IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.7, Special Issue.3, Feb. 2019 E-ISSN: 2347-2693

Comparison of Chess Engines: A Review

Rajesh Kumar Singh[1*], Satish Kumar Negi[2], Pusphendra Kumar Chandra[3]

[1,2,3]Dept. of Computer Science & Engineering, Guru Ghasidas University, Bilaspur (CG), India

[*]*Corresponding Author: rajeshkumarsingh381@gmail.com, Tel.: +91-8247697341*

Available online at: www.ijcseonline.org

***Abstract*—**The Chess engine alludes to a program that examines chess and chess variation positions. The first ever chess engine to have won against a human grandmaster was IBM's DeepBlue in 1999. Since then multiple chess engines have emerged with improved search heuristics, hardware sets and dictionaries. Though in recent years many chess engines with a machine learning approach have produced striking achievements in comparison to the typical bruteforce chess engines. This paper aims to review the selected chess engines which generate counter moves automatically. Their respective specialties will be explained and compared to give insight on direction of research on chess game in modern world.

***Keywords*—**Chess Engine, Pruning, Minimax

I. INTRODUCTION

The chess game is an intriguing mathematical problem, approached by many researchers and mathematicians over the years to find an optimal solution. The chess in itself may not hold much importance but solving it requires understanding of the chess environment, defining win conditions and working out a series of changes in the environment in counter to the uncontrollable events, also affecting the environment, to achieve the win condition. Attempts to find better solutions have been made since decades from dictionaries, endgames, brute force to finally machine learning. More successful research on this will provide insight that will help in tackling other real-world situations, like:

- Better Heuristic search in management of workforce in telecommunications companies
- Applications and improvements in medical field like Heart sound analysis, drug creation, and toxicology and disease symptoms analysis.
- In transportation, DSP transmission in automobiles with fuzzy logic, and autonomous vehicles.
- Automated reasoning, automated theorem proving, automated proof checking, reasoning by analogy induction and abduction.
- Image restorations, Mobile advertisings and many more.

In order to comprehend how chess engines generate moves and solve the problem of chess, a few required topics are to be learned. These algorithms also have significance in solving many real life problems, which is the main motive behind the research in chess computing.

A. THE PROBLEM OF CHESS

In chess, all the information (pieces, position, turn, rules and win condition) is very clear and discrete. The main goal of any player, white or black is to either win or force a draw. The white has first turn to make a move in chess. As such in the beginning it has 20 possible moves, of which, one it selects and moves. Similarly, black also has 20 possibilities [2]. The first move by white cannot be judged as good or bad because there are alternate scenarios where each move can be good or bad, but of course, we can state certain moves safer than others based on a database of previous games. So to make a move, we have two strategies to choose on move in any turn [1]:

- Always choose certain move from the list of possible legal moves, which is pure calculated strategy.
- Choose a random move from the list of possible legal moves, which isn't a pure strategy but doesn't mean will fail.

But doing any of these or having a dictionary (having a correct move for all positions based on previous games or created by a chessmaster) is equally impractical. Because the total no. of positions and moves is too big to be evaluated in a feasible time.

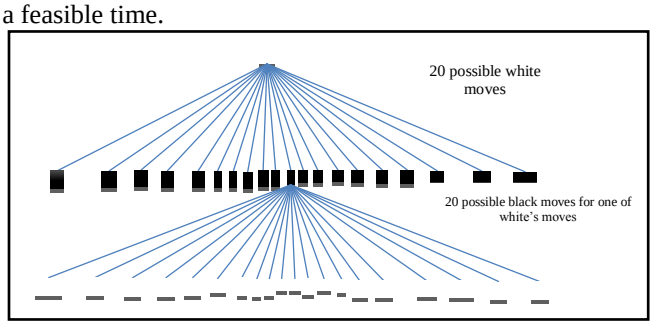

Figure 1. Possible Moves

As can be seen, the black has 20*20=400 possible moves (in strategy) depending on whites move. And the white in next turn has 400*20=800 number of possible legal moves and this number keeps increasing. A typical game lasts for 40 moves until resignation, this is inferior to the aim of the target chess engine, but even this will generate 10^{120} moves. And evaluating which will take years even if the machine evaluates one move per micro second [1].

