EVOLUTION OF COMPUTER CHESS

Yurii Moroz[1]

DOI: https://doi.org/10.30525/978-9934-26-602-7-10

Abstract. Chess has always been a subject for scientists to explore and apply new technological tools. The game has clear rules and a defined structure, which made it one of the few problems well-suited for testing new ideas in logic, computation, and machine reasoning. From early mechanical "chess players" – which created only the illusion of intelligence – through pioneers like Turing and Shannon, who introduced early algorithms and search methods for computer chess, and continuing to today, chess has provided a clear, measurable challenge inspiring exploration of machine intelligence and decision-making. *The purpose* of this paper is to demonstrate how technological progress – particularly in computing hardware, software architectures, and algorithm design – has steadily influenced the development of computer chess from its early stages to the present. It aims to show not only how advances in hardware and software shaped the evolution of chess engines but also how ideas from this field contributed to the progress of other scientific and technological domains. *The methodology* of the study is based on collecting, summarizing, and structuring key technological achievements throughout the history of computer chess. Based on original papers, the study highlights each major engine's hardware design, software structure, algorithmic innovations, tournament successes, and long-term influence on the field. *Results* of the survey showed a significant impact of technological developments on computer chess, influencing both the strength and structure of chess engines. At the same time, many ideas that emerged from building chess engines demonstrated *practical* value in other areas too. This study can inspire the development of new chess engines, tools, and heuristics to improve how artificial intelligence systems analyze the game. Moreover, the methods discussed here can be applied not only to chess, game theory, and information technology, but also to other fields such as mathematics, physics, chemistry, biology, and many

[1] Graduate Student at the Department of Computer Information Systems and Technologies, Interregional Academy of Personnel Management, Ukraine

others. The *value* of this study is in presenting a clear, structured overview of the technological side of computer chess – from early experiments to modern neural networks. This interconnection shows that the advancement of computer chess, general technology, and modern digital life are closely linked, each contributing to and benefiting from the others.

1. Introduction

For centuries people wanted to believe that the magic might be real – that there could be a robot or automaton capable of solving any problem. At the same time, chess independently was one of the most prestigious games, and those who played at a high level were regarded as brilliant minds. The peak moment when engineering was ready to build something beyond primitive but still limited came in the 18th century with the Turk, which was just a trick because technology was not ready for a real playing machine. Later, in 1900–1950, technology advanced toward building an exceptional machine; real experiments began with El Ajedrecista, and algorithms such as Minimax were invented. Then, in 1950–2000, technology evolved from early engines such as TuroChamp to Deep Blue, which defeated Garry Kasparov. Since 2000, interest in computer chess has actually increased because it became more accessible for everyone, and AlphaZero introduced new methods now used by the strongest engines. Both general technology and chess helped each other and gave something to other fields. Herbert Simon, a co-creator of one of the first-ever chess engines, NSS 1958, won a Nobel Prize in Economics. Demis Hassabis, CEO of DeepMind, whose team created a game-changer, AlphaZero, in 2016, later won a Nobel Prize in Chemistry. This work introduces and explains new approaches in computer chess that came from chess itself and different periods in its history. Using clear summaries of the most major engines and important innovations, it aims to familiarize readers with the world of chess engines and the key algorithms, which originated from computer chess experiments. The goal is to inspire readers to build better chess engines, contribute to existing ones, or even create tools that improve our lives beyond chess. Original papers by the authors were the primary sources used to keep the accuracy high. It gives short summaries of all major chess engines in chronological order to give a solid understanding of key developments in computer chess within a focused, readable report.

2. Fake Automatons (1770-1900)

Even before computers, there were attempts to build so-called "computer chess" – devices meant to mimic the human mind. The three most well-known automatons, which were actually secretly controlled by strong players, were invented before the 20th century.

The Turk was constructed by a Hungarian civil servant, Wolfgang von Kempelen, in 1770 to impress Empress Maria Theresa [1, p. 22]. It appeared as a life-sized mannequin, dressed in Ottoman robes and a turban, seated at a wooden cabinet with a chessboard on top [1, p. 23]. Before each game, Kempelen opened all the doors of the cabinet, full of mechanical gears and clockwork machinery, to convince the audience that nobody was hiding inside, claiming that the Turk was fully automated [1, p. 24]. It traveled through Europe and North America and faced many historical figures, such as Benjamin Franklin, Catherine the Great, Napoleon Bonaparte, Charles Babbage, and Edgar Allan Poe [1, p. 43]. After 84 years of existence, the automaton was destroyed in a fire in the Chinese Museum in Philadelphia in 1854, where it had been housed in its final years [1, p. 191]. While there were many attempts to explain how the automaton worked throughout its career, only three years after its destruction did the son of its last owner reveal the secret, showing exactly how it worked and how a hidden, strong player inside had been able to operate it [2, p. 3].

