15th century the game as we know it today was born, as the rules of how pieces were allowed to move were changed towards the end of the century.[1]

In the 19th century the first acknowledged World Chess Championship was arranged. With that, one could also begin to compare world champions[25] against one another, by evaluating how many of the moves they made were objectively the best ones. Although a controversial method of comparison, it has shown that the most recent world champions are objectively stronger players than the former world champions. With this in mind one can safely assert that humans have not yet reached their maximum chess playing potential.

Over time chess has become a household game, and although it is hard to pinpoint exactly how many people play the game, estimations suggest somewhere around 600 million people play the game on different levels.[13]

2.1.2 The rules of chess

Chess is played on a 8x8 board between two players. At the start of the game each player has eight pawns, two rooks, two knights, two bishops, a queen and a king. The goal of the game is to move your pieces around in such a way that you end up placing your opponents king in checkmate.

The game always starts by a move from the player with the white pieces. The players then alternate between making moves, one after the other. Each type of piece has a predetermined set of legal moves. In addition each piece has a predetermined set of squares they threaten, based on their position on the board. These squares are mostly the same as the ones the pieces are allowed to move to. A rook for instance is allowed to move as far as it wants to either side or forwards/backwards, as long as it does not go through a piece. When a player has placed his opponent's king under a threat the opponent cannot legally move his pieces to get out of, you have placed your opponent in a checkmate, and you have won the game. The players can also agree upon a draw if they think they have no winning chances without risking to lose. The game can also end by a stalemate, which means that a players king is not under a threat, but he has no legal moves. The full rules of chess are easily available online, and can be found here.[14]

2.2 Computer chess

When talking about computer chess it is important to distinguish between software and hardware. That is respectively the program playing the game and choosing moves, and the actual circuitry the program runs on. The software can be thought of as the brain of a chess computer, and is referred to as a *chess engine*. When one talks about chess computers nowadays, one usually refers to the software, but one would be remiss not to acknowledge that the hardware is a significant part of chess computers. A bad chess engine will outperform a good chess engine if the hardware limitations of the good engine are strict enough. With this in mind the quest becomes to find the best engine, given equal hardware limitations. Thus our interests lie in exploring the software of chess computers.

2.2.1 History of computer chess

Since the birth of artificial intelligence in the 1950s[15] the possibility of creating an artificially intelligent chess playing agent has been discussed. While the notion that a computer program would outplay the best humans was unthinkable at first, it became evident that this was in fact possible, largely due to improvements of hardware following Moore's law.[12]

1997 marked a turning point in the history of computer chess, as the world champion at the time, Garry Kasparov, was defeated by IBM's Deep Blue.[10] Since then the strongest chess computers have continued to improve, both with respect to software and hardware, making them way stronger than the best human players. At the time of writing Stockfish has an Elo[6] rating of around 3400[23], depending on the hardware, which is around 600 rating points more than the current world champion Magnus Carlsen. A rating difference of that many points renders the chance of Magnus beating Stockfish virtually nonexistent.

2.2.2 Approach to playing chess

From a scientific point of view one can say that chess is a finite and solvable game. It is theoretically possible to calculate all legal moves, and, if one exists, find a perfect way to play that guarantees you victory. If you picture a game of chess as a tree with the initial position as the root node, one can continue to expand the tree for every move made. It does however not take long to realize that this approach yields an unmanageable amount of positions for your program to handle. With an average branching factor of around 35, the tree would quickly become huge. Shannon's number[17] is a conservative lower bound of the number of nodes in a tree representing chess. Its numeric value 10^{120}, is unfathomably large, and rules out the option of brute forcing chess from start to finish.

Seeing as brute forcing is not an option, programmers have adapted different approaches to narrowing down the number of branches their chess engine has to explore. By implementing unique evaluation functions, search techniques and heuristics, they are able to concentrate the computing power they have on hand more effectively than a pure brute forcing technique. One can say that the search process has been made more intelligent, and in the following sections we will indeed explore these techniques.

The differences in techniques and evaluations have resulted in several different chess engines, with Stockfish being among the leading ones.

2.3 Stockfish

As previously mentioned Stockfish is a chess engine. In other words it is a computer program that takes a board representation as an input, and outputs a ply. "Ply" is a term in computer chess describing a players turn or a half-move. In other words two ply make up one move. For instance at the start of a chess game, when each player has moved one piece, that particular game has seen two ply and one move. In mathematical notation Stockfish can be described as $f(b) \rightarrow p$ where b is the board representation and p is the suggested ply.

