International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.51, October 2025

History and Milestone Developments in Computer Chess Algorithms from 1947 -1986

Evarista Onokpasa
University of Jos
Jos, Nigeria

ABSTRACT

According to Shannon there are "at least 10^{120} ways to play the game of Chess from start to finish. A computer operating at the rate of one variation per micro-second would require over 10^{90} years to calculate the first move!"[5]. This brings to the fore the importance of applying efficient algorithms, heuristics and data structures in the design of computer chess engines. Much work has been done in the development of computer chess algorithms. This paper takes a journey on the history of a few famous, published algorithms for computer chess. It considers in a concise manner, the strategies conceived and applied in designing different chess engines from 1947-1986 (the eras of foundational computer chess and the start of computer chess championships). These strategies, include chess knowledge, game tree search, tree pruning strategies, the use of evaluation, transposition and refutation tables, as well as specialized hardware designed to optimize strength of play of computers.

General Terms

Algorithms, Heuristics

Keywords

Chess, strategies, game tree, tree pruning and evaluation functions

1. INTRODUCTION

The first true Chess automaton was built in 1914 by Leonardo Torres y Quevedo in Spain [5]. This machine had the ability to play chess end games with a king and a rook and it ensures it checkmates the opponent in a few moves. Prior to this was an automaton built by Baron Wolfgan Von Kempelen, which seemed to have the capability to play the game of chess autonomosly, but was later found to be a sham. A human operator was hidden in the machine [16]. Starting from Shannon's work to date, computer chess has involved strategies and game tree search, tree prunning, game state evaluation and statistical analysis. The next section takes each one of these and brings out the key elements involved in the computer chess design.

2. DEFINITION OF CHESS TERMS

Chess Positions
A chess position contains all the information of a game's state, for example:

(1) A statement of the positions of all pieces on the board.
(2) A statement of which side, White or Black, has the move.
(3) A statement as to whether the king and rooks have moved (this determines if a castling move can still be done or not)
(4) A statement of the number of moves made since the last pawn move or capture.(50 moves after this, the game ends in a draw, this is called the 50 move draw rule).

Using algebraic notation(every square is labeled using the intersection of the files (defined with the lower case letters a, b, c, ..., h) and ranks (defined with the numbers 1, 2,...,8)), we can represent the chess game from start to the current state shown in Figure 1. King, Queen, Knight, Bishop, Rook represented with the alphabets K, Q, N, B and R respectively. There are no letter representation for Pawns. A label for a square on the board e.g. e4 suffices to indicate a pawn moved to square e4. To denote moves for other pieces the alphabets are followed by the target square. Below is the notation of the first 2 ply for the chess game state shown in Figure 1

white moves	black moves
e4	e5
f6	Nf3

Strategies
A Strategy is a sequence of planned moves a player takes to attain a game state with a better evaluation than his opponent. This strategy must take into consideration the possible responses from the opponent. Thus if at game state P, White's choices for a good play are dependent on the stage of the game and the stages are classed in 3: The opening, the mid-game and end-game.
Ply
A half-move i.e. a single player's turn.
Game tree
This is a tree- like representation of all possible move sequences in a game.
Variation
A variation in chess is simply a branch of possible moves—either theoretical (openings), practical (middlegame plans), or forced (tactical/endgame sequences).
En prise

This means a piece is undefended and able to be captured by the opponent.

En passant

This describes the capture by a pawn of an enemy pawn on the same rank and an adjacent file that has just made an initial two-square advance.

Quiescence search

A deeper search performed at the end of a standard search (like alpha-beta pruning) to resolve tactical situations before evaluating the position.

Refutation move

A refutation is a move or combination of moves that exposes the weakness in an opponent's strategy.

Move generation Move generation is the process of creating a list of all legal chess moves for a given board position

3. EARLY WORK ON COMPUTER CHESS ALGORITHMS FROM 1947 TO 1969

3.1 Shannon's work

In 1949, Shannon presented a paper that described designing computers to play chess, with a chess strategy. These would include rules of thumb already known to human players. He defined a strategy for chess as "a process for choosing a move in any given position."

He described 2 strategies, "type A" and "type B". The type A strategy, the program that did a complete search of the chess game tree (using a simple evaluation function) to a given depth, this search type is predictable, the same game position always produces the same move. The type B program searches selectively by eliminating branches (using forward pruning) that appear to be less advantageous and investigating only a few promising branches and investigating each one to deeper plies. Shannon went further to explain some computer chess concepts of chess positions, evaluation functions, variations in play and style.

Approximate Evaluation Functions The term "approximate" is used to describe evaluation function in chess as there are no exact functions to rate or evaluate a chess game state. The evaluation functions defined for the game of chess are based on assertions and information gathered from a study of countless games, to produce principles which can define the strength of a play or value of a game state. Some examples of these principles are:

(1) Assigning values numeric values to each chess piece, e.g. King = 200, Queen = 9, rook = 5, bishop = 3, etc.

(2) Material Advantage i.e. Summing up the numeric value of each players' pieces and obtaining the difference.

(3) Mobility of chess pieces (i.e. how many possible moves each piece can make), higher mobility attracts higher rating.

(4) Backward, isolated and doubled pawns have lower ratings.

(5) An undefended or exposed King is assigned a lower value than otherwise.

