International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 689-694 www.ijaem.net ISSN: 2395-5252

Comparative analysis of minmax algorithm with alpha-beta pruning optimization for chess engine

Rashi Sahay
Assistant Professor Apex Institute of Technology, Chandigarh University, Mohali, India

Date of Submission: 05-02-2023 Date of Acceptance: 20-02-2023

ABSTRACT—The chess game engine is introduced which makes use of a Minmax algorithm for searching game trees and makes use of alpha-beta pruning algorithm to reduce the search space of game states in the game state tree. The first part of this paper is an expository presentation that demonstrates and explains the Minmax algorithm. The paper also proves that the alpha-beta pruning method is shown to make the chess engine algorithm optimal and bounds are obtained for its running time with various randomized data.

Keywords—Chess engine, Artificial intelligence, Minmax algorithm, Alpha-beta pruning algorithm

I. INTRODUCTION

Computer programs for playing the chess game usually search a large number of states from the game state tree where each node is a state that represents the state in the game. An algorithm called the "Minmax algorithm" is used to select the best possible state, which may or may not be correct (guarantees a checkmate) but is the best optimal state from the current state to all possible future states. Additionally, the Alpha-beta pruning algorithm is used to reduce the number of iterations in the Minmax algorithm.

The purpose of this paper is to create and analyze the performance of a chess game engine that makes use of this algorithm [9]. Additionally, we also analyze the performance of the game engine that reduces the search space for the Minmax algorithm using the Alpha-beta pruning algorithm.

Section II of this document describes the literature review for the Minmax algorithm and the Alpha-beta pruning algorithm. Section III describes the proposed methodology to create the AI chess game engine. Section IV contains the results and a discussion on recording the results. Section V gives the comparative analysis of the game engine with and without the Alpha-beta pruning function. Finally, Section VI gives the conclusion derived from the results.

II. LITERATURE REVIEW

A. Minmax Algorithm

The Minmax algorithm [1,2] is a simple recursive algorithm that determines the next state by analyzing the different states in the game tree.

The maximizing player tries to maximize the score of the next state whereas the minimizing player tries to minimize the score of the next state. However, since the Minmax algorithm cannot search the entire game tree to find the best next state, it uses a fixed depth search to determine the next state[23].

The Minmax algorithm works by generating a game tree up to a certain depth with the current state as the root node. Then we generate a score value for each leaf node using the evaluation function. Depending on the type of parent node (maximizing or minimizing), we select the leaf node's value as the parent node's value. So, if the parent node is a maximizing player, it will always select a state with the highest score value from its child nodes [10,22].

In this the process is repeated until we reach our root node at depth 0, and we can determine the optimal path that maximizes (or minimizes) our state value.

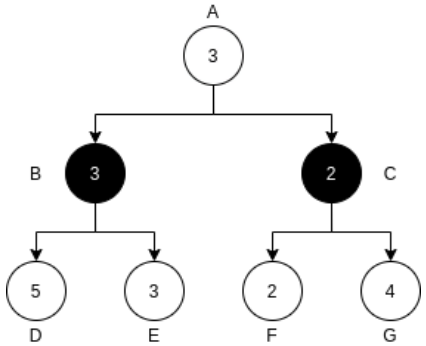

Fig. 1. Ex. Game tree

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 689-694 www.ijaem.net ISSN: 2395-5252

Considering the example in Fig. 1, where each node is the state is the game of chess. A white state represents a maximizing player and a black state represents a minimizing player. The leaf nodes are D, E, F, and G which have been assigned score values according to the evaluation function.The current state is the root node A [12,13].

Now, considering state B and its child nodes D and E, node B is a minimizing node and hence it will be assigned the minimum value from its child nodes. Hence, node B will be assigned the value 3. Similarly, for the nodes, C, F, and G, node C will be assigned the value 2. Node A is a maximizing node and hence it will select the maximum score value from its child nodes. Hence, node A will be assigned the value 3 [18].

Looking at the above example we can see that the next best state from the current state A is node B. We can assume that the opposing player plays optimally and selects node E.

