Game-Theoretical Analysis of Chess: The Sixth Game Between Kasparov vs. Deep Blue, 1997

Ayça Begüm Taşçioğlu, ID:2020531
Game Theory, University of Padova
Submitted to: Leonardo Badia
January 29, 2020

Abstract

Chess is a dynamic game, involves some players moving first and others moving later [1]. Every player is assumed to have knowledge of the rules, the whole play involves on the principle that there is no hidden information; each player has full awareness of component's move [2]. This paper will focus on the game-theoretical analysis of chess, and a historical scenario will be investigated: the sixth chess game between Kasparov and Deep Blue, which is played in 1997. General information about chess, analysis of chess from a game-theoretical perspective will be included in this paper. Decision algorithms that were used by Deep Blue version 1997 will be discussed and will be compared with the Stockfish chess engine. Further, simulating the sixth game between Kasparov and Deep Blue in 1997 with Stockfish chess engine will be discussed.

Introduction

The setup of the chessboard and initial state is the same for each game. The chessboard is an 8x8 board, includes a total of 64 squares, 32 white, and 32 black squares. There are 16 pieces for both players; 6 pieces of these total have unique movements which are also included in the general rules of chess. According to Zermelo's Theorem, the game should be finite; even though there are many strategies, each player has a finite amount of moves that can be performed in a given chess board state [3]. Assuming that chess is a finite game allows analyzing the game.

Players of chess perform sequential moves, which are allowed in the given state. Players are assumed to know the rules and how the pieces move, all moves are open, players of chess are assumed to have full awareness during the game hence, players are assumed to have perfect information; one person's defeat is others' winning, in that sense chess can also be defined as a zero-sum game [4]. For instance, player 1 moves and then, player 2 knows which piece player 1 moved to which square location on the chessboard, after that player 2 moves; the game continues sequentially until it reaches one of the end states.

A chess game can be ended as a win, draw, or loose state for players. If a player's end state is "win", then the rival ends with a "loose". However, the result can be also a draw for both players. The complexity of chess also occurs because of that; Elkies states that combinatorial game theory is not proper to apply to a chess game since the result of the chess game can be independent of the player who makes the last move; a game can result in a win, end or drawn state [5]. Also, dividing a game state into sub-games is difficult and this tactic may end in a false result since pieces of the chess do not have the same effect on the board and game state [5].

Furthermore, since chess is a game between two people, the analysis should focus on the rival's move too. Because of this, the minimax optimization approach is used to build a search tree; we also need the rival's strategy to maximize our utilization.

In this paper, a specific scenario of a historical chess game will be discussed with focusing on the following question: "Can we alternate the history of the sixth game between Kasparov and Deep Blue, which is played in 1997 by using game theory?". For this study, this paper will investigate Stockfish chess engine to determine the best move and how Stockfish and Deep Blue work; the algorithms such as minimax algorithm, alpha-beta pruning and decision making systems will be discussed and analyzed.

It can be observed that even different pieces have different values, the total value in the middle stages of the game does not represent which player is close to winning. To exemplify, a player who took his/her rival's queen may take advantage as counting points yet, it is not a shred of evidence for this player will win the game; s/he can be loose even the next stage after taking a queen. In that way, we can conclude that chess is not simple as counting points in the middle stages. For this reason, we also should focus on backward

induction, and see what is the best move to reach (if any) the best solution for a specific player.

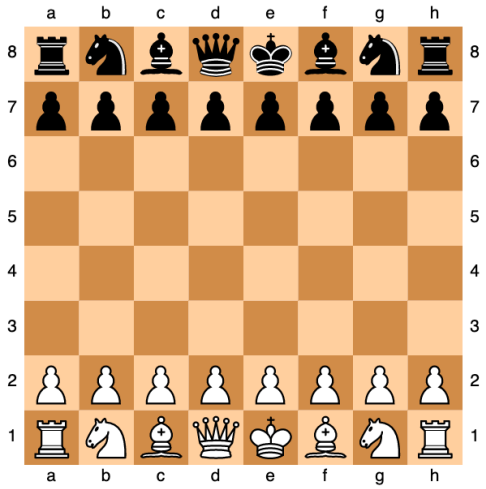
Deep Blue

Deep Blue was a chess engine developed by IBM [6]. There were two versions of Deep Blue released in 1996 and 1997. Kasparov, who was the world chess champion and Deep Blue held two different matches in 1996 and 1997. This paper will be focused mainly on Deep Blue version 1997. The first version, Deep Blue 1996, had 216 chips where each chip searched 50-100 million positions per second. However, there was no certain defeat in the matches between Kasparov and Deep Blue version 1996; IBM improved this version and in the next year, 1997, Kasparov and Deep Blue held six matches again [6]. For analyzing a chess game in a given state, the engine should be able to analyze the current chess board situation, attend some scores for each move in order to find the best move for a player [7]. In that way, Deep Blue version 1997 has three important parts in its chip, evaluation function, search control, and the move generator [6].