Ajeeb was created by a cabinetmaker, Charles Alfred Hooper, in 1865 and followed the same principle as the Turk: concealing a strong player inside. In addition to chess, it could play checkers [3, p. 2]. It also toured the world and faced many famous individuals, including Harry Houdini, Theodore Roosevelt, George Dewey, Sarah Bernhardt, and O. Henry. Unfortunately, it was destroyed by fire in Coney Island in 1929, repeating the Turk's tragic fate.

Mephisto was built by a manufacturer of artificial limbs, Charles Godfrey Gumpel, in 1876 and was technically more advanced. Unlike the Turk and Ajeeb, there was no player inside; it was remotely controlled by electromagnetic devices, primarily by Isidor Gunsberg, the runner-up in the first-ever Chess World Championship match, where he lost to Wilhelm Steinitz, marking a step toward real automation [4, p. 46]. The automaton's lifetime was significantly shorter: it was dismantled in 1889.

There were several other lesser-known automatons from that period, but all followed a similar principle or were almost exact copies of the three

described above. While these machines were nothing more than mere hoaxes, they still played a vital role in the history of computer chess by pushing engineers to create something increasingly close to a true chess engine.

3. Real Automaton and Theory (1900-1950)

The first half of the 20th century is remembered for its theoretical work, but the necessary technology was still lacking to apply these ideas in practice. Nevertheless, during that time, the first true automaton was built, and the most important theoretical concept was discovered.

El Ajedrecista, built in 1912 by Leonardo Torres Quevedo, was the first automaton capable of playing without human intervention, relying solely on electromechanical components, and could win a king and rook versus king endgame by following a simple algorithm. For example, when the black king came close enough to capture the rook, the rook moved to the far side of the board, keeping it cut off horizontally; when the kings stood in vertical opposition, the rook advanced one square vertically towards the black king, pushing it back to the last row to deliver checkmate [5, p. 4-6]. Torres mentioned that it was just a scientific toy and had no practical purpose, but he believed that any device – regardless of complexity – could be built by defining a set of rules [5, p. 2]. Although the problem was relatively simple, the machine was quite advanced for that time and can be considered the first chess computer.

Minimax is a foundational theorem in game theory for two-person zero-sum games, proven by John von Neumann in 1928. The core idea is to minimize the maximum possible loss by selecting the optimal strategies for both players. It is expressed as:

$$\max_x \min_y g(x, y) = \min_y \max_x g(x, y) = M, \tag{1}$$

where *g(x, y)* is the payoff function, *x* and *y* represent the strategies of the two players, and *M* is the value of the game [6, p. 303]. This concept was a revolutionary development for both chess and game theory and continues to serve as a primary algorithm in many strong chess engines today.

The main barriers that prevented scientists from applying their ideas to working chess engines in the first half of the previous century were no programmable machines, slow early devices, and unfinished minimax. But fortunately, things started to change in the second half.

4. Early Chess Engines (1950-1960)

Real computer chess development began after World War II. Back at that time, the first digital computer was built, and scientists started to apply their ideas into practice in many fields of life, including chess. Turing and Shannon were pioneers of experimenting with chess.

Turochamp was a computer chess program started being developed by Alan Turing and David Champernowne in 1948 but was never physically implemented due to limited computational power. In addition to piece values, the evaluation function included features such as mobility, piece safety, king mobility, king safety, castling, pawn credit, checks, and mate threats [7, p. 291-292]. The only recorded game, played in 1952 against Alick Glennie, was lost by Turochamp and took several weeks to complete because Turing acted as a human processor, manually calculating each move with paper and pencil following the algorithm. According to Champernowne, sometime earlier, Turochamp defeated his wife Wilhelmina, a beginner at chess, marking it as the first chess program known to beat a human, yet, unfortunately, the game was not recorded [8, p. 563]. Although Turochamp never became a functioning chess engine and remained a 'paper machine,' it is widely considered the first chess program.