Figure 2.2: An example of Stockfish implemented in a chess UI. It is white to move and the red arrows are the suggestions from Stockfish of the most beneficial ply for white. On the right you can see Stockfish' evaluation for white's possible moves.[16]

Stockfish saw the light of day in 2008, and was a modification of a preceding chess engine - *Glaurung*. It has continuously evolved through contributions from the computer chess programming community.[20] At the time of writing Stockfish 9 is available, performing at Elo 3438 on a 64-bit quad core CPU. This narrowly places Stockfish as the currently highest rated chess engine.[23] It is important to note that Google's Alpha Zero has not yet received an official rating, and therefore is not taken into account.

The product Stockfish offers is itself - a highly competitive chess engine that outperforms all humans and most other chess engines.

Stockfish is available for download to most platforms, and is released under the GPL license. This means that you are free to use and alter the source code as you so please, even in your own commercial products, with very few restrictions.

3. *Techniques of Artificial Intelligence*

In order to understand how Stockfish is able to play chess at such a high level, we must try to grasp the underlying techniques the engine uses for choosing its next move. We highlight the techniques we deem most significant for Stockfish' performance, but do not make the mistake of believing that this is a comprehensive list.

3.1 The minimax algorithm

The minimax algorithm lies as a foundation of most chess engines, including Stockfish. It lies as a foundation of many turn-based two player games, in fact. It utilizes the concept of zero-sum games, which, as the name suggests, are games in which the total score always adds up to zero. In chess for instance, a player can lead by three pawns if and only if the opposing player is down three pawns. The zero-sum property renders the minimax algorithm an effective decision-making tool for making the most advantageous choice.

The algorithm calls one player the maximizer and the other the minimizer. They have the corresponding goals of maximizing and minimizing the score. The challenge minimax solves is that of choosing the optimal choice for yourself, given the fact that you know your opponent will attempt to counteract your choices.

The game tree in figure 3.1 shows us how the minimax algorithm makes it's decisions. The algorithm explores nodes down to the leaves or an otherwise predetermined depth. Once these are explored it will maximize or minimize, depending on which depth it is on, and pass its value upwards towards the root node. In the figure you can see the maximizing player being at depth zero, the root node, and having to choose between two paths. In order to determine the best choice a minimax algorithm probes the game tree. At depth one, it would be the minimizing players turn, hence the nodes at this depth minimize the values from its children at depth two, which have maximized the values of its children at depth three, which have minimized the values of its children at depth four. In the end, the maximizer at our root node is left with a choice between -10 and -7, which he of course maximizes, and chooses the rightmost path of the game tree.

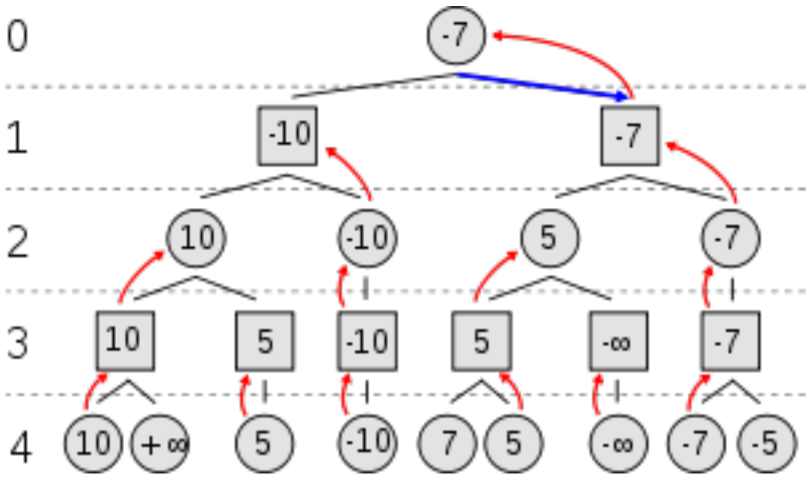

Figure 3.1: An example of a game tree using the minimax algorithm. Source:[19]

The pseudocode below illustrates how the minimax alogrithm might be implemented. [19]