For more of these principles see [5] Thus an evaluation $f(S)$ for a game state S can be summarised as

$$f(S) = r(W) - r(B)$$

where $r(W)$ and $r(B)$ are the ratings for White pieces and Black pieces respectively, based on the evaluation principles. Thus if $f(S) > 0$ white has a more favorable evaluation, if $f(S) < 0$ black is at an advantage. Shannon's evaluation function for a player Y is summarized below:

$$r(Y) = 200K+9Q+5R+3(B+N)+P-0.5(D+C+I)+0.1M$$

where Y has K, Q, R, B, N, P number of Kings, Queens, Rooks, Bishops, Knights and Pawns respectively. D, C and I stand for Doubled, backward and Isolated pawns which have a negative effect on the ratings and M are the total number of legal moves for all the pieces for Y. Using this evaluation function, the evaluation for the game state for Figure 1, indicates an advantage for black:

$$r(W) = 200+9+5+3+8-0.5(0+0+1)+0.1\times 24 r(B) = 200+9+5+3+8-$$

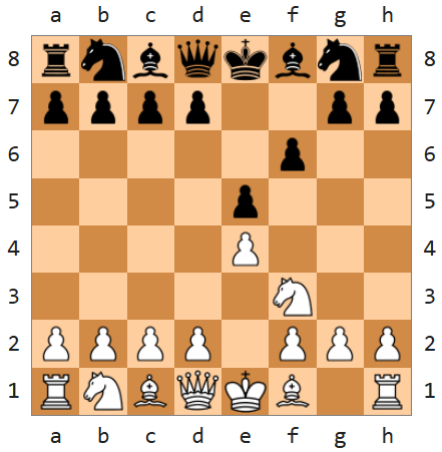

Fig. 1. A chess game state example

Shannon's strategy

Shannon [5] details a strategy which uses a one-move look ahead. This involves evaluating the next game state $P_x(S)$ for all possible legal moves P_x, $x = 0, 1, ..., r, r \in N$, for the current player.

With this one-move look ahead the white player can choose the move for which $f(P_x(S))$ is maximum, or the minimum in the case of a black player. This look ahead strategy of evaluating possible game board states for every legal move can be extended to 2, 3, 4, ... depending on how much resources (cpu time, computational speed and memory) are available to carry out the evaluations inorder to make a move of maximal advantage. Figure 4 shows an example of a two-ply look ahead for which the white player can make the move of maximal advantage, while considering the optimal move for his opponent. After 1-ply it appears the move of maximal advantage for player Y is P_2 which gives the highest score of 21. However the opponent can respond with move P_{2A} to obtain a maximum score of -4. For player Y this would have resulted in the worst game state after the second ply. With the evaluation of the game states after 2-ply it is clear that if both players aim at obtaining an optimal game state. Player Y must make the move for which the minimum value is maximum (this is the minimax algorithm [27]), that corresponds to P_3 for player Y. Thus, it is clear that the deeper the search the better the strength of play. Shannon's type A strategy involves a tree search of depth 4 (i.e. 4-ply look ahead) but this is basically brute-force approach as experienced human chess players would not need to consider all possible moves to a given depth to make strong moves. So, Shannon proposed a Type-B Strategy which involves eliminating branches of the tree which are not considered viable options to obtain a favorable game state.

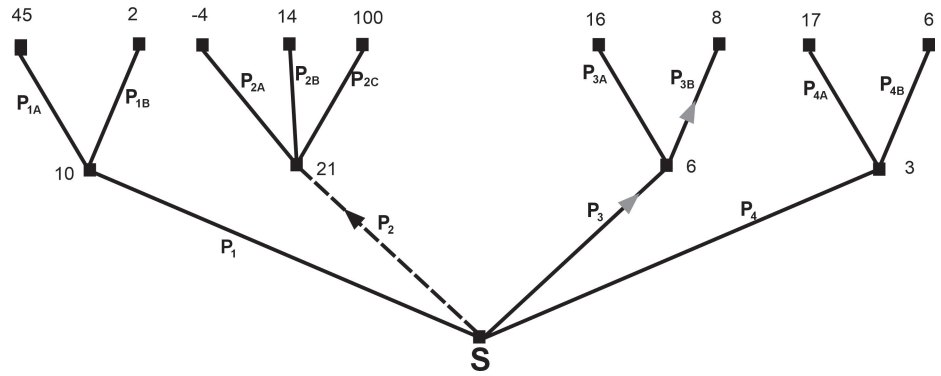

Fig. 2. A 2-ply game tree. Scores are shown at the nodes after 1-ply and after 2-ply. $P_{1,2,...,4}$ show the possible moves for the player Y from game state S. P_{NX}, $N \in \{1,2,3,4\}$, $X \in \{A,B,C\}$ show the possible moves for the opponent. The numbers at the nodes are the game state values after the moves.

3.2 Turing's hand simulation

Turing defined a computer chess algorithm which was not implemented on a computer but was simulated by hand. He defined a scoring function based mainly on material advantage (with value attached to each piece) and if a tie occurs for different moves, then a positional value is calculated based on the side to move. This positional value is calculated based on mobility, piece safety, king safety, castling etc. Turing's strategy analyzes all moves to a depth of 2 plies. Recapturing moves, capture of an undefended piece, capture of a piece of higher value, checkmates are giving priority for further analysis. He also defined a "dead" position as one in which neither side can immediately gain my making a capture.

3.3 Bernstein Chess Program

In 1958 a program for IBM 704 was written and the evaluation function was based on human-like chess moves and decisions. Moves where based on questions skilled human players ask[1]:

(1) Is the King in Check? If yes, can the king safely capture the checking piece(offense), interpose (defense) or move away?
(2) Are there exchanges to acquire material advantage for me or my opponent?(This may require an offensive (capturing a piece) or defensive (moving one's piece away or interposing a piece of lesser value))
(3) Can I castle?
(4) Can I advance a minor piece?
(5) Can I occupy an open file?
(6) Can I create a pawn chain using my piece?
(7) Can I move a pawn?
(8) Can I move a piece?

These questions were the basis for the game state evaluation and look-ahead in the game tree (restricted to 4-ply maximum look-ahead). The Bernstein algorithm is similar to Shannon's as it considers factors such as material advantage, mobility of pieces, king safety and control of key squares on the board.