A revolutionary paper, "***Programming a computer for playing chess***," written by Claude Shannon, was published in 1950. In his work, Shannon suggested a board representation, minimax routine, and two types of programs: Type A – a brute-force search exploring all moves; and Type B – a selective search focusing on promising moves. To evaluate the score of a position, he considered three factors: material, pawn structure, and mobility, and suggested a crude evaluation function for illustrative purposes:

$$f(P) = 200(K - K') + 9(Q - Q') + 5(R - R') + \\ +3(B - B' + N - N') + (P - P') - \\ -0.5(D - D' + S - S' + I - I') + 0.1(M - M') \quad , \qquad (2)$$

where K, Q, R, B, N, and P are piece values; D, S, and I are doubled, backward, and isolated pawns; M is mobility; and primed letters refer to Black [9, p. 260]. Shannon also estimated that there are 10^120 possible sequences of moves from the initial position – known as the ***Shannon number*** – which exceeds the number of atoms in the universe; therefore, solving chess by brute force would be impractical in the foreseeable future

[9, p. 259]. His contribution has remained enormously influential for the following computer chess engineers and enthusiasts, and most engines developed thereafter have followed Shannon's ideas.

Maniac I was a chess variant program written in 1956 at the Los Alamos Scientific Laboratory by Stanislaw Ulam, Paul Stein, Mark Wells, James Kister, William Walden, and John Pasta for the MANIAC I computer, which had a memory of 600 words, storage of 80K, 11KHz speed, and 2,400 vacuum tubes. In order to allow the machine to reach a depth of four plies, the scientists reduced the board to 6 × 6, omitting the bishops, castling, and two-square pawn moves. They used pure minimax search (Shannon's type A) with an evaluation function that scored material and mobility, resulting in an average of 12 minutes per move [10, p 174]. The first game it played was against itself, with White winning; second, it lost to a strong player who gave a queen handicap; third, it defeated a beginner who had learned the game just a week earlier [10, p 175-176]. Even though Maniac I is considered a chess variant, it was the first functioning engine – unlike Turochamp – to beat a human, albeit in a simplified chess game.

The Bernstein chess program was developed by Alex Bernstein, Michael de V. Roberts, Timothy Arbuckle, and Martin Belsky in 1957 for the IBM 704 – one of the last generations of vacuum-tube computers – with 42,000 instructions per second, 4,096 words of memory, and 70K storage. Bernstein presented the program under construction at the Dartmouth Workshop, which is widely recognized as the birthplace of artificial intelligence. Although there are about 30 legal moves in a typical chess position, the authors decided their machine would select only seven for detailed analysis on the basis of eight questions, such as "Can minor pieces be developed?" and "Can open files be occupied?" [11, p. 100]. They used selective forward pruning (Shannon's type B) with an evaluation function considering material, king defense, mobility, and area control, which took close to eight minutes to reach a depth of four plies [11, p. 103]. Bernstein's program might be considered the first known fully automated chess engine, but IBM's executives were unhappy with company resources being dedicated to chess research, and the project was discontinued.

The NSS chess program was written in the high-level Information Processing Language by Allen Newell, Herbert Simon, and Cliff Shaw

for the early JOHNNIAC computer in 1958. While earlier engines simply picked the move with the highest minimax value, the authors of NSS created a conceptually and technically much more sophisticated analysis procedure, which might be described as an approximation to the alpha-beta algorithm. The program treated all evaluation factors as goals and employed specific routines to reach them. For example, to reach the goal "material balance," a whole series of possible moves was generated, including "capture the attacker," "add a defender," and "move the attacked piece," among others [12, p. 327]. NSS became the first full-game engine to defeat a human player, a secretary who had been taught to play just one hour prior to the game. Notably, the authors' approach and techniques were a genuine breakthrough in the field – especially the combination of minimax and alpha-beta pruning – which is now used in many strong engines.

The alpha-beta algorithm is a technique for reducing unneeded branches in a game tree. John McCarthy was the first to suggest it to Alex Bernstein at the Dartmouth Workshop, but the latter was unconvinced. Then the authors of the NSS chess program used the idea in their implementation, which McCarthy later described as an "approximation".

In alpha-beta pruning, the maximizing player stops evaluating a branch if its value is greater than or equal to a previously encountered minimum (beta), as the minimizing player would not allow that line of play [13, p. 2]. Likewise, the minimizing player prunes a branch if its value is less than or equal to a previously encountered maximum (alpha), since the maximizing player would avoid it, as shown in Figure 1.

