2022 IEEE Pune Section International Conference (PuneCon)
International Institute of Information Technology ((I²IT), Pune, India. Dec 15-17, 2022

Vecma : An advance chess engine

1st Manavkumar Patel
Dept. of Computer Science(RMDSSOE)
Savitribai Phule Pune University
Pune, India
manavp347@gmail.com

2nd Harshit Pandey
Dept. of Computer Science(IIST)
Rajiv Gandhi Proudyogiki Vishwavidyalaya
Indore, India
harshitpandey251@gmail.com

3rd Tarush Wagh
Dept. of Computer Science(JSPMNTC)
Savitribai Phule Pune University
Pune, India
tarush390@gmail.com

4th Ameya Deepak Hujare
Dept. of Computer Science(PVGCOET)
Savitribai Phule Pune University
Pune, India
ameya.hujare@gmail.com

5th Rahul Dangi
Dept. of Electrical Engineering(IIT)
Indian Institute of Technology Delhi
Delhi, India
rahul@turnkey.work

2022 IEEE Pune Section International Conference (PuneCon) | 978-1-6654-9897-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/PUNECON55413.2022.10014895

***Abstract*—With the rapid development of personal computers and the widespread use of the world wide web, the Internet has emerge as an information carrier and has gradually replaced the traditional media such as newspapers and television, becoming the main source of information. Thus effecting the popularity of traditional games like chess. As, various chess website and app appear to be on web, but there is a gap between players and Ai bots as various challenges occurs like efficiency, intelligence of the engine and engine level to mimic different player's depth. This paper propose a unique implementation on the previously based chess AI engines which increases the efficiency of computation of every move of the game tree.**

Vecma engine is unique Negamax implementation of game tree along with the Ideal vector value map. Proposed model focuses on parameters which are usually excluded from the other AI engines like positional king safety, optimal Queen positional advantage and the complex pawn advancement with the vector map integration for every piece. This engine also keeps a check on heuristic functional values of the future checkmate or stalemate probabilities. We have also develop a GUI which can be used as a base for developing other AI variations. With both Vector map and Negamax tree the chess engine is far more computationally efficient as compare to other chess engines.

***Index Terms*—Alpha-beta pruning, Chess, GUI, Google colab, Min-max algorithm, Negamax, Positional Value Tables (PVTs), Python, Stockfish, Vector hash map**

I. Introduction

In recent decades the role of computers has been a backbone for development. In every sector the role of computers has led to the development and ease of living, but 2022 pandemic also brings a light on the over dependence on computer. But, recent study shows, every two child out of 3 in America is addicted to one or the other type of electronic device. As our dependence on computer is not only used for stimulating but it's became the necessity, as the every sector in pandemic was going in remote based, even education which is the most crucial, opted online platform. Thus computer are now considered as a necessity to access daily needs.

Games like checkers, snake and ladders and chess which previously were major part in the childhood is diminishing, but new approaches like online chess have thus have spark a new interest in such games. Chess is a strategy based game which revolves around the position of the pieces. It is a relative quite accident game and has closed link to various traditional practices. Also chess was used as a matrix to qualify for various events.

The chess has fascination history associated to it, the automation of chess can be recorded as back as 1952. But, throughout the decade chess was portrait as a game of intelligence. Chess has seen some of the best player with their unique playing approach. Thus, the race begins to mastery the game of intelligence. In 1958, first iteration of Alpha-beta algorithm was used as a module for a chess engine. Fast forward to the 1963, Grandmaster David Bronstein became first person to defeat a chess engine, back then almost all the chess engine was developed on knowledge-based approach and CSP. Advancements in the algorithm with increasing computational power of the devices made possible for developing a complex engine which lead to a definite win.

Here, we proposed a model which implements a new algorithm along with a positional vector model. As, for tracking the position of every move, we are going for a visual tracker for both moving the pieces and tracking the indices. After observing the moves and studying on the current chess engines like Stockfish , Fat Fritz and RubiChess. All of the chess engines uses alpha-beta pruning as a base along with the neural network. We followed some of the practices which makes a common ground for evaluating the results of different engines. The base code was taken from previous iteration where we enhance the efficiency of engine by using alpha-beta algorithm.