3.4 Newell et al and the Alpha-Beta algorithm

Alan Newell, John Shaw and Herbert Simon [2] produced a chess program which based its move on specific goals e.g

—king safety,
—material balance,
—center control,
—piece development,
—king-side attack,
—promotion etc

. It also uses a set of move generators "similar to Bernstein's program, except that here they are used exclusively to generate alternative moves not continuations...as these are generated by a separate set of analysis generators". An evaluation of each move is explored using alpha-beta pruning to optimise the minimax game tree search. This involves eliminating branches for moves which are meaningless or disadvantageous. To illustrate the alpha-beta pruning, consider the game tree in 4. Suppose the tree from S is examined breadth-wise, from P_1 to P_4 and the possible scores for each response by the opponent, are considered in the same order. The best possible reply to P_1, P_{1B} gives the score 2, this is retained as the backup score. Next P_2 is examined, the first move P_{2A} gives a score -4 which is worse than P_{1B}, thus all other possible responses to P_2, P_{2B} and P_{2C} are eliminated/pruned. Moving on to P_3,P_4 all responses have a minimum greater than the backup score thus the branches are retained. Alpha cut-offs and Beta cut-offs are pruning which occur at odd and even plies respectively.

3.5 A Chess Mating Program

G. Baylor's, H. Simon's and P. Simon's work produced MATER I[32]. This program was then revised during 1964 by Baylor into a second version, MATER II. Both programs are designed to analyse chess positions, with checkmating the opponent as the goal.This mater program employs heuristics in move selection in a manner similar to those utilized by human players. Some of these strategies include:

—Restricting the opponents mobility(e.g.in a checkmate the opponents king has zero mobility)
—Gaining control over any connected squares around the opponents king that are free of enemy pieces.

Mater I primarily searched for checking moves, while Mater II focused on moves that either threatened checkmate in one move or most effectively restricted the opponent's mobility. The search would only continue along a chosen path as long as the opponent's mobility continued to decrease. In Mater II if a number of checking moves $(m_1, m_2, ...)$ are generated, they are ordered based on

Table 1. Safe checking moves

move	response	n_value
Qh1#	Kf2, Ke2, Kd2	3
Qa5#	Nc3, Kf1, Ke2, Kf2	4
Qe4#	Nc3, Kf1, Ke2, Kf2	4
Qe5#	Ne3, Kf1, Kd2, Kf2	4
Qe6#	Ne3, Kf1, Kd2, Kf2	4

the number of legal replies $(n_1, n_2, ...)$ that are available to the opponent. Moves with fewer legal replies are rated higher than those with more legal replies (in a way this was the basis for evaluating a move). If the number of legal replies to m_1 and m_2 are n_1 and n_2 respectively and $n_2 < n_1$, then m_2 is rated higher than n_2. If a move has zero replies from the opponent, a checkmate has been achieved and this move is selected, otherwise the move with the highest rating is further explored, provided the opponents mobility is further restricted. However, if along the path all possible checking moves have a rating lower than preceding moves, this move or node is discarded, the search goes back up the tree, thus the search is carried out in depth-first search and the discarded node is considered a dead end. Consider the end-game situation in Figure 3 with Black to move with the aim of either checking or exchanging White pieces. The analysis is as follows using Standard Algebraic Notation (SAN)[12]. Safe checking moves or exchanges are Qh1#, Qa5#, Qe4#, Qe5#, Qe6# the corresponding moves to relieve the checking are shown in Table 1.

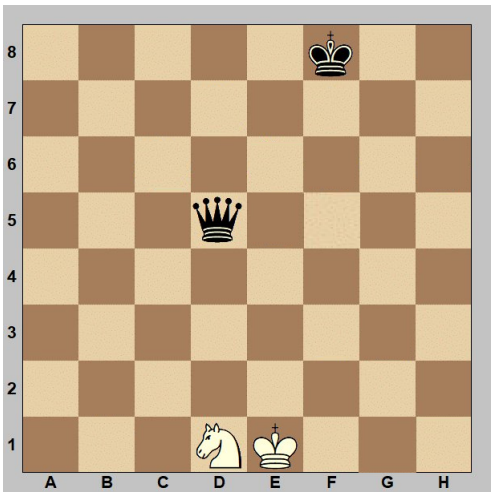

Fig. 3. Sample end game situation

3.6 The Kotok-McCarthy Chess Program Versus ITEP

John McCarthy and Alan Kotok [17] developed a chess program which was in 1967, matched against the ITEP Chess Program developed in Moscow [19]. The Kotok-McCarthy chess program utilized Shannon's type-B strategy, alpha-beta pruning with forward pruning. It also applied an evaluation function which was based on material balance, center control and piece development. The program had the capability of searching to an 8-ply depth. The ITEP program was based on Shannon's type-A strategy. The match between these 2 machine was a test of the two strategy's by Shannon - type-A and type-B. Out of 4 games played ITEP drew 2 and won 2. With the outcome of the match, one might be drawn to conclude that the type-A strategy is superior to the type-B, however, the ITEP program used a search depth of 3-ply to obtain the draw and 5-ply for the wins. On the other hand, it seems that the Kotok-McCarthy program's tree search technique was modified just before the match, but the adjustments made were not effective in selectively pruning moves which led to weak moves [20]. Create a table to show development in each era, do not forget to mention the weaknesses in each of the algorithms

3.7 Mac Hack Six Chess Program

Mac Hack Six was produced at MIT's Artificial Intelligence Laboratory by Richard Greenblatt, Stephen Crocker and Donald Eastlakein 1967 [7]. Shannon's type-B strategy was applied to Mack Hack Six along with forward prunning and alpha-beta algorithm. The program also incorporated book openings by chessmasters. The main components of Mac Hack Six's design are:

—**Plausible move generator** According to [7] about "50 identifiable heuristics" are used in computing the plausibility. Each square on the chess board is assigned a value e.g squares close to the center and the opponent's king are assigned higher values. Pieces are also assigned developmental values i.e. the value of a "piece at a position is the sum of all the values of the squares it attacks plus values of actual attacks on enemy pieces" as well as putting an opponent's piece en prise. Thus a plausible move involves calculating the difference between the current developmental value and new developmental value. This is one key component used in the tree evaluation.

—**Evaluation of the Board** The board value S is given by:

$$S = B + R + P + K + C$$

B = material balance = total value of white pieces - value of black pieces

R = piece exchange ratio=$\{N/(T-1)\} * 1/8 * M$ (where N is the ratio of white material to black material at the node evaluated, T is ratio of white to black materia at the top node of the tree, M is the material for one side at the beginning of the game.)