Move Generator

Move generator generates all possible moves, by ordering them as a low-valued piece takes a high-valued piece, a low-valued piece takes a low-valued piece, a high-valued piece takes a high-valued piece, a high-valued piece takes a low-valued piece, and moves which do not include taking pieces [6].

```
import chess
board = chess.Board()
board
```




```
board.legal_moves #first possible moves for the white player
```

```
<LegalMoveGenerator at 0x7f99e9ab8c88 (Nh3, Nf3, Nc3, Na3, h3, g3, f3, e3, d3, c3, b3, a3, h4, g4, f4, e4, d4, c4, b4, a4)>
```

Figure 1: Initial chessboard setup and all legal moves are listed for the next player with given command

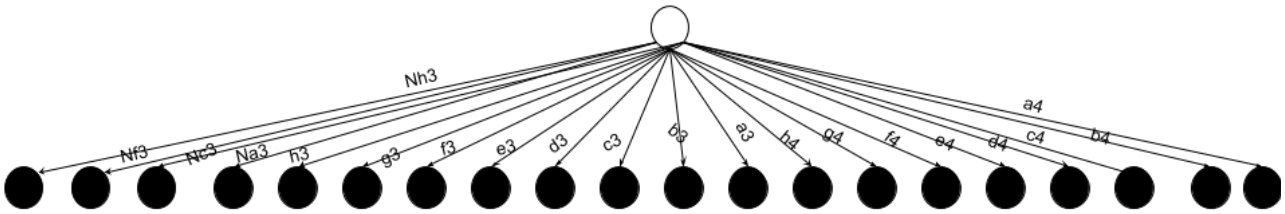


Figure 2: Tree representation of the Figure 1

Search Algorithm

The search algorithm was used to allow Deep Blue to look ahead at possible future positions before what move can be performed in the current position; in that way, the current move will be strengthened with foresighted strategy since the program could estimate the potential counter moves and the results [6]. We can perform the given step in Figure 1 to visualize the chessboard and given the command to see all possible moves in the given chessboard state. In Figure 2, all possible moves in the initial state are visualized in tree format.

The white player will have 20 potential moves initially. If we want to represent it by a tree structure, nodes represent white player's or black player's states and arrows will be labeled with the move.

There are twenty possible starting moves for the white player and there will be twenty counter move from the black player; there will be 400 possible moves after two play and 96889010407
after the fifth turn; the tree will grow exponentially [8].

While the tree will be grown in each turn of a player, it will not be feasible and effective to search full depth, Alpha-Beta Algorithm eases this situation. While using Alpha-Beta Algorithm, there is no need to investigate every node in the search tree for finding the best move, pruning a potential position/state node if the utilization of that node is not reaching the window, it can be omitted and the beyond subtrees can be excluded from the search. If the value of the move is smaller than the alpha value for maximizing player, then it can be omitted; if it is greater or equal to the beta value it can also be omitted since the opponent who is the minimizing player, has better alternatives than choosing that state; alpha value is set as the maximum value of a node encountered so far, and beta value is set as the minimum value encountered [9]. Based on this algorithm, the nodes which are not greater than or equal to the alpha value for maximizing player, white in our case, will be cut off, and for the minimizing black player, the nodes which are not greater than the beta value will be cut off and their further depth leaves will not further be investigated. These cutoffs will increase the efficiency, and result in lower durations while deciding the best move. For calculations of the node value, which refers to the result of the utilization function in our case, the 1997 version of Deep Blue searched over six and eight depth pairs of moves [10]. Depth of 6 and 8 is the minimum software depth yet special counter-attacks might result in further depth searches around 26-36; also, for the hardware search, Deep Blue used 4 - 5 play depth search [6].

Evaluation Function

Deep Blue's evaluation function is implemented on hardware, constant evaluation time reserved yet new behaviors were not allowed to be added since the evaluation function is on hardware [6].

The piece placements affect the evaluation function at most for Deep Blue [6]. By piece placement term, the positions of the peaces, number of peaces, and the resulting threats or advantages are included.