Proposed implementation helps in this direction as we developed a GUI to plays chess but also implemented the alpha beta pruning with negamax implementation which reduces the computation for AI bots. This overcome two major problems the lesser modular devices like smart phone due to restrain in power also provide a base to further improve or implement another algorithm. Alongside we developed an ideal vector

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.

map for every pieces as this increases the accuracy and also help the engine to gain a sense of direction to obtain a good position from a stuck or losing one.

There is also a major emphasis on the getting better position in early game. Parameters like king safety, Queen positional advantage and the complex pawn structure is also held in account. While getting results we also observed that on a particular move a fixed set of position is obtained in early games. To overcome this we developed a code which choose a random opening position out of a Grandmaster game library which ticks the objective.

II. Related Work

Approach of using database of chess matches played in various events is implemented in the paper, thus the engine does not compute for next move rather pick the move which was best for the player in particular position, which was previously encountered [1]. Another similar study was conducted on the chess opening moves based on two book. The author concluded the association of human player and a bot for choosing the same move for a particular opening in a game [2].Paper titled 'Programming a computer for playing Chess 'cited the first use of Min-max algorithm along with heuristics values which provided a base for future development in this direction [3]

Paper titled 'Metamorphic Testing of an Artificially Intelligent Chess Game 'uses the metamorphic testing as an alternate way for testing for game Ai engine. Metamorphic testing gives error free testing results comparatively to traditional methods [4].In [5] the author compare the chess game with no AI involvement. The test was conducted between the two human players. Thus stating the relation of metamorphic nature of only targeting the legal moves.

The analysis was conducted in which the data was compared between the choice of moves and the engines distribution of top moves. The parameter were non parametric association measure [6]. Here the Multi-Niche clustering is used along with genetic optimization. The paper focuses on choosing Positional Value Tables (PVTs) for every move, based on the PVT's the engine chooses the best move along with the position in account [7]. Vecma has a similar method for positional vector, but instead of relying on the value proposed method focuses on the position itself along with different parameter like king safety held accountable. In [8] author cite the initial idea of implementing Alpha-beta pruning for traversing the tree. As exhaustive search lead to optimizations of every move. The index of getting a 0 for a positional vector and evaluating method to stalemate or draw [9].

Another paper titled 'Candidate Moves Method Implementation in MiniMax Search Procedure of the Achilles Chess Engine' implemented the candidate move principle strategy along with the parallelization of search procedure in Min-max algorithm. The paper concluded the study by evaluating the Elo rating of the engine and concluded the parallel processing is similar to thinking of Grandmaster [10]. Similar paper 'Enhanced parallel NegaMax tree search algorithm on GPU' in this paper the parallel computation is implemented with Negamax algorithm as a base which obtained 40 times the results in traversing the tree compare to cpu threads. Also methods like shared GPU table, dynamic parallelism and no divergence are compared in the paper [11].

In [12] author proposed adaptability of a neural network which could be applied on board games like chess and checkers. The paper cites their implementation of training the engine to play N-dimension board and trained a model. image synthesis.

III. Methodology

Proposed method is the unique implementation of negamax variation of min-max, alpha-beta algorithm along with the desired vector maps of every pieces, name Vec-ma derived from the two method used. Thus it is both efficient also has a sense of direction of game play which proves to be a deciding factor in end games. The following flowchart shows the flow of evaluation of game position.

Along with this two methods there is also a new decision tree structure used which can be tuned for different difficulty and different playing style of the bot.