P = pawn structure (values are attached to pawns based on factors e.g. defended, doubled, backward etc)

K = king safety (If the queens have been captured this value is zero otherwise this is calculated as 8 times the difference between the ranks of the kings)

C = center control(+1 if there is at least one white pawn and no black in the center-four-squares and - 1 if there is at least one black pawn and no white pawn in the center-four-squares and zero otherwise)

—**Feedover conditions**
These are conditions where a player must make one or more of the following decision: save a piece, exchange a piece, sacrifice a piece or relieve a check when one or more of his pieces are en prise. The program identifies 3 possible conditions:

(1) The side to move has a piece enprise and is in check or the enprise is pinned
(2) The side to move has more than one piece enprise
(3) Both sides have one piece enprise, the enprise piece of the side to move is not pinned but that of the opponent is pinned.

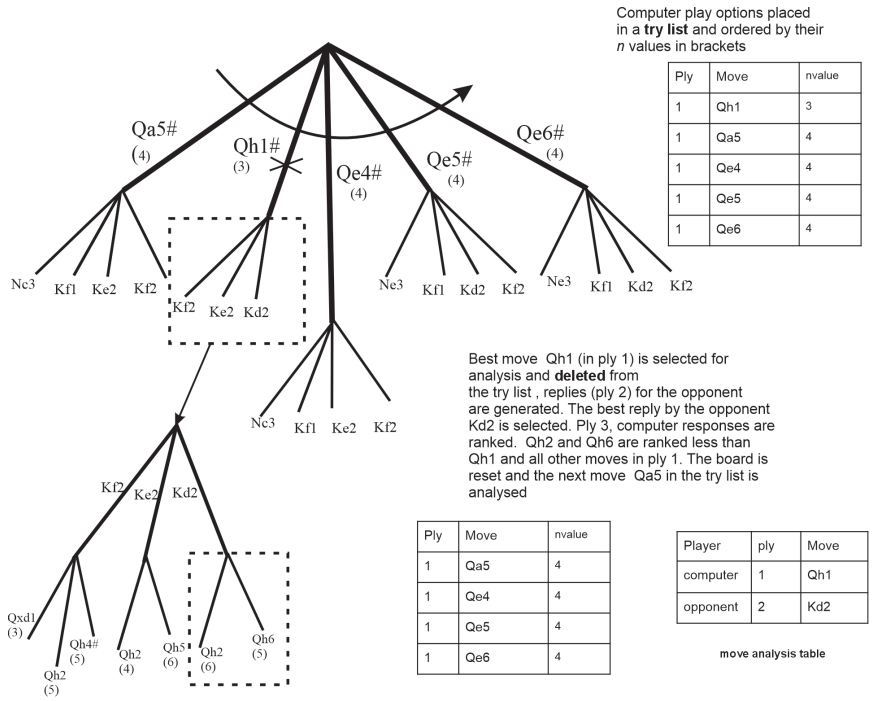

Fig. 4. Game tree analysis for Mater II, using the end game state in Figure 3. Moves which place the opponent's king in check, with minimum reply options for the opponent are given priority for analysis. Qh1# in ply 1 is the best move with minimum reply (3), if it further restricts the opponents mobility, the path is investigated in more depth, otherwise the path is retraced (Qh1# is later discarded because further investigation leads to Qh2# and Qh6# with minimum reply of 5 and 6 respectively - which is worse than the rating of Qh1#). With Qh1# discarded, Qa5# is analysed next.

—**Width of the Search**
The width of the search of the game tree is fixed at 6 for normal play at any ply. For tournaments the width is place at 15, 15, 9, 9, 7, for each node at ply 1, 2, 3, 4, 5 repspectively and 7 at depths 6 and greater. However these numbers can vary based on the following heuristics/conditions:

(1) All safe checks have been examined and all captures investigated at ply 1 and 2.
(2) Moves which expose the side to move to a checkmate are ignored.
(3) Some key pieces are prioritized to investigate a minimum number of their plausible moves.

—**Alpha-beta pruning** The alpha-beta algorithm is applied in the Mac Hack Six program. This algorithm as discussed in Section 3.4 discards a branch in the game tree if the move leads to a worse value for the side to move than a previously considered move. This pruning technique speeds up the game tree search tremendously!

—**Hash coding** The application of hash coding is a major advancement in the Mack Hack Six, same board game positions are never evaluated twice. if a board state is revisited its evaluation is simply retrieved from the hash table which speeds up the computer play.

In 1967, Mac Hack VI competed in the Boston Amateur championship against human players, winning two games and drawing two games. Mac Hack VI made record: This was the first time a computer won a game in a human tournament! In 1968 Mac Hack VI achieved a rating of 1529, above the average rating in the USCF of about 1500 [24].

4. COMPUTER CHESS CHAMPIONS IN THE 1970S AND 1980S

4.1 CHESS 3.0, 3.5, 3.6, 4.0 and 4.5

CHESS 3.0 was written by Larry Atkin, Keith Gorlen and David Slate in 1969. It was top both in the First United States Computer Chess Championship and ACM (Association for Computing Machinery) championship in 1970 [18]. CHESS 3.0 search engine was designed using the alpha-beta algorithm and an opening book program which was used to select opening moves.

CHESS 3.5 came as an improvement to CHESS 3.0 and like its predecessor, it won the ACM 1971 championship. Its design performs a depth first search to a fixed number of plies, and uses the evaluation function score to select the best moves to be searched at each depth. It uses only evaluation function as the basis for move selection and as a result misses some moves such as "sacrificial or combinatorial" moves which could lead to greater positional advantage at deeper plies. It also uses fixed width w search, at a given depth d except for a "fallback" search at ply 1 which increase the number of moves searched to a value $v > w$ to ensure that a defensive move is found against an opponents threat [9]. Next came CHESS 3.6 which employed the Shannon type-B strategy using "depth-first search, α-β and fixed depth trees" [26]. However, it performed woefully in the ACM 1973 championship. This was mainly as a result of having a "A primitive position evaluation function" and its plausible move generator which explored only the top best moves.