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := -infinity
        for each child of node do
            value := max(value, minimax(child, depth - 1, FALSE))
        return value
    else (* minimizing player *)
        value := +infinity
        for each child of node do
            value := min(value, minimax(child, depth - 1, TRUE))
        return value
```

3.2 Evaluation functions

As stated in the previous section, the minimax algorithm makes choices based on a score give to each node. But where do we get that score from?

A common way of scoring nodes is through the use of heuristic evaluation functions. The aim of such a function is to give an estimation of the utility at a given state. If you manage to give precise estimations, these will guide your minimax algorithm towards the optimal choice. A normal approach is to use a weighted linear function such as

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

[15]

where different properties of the state are multiplied by a predetermined weight and summed up to produce the final heuristic value. Examples of properties can be to have two x'es next to each other on a board of tic tac toe, or to have a queen on the board in a game of chess.

Evaluation functions come especially in handy when your algorithm, due to lack of time or rescources, is unable to calculate to all terminal states. That is, it is not able to determine whether or not the choice will lead to a win, draw or loss. The evaluation function still scores reached state, giving the algorithm an indication of of how likely an outcome is at the state it reached.

3.3 Alpha-beta pruning

When looking at an arbitrary game tree the minimax algorithm is about to explore, it becomes evident that not all of its branches will affect the outcome of the probe. This is the property the concept of alpha-Beta pruning makes use of. Consider the following figure 3.2:

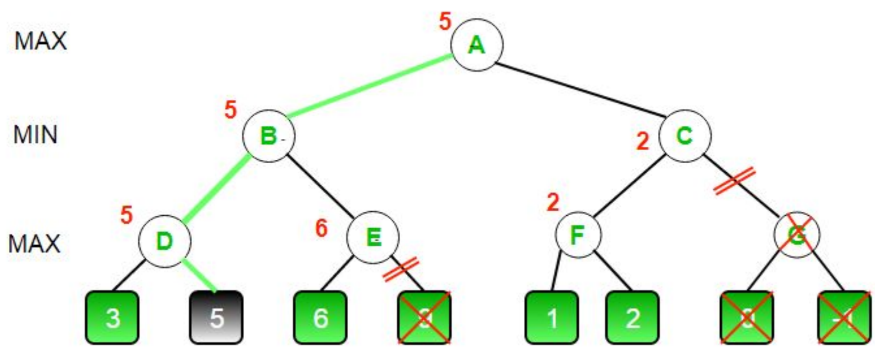

Figure 3.2: Example of alpha-beta pruning

If we imagine following the path of our probe down the game tree we venture from down the leftmost branch of the tree and calculate the value, using our evaluation function, of the children of node D. D will maximize the value and thus passes the value 5 to its parent B. We know that B will want to minimize the value of its children. When we then therefore venture down to E and explore its children we can stop exploring after the first child, which returns six. The reason being that we know E will maximize the value of its children, and thus returning a number no less than 6. B will therefore never choose E making it a waste of resources to explore all of its children.

In order to implement the alpha-beta pruning we add two new variables to our minimax algorithm, and the corresponding logic to check whether or not we can prune. This leaves us with this algorithm: [18]