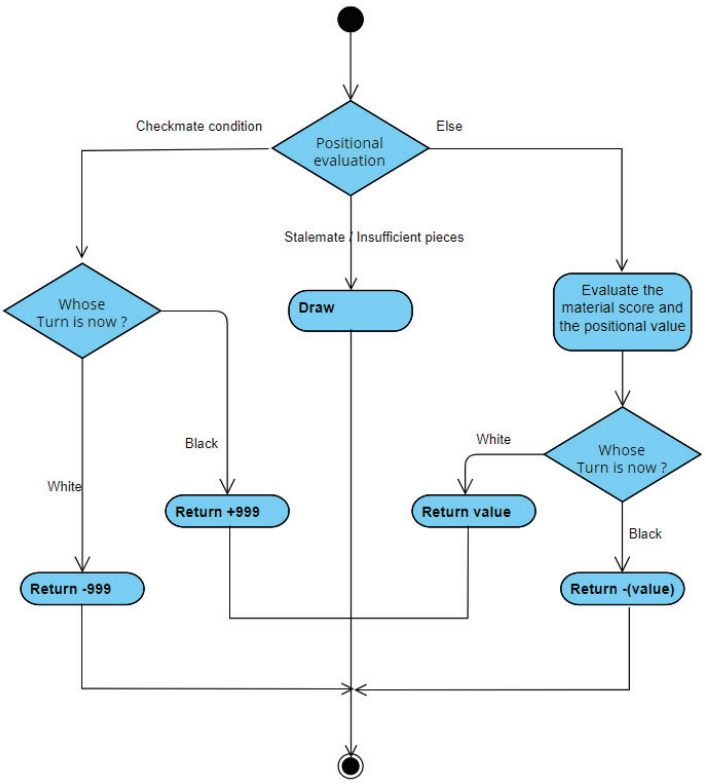

Fig. 1. Positional Evaluation Flowchart

A. Algorithm

Previously the chess AI engines used a basic game tree search, as advancement in the algorithms the efficiency of

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.

algorithms where also improved. In many chess engines now-a-days, min-max algorithm is used, STOCKFISH 8.0 use a custom variation with depth of 18. The algorithm focuses more on efficient way to search and reduce the time complexity. As the basic min-max algorithm takes around O(bm̂) where b is number of legal moves, which is very time consuming. In 1963, a new variation of min-max was developed. Alpha-beta variant of min-max algorithm was developed to drastically reduce the time complexity [13]. O(b̂(m/2)) is the time complexity of Alpha-beta pruning which is half of the min-max algorithm.

Even though this reduces the main problem which was the time complexity, but new problem arise which was, the assumption that main beta will also cut the lowest possible values every time. This introduced a problem recursive in nature.

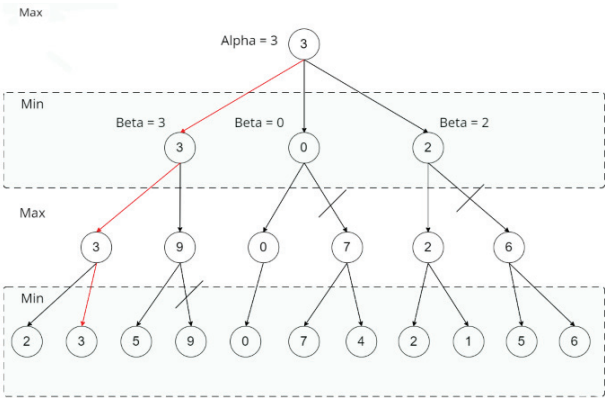

Fig. 2. Alpha-beta pruning tree

The traversing and pruning of is done to the branches in alpha-beta, but in negamax due to initial indexing the possibility of entire tree to be prune is also there, which narrows the time complexity drastically. In the game tree the Negamax implementation is shown, we clearly see that initial indexing to find the optimal path is traversed, also compare with the traditional Alpha-beta pruning.

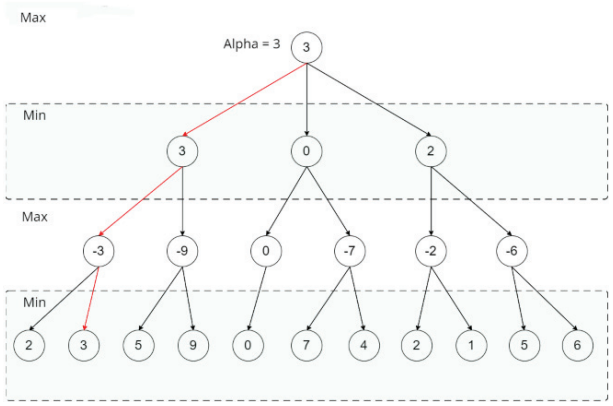

Fig. 3. Negamax implementation of Alpha-beta pruning tree

Negamax implementation of Alpha-beta pruning along with the Positional vector decrease the depth of game tree to traverse and find ideal move. This approach is ideal for real world scenarios. We clearly see the pruning of entire branches in Negamax game tree which decreases the traversing to be done.

$$Max(a, b) = -Min(-a, -b))$$

Along with Negamax implementation we also proposed an integration of vector maps, which encourages the engine to play positionally from very early stages. This is done using the vector map of 2D array which contains the heuristic values of every in every position. In fig. 4, 'Positional vector value heatmap' shows the ideal position for every piece.