CHESS 3.6 was abandoned, and in 1973 CHESS 4.0 was released, which came second in the First World Computer Championship in 1974 . According to [26], modifying the evaluation program in 3.6

was a challenge, to overcome this, CHESS 4.0's tree search and evaluation function where embedded in one program and the tree search employed full width search.
Next in line was CHESS 4.5 which used bit boards (i.e. a set of 64 binary bit string which contains information of a current chess position e.g. a bit board can contain information of all squares attacked by the white queen, all squares set to 1 are the squares attacked by the white queen, thus each bit represents a square). Bit boards helped to improve speed of chess position evaluation. Hash tables were used to store computed values for functions with high CPU overhead, while ensuring that a quick table assessor method was implemented. In 4.5 the evaluation function embedded several factors e.g. material balance, hung pieces(i.e. pieces that are in danger of being captured by the player to move), positional evaluation (this funcion assigns values to pieces based on the squares they occupy), piece and position centered evaluation e.g. Bonuses for rooks for square control(number of squares attacked by the rook), kings are evaluated for safety from attack. The tree search technique in 4.5 is split in 2 parts: the **base** module and the full module. Base module conducts its search from the current board position (i.e. ply-0) by searching book opening moves, if not found all legal moves are generated and captures and hung pieces are analysed, then the ply-1 search is done using the evaluation function and the moves are ordered by the scores obtained. Next is ply-2 which is searched using the **full** module, the full module does a repetition check which ensures that moves down a line are nor repeated. It checks the transpositions/refutations (the hash table which stores best response (moves/refutations) for unique chess position explored) for a match, if found retrieves the value. The full module gives priority to searching moves with check evasion, captures, killer moves, and a considerable number of other heuristics of which the details can be found in [18]. CHESS 4.5 was designed to search to a depth of up to 10 plies and thus had a degree of brute-force advantage in tree search depth. The use of transposition/refutation tables aided in improving its search speed for best moves along with various heuristics employed. However according to the authors, CHESS 4.5 had very little knowledge base (of the game of chess), likely as a result of the computational limitation at the time.

4.2 Mr. Turk

Work began on Mr. Turk in 1967 by Garry Boos, James Mundstock and their team at the University of Minnesota. This chess engine is particularly interesting as it attempts to imitate human chessmasters' style of play. The program included opening game and middle game evaluation and human chess player look-ahead analysis i.e. analyse one line (seemingly the best for both sides) as deeply as possible "until it is shown it is no longer the best line"[9]. This one line look-ahead, according to [10] "is better than alpha beta pruning" as it is able to find the main variation relatively quicker. Its main weakness though, was its search tree, which has a maximum width and depth of 5. This placed a major limitation on the number of good moves it could find.

4.3 TECH

TECH, created by James Gillobly was the second best performing chess engine in the 1971 and 1972 ACM (Association for Computing Machinery) chess championship [18]. It uses brute-force exhaustive search to investigate all possible plays 6-ply deep, before selecting the move with maximum advantage. It does this until the endgame. TECH was not endowed with knowledge of opening moves, nor end-game moves, it simply applies material balance for its position evaluation and utilizes "the brute force of the computer in analyzing numerous positions, rather than attempting to apply 'intelligence'"[15]. Thus TECH set the standard for chess engines which enforce exhaustive search to specific depth by taking advantage of the processing speed of computers at the time.

4.4 OSTRICH

The chess program OSTRICH, was first developed in 1971 at Columbia University by George Arnold and Munroe Newborn and later refined in 1975 by Newborn [23]. It was 3rd best in the ACM 1972 championship [18]. It uses Shannon's type-B strategy along with the alpha-beta and gamma algorithms. OSTRICH has 3 separate programs: a BOOK program (for opening moves which was restricted to a maximum of 5 moves), CHESS playing program (for opening (if seleced by the human operator) and mid-game play) and END GAME program (for end game play to force a checkmate). According to [23], CHESS was built in 5 main parts:

—An input/output controller for user communication and to control tree size which would be used to determine moves.

—A move tree generator, used during a game play to search through the game tree.

—A move rating segment, which assigns a **plausibility score** to each move and ranks them in a list and maximizes cut offs of dead end moves.

—An update part for the entire program, for all pointers and lists.

After legal moves are generated, they are assigned values or a plausibility score. This score is based on a number of factors, such as captures, castling, attacks etc. After the plausibility scores are obtained, the moves are ordered and updated using this score, thus the search tree is traversed breadthwise in this order. At terminal nodes, a special scoring is done based on components such as board material, material ratio, castling, board control, minor piece development, king defense etc. Another key feature in the design of OSTRICH is the implementation of the gamma algorithm which involves terminating a search down a path for which the move sequence down that path leads to a worse position than the best move found at a branch higher up the tree. The node at the point of termination is assigned a terminal node score.

4.5 KAISSA

KAISSA was written by Mikhail Domskoy and his team as an improvement to the ITEP program in 1972 and won the First World Computer Chess Championship in 1974 [18, 23]. This program was designed to search the game tree extensively (maximum depth of 30) and apply tree pruning techniques at every level of the game tree search (i.e. increase depth and reduce width of the game tree). It applies the min-max algorithm and α-β heuristics to prune the tree at non-terminal nodes. One of these heuristics involves setting an evaluation limit for possible chess moves before the game tree search begins, thus all moves with values outside this limit are not included in the game tree. However, setting this limit may result in a decrease in a number of forcing moves and other tactical moves. The program also employed a move ordering consideration. This ordering is based on the assumption that if a move is best in several variations, then it would be the best move for the current variation being considered [3]. For each level 10 moves are stored as best moves and placed at the head of the move list. This greatly helped to reduce CPU-time in the tree search.