Alpha-beta pruning can dramatically reduce the number of nodes explored. For a standard minimax search tree with x nodes, the best-case scenario of the moves that are left to evaluate can be expressed by the following formula:

$$x = b^{\frac{n}{2}} + b^{\frac{n}{2}} - 1, \quad (3)$$

where b is the assumed number of legal moves per position and n is the depth of the search tree.

The number of nodes that may be cut largely depends on the move ordering. Assuming there are always 40 moves available in each position and the best move is always considered first, the number of nodes that are left to explore using alpha-beta pruning over the minimax algorithm is shown in Table 1.

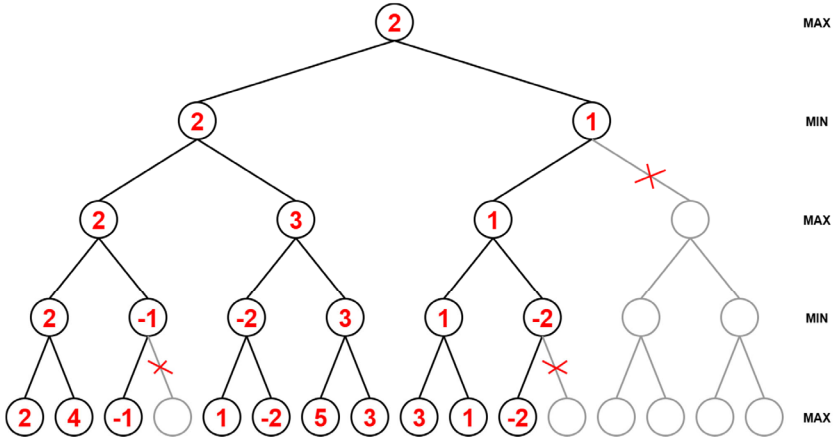

Figure 1. Alpha-beta pruning

Table 1

Minimax and Minimax with Alpha-Beta pruning (best-case)

depth	Minimax	Minimax + Alpha-Beta pruning (best-case)
0	1	1
1	40	40
2	1,600	79
3	64,000	1,639
4	2,560,000	3,199
5	102,400,000	65,569
6	4,096,000,000	127,999
7	163,840,000,000	2,623,999
8	6,553,600,000,000	5,119,999

Despite significant hardware limitations of the vacuum tube computer era, this decade is remembered for taking computer chess from theory to practice with the creation of the first chess engines and the development of foundational algorithms.

5. Transistorized computers (1960-1970)

While transistors were invented in 1947, it took over a decade to replace vacuum tube computers entirely, and by 1960 new designs were fully transistorized. It made computers more capable and practical for chess.

The Kotok-McCarthy Program was the first chess program to run on a transistorized computer – the IBM 7090, which had 36-bit architecture and 32,768 memory words and was much faster than vacuum tube computers – developed in FORTRAN and FAP by Alan Kotok, Elwyn R. Berlekamp, Michael A. Lieberman, Charles Niessen, and Robert A. Wagner in 1962 [14, p. 1]. While testing their program, the authors noted it searched too many irrelevant positions and decided to prune the move tree by combining minimax with McCarthy's alpha-beta algorithm – more advanced than the one-sided technique in NSS and involving upper and lower bounds – as he had suggested to Bernstein years earlier [14, p. 5]. The evaluation function was a weighted sum of material balance, center control, pawn structure, tempo advantage, and development [14, p. 2]. Based on Bernstein's selective search routine, the algorithm considered four moves at the first ply, three at the second, two at the third and fourth, and one move at plies five through eight, enabling a search depth of eight plies [14, p. 8]. Building on earlier work, the Kotok-McCarthy Program refined and combined existing ideas to reach amateur-level play and became a benchmark for future innovations in computer chess.