```
function alphabeta(node, depth, alpha, beta, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := -infinity
        for each child of node do
            value := max(value, alphabeta(child, depth - 1, alpha, beta, FALSE))
            alpha := max(alpha, value)
            if alpha >= beta then
                break
        return value
    else
        value := +infinity
        for each child of node do
            value := min(value, alphabeta(child, depth - 1, alpha, beta, TRUE))
            beta := min(beta, value)
            if alpha >= beta then
                break
        return value
```

It is important to note that the effectiveness of alpha-beta pruning is strongly dependent on the ordering of the choices. If the presumed good choices are explored first, the probability of our algorithm pruning away parts of the game tree increases drastically. In theory a minimax search tree can be reduced down to $O(b^{m/2})$ of nodes to explore, compared to the normal $O(b^m)$. In other words a good ordering can significantly increase the search's performance. Using figure 3.2 as an example, one can see that the rightmost part of the tree would not have been pruned away, had it been explored before the left branch of the tree.

3.4 Iterative deepening

Iterative deepening depth-first search is a search technique that combines the low memory demands of a depth-first search and the sideways exploration of a breadth-first search. As the name implies, the search is performed by exploring one layer of the search tree at each iteration, until the search reaches a goal state, or a predetermined depth.

4. *Stockfish' Use of Artificial Intelligence*

The artificial intelligence of Stockfish can be thought of as two separate components: The search component and the evaluation component. The search component tries to maneuver the search tree as effectively as possible. The evaluation component tries to predict the likelihood of an outcome based on a static position. Each of these are essential to the final product of Stockfish, but differ greatly in their techniques.

4.1 Search

As previously mentioned, chess is a zero-sum game, meaning that if one player is loosing, the other must be winning. With this information it is safe to assert that chess is a perfect game for the minimax algorithm, which Stockfish has implemented in its program. Actually Stockfish uses a slightly simplified version of the minimax algorithm, called negamax [2], which makes use of the fact that $max(a, b) == -min(-a, -b)$. This does not change the logic of the algorithm, but simplifies the code a little bit. The implemented negamax-algorithm of Stockfish does indeed include alpha beta pruning as well.

Stockfish assigns the role of the maximizing player to white and the minimizing player to black. For every ply it then does a probe down the game tree as far down as time and hardware restrictions allow it. Iterative deepening is here a key concept, as this technique is used for the probe down the game tree. This allows for the search to iteratively update the move it so far has found to be most promising, which ensures that the engine will provide the user with the best ply, under varying time and hardware restrictions. Figure 4.1 shows the result of an analysis performed by Stockfish. The number on the left hand side indicate the depth of the search performed. As one can see from the suggested line, the recommended ply changes as the search reaches deeper down the tree.

The technique of iterative deepening serves another important purpose in Stockfish' search algorithm. By searching one level of the tree at a time, it can decide which of the moves are the most promising. This information is then used for the consecutive search on the next level, making sure the most promising moves are explored first. This allows for the alpha-beta pruning to be much more efficient than it would be if the moves were explored in random order. In practice this is implemented by storing the most promising moves in a transposition table, and looking them up once the new iteration has begun.

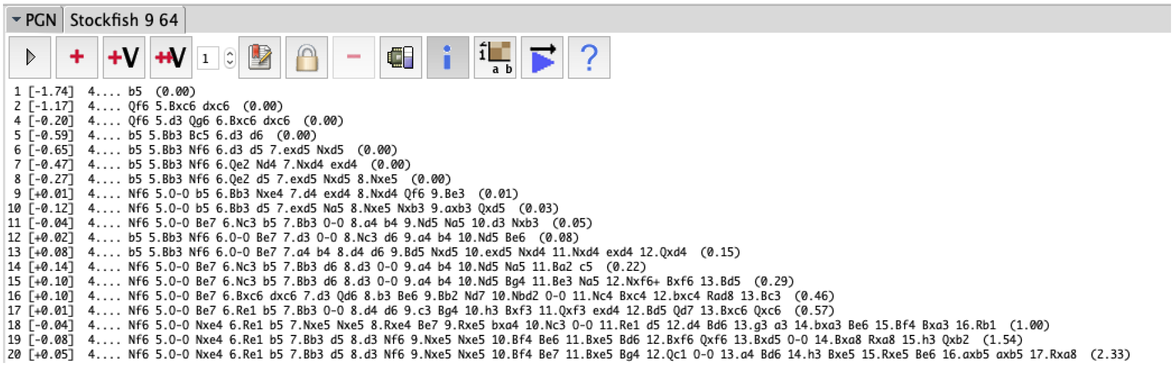

Figure 4.1: The result of iterative deepening. From left to right: Depth, evaluation, move number, suggested line, calculation time. Source: [5]

4.2 Evaluation

In order to properly guide the search, Stockfish must be able to perform a precise static evaluation of the position that each node of the game tree represents. In order to perform this evaluation Stockfish considers several factors of the position. The number of pieces, and which type of pieces each player has in a given position. The placement of these pieces relative to each other. How safe the king is. How mobile the pieces potentially can be. The list of properties to consider is quite extensive.

The following excerpt is from Stockfish' source code and is an evaluation table for knights in the middle- and endgame.[7] The table is set from white's perspective and the columns represent file A-D on a chess board. The scores for E-H are symmetrical, and therefore unnecessary to write out.