4.6 Belle

The first version of Belle was written in 1973 and was revised in 1974, 1978 and 1980 [6]. It won the 1978, 1980, 1981 and 1982 ACM championships as well as the 1982 world championship [18]. It was the first computer to become a Chess master. The 1980 version had specially designed hardware, utilizing the modern VLSI microchip technology, which gave it speed advantage, having the ability to search about 160,000 chess positions per second.
The custom Belle hardware has the α-β search and move generation encoded in the hardware. Belle is also endowed with opening book moves that are used for the first few moves in a game. It is designed such that there are specific registers to store material values for all pieces on the board, accumulators for incremental evaluation, which is updated each time a move is made, board positions are stored using 48-bit hash code, which aids a fast evaluation of the current board position. It also employs a transposition table, which aids quick retrieval of evaluated positions and checks for already evaluated positions to prevent repetitions. In order to select a best move, it does a full width α-β search and at terminal nodes it does a quiescence search for captures, check reliefs etc. A look up on the transposition is done at non-quiescent terminal nodes, to check for repetitions in a descendant node inorder to call a draw or to update the α-β parameters. Belle's major advantage over other computer chess engine was its processing speed. This advantage was also taken by the next chess engine to be considered - Cray Blitz.

4.7 Cray Blitz

Cray Blitz developed in 1975 became the 4th and 5th World Computer Chess Champion. Similar to Belle's engine, Cray Blitz does a full-width tree search. However, it uses "iterative deepening algorithm for timing control" [14]. Cray Blitz beat Belle at the 4th World Computer Chess Championship, even though Belle had a faster node search speed per second (160,000 nodes/sec against 50,000 nodes/sec), Cray Blitz had a unique method of eliminating quiescent searches (at the depth limit) which it considers as futile, this reduces the search tree width at the quiescent depth. However it goes 2 plies deeper at quiescent depth to find checks and avoid mates.
Cray Blitz maintains 3 hash tables: a transposition table, a pawn hash table and a king safety table. The transposition table prevent repeated evaluation of an identical or similar chess positions. It maintains a lower/upper bound value for good positions, any position which falls out of range is cutoff. This helps to save time to generate moves for positions which are not fit for consideration. "It stores the position, its value and a suggested best move for the position" [14].
Pawn structure information is stored separately in the pawn hash table. This helps to avoid repeating pawn structure evaluation, saving evaluation time. Evaluating the King's safety, defense and attack can result in considerable overhead cost. Having this information stored in the King safety hash table greatly reduces this cost, as many of the king safety positions are similar.
Another key feature is its multiprocessing algorithm. The search tree is split into subtrees at ply-1 and shared between the multiprocessors and examined for best moves. Every ply move which has been examined is marked as a tried move, if it is better than the current best, this is saved as the new current best move and the score is shared with the other processor to increase alpha-beta cutoffs. One down-side to this algorithm is that, when one processor examines the current best move, the other has to examine the second, third, fourth, etc. If the latter obtains a better score before the timing elapses, this move is stored as the best, even when the former has not been fully examined and may have obtained the best score. This challenge can be handled by giving extra time to examine the first move to completion.
A cleverly designed scoring function for endgames involving pawns. These functions are designed according to different endgame variations e.g. A pawn and king vs. king, passed pawns and King vs. King, passed pawns and king on both sides etc. The function includes counting steps for pawns to reach promotion and determine if this can be stopped by opponent king or protected by its own king, pawn races(first side to promote a pawn) was also factored into the scoring function.
This is from Ebeling Carl's book about Hitech. There are 2 schools of thought on how to play chess 1. Examine a few lines of play and a large amount of chess knowledge and experience to reason out the effects of every available course of action. difficult to code, human like play 2. Apply brute-force approach and search a large part of the game tree and apply little chess experience. This employs the strength of computers. easier to code and plays tactical chess, but no long range strategic planning

4.8 HiTech

HiTech was built in Carnegie Mellon University in the 1980s and it became the first computer to achieve a USCF(United States Chess Federation) Senior master rating [4]. It won the North American Computer Chess Championship (NACCC) (formerly known as ACM) in 1985, 1987 and 1988. It had the ability to search 175,000 chess positions per second—[13]. In HiTech, the search engine was etched into its specialized hardware with the use of Very Large-Scale Integration (VLSI) design principles. Generating moves, evaluating positions, ordering and selecting moves were done using parallel processing, which greatly sped up the processes [8]. A typical move generation could take up to 4000 move computations using about 16,000 wires of cpu hardware for the logic computations. These computations would require about 2000 packages with the TTL(Transistor-Transistor Logic) technology at the time but with the VLSI technology (where several thousand gates could be implemented on one chip) move generation computation could be split between just 64 chips! Hitech was more than a brute-force machine which had the ability to search through many variations quickly, it was also blessed with chess knowledge which enabled it play good tactical chess.
The key components of the Hitech chess engine are its **move generation**, **ordering** and **selection**. All possible moves to a square on the board is handled by a chip, thus 64 chips generate the moves for the 64 squares on the board. The generated moves are then ordered based on a number of heuristics which determines the value of moves e.g. moves involving a capture has a value equal to the value of the captured piece, if a recapture occurs, then the value of the move is the difference between the piece and its capture, moving to safe squares have a higher priority over recaptures, moving enprise pieces to safe squares are also highly valued as well as moving pieces to centre squares to gain more board control. The move ordering is done per position basis, so each time a move is made, the move ordering is recalculated and updated. Here is an example of one heuristic order value calculation of a move V_o based on the destination square.

$$V_o = \{\, 0\ , \quad safe\ squares C, \quad a\ piece\ is\ captured C-M, \quad a\ recaptur \tag{1}$$

where C is the value of the captured piece, M is the vale of the moving piece, E_x is the material value difference when a sequence of exchanges occur.
In move selection, the Hitech engine has a detailed evaluation function for its positional evaluation. Some parts of its evaluation function has components similar to that of Mac Hack VI such as material balance, pawn structure (isolated pawn, backward pawn, doubled pawn, open and semi-open files, passed pawn and king safety). Also included is board control (calculated as $\sum_i w_i \cdot v_{pi} \cdot c_i$, where i ranges over all squares w_i is the weight of the square, v_{pi} is the value assigned to the piece occupying the square and c_i is the control factor) and mobility (given by $\sum_p f_p(\sum_d c(d))$ where p ranges over all pieces, d ranges over all legally accessible squares and f_p the number of safe squares p can access)[8].
Hitech's performance rests mainly on its detailed position evaluation function which embeds considerable amount of chess knowledge and its search speed