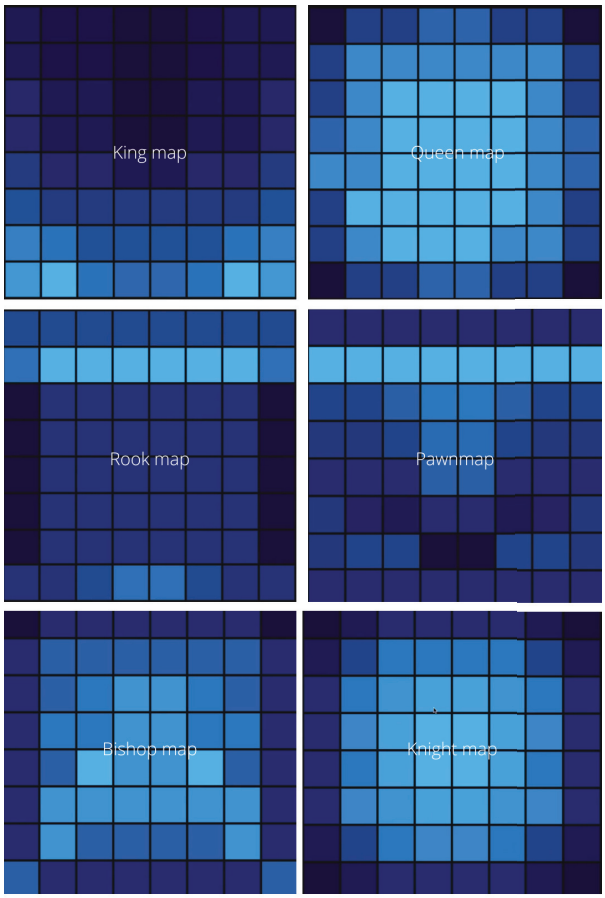

Fig. 4. Positional vector value heatmap

Every Lighter square show's higher vector value, which is a desired position over the other temporary positional value. This encourage a better positional development of pieces in early stages of games which is more important than a temporary threat. The fig. 5, shows the flow chart of the chess engine. Vecma uses matrix is used to determine whether the denoising is done towards the actual desire direction. TF-IDF and prepositions model layer sits between the pre-processing of input tax and selection of image where it vectorize the word which are relationally useful for original text.

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.

B. Model implementation

Entire engine is based on previous version of a min-max version. The programming language chosen is python for quick development as it has a large community which also bring a huge library support. We use google colab as an interface to develop and test the engine.

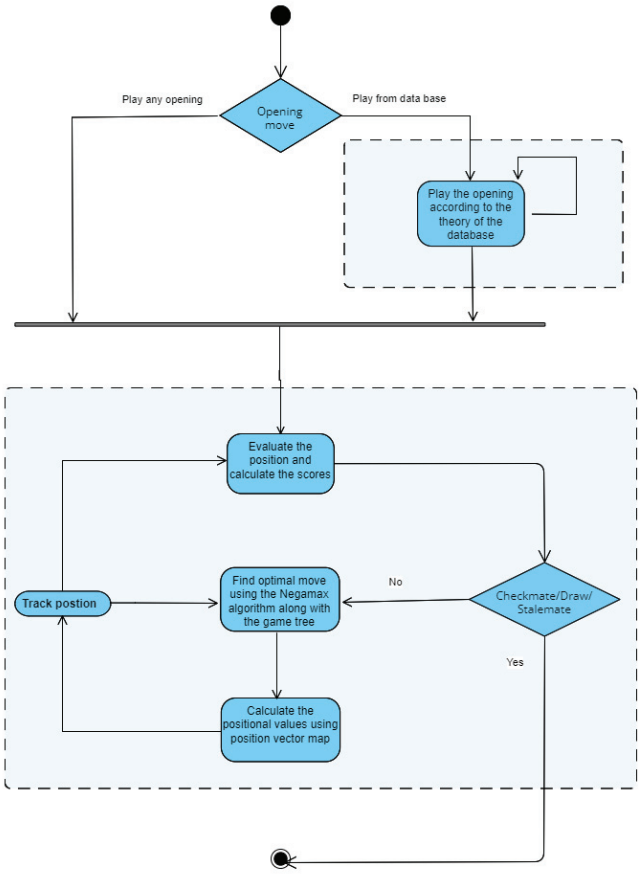

Fig. 5. Engine predictional move flowchart.