B. Alpha-beta pruning algorithm

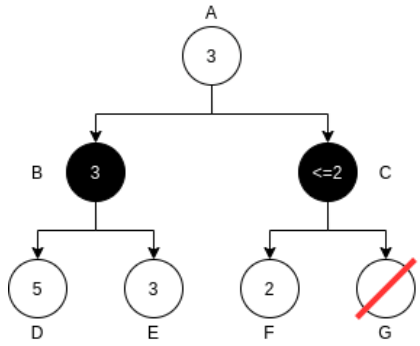

Fig. 2. Ex. Game tree with □-β pruning

Considering the example in Fig. 2, we can see that node F has a score value of 2. We know that the parent node C is a minimizing node and hence its score value will be at most equal to 2 (from node F) [11,20]. Now considering the left subtree (node B, D, and E), the value of node B will be equal to 3. We can hence determine that node A being a maximizing node will select node B (value 3) over C (at most 2). Thus, we would never have to calculate the score value of node G, since its parent node will always select the minimum value, which can at most be 2[21].

This method of reducing the search space in the game tree is known as □-β pruning [1,2,3], where □ and β are the upper and lower bounds. With □-β pruning, we only have to explore the most optimal nodes which can lead to a win (considering the opposing player plays optimally) and we can stop exploring nodes that do not change the score of their parent nodes.

□ and β are the best scores that either player can achieve where □ corresponds to the maximizing player and β corresponds to the minimizing player. Initially, □ is assigned the minimum possible score for the maximizing player which in -∞ and β is assigned the maximum possible score for the minimizing player which is ∞ [14,17].

Now, consider the following example using the □ and β variables, we can determine which node should be explored further and which can be pruned. The condition to terminate the recursive call is when the value of □ is less than or equal to β (β <= □). For minimizing node B, □ would remain the same (-∞) and β would be min(node D, node E) which is 3 [19].

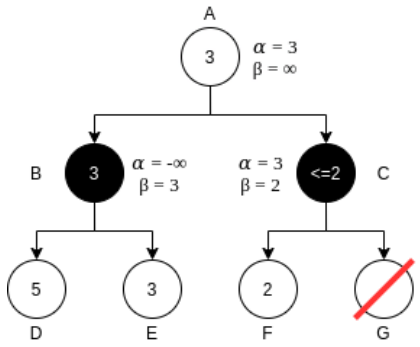

Fig. 3. Ex. □-β pruning

Since the left subtree is now explored, the values of □ and β for node A will now be (□ = 3 and β = ∞). The algorithm will now start evaluating the right subtree with modifies □ and β. Node C is a minimizing node and hence, for node F value of β for node C will be 2 (due to F). As the termination condition is now satisfied the recursive call will be terminated and the current score value will be returned to the parent node [16,24].

Following is the pseudocode for Minmax with Alpha-beta pruning, For maximizing nodes, we want to visit the child with the highest score value first. Similarly, for the minimizing nodes, we want to visit the nodes, we want to visit the child with the lowest score value first.

III. PROPOSED METHODOLOGY

In this paper, we will be comparing the effectiveness of the Alpha-beta pruning technique in improving the performance of the Minmax algorithm in chess. Additionally, we will analyze the performance results to obtain a quantitative measure for the effectiveness of the Alpha-beta pruning technique.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 689-694 www.ijaem.net ISSN: 2395-5252

A. Board Representation

The chessboard is represented in the simplest possible manner - as an 8 by 8 matrix, each containing a piece or a blank space. For board representation, we use the chessboard.js library. This library provides the chessboard layout, its animations, and different API calls to interact with the chessboard.

B. Evaluating the next move

The chess engine uses Flask to interact with the chessboard. We use the python-chess library to determine all the legal next moves from the current state. Using the library, we first create a chessboard object with starting state representing the start of the game. In order to make a move, the chess object can be passed an alphanumeric string which tells the board to move a piece from position A to position B. If the move is illegal, the method call returns an error. The board object also has a method legal_moveswhich returns an iterable list of all possible next moves [15].

These moves are used by the Minmax algorithm to determine the next best state. The board object also implements a method fen, which returns a FEN string that represents the state of the board and this string can be used by the chessboard.js to update the state to the user.

C. Heuristic evaluation function

TABLE I. HEURISTIC EVALUATION TABLE

BLACK	SCORE	WHITE	SCORE
Pawn	-10	Pawn	10
Bihop	-30	Bihop	30
Kight	-40	Kight	40
Rook	-60	Rook	60
Quen	-100	Quen	100
Kin	-800	Kin	800

The Minmax score value for each state will be the sum of values of all the pieces present on the chessboard. Hence, the following is the equation to calculate the score value,

Score = (∑BLACK) + (∑WHITE)

Before calling the Minmax function for making a move the □ and β variables will be initialized to -∞ and ∞ respectively. Once the score is calculated for the child node, it will be assigned to the parent node's □ or β. The □ and β variables are maintained by each state and when the terminating condition is true (β <= □), the recursive call in the Minmax algorithm is terminated.