5. DISCUSSION

Shannon classified game tree search into 2 basic classifications. One was a brute force search of the entire game tree width and depth to determine the best move using min-max algorithm (machine style). The other was a strategic in-depth line of investigation of a series of moves to determine it's viability (human style). Both search styles were further improved with the use of Newell et al's application of alpha-beta pruning to the game tree search. Building in chess knowledge: opening, mid-game and end-game moves into chess engines, endowed machines with more strategic style of play as seen in the case of Ostrich, but in championships Ostrich has been beaten by machines with less chess knowledge and more brute force search. In brute force search of the chess game tree some computations are a waste of computational effort e.g. numerous variations in chess have same evaluation, best-response and refutation, evaluating similar variations is simply a duplication of effort. The use of Hash tables to store these computed results eliminates re-computation for similar variations, thus improving search and response speed.
VLSI micro chip technology, equipped chess engines with greater search and response speed. The computer chess tournament results from the early 1980s, showed the superiority of chess engines designed with the VLSI hardware. They could traverse the width of game tree to deeper plies at a faster speed than their rival chess engines. Crayblitz, Belle, HiTech took advantage of this technology in their design. It is interesting to note that most of the chess engines which dominated the championships are those which took advantage of improved computational speed to search deeper down the game tree, with the inclusion of pruning techniques as in the cases of ITEP, Kaissa, Chess 4.5, Cray Blitz, Belle and HiTech. Mr. Turk was designed to analyse one line down the tree as deeply as possible until it is found not to be viable. The authors of Mr. Turk claim that it's more efficient than the alpha beta pruning [9] as it finds the main variation relatively quicker however, it was the least performing chess engine in the 1971 ACM championship. Its poor performance may have been as a result of the limitation placed on the tree depth (5 plies). This gives rise to an important question: Is a full width, alpha-beta and relatively deeper ply search of the game tree, a necessary condition in the design of a computer chess champion? Can a chess engine designed with Shannon's type B strategy outperform that designed with type A of same tree depth and other conditions being equal? Modern machines may already have answers to these questions.

6. SUMMARY OF CHESS PROGRAMS AND ALGORITHMS FROM 1947-1986

The first 25 years in computer chess development have witnessed impressive developments, from Shannon's simple chess position evaluation function involving just material advantage to HiTech's detailed move generation, ordering and selection, and intricate position evaluation function. At the time Turing penned down his conceptual chess algorithm which analyzed moves to a depth of 2 plies, he may never have imagined that in less than 2 decades machines would be able to analyze chess game trees to a depth of 30 plies (as in KAISSA)! These developments in chess engine designs are summed up in Table 2.

Table 2. Chess engines (Implemented/Conceptual), major features and win rate

Chess Engine (Implemented/Conceptual)	**Main Features**	**Win ratio (calculated using beta-binomial model ($Beta(5,5)$ [21])**
Shannon's Type A	Simple game state evaluation based on material advantage and one move min-max look-ahead	–
Shannon's Type B	Same Evaluation as type A with application of forward pruning of the game tree, to investigate only a few promising branches	–
Turing's hand Simulation	Material advantage for state evaluation and positional evaluation for ties in state evaluation and 2-ply deep look-ahead.	–
Bernstein's	4-ply look-ahead with game state evaluation based on human-like reasoning.	–
Newell et al's	Goal based evaluation and game tree search optimization using alpha-beta pruning	–
Baylor et al's	Goal based evaluation to achieve a checkmate	–
Kotok-McCarthy's	Shannon's type-B strategy with alpha-beta and forward pruning	≈ 0.43
ITEP	Shannon's type-A strategy with full width search	≈ 0.57 [25]
Mac Hack Six	Detailed game state evaluation involving material balance, king safety, pawn structure, center control and other tactical factors, fixed width search and alpha-beta pruning	≈ 0.64 [11]
CHESS 3.0	In-built chess knowledge for opening moves and alpha-beta pruning.	0.62 [30]
CHESS 3.5	Depth first-search of game tree to fixed depth and fixed width	0.62 [31]
CHESS 3.6	Same as 3.5, with the addition of alpha-beta pruning	0.62 [31]
CHESS 4.0	Same as 3.6 but with an improvement to program structure to improve execution speed	≈ 0.64 [28]
CHESS 4.5	Bit boards for game state representation, hash tables for computed state evaluation and more detailed (compared with 4.0) evaluation function	0.64 [29]
MR. TURK	Human-like look-ahead analysis with opening and middle game evaluation	0.38 [22]
TECH	Fixed depth, full width brute-force search	≈ 0.55 [15]
OSTRICH	In-built chess knowledge of opening, middle and end game moves. Evaluation of moves involves plausibility analysis and reordering in the game tree. Alpha-beta and gamma algorithms optimize the game tree search.	≈ 0.45 [18]
KAISSA	Extensive game tree search to maximum depth of 30 plies, with alpha-beta pruning and move ordering	≈ 0.64 [18]
Belle	Full width alpha-beta search with the use of transposition tables for retrieval of evaluated positions. Specialized hardware for quick evaluations	≈ 0.66 [18]
Cray Blitz	Full-width tree search with iterative deepening algorithm. Use of transposition tables for quick retrieval and custom hard ware for speedy evaluations	≈ 0.71 [18]
Hitech	Serial and parallel position evaluation using VLSI chips, detailed evaluation	0.72 [18]

7. FUTURE WORK

Computer chess is a field that is continually improving. The history considered so far, covers the early days of programming computer chess. This research has shown that brute force machines have dominated computer chess championships in this era. Are the more modern engines able to apply better heuristics and use less force in their search of good moves? The next research paper would bring the history of computer chess engines up to date and attempt to answer this question.