Evaluation function requires feature values, which can be static or dynamic; static values were set at the beginning of the search yet dynamic values need more time to evaluate since they are not stable, because, dynamic values were related to the current position of the board and pieces' safety in given state [6]. King safety, pawn structure, type or structure-based independent evaluation of pieces are the example of dynamic values, which have a significant part of the evaluation function of the Deep Blue. Also, previous matches were recognized by Deep Blue for selecting patterns, decide the strategy of the game so we can say Deep Blue also used prior information to play.

Stockfish

Stockfish project was started by Tord Romstad in 2004; in November 2008, Marco Costalba released Stockfish 1.0, which will become one of the most famous open-source chess engine [11]. One of the main reasons Stockfish is selected for performing the experiment is that since Stockfish is an open-source chess engine, it is easy to try and reach the results of the experiment without further permits. Also, Stockfish is defined as one of the most successful chess engines in the world, and its alternative and more successful rival AlphaZero is not provided as an open-source engine.

Like Deep Blue, Stockfish also searches in numerous positions to find the best alternative. Stockfish scans 70 million positions per second to find the best move [12]. Even though Stockfish has an attribute for scanning a high number of position variants which seems like an advantage, the efficiency of this search is improved by further chess engine developments such as AlphaZero because time is another important constraint for a chess game and expanding every node with brute force required large amount of time.

There is no significant difference between Deep Blue and the Stockfish chess engine since both of them using brute force to build search tree [13]. In that way, it is thought that if we simulate the sixth match between Kasparov and Deep Blue in 1997, we can expect to have closer results for imitating the history.

Also, like Deep Blue, Stockfish uses the data which is collected by the human players' matches as learning set to build its artificial intelligence and to train on behavioral patterns of humans who play chess [13]. This attribution of Stockfish is also improved in next-generation chess engines since they can learn themselves without training on real human data.

Stockfish's Evaluation Function

Like Deep Blue, Stockfish's evaluation function depends on different parts as mentioned below:

- Pieces and number of pieces which has a presence on the given chessboard state,
- Structure of the pieces like focusing on whether a piece is also defended by other pieces or not gives penalty or extra points,
- Structure of the pawns, whether they are defending other pieces or not,
- Type of structure-specific independent evaluation, for instance, specific movements such as castling can affect the function,
- Number of moves that a piece can perform, like distance a piece can move,
- Safety of the king,
- Threads by the rival,
- Safe and empty squares, more specifically safe and empty squares on the player's side,
- Certain draws [14].

Experiment

n the sixth game between Kasparov and Deep Blue that was held in 1997, which is the main discussion of this paper, one of the moves of Kasparov is criticized by the chess societies, which was performed in 7th turn. Daniel King also stated that this move may lead to the defeat of the undefeatable chess champion Kasparov [15]. We will try to simulate moves after the 6th turn and discuss whether the game result is the same or not. Stockfish version 12 will be used for simulations (See Appendix A).

It can be observed that Stockfish decides through its evaluation function; first, threatening rival's pieces has the highest priority such as checking the rival's king. The second realization is that Stockfish gives importance to the pieces according to their values; losing a bishop or knight which has 3 points importance is highly decided rather than losing a rook which is equal to 5 value points importance.

This match can be evaluated as one subtree, in the all possible strategies tree of a chess game. The backward induction is necessary to solve such problems yet under the limitations of efficiency problems such as the searching every node and state to find the state which can be the main reason for the defeat of Kasparov, it is assumed to have investigated after the sixth turn of the match. Moreover, in this match Kasparov resigned since he thought he will defeat, other alternative history is also investigated with Stockfish version 12 (See Appendix B).

Conclusion

In this paper, basic information about chess and general game-theoretical views on chess games is discussed. The selected scenario, "The sixth game between Kasparov and Deep Blue, in 1997" is evaluated under Deep Blue's technical attributions.

Deep Blue used a move generator, a search algorithm, and an evaluation function. The move generator lists the possible moves for a specific state and orders possible moves. The search algorithm dynamically searches for finding the best move in a given state by using Minimax Algorithm and Alpha-Beta Pruning. Evaluation Function is for deciding how a move will be selected, with consideration of chessboard situation, threats, positions that can strengthen the player's play, and potential defense mechanisms.

After Deep Blue is investigated, the research is extended to investigate the alternative history of the game with Stockfish version 12. There are multiple variants of chess engines with satisfying results, Stockfish is also one of them and it is provided as open-source, which gives developers to build while using free software. Also, as researched, Stockfish and Deep Blue work similarly, since both of them using brute force to expand the game tree. Hence, the experiment is held with Stockfish version 12. In that way, evaluation function and its effects can be observed clearly such as most preferred moves or strategies in the game.