D. Game over condition

The method is_game_overprovided by the python-chess library is called before every move to determine if the game is over if any of the following conditions happen:

- Checkmate.
- Stalemate (Not in check state but no legal moves are available).
- Insufficient material.

Other conditions for the game to end are not considered (resignation, timeout, repetition, agreement).

IV. RESULTS

To obtain performance analysis for the Minmax algorithm, we considered the time required to make a single move in a game of chess in milliseconds. The Minmax algorithm used by the chess engine evaluates the game tree with a depth of 3. The chess engine is played against the Stockfish 14 game engine. The Minmax algorithm plays 10 games with the Stockfish 14 game engine and additionally, 10 games are played with the alpha-beta algorithm.

A. Minmax algorithm

Following is the bar graph for 1 game played against the Stockfish game engine using the Minmax algorithm,

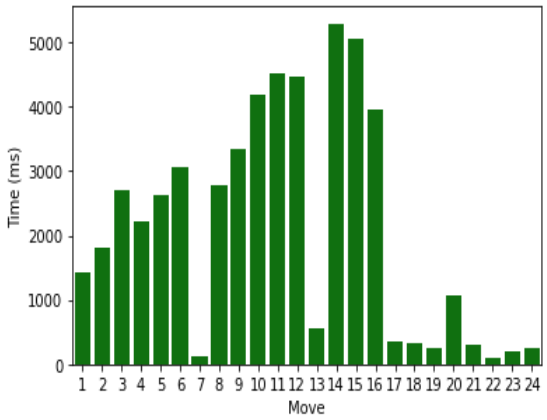

Fig. 1. **Minmax bar graph for 1 game**

Following is the line graph for a set of 10 games played against the Stockfish game engine using the Minmax algorithm,

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 689-694 www.ijaem.net ISSN: 2395-5252

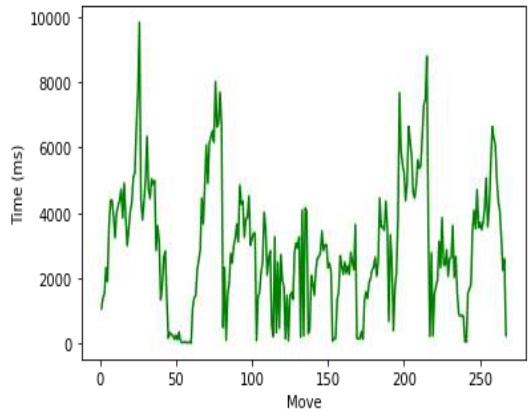

Fig. 2. Minmax line graph plot for 10 games

Following is the bar graph for the number of moves played each game for a set of 10 games against the Stockfish game engine using the Minmax algorithm,

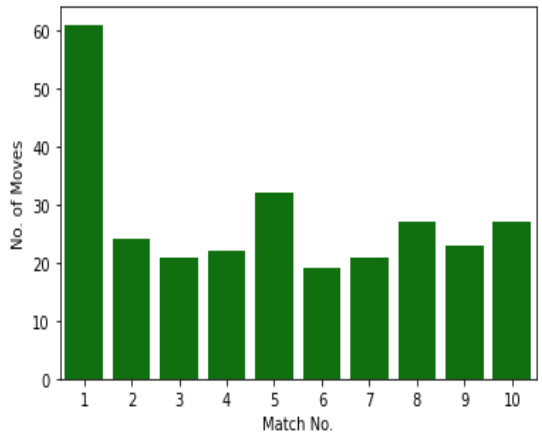

Fig. 3. Minmax bar graph plot for moves each game

TABLE II. MINMAXALGORITHM

No.	Minmax algorithm		
	Property	**Result**	**Unit**
1	Average time taken per move for 10 games	2965.0524	ms
2	Average no. of moves for each game	27.7	-

B. Minmax algorithm with alpha-beta pruning

Following is the bar graph for 1 game played against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning,