8. REFERENCES

[1] Bernstein A., Roberts M. de V., Arbuckle T., and Belsky M. A. Chess Playing Program for the IBM 704. *Western Joint Computer Conference, 157-159*, 1958.

[2] Newell A., Shaw J. C., and Simon H. A. Chess-playing Programs and the Problem of complexity. *IBM J. Res. Dev.*, 2(4):320–335, October 1958.

[3] Arlazarov V.L. Donskoy M.V. Adelson-Velskiy, G.M. Some methods of controlling the tree search in chess programs. *In: Levy, D. (eds) Computer Chess Compendium. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-1968-0_14*, 1988.

[4] Berliner H. J. Hitech Becomes First Computer Senior Master. *AI Magazine*, 9(3):85, Sep. 1988.

[5] Shannon C. Programming a Digital Computer for Playing Chess. *Philosophical Magazine Ser.7, Vol. 41, No. 314 - March*, 1950.

[6] Condon J. and Thompson K. Belle Chess Hardware. *In: Levy, D. (eds) Computer Chess Compendium. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-1968-0_28*, 1988.

[7] Greenblatt R. D., Eastlake D. E., and S. D. Crocker. The Greenblatt Chess Program. *Proceedings of the FJCC, 31, 801-810.*, 1967.

[8] Ebeling C. *All the Right Moves: A VLSI Architecture for Chess*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 1986. Ph.D. Thesis. Winner of the 1986 ACM Doctoral Dissertation Award. Later published by MIT Press in the ACM Distinguished Dissertation Series (1987).

[9] Boos G., Cooper D. W., Gillogly J. J., Levy D. N. L., Raymond H., Slate D. J., Smith R. C., and Mittman B. Computer Chess Programs (panel). *Proc. 1971 Annual ACM Conference, 25, 97-102.*, 1971.

[10] Boos G., Cooper D. W., Gillogly J. J., Levy D. N. L., Raymond H., Slate D. J., Smith R. C., and Mittman B. Computer Chess Programs (panel). *Proc. 1971 Annual ACM Conference, 25, p.100. par. 8*, 1971.

[11] Richard D. Greenblatt and Donald E. Eastlake III. The greenblatt chess program (mac hack vi). *SIGART Newsletter*, (6):8, October 1967.

[12] Davidson H.A. A Short History of Chess. *David McKay: Dysart, UK*, 2012.

[13] Hsu, F. Two designs of functional units for vlsi based chess machines. Technical Report CMU-CS-86-103, Carnegie Mellon University, Department of Computer Science, Jan 1986.

[14] Hyatt, R.A., Gower, A.E. and Nelson, H.L. Cray blitz. *In: Levy, D. (eds) Computer Chess Compendium. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-1968-0_10*, 1988.

[15] Gillogly J.J. The Technology Chess Program. *Artificial Intelligence*, 1972.

[16] Harkness K. and Battell J. S. This made Chess History. *Chess Review. February-November*, 1947.

[17] Kotok, A. *A Chess Playing Program for the IBM 7090 Computer*, pages 48–55. Springer New York, New York, NY, 1988.

[18] Levy, D., editor. *Computer Chess Compendium*. Springer-Verlag, Berlin, Heidelberg, 1988.

[19] Adel'son-Vel'skii G. M., Arlazarov V. L., Bitman A. R., Zhivotovskii A. A., and Uskov A. V. Programming a Computer to Play Chess. *Russian Math. Surveys, 25:2 (1970), 221–262*, 1970.

[20] Botvinnik M. M. Computers, Chess and Longrange planning. *New York: Springer Verlag*, 1970.

[21] Taboga M. Beta-binomial distribution, 2021. Accessed: 2025-10-13.

[22] B. Mittman. Computer chess programs (panel). PDF archived at The Computer History Museum, 1971.

[23] Newborn M. Computer Chess. *New York: Academic Press*, 1975.

[24] Newborn, M. Mac Hack and Transposition Tables.In: Kasparov versus Deep Blue. *Springer, New York, NY*, 1997.

[25] SIGART. Progress report on the kotok–mccarthy vs. itep chess match. *SIGART Newsletter*, (4):11, June 1967.

[26] Slate D. J. and Atkin L. R. *CHESS 4.5—The Northwestern University chess program*, pages 82–118. Springer New York, New York, NY, 1983.

[27] Van Den Herik, J. Computer Chess, the Chess World, and Artificial Intelligence. *Ph.D. thesis, Delft University of Technology. Academic Service, The Hague. ISBN 90 62 33 093 2. (in Dutch)*, 1983.

[28] Atkin L. W. and Slate D. J. Chess 4.0 at the 1973 ACM north american computer chess championship. In *Proceedings of the 4th Annual ACM North American Computer Chess Championship*, New York City, NY, 1973. Association for Computing Machinery (ACM).

[29] Atkin L. W. and Slate D. J. Chess 4.5 at the 2nd world computer chess championship. In *Proceedings of the 2nd World Computer Chess Championship*, Toronto, Canada, 1976. International Federation for Information Processing (IFIP).

[30] Atkin L. W., Slate D. J., and K. D. Gorlen. Chess 3.0: Northwestern university chess program. In *Proceedings of the 1st Annual ACM North American Computer Chess Championship*, pages 1–3, ACM National Conference, Houston, Texas, 1970. Association for Computing Machinery (ACM).

[31] Atkin L. W., Slate D. J., and Gorlen K. Chess 3.5 and 3.6 at the 1972 ACM north american computer chess championship. In *Proceedings of the 3rd Annual ACM North American Computer Chess Championship*, Sheraton Boston Hotel, Boston, MA, 1972. Association for Computing Machinery (ACM).

[32] Baylor G. W. A Computer Model of Checkmating Behaviour in Chess. In A. D. De Groot and W. R. Reitman, editors, *Heuristic Processes in Thinking*. Nauka, Moscow, 1966.