```
  { // Knight
  { S(-169,-105), S(-96,-74), S(-80,-46), S(-79,-18) },
  { S( -79, -70), S(-39,-56), S(-24,-15), S( -9,  6) },
  { S( -64, -38), S(-20,-33), S(  4, -5), S( 19, 27) },
  { S( -28, -36), S(  5,  0), S( 41, 13), S( 47, 34) },
  { S( -29, -41), S( 13,-20), S( 42,  4), S( 52, 35) },
  { S( -11, -51), S( 28,-38), S( 63,-17), S( 55, 19) },
  { S( -67, -64), S(-21,-45), S(  6,-37), S( 37, 16) },
  { S(-200, -98), S(-80,-89), S(-53,-53), S(-32,-16) }
 },
```

According to the table if white has a knight on the square D4 in the middle game, this contributes positive 52 to his score. On the other hand if a knight is placed on H8 during the endgame, it contributes with a score of -105. This table is only one of several that is implemented in Stockfish and that contribute to the final evaluation function. The approach to all the evaluation criteria is the same: A given property is assigned a numeric value based on a table. All the numeric values are summed up, and then black's total value is subtracted from white's total value. The final number indicates who has the objectively better position. A negative value indicates black, while a positive value indicates white.

5. *Stockfish and Innovation*

As mentioned in previous chapters Stockfish makes use of several different techniques of artificial intelligence. These techniques themselves are not something one would consider innovative, as they have existed prior to Stockfish' implementation of them. One could however argue that the way in which these techniques are incorporated into the engine is very innovative. With Stockfish being an open source project, everyone who wishes can attempt to improve the engine by tweaking the source code and suggesting improvements. This crowd funding of knowledge has resulted in the very specific combination of techniques found within Stockfish, and have proven to outperform all other chess engines, commercial or not.

One of the major leaps forward Stockfish has made in recent years came in 2013, with the implementation of Fishtest. [9] This boosted Stockfish' rating 120 ELO points the following year. Fishtest is a distributed testing network where everyone can donate CPU time to run suggested improvements to the engine. The system then plays the suggested version against the current version to determine whether or not they are actually improvements. While Fishtest itself is not a technique of artificial intelligence, it greatly helps the developers determine which of the techniques they use that are effective.

In summary one can say that Stockfish is innovative in its method of developing the engine, more so than the actual techniques of artificial intelligence themselves.

6. *Business Impact*

As Stockfish is not a commercialized business, the impact of its product differs somewhat from what one normally would see in a commercialized business. With this being said, it is still relevant to analyze how the product affects the community behind Stockfish and the market it operates in.

6.1 Benefits

The most prevalent benefit Stockfish gains from its way of development is a vast pool of knowledge. Where ordinary businesses are normally confined to hired intellect, Stockfish as an open source project, is confined only by the level at which people are willing to participate. This gives Stockfish a potential that is much greater than other commercialized competitors.

With people from all over the world, 120 at the time of writing [20], having contributed into the official repository for Stockfish, it has benefited from the variety of competence people posses. It is worth to note that these 120 contributors are only the number of people who have actually managed to improve the engine with their suggested altercations.

Regarding the techniques of artificial intelligence, Stockfish as a organization would be useless without them. Stockfish' engine is the base of all of the applications Stockfish as a organization offer. It is therefore an understatement to say that the use of these techniques benefit the organization, as the organization itself would have never existed if it were not for them.

6.2 Risks

While the open source code of Stockfish allows anyone to contribute to the chess engine, it also allows anyone to copy it and use to their own liking. This is also legal as per the license under which Stockfish is released. With Stockfish being a non-profit organization, this is not really a big issue. However if Stockfish ever were to become commercialized this would likely be a deal breaker.

Furthermore does the openness allow for intellectual theft. While the code for Stockfish is open for public use, it is not open for anyone to take credit for the code. With the model as it is today, this is solely based on a gentleman's code between programmers online.

Stockfish' main strength could also be one of the biggest risks for the organization. It is strictly dependant on voluntary work from strangers on the internet. Should the engagement from the online community cease to exist Stockfish would not have a leg to stand on. Unlike commercial businesses that save for a rainy day during good times, Stockfish lives and dies with the trends of its time.

6.3 Market position