B. THE EVALUATION FUNCTION

The way to deal with choosing the decency or disagreeableness of a move can be summed up in view of specialists' surveys. The most widely recognized component to choose which group is winning is the quantity of pieces in every player still in play. The more pieces a player has, progressively the odds of winning. In any case, not all pieces are similarly important, so a coefficient esteem can be given to each class of piece. For instance, ruler can be given 9, Rook be given 6, bishop and knight 4 and pawn 1. Then for a given position, Evaluation function E(P) can be calculated:

$$E(P) = 9(Q'-Q)+6(R'-R)+4(B'-B)+4(K'-K)+(P'-P)$$

The unprimed letters are pieces of the opponent. If the function evaluates a positive value the team is winning, if it's negative the opponent is winning and if it's 0 then leading to a draw.

C. DESCRIPTION OF THE ALGORITHMS

These algorithms show how evaluation function is used to predict moves and also self asses own moves and how the searching problem can be minimized in a way.

1) Minimax

This is the most normally utilized calculation for turn based games to choose a victor. In this calculation, the Max tries to expand its assessment by choosing a move which favors it. Thus, the Min additionally tries to limit the value of evaluation function, choosing a move favoring it most. Since the value of evaluation function is a negative ramifications to rival.

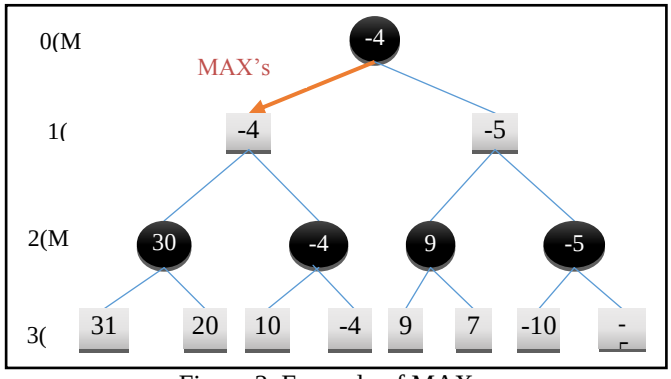

Figure 2. Example of MAX

As can be seen in example figure, at 0, Max will try to maximize the function value so it'll choose -8 move. While min will try to minimize it so it'll choose -8 over 20. In this way they both work up ideally to win.

2) Alpha Beta Pruning

This is like an extension to MiniMax. We have alpha and beta, a maximum assured value for minimizer and a minimum assured value for the maximizer. The idea is, in case of a maximizer, to not search further children nodes if any 1 children node gives a value which is higher than the value of sibling of the maximizer, because the parent minimizer will never select values from any children nodes of the current maximizer. Conversely, in the case of a minimizer, is to not search further children nodes if any one children returns a value less than the value of the sibling of the minimizer, because the maximizer will always select the maximum one [4].

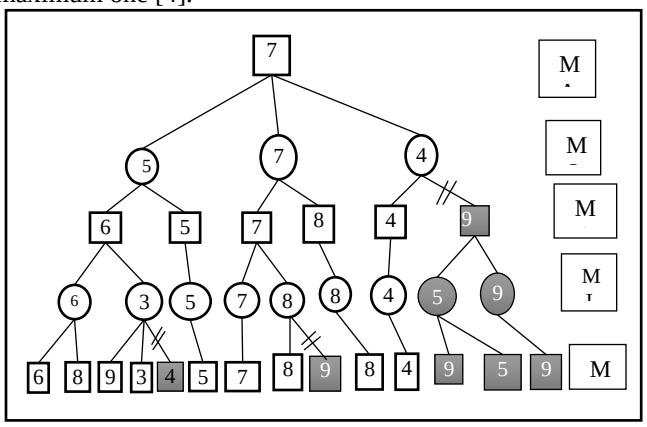

Figure 3. Alpha Beta Pruning

As can be seen in the given example, in the minimizer node with value 4, its sibling has value 5, first exploration returns 7, which is higher than 5, but it has to minimize it, it searches further to find 4 this is less than 7 but also less than 5, which means going further will either return unacceptable (greater than 4 which the minimizer will discard) or values less than 4 which the parent maximizer will discard. So it doesn't need to search further. Alpha beta pruning can, in this way, reduce the branching factor and increase search depth in limited time.