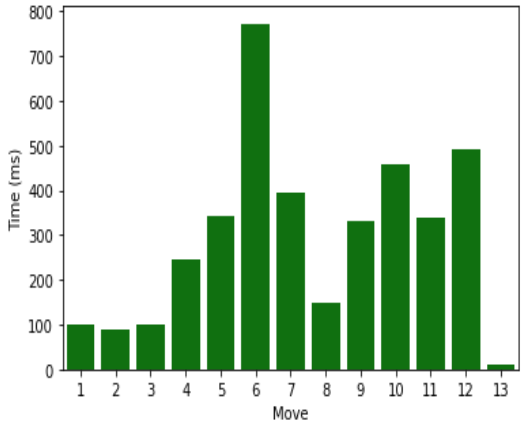

Fig. 4. Minmax with alpha-beta bar graph plot for 1 game

Following is the line graph for a set of 10 games played against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning,

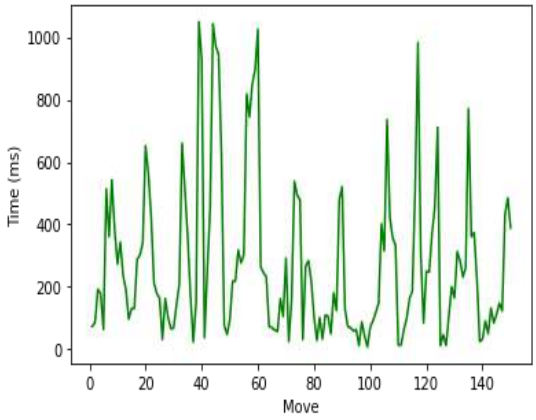

Fig. 5. Minmax line graph plot for 10 games

Following is the bar graph for the number of moves played each game for a set of 10 games against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning,

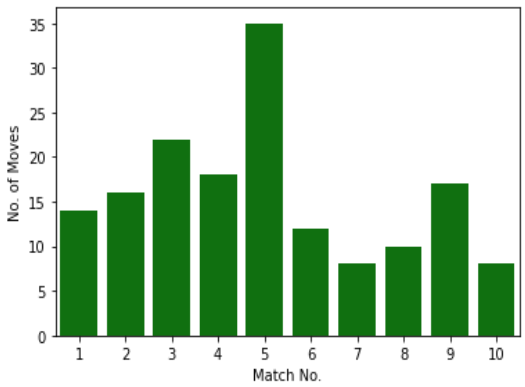

Fig. 6. Minmax with alpha-beta bar graph plot for moves each game

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 689-694 www.ijaem.net ISSN: 2395-5252

TABLE III. MINMAX ALGORITHM WITH ALPHA_BETA

No.	Minmax algorithm with alpha-beta		
	Property	**Result**	**Unit**
1	Average time taken per move for 10 games	274.4000	ms
2	Average no. of moves for each game	16.0	-

V. ANALYSIS

Hence, looking at the above results, we can conclude that the Minmax algorithm takes more time to calculate each move since it looks at all the states in the game state tree for a certain depth. This results in higher time required for each move as compared to the alpha-beta pruning optimization which does not have to look at all the states.

VI. CONCLUSION

In this paper, we implemented the Minmax algorithm to create a chess engine and also compared its performance with the Alpha-beta optimization. The Minmax algorithm generated the game tree with a depth of 3. The chess engine was tested against the Stockfish 14 chess engine. The results show that the Alpha-beta optimization significantly reduces the time required to determine the next move.

Thus, the alpha-beta pruning algorithm optimizes the Minmax algorithm and is 90.7455% faster than the Minmax algorithm.

VII. ACKNOWLEDGEMENT

The work would not have been possible without the academic support of my organization and colleagues. I had full support and guidance from my family, colleagues and my work environment so that I can indulge in my research. I was also motivated to analyze the Min-Max algorithm and move forward with my work for which I am grateful.

REFERENCES

[1] Tiantian Guo, HongkunQiu, Bairui Tong, YajieWang :. ”Optimization of Multiple Game Algorithms in Amazon Chess” (2019 CCD

[2] Cong Hu, Xiangcheng Wu, Tonghui Qian, Hai Luo, Jiaqian Wang, “An Improved Knowledge Base for Chinese Chess Game” (2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2020))

[3] Knuth, D.E. and Moore, R.W. (1975). An Analysis of Alpha-Beta Pruning. Artificial Intelligence Vol. 6, No. 4, pp. 293-326. ISSN 0004-3702.

[4] T.A. Marsland and Y. Bjrmsson, From minmax to manhattan, Games in AI Research, pp. 5-17.