The tracking of pieces is done with the observing the moves on the board, and tracking moves as an input and the compute different value like positional and heuristic values further.

In idol case, two agents are equally intelligent. Which are playing for their own benefits in positional play. This problem is solved with the negamax approach which realize on the facts to proceed the game tree traverse further. Negamax considers both the player as equal, rather for considering one player as a wining and other for loosing. Negamax algorithm evaluates in the search tree to better prune the game tree [14].

The use of PVT's was also considered but, PVT's needs comparatively more computation power. Our approach not only is efficient but is also more accurate in real world. The main concern of mimicking the bot was also addressed using varying the game tree depth. Proposed model also has new decision tree structure used based on ideal vector map which can be tuned for different difficulty and different playing style of the bot, which is a unique.

IV. Results

There were two type of results obtain. The first batch shows the comparison of Vecma engine and stockfish engine for 3 grandmaster games played across decade. This will bring not only year of tactics but also shows the evaluation will stockfish8. Refer TABLE I for second batch, 300 games were played on 'chess.com' to evaluate the rating and real world scenario.

1) Bobby Fischer (W) vs Boris Spassky, Reykjavik, 1972
2) Viswanathan Anand vs Magnus Carlsen (W), Chennai, 2013
3) Kasparov Kasparov (W) vs Viswanathan Anand, 10th game, 1995

Bobby Fischer (W) vs Boris Spassky, Reykjavik, 1972

1. c4 e6 2. ♘f3 d5 3. d4 ♞f6 4. ♘c3 ♝e7 5. ♙g5 O-O 6. e3 h6 7. ♙h4 b6 8. cxd5 ♞xd5 9. ♙xe7 ♛xe7 10. ♘xd5 exd5 11. ♖c1 ♝e6 12. ♕a4 c5 13. ♕a3 ♜c8 14. ♙b5 a6 15. dxc5 bxc5 16. O-O ♜a7 17. ♙e2 ♞d7 18. ♘d4 ♛f8 19. ♘xe6 fxe6 20. e4 d4 21. f4 ♛e7 22. e5 ♜b8 23. ♙c4 ♔h8 24. ♕h3 ♞f8 25. b3 a5 26. f5 exf5 27. ♖xf5 ♞h7 28. ♖cf1 ♛d8 29. ♕g3 ♜e7 30. h4 ♜bb7 31. e6 ♜bc7 32. ♕e5 ♛e8 33. a4 ♛d8 34. ♖1f2 ♛e8 35. ♖2f3 ♛d8 36. ♙d3 ♛e8 37. ♕e4 ♞f6 38. ♖xf6 gxf6 39. ♖xf6 ♔g8 40. ♙c4 ♔h8 41. ♕f4 1-0

Fig. 6. First game-play moves

The first game we compare is the one of the most famous games in chess history, Bobby Fischer (W) vs Boris Spassky, Reykjavik, 1972 4th game. At the time chess was relative clam in nature where the mid-games were in development of pieces. In this game, we saw the excellence from both the grand masters, the games followed with both player having a equal positional play going on board but, the aftermath indicates that Boris was in advantage on 19th move, but soon felt behind in development. Stockfish evaluates the game as same. While going through the line here is where proposed engine shines as the move matric show that the game was dominated by Fischer, with the mistake on 19th move, which gave advantage to Boris but, stayed relatively short with a sudden blunder on 35th move.

Here despite Stockfish game tree is indexing at 18, our engine is much more responsive to every move and accurate in the nature of the game.

Viswanathan Anand v Magnus Carlsen (W), Chennai, 2013

At this game we again see the efficiency and accuracy of the engine as compare to stockfish analysis. The game was world chess championship, 2013 when Carlsen took the title from Anand. In this game we clearly see the sharp opening

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.

Fig. 7. Chess board on move 19th and 35th.

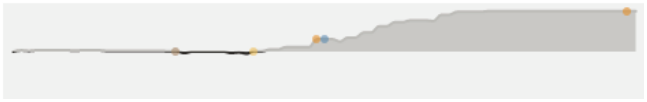

Fig. 8. Evaluation graph of stockfish.

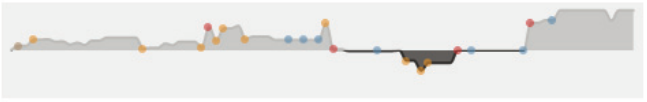

Fig. 9. Evaluation graph of proposed engine.