By continuously being among the highest rated chess engines for over a decade, Stockfish has come to be the "king" of chess engines. While it's dominance is currently being challenged by AlphaZero, Stockfish is so well established that it will remain in fashion even if it has to concede the title of strongest engine. AlphaZero is a part of proprietary reserach and development at Google, and there have been no indications that the code and techniques behind AlphaZero will be accessible to the public in the same way that Stockfish is.

As testament to Stockfish' market position the two largest online chess pages, chess.com and lichess.org, both use Stockfish as their engine when providing their users with game analysis and other computer assisted features.[21] [22].

7. *Social Impact*

7.1 Users

When Kasparov was beaten by Deep Blue in 1997 [10] chess players realized that they might actually benefit from studying so called "computer lines" in chess. They are characterized by moves that a human normally would not be able to calculate themselves, and are the closest one can come to a sequence of objective best moves in a game of chess. With the rise of the chess engines and their increased availability, specifically through Stockfish, one began to see a shift in how chess was being played. While games prior to the chess engine era were about being the best players in the moment, the games of today are also about pushing your opponent out of their computer preparations. If you are unable to do so, you virtually end up playing a computer rated several hundred ELO points higher than yourself for the first moves, possibly giving your opponent an edge you will never recover from for the rest of the game. It is not uncommon for the Grandmasters [24] of chess to have memorized 15-20 moves ahead of the most common lines during computer preparation for big games.

7.2 Society

With the release of Stockfish, suddenly everyone had access to a quality chess engine for free. This has affected the chess community in several ways. One of which is that it allows average Joes to follow and understand the moves Grandmasters play during their games, as the engine explains the line and the position it leads in to. During coverage of chess events nowadays, it is quite common for the broadcasters to include a computer's evaluation (usually running Stockfish) of the position on the board, in the same way as the score is displayed during a football game. This has helped chess become a more spectator friendly sport.

It has become evident the later years that in order to perform as a chess player on a high level, you need to include engines such as Stockfish into your preparations and to develop your skills. Magnus Carlsen is said to be the first product of this generation of chess players, playing more like a computer than any player before him. [11].

While the moves the players make objectively are better and the players are displaying more precise chess than ever before, some argue that engines like Stockfish have ruined the "brain vs brain" element of chess that previously was all important in the world of chess. There have therefore been discussions of including alternative types of chess games such as chess960 [3] to minimize the possibility for the players to prepare. Thus recreating the "brain vs brain" scenario for both the players and spectators. While some online tournaments such ass chess.com's Speed Chess Championship have experimented with the game type, it is a long way from being a reality in any FIDE-official tournament.

Bibliography

[1] *Chess history*. https://simple.wikipedia.org/wiki/History_of_chess.

[2] *Chess programming wiki negamax*. https://www.chessprogramming.org/Negamax.

[3] *chess960*. https://en.wikipedia.org/wiki/Chess960.

[4] *Chessboard image*. http://www.stickpng.com/cat/objects/chess?page=1.

[5] *Desktop interface using Stockfish*. http://scidvspc.sourceforge.net/.

[6] *Elo rating*. https://en.wikipedia.org/wiki/Elo_rating_system.

[7] *Evaluation table*. https://github.com/official-stockfish/Stockfish/blob/master/src/psqt.cpp.

[8] FIDE. *World Chess Championship 2013*. http://chennai2013.fide.com/.

[9] *Fishtest*. https://en.wikipedia.org/wiki/Stockfish_(chess)#Fishtest.

[10] *History of computer chess*. https://thebestschools.org/magazine/brief-history-of-computer-chess/.

[11] *How computers changed chess*. http://theconversation.com/how-computers-changed-chess-20772.

[12] *Moore's law*. http://www.mooreslaw.org/.

[13] *Number of chess players*. https://www.fide.com/component/content/article/1-fide-news/6376-agon-releases-new-chess-player-statistics-from-yougov.html.

[14] *Rules of chess*. https://www.chess.com/learn-how-to-play-chess.

[15] Stuart Russel and Peter Norvig. *Introduction to Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey 07458: Pearson Education, Inc, 2010.

[16] *Sesse Stockfish implementation*. analysis.sesse.net.

[17] *Shannons number*. http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf.

[18] *Source of alpha-beta pruning pseudocode*. https://en.wikipedia.org/wiki/Alpha-beta_pruning.

[19] *Source of minimax image*. https://en.wikipedia.org/wiki/Minimax.

[20] *Stockfish official repo*. https://github.com/official-stockfish/Stockfish.

[21] *Stockfish on chess.com*. https://www.chess.com/analysis-board-editor.

[22] *Stockfish on lichess.org*. https://lichess.org/analysis.

[23] *Strength of chess engines*. https://www.ichess.net/blog/best-chess-engines/.

[24] *What is a chess Grandmaster?* https://en.wikipedia.org/wiki/Grandmaster_(chess).

[25] *World Chess Champions over time*. https://www.thesprucecrafts.com/world-chess-champions-611274.