D. FACTORS IN EVALUATION

1) Distributed Computing

Generally, in a large system, the processing work is divided and distributed among different node processors in the form of a client-server model(very large scale projects) or in the form of threads and processes(in the case of chess engines). This is called distributed computing, as it saves time by searching tree nodes in parallel.

2) Race Condition

When in context of computing, race is an anomaly that occurs due to improper synchronization between the threads of a process [8].

3) Non Deterministic Algorithm

It is an algorithm that can, for same inputs, in different iterations, execute differently or have different behavior, as opposed to in deterministic algorithm. For example, concurrent algorithm performs differently in different runs because of race condition [9].

II. VARIOUS CHESS ENGINE

These are the various chess engines under consideration. They have been selected based on their performance criteria in tournaments and their respective unique characteristics.

A. ASMFISH (FORK OF STOCKISH)

It is a free open source UCI compatible chess engine which keeps improving with the help of its huge community contribution. In order to not analyze the same position several times, a transposition table is used. This is essentially memorization applied to the search function.

B. KOMODO

Komodo is a UCI compatible chess engine developed by Mark Lefler and Don Dailey, improved by GM Larry Kaufman. Komodo is a business chess engine however more seasoned versions are free for non-profit utilization. Komodo intensely depends on assessment instead of profoundity, and therefore has an unmistakable positional style.

C. HOUDINI

Houdini is a UCI compatible chess engine created by Robert Houdart. It has been inspired by open source engines like Stockfish, and Crafty. Houdini 5 utilized aligned assessments in which engine scores connect specifically with the win hope in the position.

D. RYBKA

Rybka is a PC chess engine outlined by International Master Vasik Rajlich. Rybka is a shut source program, yet at the same time a few points of interest have been uncovered: Rybka utilizes a bitboard portrayal and is an alpha-beta searcher with a generally expansive desire window. It utilizes exceptionally forceful pruning, prompting imbalanced search trees.

E. GIRAFFE

A trial open source chess engine by Matthew Lai under the GNU General Public License in 2015. Giraffe utilizes the Eigen linear variable based math library, and Pradyumna Kannan's magic move generator. Giraffe's assessment work is a profound neural system prepared by TDLeaf.

III. COMPARISON AND ANALYSIS

A. COMPARISON TABLE OF THE OF CHESS ENGINES

The beginning, mid game and endgame of chess engines matter less in the overall result of the match but it's still worth noting their respective natures for a personalized review.

Table 1. Comparison of Chess Engines

	Houdini	**Rybka**	**Komodo**	**Stockfish**	**Giraffe**
Positional	Can evaluate well in closed positions	Applies tricks and tactics to avoid quiescent position	Good calculating power makes it good in semi-closed positions	Precise calculation and understanding of nuances of position	Has good understanding of different positions (depends on training data set)
Deep Search	Piece sacrifice and recovery ability involving deep search	Tends to not search very deep	Performs best when allowed to search deep	Can calculate long and precise	It uses slow neural networks so its search depth in limited time is limited further
End Game	Very good defense, tends to draw more in end game	Good end game with flexible play style	Relatively best end game skills	little tendency to draw positions	Plays exceptionally well in both endgames and openings

As can be seen, there are sufficiently capable in all beginning, middle and endgames but their nature changes a little which may result in different choices and possible mistakes.

B. TEST RESULTS STATISTICS FROM CCRL DATABASE

CCRL 40/40 Rating List is a Chess engine database where users can custom select engines to test and compare between them under following conditions:

Ponder is turned off, General book is allowed up to 12 moves, End Game Table Base for 3-4-5 pieces is allowed. Time control: Equivalent to 40 moves in 40 minutes on athlon 64 X2 4600+ (2.4 GHz), about 15 minutes on a modern Intel Central Processing Unit (CPU).

Computed on December 30, 2017 with Bayeselo based on 774'825 games.
The engines to be compared, mentioned above were custom selected in CCRL and the following results were obtained:

1. Results Matrix
It shows the number of wins, losses and draws of each engine. Here for each win, 1 score is awarded to winner, 0.5 to both engines in case of draws. This table shows how many wins each of the pair of engines have won. The score of the engine in row is taken as +ve and the one in column is taken as –ve.