MacHack was written in the macro assembly language MIDAS by Richard Greenblatt in 1966 for the PDP-6 computer, which had 16K of memory and could execute 225,000 instructions per second [15, p. 802]. Greenblatt was convinced that Kotok and McCarthy were fundamentally mistaken in limiting their search to 4 moves at the first ply, so he decided to search less deeply but more broadly. His program considered six moves at each ply to a depth of five for normal play, increasing to 15 moves at the first and second plies, nine moves at the third and fourth, and seven at the fifth ply for tournament play. The evaluation function summed material balance, piece ratio change, pawn structure, king safety, and center control. Greenblatt incorporated about 50 heuristics into the plausible move generator, such as discouraging non-positional moves and encouraging attacks on weak spots (weak pawns, pinned pieces, pieces defending other pieces, etc.), among others [15, p. 804-805]. Greenblatt's program was the

first to use a ***transposition table***, a database that stores results of previously performed searches, which helped to avoid re-evaluating approximately 25% of positions [15, p. 806]. Additionally, with the assistance of MIT students Larry Kaufman and Alan Baisley, who added over 5,000 moves, it was the first program to utilize an ***opening book***, a database storing the best moves at the beginning of the game [15, p. 807]. MacHack became the first chess program to defeat a human in tournament play and to receive a chess rating [15, p. 801]. It became an inspiration for computer chess developers, which led to a rapid rise in chess programs and the idea of holding championships between chess engines.

As computers got better, so did chess engines. In 1965, Gordon Moore observed that transistor counts had been doubling every two years, causing hardware speed to increase exponentially; this is known as Moore's law and still holds today [16, p. 114-116]. But chess engines grew stronger not just because of faster hardware but also thanks to smarter software and better heuristics.

6. Competition and Brute Force (1970-1980)

In this decade many new engines appeared on the stage, and computer chess championships were started: ***the ACM North American Computer Chess Championship*** in 1970 and **the** ***World Computer Chess Championship*** in 1974.

Chess, the Northwestern University program, started being developed in 1968 by engineering students Larry Atkin, Keith Gorlen, and David Slate, who joined a year later, in FORTRAN and assembly language for supercomputers – the CDC 6600 and the CDC Cyber – that executed up to 3 million instructions per second – more than 10 times faster than earlier machines. It was a long-running project that dominated the computer chess scene for almost a decade. Until 1973, the program versions from 1 to 3.6 were based on Shannon's type B selective routine, like most earlier programs, to cope with hardware limitations, but then, frustrated by the code's complexity, the authors decided that big changes had to be implemented [17, p. 84]. Chess 4.0 was fundamentally rewritten in COMPASS, the CDC assembly language, switching to Shannon's type A brute-force search of all positions, taking advantage of CDC supercomputers' speed. Central to its strength were ***bitboards*** – 64-bit integers encoding board states for fast

move generation – and incrementally updated square-centric ***attack tables*** improving evaluation efficiency; ***iterative deepening***, which increased search depth over time; a sophisticated ***transposition table*** to avoid redundant calculations; and ***the killer heuristic***, prioritizing moves that previously caused beta-cutoffs [17, p. 85-88]. Proving its dominance, the Chess program won eight of 10 ACM North American Computer Chess Championships in the decade and the second World Computer Chess Championship in 1977. By the end of the decade, the last versions of the program reached the level of Expert and became the first to achieve a victory against an International Master – David Levy – in classical time control and a Grandmaster – Michael Stean – in blitz. Combining advances in hardware and innovative techniques, the Chess program became an inspiration for the next programs.

Belle was a chess machine developed by Ken Thompson, co-author of the UNIX operating system and B programming language, and experimental physicist and scientist Joe Condon. It was not just a chess program but special-purpose hardware and associated software. The origins of the program date back to 1972, when Thompson wrote a software-only version for the PDP-11. After playing in tournaments, the authors realized that full-width search produced the best chess play and the largest computer would usually win, so they decided to build special-purpose hardware and software [18, p. 45]. The first version of the chess-specific hardware Belle was built in 1976, when Condon added a hardware move generator that scanned moves using registers and microcode, supporting a full-width fixed-depth iterative search, a sophisticated evaluation function, and a transposition table. A second generation of the machine in 1978 had several refinements: internal move stacks, a static position evaluator, and a transposition memory. A third generation of Belle in 1980 had hardware move generation (64 transmitter/receiver circuits), finite-state machine evaluation for pins and pawn structure, and a 1MB transposition table with 48-bit hashing. The system ran alpha-beta search microcode on a 64-bit machine, achieving 100k–200k positions/sec, 8–12 ply depth (up to 30 plies in endgames), and a 360,000-position opening book. From the late 1970s to the mid-1980s, Belle won five ACM North American Computer Chess Championships and the third World Computer Chess Championship in 1980. Belle became the first chess engine to achieve a level of Master, and

Thompson and Condon were awarded a $5,000 Fredkin Prize. Belle proved the power of brute force search combined with hardware and influenced the next generations of programs.