1. d4 ♘f6 2. c4 e6 3. ♘c3 ♗b4 4. f3 d5 5. a3 ♗xc3+ 6. bxc3 c5 7. cxd5 exd5 8. e3 c4 9. ♘e2 ♘c6 10. g4 O-O 11. ♙g2 ♘a5 12. O-O ♘b3 13. ♖a2 b5 14. ♘g3 a5 15. g5 ♘e8 16. e4 ♘xc1 17. ♕xc1 ♜a6 18. e5 ♘c7 19. f4 b4 20. axb4 axb4 21. ♖xa6 ♘xa6 22. f5 b3 23. ♕f4 ♘c7 24. f6 g6 25. ♕h4 ♘e8 26. ♕h6 b2 27. ♖f4 b1=♛+ 28. ♘f1 ♛e1 0-1

Fig. 10. Second game-play moves

d4 from Anand and with time it weathered away the chances he build with the early setup.

Fig. 11. Chess board on move 9th and 12th.

At move 12th we a blunder by Anand which by time the evaluation gap got bigger. Anand prefers more stable games where with time the position gets stronger and stronger. At last the immense pressure builds up for the opponent to break through the defence of Anand has created, But at this games we clear see Anand choosing a different approach which cost him the title.

At the evaluation graph of both stockfish and Vecma, we see clearly that the mid-game was also favouring Carlsen with our engine shows gradual losing value for Anand. There was a favourable position obtained on 9th move but 12th move cost Anand. The results show that the evaluation done by stockfish is at 18th depth.

Fig. 12. Evaluation graph of stockfish.

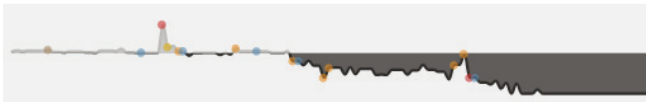

Fig. 13. Evaluation graph of proposed engine.

Kasparov Kasparov (W) vs Viswanathan Anand, 10th game, 1995

1. e4 e5 2. ♘f3 ♘c6 3. ♙b5 a6 4. ♙a4 ♘f6 5. O-O ♘xe4 6. d4 b5 7. ♙b3 d5 8. dxe5 ♗e6 9. ♘bd2 ♘c5 10. c3 d4 11. ♘g5 dxc3 12. ♘xe6 fxe6 13. bxc3 ♛d3 14. ♙c2 ♛xc3 15. ♙b3 ♘xb3 16. ♙xb3 ♘d4 17. ♕g4 ♛xa1 18. ♙xe6 ♜d8 19. ♙h6 ♛c3 20. ♙xg7 ♛d3 21. ♙xh8 ♛g6 22. ♙f6 ♗e7 23. ♙xe7 ♛xg4 24. ♙xg4 ♔xe7 25. ♖c1 c6 26. f4 a5 27. ♔f2 a4 28. ♔e3 b4 29. ♙d1 a3 30. g4 ♜d5 31. ♖c4 c5 32. ♔e4 ♜d8 33. ♖xc5 ♘e6 34. ♖d5 ♜c8 35. f5 ♜c4+ 36. ♔e3 ♘c5 37. g5 ♜c1 38. ♖d6

Fig. 14. Third game-play moves.

Fig. 15. Chess board on move 12th and 32th.

Again a classic game between Garry Kasparov and Viswanathan Anand, The two are considered as the top player of their eras. This game was played in the year 1995, where after 8 consecutive draw 9th games was led by Anand, but soon this games show how good of a Grandmaster Garry Kasparov

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.

was. The bold opening with Ruy Lopez was chosen by Anand which proved to be fatal. Garry playing black has an advantage throughout the game and with the first blunder on move 12th Garry clutch the position slowly to his side.

Fig. 16. Evaluation graph of stockfish.

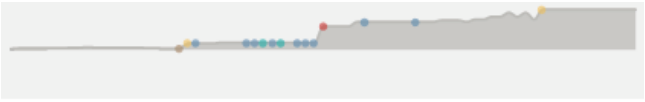

Fig. 17. Evaluation graph of proposed engine.

The evaluation charts of both the engine shows similar results but at move 19th again a blunder by Anand slides the rating toward Garry again, here our engine shows clear positional advantage points gained by the moves whereas at the move 32th stockfish starts to pivot the game toward Garry.