Table 2. Result Matrix

#	Name	Elo	1 asmFi	2 Houdi	3 Komod	4 Rybka	5 Giraf
1	asmFish 051117 64-bit 4CPU	3422		25 – 21 +6–2=38 +10	29.5 – 19.5 +11–1=37 +34		
2	Houdini 6 64-bit 4CPU	3410	21 – 25 +2–6=38 –10		33.5 – 24.5 +10–1=47 +32		
3	Komodo 11.2 64-bit 4CPU	3404	19.5 – 29.5 +1–11=37 –34	24.5 – 33.5 +1–10=47 –32			
4	Rybka 4 64-bit 4CPU	3154					
5	Giraffe 20150908 64-bit	2461					

As can be seen, asmFish won 6 matches against Houdini, lost 2 matches and 38 matches were drawn, hence its score was 38*0.5+6=25. But since a lot of the matches were draws with a few exceptions, asmFish gained 10 points in its elo rating after the assessment.

B. Score Matrix
This matrix shows the percentage of wins of chess engines in row against the one in column. The number of wins and the total number of matches are also shown.

Table 2. Score Matrix

#	Name	Elo	1 asm	2 Hou	3 Kom	4 Ryb	5 Gir
1	asmFish 051117 64-bit 4CPU	3422		54% 25/46	60% 29.5/49		
2	Houdini 6 64-bit 4CPU	3410	46% 21/46		58% 33.5/58		
3	Komodo 11.2 64-bit 4CPU	3404	40% 19.5/49	42% 24.5/58			
4	Rybka 4 64-bit 4CPU	3154					
5	Giraffe 20150908 64-bit	2461					

As can be seen, asmFish has greater number of wins against both komodo 11.2 and Houdini 6. It wins 25 out of 46 matches against Houdini and won 29 matches and 1 draw out of 49 against komodo 11.2. asmFish wins all its matches against Giraffe, So do Houdini and Komodo.

C. ANALYSIS
Stockfish has obviously shown that straightforward, brute-force methodologies ought not be immediately disposed of. Also, iterative procedures, specifically, thoughts created for alpha-beta pursuit and iterative developing, are pertinent to other search areas. Stockfish has clearly exhibited the insufficiency of ordinary AI strategies for realtime calculation. Stockfish does not utilize AI languages or learning portrayal strategies, for these traditions are too moderate for a continuous, high performance application. But still stockfish needs to be improved in its endgames phase.

Komodo is reasonably solid in all phases of the game. Usually does not favor sacrificial lines but it is more susceptible to sacrifice when compared to other engines, which is a good thing in a few cases.

Houdini's understanding of beginning is not as greater than others in comparison. But its endgames are where it shines. In terms of tactics, Houdini performs well, but still tends to defend more in complicated positions. Houdini despite being very good at strategic moves loses to stockfish due to horizon effect.

Rybka loses the competition in the early phases, its understanding of early game pawn structures is poor. Despite being good in endgames, it is too slow for competition. Its biggest advantage is that it has very good positional understanding. It also loses to other chess engines due to horizon effect.

Giraffe has good positional understanding which is verified by STS tests, but it is outperformed by other chess engines by a long shot. But its real strength lies in being able to see the positions in a unbiased way given enough hardware support, which is also why it cannot achieve grandmaster level of chess game play with its current neural network. Along with other chess engines giraffe is also subjected to STS test under the same conditions. As giraffe learns reflects on its score of STS test and improves and after thousands of iteration of learning, converges to 9641 score. Here is the score of other chess engines on same test.

Table 3. STS Score

Engines	Avg. ELO	STS score
asmFish 051117	3422	9420
Houdini 6	3410	9378
Komodo 11.2	3404	9350
Rybka 4	3154	8561
Giraffe 20150908[3]	2461	9641

As can be seen in the above table, giraffe also has attained STS score rivaling the top chess engines, in a comparatively very short amount of time.

IV. CONCLUSION

Overall, in terms of performance and win rates, asmfish 051117 is the best engine among the chosen chess engines with a win rate of 17.89% and a draw rate of 78.94% in 95 matches. Houdini 6 has a winrate of 11.53% and a draw rate

of 81.7% out of 104 matches. Komodo has winrate of 1.86% and draw rate of 78.5% out of 107 matches. Although the number of matches aren't nearly as enough to evaluate engines performance independently but for comparison it suffices. Giraffe is out of their league because of the time constraints in each move. However except giraffe, all of them rely on heuristics and simple search algorithm with bruteforce to attain deeper searches in same time, so it hasn't actually solved the problem of chess. In terms of solving the problem of chess, the only chess engine that has learned how to play chess is Giraffe. It gives an example of how a unbiased solution however slow, to a unknown problem using machine learning, can be reached. Giraffe learned to play chess in a relatively less time too. So further improvements in giraffe performance can make it beat other grandmaster level chess engines, and hence further research on chess with machine learning approach is encouraged.