Moreover, besides that investigating the game with interrupting 6th turn of the play was analyzed in the paper, also the original version which is ended in 19th turn is also investigated with Stockfish version 12.

As a result of the research, the game can be ended in a draw unexpectedly, since it is a sample practice of the same chess engine trying to beat itself. While using the minimax algorithm with alpha-beta pruning, given scenario of the defeat of Kasparov, who is a famous chess champion all over the world, can be changed.

References

[1] L. Badia, “Dynamic Games” presented to Game Theory to University of Padova, PD, IT, 2020 [PowerPoint slides]. Available: https://elearning.dei.unipd.it/pluginfile.php/562143/mod_resource/content/4/Gth2008.pdf, Accesed on: Jan. 29, 2020

[2] L. Gallego, “Game theory I: Perfect information,” Policonomics. [Online]. Available: https://policonomics.com/lp-game-theory1-perfect-imperfect-information/#:~:text=Perfect%20information%20refers%20to%20the,player's%20pieces%20on%20the%20board. [Accessed: 29-Jan-2021].

[3] K. Kunal, “Chess”, 30-Aug-2002. [Online]. Available: https://www.cse.iitd.ac.in/~rahul/cs905/lecture5/gametheory.html#SECTION00021000000000000000. [Accessed: 29-Jan-2021].

[4] A. Iqbal and M. Yaacob, “Computer recognition of aesthetics in a zero-sum perfect information game,” 01-Feb-2008. [Online]. Available: https://dl.acm.org/doi/10.5555/1415881.1415895. [Accessed: 29-Jan-2021].

[5] N. D. Elkies, “On Numbers and Endgames: Combinatorial Game Theory in Chess Endgames,” 1996. [Online]. Available: https://www.link.cs.cmu.edu/15859-s11/notes/Elkies-Chess.pdf. [Accessed: 29-Jan-2021].

[6] M. Campbell, A. J. Hoane, and F.-hsiung Hsu, “Deep Blue,” Artificial Intelligence, 09-Aug-2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370201001291. [Accessed: 29-Jan-2021].

[7] “Deep Blue,” CS221. [Online]. Available: https://stanford.edu/~cpiech/cs221/apps/deepBlue.html. [Accessed: 29-Jan-2021].

[8] R. Djurhuus, “Chess Algorithms Theory and Practice”, 4-Oct-2018. [Online]. Available: https://www.uio.no/studier/emner/matnat/ifi/INF4130/h18/slides/chess-algorithms-theory-and-practice_ver2018-2.pdf. [Accessed: 29-Jan-2021].

[9] A. Kishimoto, “Tutorial 3: Alpha-Beta Search and Enhancements”, [Online]. Available: https://webdocs.cs.ualberta.ca/~mmueller/courses/2014-AAAI-games-tutorial/slides/AAAI-14-Tutorial-Games-3-AlphaBeta.pdf. [Accessed: 29-Jan-2021].

[10] L. Greenemeier, “20 Years after Deep Blue: How AI Has Advanced Since Conquering Chess,” Scientific American, 02-Jun-2017. [Online]. Available: https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/#:~:text=The%201997%20version%20of%20Deep,more%20pairs%20in%20some%20situations. [Accessed: 29-Jan-2021].

[11] “Stockfish Chess,” Stockfish. [Online]. Available: https://stockfishchess.org/about/. [Accessed: 29-Jan-2021].

[12] S. SR, “Evaluation of Strategy by AlphaZero and StockFish on Chess Game ,” Journal of Basic and Applied Engineering Research, vol. 5, no. 5, pp. 443–444, Jul. 2018.

[13] L. Gallego, “Game theory I: Perfect information,” Policonomics. [Online]. Available: https://policonomics.com/lp-game-theory1-perfect-imperfect-information/#:~:text=Perfect%20information%20refers%20to%20the,player's%20pieces%20on%20the%20board. [Accessed: 29-Jan-2021].

[14] M. Lai, “Giraffe: Using Deep Reinforcement Learning to Play Chess,” , 2015. [Online]. Available: https://arxiv.org/pdf/1509.01549.pdf [Accessed: 29-Jan-2021].

[15]D. King, Kasparov V Deeper Blue. London: Trafalgar Square, 1997.

Appendix

Appendix A

https://colab.research.google.com/drive/1z0NIL5z-GWpdTGG2YOvwfLlR71opvWym?usp=sharing

Appendix B

https://colab.research.google.com/drive/1ZclaFqEhrx-KUnMAh5SB7WVgRX9dazhq?usp=sharing