TABLE I
TABLE TYPE STYLES

Total game	Win	Loss	Draw
300	216	63	21

Refer TABLE I, with 300 games over 6 weeks, Engine secure 216 wins with 63 loss and 21 draw. At the end the rating of engine was 1726 with accuracy of 96.4% according to stockfish at the depth of 5. The initial games were majority of loss due to fixed opening moves played by the engine, after the database was connected for opening engine started giving desired results. Due to our unique implementation of Negamax and Ideal positional vectors, Vecma has obtain the graph sensitive and smooth in nature.

V. CONCLUSION

In comparisons with other engine, Stockfish tends to beat Vecma every time. This could be due to neural networks used in the engine and relative higher depth of game tree the stockfish traverse (50). At, the end the results were clear that our unique implementation of negamax along with ideal positional vector boost the performance of the previous engine which used alpha-beta pruning which is used in current state-of-the-art chess engines. As a result, Vecma was found to be superior to other systems in terms of simulating players of varying ratings and playing styles in the real world. Engine is focused on efficiency and better positional moves. Proposed engine is best suited for low powered device. Our implementation is done in python which increase the computational time, in future the engine would be implement in c# and c++ to get best time complexity for traversing the tree.

Further task would be to use threads and sockets which will enable to parallel computation for game tree increasing the efficiency. Also, another approach of using of database, graph and neural networks would likely also decrease the game tree. The batch size of the survey also should be increased for accurate results.

ACKNOWLEDGMENT

Thanks to J.P. Fishburn for explaining the version of Negamax in the research paper [15]. As Negamax was a great jump in efficiency for the game tree and it gave a huge boost in the computation of the engine. Special Thanks to Sebastian Lague, who is a well-known youtuber based on which the idea spark, He also explained a complex mechanism of another alternative to traverse for deeper depth of game tree. Also, thanks to Tord Romstad, Marco Costalba, Joona Kiiski and Gary Linscott for developing the open-source chess engine Stockfish. At last, we would like to thank our mentor and guide Rahul Dangi for guiding us throughout.

REFERENCES

[1] O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu, "Genetic algorithms for evolving computer chess programs," *IEEE transactions on evolutionary computation*, vol. 18, no. 5, pp. 779–789, 2013.

[2] M. Levene and J. Bar-Ilan, "Comparing typical opening move choices made by humans and chess engines," *The Computer Journal*, vol. 50, no. 5, pp. 567–573, 2007.

[3] C. E. Shannon, "Xxii. programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[4] A. Liaqat, M. A. Sindhu, and G. F. Siddiqui, "Metamorphic testing of an artificially intelligent chess game," *IEEE Access*, vol. 8, pp. 174179–174190, 2020.

[5] A. Liaqat and M. A. Sindhu, "A metamorphic relation based approach for testing a chess game," in *2018 14th International Conference on Emerging Technologies (ICET)*, pp. 1–6, IEEE, 2018.

[6] M. Levene and J. Bar-Ilan, "Comparing move choices of chess search engines," *ICGA Journal*, vol. 28, no. 2, pp. 67–76, 2005.

[7] A. Rahul and G. Srinivasaraghavan, "Phoenix: A self-optimizing chess engine," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 652–657, IEEE, 2015.

[8] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[9] V. Vučković, "The compact chessboard representation," *ICGA Journal*, vol. 31, no. 3, pp. 157–164, 2008.

[10] V. Vuckovic, "Candidate moves method implementation in minimax search procedure of the achilles chess engine," in *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, pp. 314–317, IEEE, 2015.

[11] A. A. Elnaggar, M. Gadallah, M. A. Aziem, and H. El-Deeb, "Enhanced parallel negamax tree search algorithm on gpu," in *2014 IEEE International Conference on Progress in Informatics and Computing*, pp. 546–550, IEEE, 2014.

[12] S.-N. Orzen, "Interaction understanding in the osi model functionality of networks with case studies," in *2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 327–330, IEEE, 2014.

[13] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice," *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 11, pp. 1203–1212, 1989.

[14] G. Heineman, G. Pollice, and S. Selkow, "Path finding in ai in algorithms in a nutshell," 2008.

[15] J. P. Fishburn and R. A. Finkel, "Parallel alpha-beta search on arachne," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 1980.

Authorized licensed use limited to: Vivekanand Education Society's Inst of Tech. Downloaded on September 13,2024 at 06:24:28 UTC from IEEE Xplore. Restrictions apply.