[5] Xi-Zhao Wang, Yu-Lin He, Pan Su, and Wen-Liang Li, "Two-ply iterative deepening in Chinese-chess computer game," 2009 International Conference on Machine Learning and Cybernetics, 2009, pp. 2020-2026,doi:10.1109/ICMLC.2009.5212141

[6] J. J. Gillogly. The Technology Chess Program. Artificial Intelligence, 3, 1972, pp: 145-163.

[7] C. E. Shannon. Programming a computer for playing chess. Philosophical Magazine, Vol. 41, No. 7, 1950, pp: 256-275.

[8] Richard E. Korf. Depth-First Iterative Deepening: An Optimal Admissible Tree Search. Artificial Intelligence, 27, 1985, pp: 97-109.

[9] Ghosh, Gopal et al. "Internet of Things Based Video Surveillance Systems for Security Applications" in Journal of Computational and Theoretical Nanoscience, Volume 17, Number 6, June 2020, pp. 2582-2588(7)

[10] Varun Dogra, Aman Singh et al. “A research paper on “Banking news-events representation and classification with a novel hybrid model using DistilBERT and rule-based features” in Turkish Journal of Computer and Mathematics Education, 10.17762/turcomat.v12i10.4954

[11] Ghosh, G. et al. “Secure Surveillance Systems Using Partial-Regeneration-Based Non-Dominated Optimization and 5D-Chaotic Map” in Symmetry 2021, 13, 1447.

[12] Batth, R.S., Gupta, M. et al. “Comparative Study of TDMA-Based MAC Protocols in VANET: A Mirror Review.” In: Khanna, A., Gupta, D., Bhattacharyya, S., Snasel, V., Platos, J., Hassanien, A. (eds) International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing, vol 1059. Springer, Singapore. https://doi.org/10.1007/978-981-15-0324-5_10

[13] Sharma, T. et al. “Intelligent Heart Disease Prediction System using Machine Learning : A Review.” In International Journal of

Recent Research Aspects ISSN: 2349-7688, Vol. 4, Issue 2, June 2017, pp. 94-97

[14] Tanvi Sharma et al. "Prediction of Heart Disease Using Cleveland Dataset: A Machine Learning Approach" in International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 4, Issue 3,Sept 2017, pp. 17-21

[15] Arora, Mohit et al. "An efficient effort and cost estimation framework for Scrum Based Projects" in International Journal of Engineering & Technology, [S.l.], v. 7, n. 4.12, p. 52-57, oct. 2018.

[16] Singh, Puneeta et al. "Analysis on Different Strategies Used in Blockchain Technology" in Journal of Computational and Theoretical Nanoscience, Volume 16, Number 10, October 2019, pp. 4350-4355(6)

[17] Gaba, Seema et al. "Analysis on Fog Computing Enabled Vehicular Ad hoc Networks" in Journal of Computational and Theoretical Nanoscience, Volume 16, Number 10, October 2019, pp. 4356-4361(6)

[18] Dipta Datta et al. "UAV Environment in FANET: An Overview" in Applications of Cloud Computing (pg. 16)

[19] I. Batra et al. "Performance Analysis of Data Mining Techniques in IoT," 2018 4th International Conference on Computing Sciences (ICCS), 2018, pp. 194-199, doi: 10.1109/ICCS.2018.00039.

[20] Gopal Ghosh, Divya Anand et al. "A research paper on "A Systematic Review on Image Encryption Techniques" in Turkish Journal of Computer and Mathematics Education, 10.17762/turcomat.v12i10.4956

[21] Kaur, Navneet et al. "Detection of Plant Leaf Diseases by Applying Image Processing Schemes " in Journal of Computational and Theoretical Nanoscience, Volume 16, Number 9, September 2019, pp. 3728-3734(7)

[22] Parteek Kumar et al. "Implementation and Analysis of Detection of Wormhole Attack in VANET" in JNCET Volume 8 Issue 3 MARCH. 2018, pp. 5-12, ISSN:2395-5317.

[23] Sharma, S. et al. "Hybrid bat algorithm for balancing load in cloud computing" in Int. J. Eng. Technology, 7(4.12), pp.26-29.

[24] Dogra, V., Singh, A. et al. "Analyzing DistilBERT for Sentiment Classification of Banking Financial News." In: Peng, SL., Hsieh, SY., Gopalakrishnan, S., Duraisamy, B. (eds) Intelligent Computing and Innovation on Data Science. Lecture Notes in Networks and Systems, vol 248. Springer, Singapore. https://doi.org/10.1007/978-981-16-3153-5_53