REFERENCES

[1] C. E. Shannon, *"Programming a Computer for Playing Chess"*, Computer Chess Compendium, **pp.2-13**, 1983.

[2] E. Okur and S. Kavuzlu, *"Developing an Adaptive Chess Program"*, Boazii University, 2011.

[3] M. Lai, *"Giraffe: Using Deep Reinforcement Learning to Play Chess"*, Arxiv.org, 14 Sept. 2015.

[4] G. M. Baudet, *"An analysis of the full alpha-beta pruning algorithm"*, Proceedings of the 10th annual ACM symposium on Theory of computing, 1978.

[5] P. Cunningham, M. Cord and S. J. Delany, *"Supervised Learning. Machine Learning Techniques for Multimedia Cognitive Technologies"*, Springer, **pp. 21-49**, 2008.

[6] E. Robert, Schapire, *"Recent Advances in Reinforcement Learning"*, Springer, pp.99-121, 1996.

[7] D. Peleg, *"Distributed Computing: A locally sensitive approach"*, Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2000.

[8] R. H. Netzer, B. P. Miller, *"What are race conditions?: Some issues and formalizations"*, *ACM Letters on Programming Languages and Systems,1*(1), **pp.74-88**, 1992.

[9] R. W. Floyd, *"Nondeterministic algorithms"*, *Journal of the ACM (JACM)*, Vol.14, Issue.4, **pp.636-644**, 1967.

[10] A. Rahul, G. Srinivasaraghavan, *"Phoenix: A Self-Optimizing Chess Engine"*, International Conference on Computational Intelligence and Communication Networks (CICN), 2015.

[11] J. Schaeffer, H. J. V. D. Herik, *"Games, computers, and artificial intelligence"*, Artificial Intelligence Elsevier Science, Vol.**134**, Issue.1-2, pp.**1-7**, 2002.

[12] J. Esch, *"A Self-Learning Evolutionary Chess Program"*, Proceedings of the IEEE, Vol.92, Issue.12, **pp.1946-1946**, 2004.

[13] E. Hearst, *"Man and machine: Chess achievements and chess thinking"*, Chess Skill in Man and Machine, **pp.167-200**, 1983.

[14] N. Ensmenger, *"Is chess the drosophila of artificial intelligence? A social history of an algorithm"*, Social Studies of Science, Vol.42, Issue.1, **pp.5-30**, 2011.

[15] C. J. Tan, *"Deep Blue: A computer chess and massively parallel systems"*, In Proceedings of the 9th international conference on Supercomputing (ICS'95), Barcelona, Spain, **pp.237-239**, 1995.

[16] K. Dhou, *"Chess software and its impact on chess players"*, University of Northern British Columbia, 2008.

Authors Profile

Mr. Rajesh Kumar Singh pursed Bachelor of Technology from Guru Ghasidas Vishwavidyalaya, Bilaspur (CG), in year 2018. He has done project on Chess Engine during Final Year of B.Tech.He is currently working in software industry.

Mr. Satish Kumar Negi pursed Bachelor of Engineering from Guru Ghasidas Vishwavidyalaya, Bilaspur (CG), in year 2005 and Master of Technology from National Institute of Technology, Hamirpur (HP), in year 2009. He is currently working as Assistant Professor in Department of Computer Science & Engineering, SOS (Engg.&Tech.), Guru Ghasidas Vishwavidyalaya, Bilaspur (CG). He has published more than 6 research papers in reputed international journals and international conferences. He has 8 years of teaching experience.

Mr. Pushpendra Kumar Chandrai pursed Bachelor of Engineering from University, Raipur, (CG), in year 2005 and Master of Technology from National Institute of Technology, Rourkela (Odisa), in year 2008. He is currently working as Assistant Professor in Department of Computer Science & Engineering, SOS (Engg.&Tech.), Guru Ghasidas Vishwavidyalaya, Bilaspur (CG). He has published more than 6 research papers in reputed international journals and international conferences. He has 8 years of teaching experience.