As hardware improved, brute-force search quickly became a far better option than selective search. Some chess engines were able to explore a hundred thousand positions per second and were already beating grandmasters in faster time controls.

7. Microprocessors and Parallelism (1980-1990)

While microprocessors were first designed in 1968, they weren't widely used until the 1980s, when they greatly boosted the speed and depth of chess engine searches. Parallel computing also became a thing; now chess engines could evaluate many positions at once, making them more powerful.

Cray Blitz was written in FORTRAN and Cray Assembly Language by Robert Hyatt and Albert Gower in 1980 for the Cray-1 supercomputer. Hyatt started working on his undergraduate project, called **Blitz**, in 1968, but after more than a decade of moderate progress and average results in tournaments, the authors realized the necessity for a fast computer and approached Cray Research [19, p. 2]. With the help of Harry Nelson, the program was adapted to Cray architectures, including the Cray-1, X-MP, Y-MP, and finally C916, which had 16 processors and 8 gibibytes of memory. As a ***parallel search*** algorithm, Cray Blitz used ***root splitting***, then ***principal variation splitting***, ***enhanced principal variation splitting***, and finally ***dynamic tree splitting***, which divided the search tree among several processors on a shared memory parallel machine to significantly increase search speed. It also used ***vector operations*** and a hybrid ***bitboard/mailbox*** chessboard representation to accelerate attack detection and game evaluation. Cray Blitz won two ACM North American Computer Chess Championships in 1983 and 1984 and two World Computer Chess Championships in a row: in 1983 and 1986. The program played the level of Master in the 1980s and the level of Senior Master in the early 1990s. Cray Blitz's success was due to careful low-level coding, innovative ideas that saved significant time during the search, and efficient use of powerful supercomputer hardware.

HiTech was developed in 1985 by the World Correspondence Chess Champion Professor Hans Berliner and a team of researchers – Carl

Ebeling, Murray Campbell, Gordon Goetsch, Andrew James Palay, Andy Gruss, Larry Slomer, and Chris McConnell – and ran on Sun-3 and later Sun-4 workstations. For move generation and evaluation purposes, the authors built 64 special-purpose very-large-scale integrated chips, one for each square of the board, enabling analysis of 175,000 positions per second [20, p. 246-247]. In addition to alpha-beta, HiTech also used the ***B* probability-based search algorithm***, created by Berliner and McConnell, which selectively focused on promising moves by estimating uncertainty and stopped early when a clearly superior option was found [21, p. 97]. HiTech won two ACM North American Computer Chess Championships in 1985 and 1989 (tied with Deep Thought in the latter) and became the first program to achieve a Senior Master level. HiTech's dominant performance was driven by an architecture that combined large-scale search to examine all promising nodes within a wide horizon with pattern recognition that identified complex positional features.

ChipTest was built in 1986 by Carnegie Mellon University doctoral students Feng-hsiung Hsu, Thomas Anantharaman, and former HiTech co-developer Murray Campbell. Originally, Hsu planned to work on a thesis about printer controllers, but when he was asked to help with the design of HiTech, he realized that a completely different approach might be possible and decided to change the direction of his study to pursue a dream of making history by defeating a world chess champion [25, p 32]. Unlike the 64 chips used in HiTech, Hsu built a single-chip VLSI parallel move generator to significantly speed up the search [22, p. 278-279]. The team implemented ***singular extensions***, a selective search technique that deepened the search on moves that appeared much better than alternatives by performing a reduced null-window search to confirm their singularity before extending the search depth. The first version of the program, running on a Sun-3, searched 30,000 positions per second, while the second version, named ChipTest-m, running on a faster Sun-4 and already using singular extensions, reached 400,000 positions per second. ChipTest-m won the ACM North American Computer Chess Championship in 1987 and eventually evolved into Deep Thought.

Deep Thought, successor to ChipTest, was built in 1988 by the original team – Feng-hsiung Hsu, Thomas Anantharaman, and Murray Campbell – joined by Andreas Nowatzyk and Mike Browne, with Joe Hoane and Peter

Jansen contributing later. Deep Thought's main improvement over ChipTest was a hardware evaluation function that recognized dynamic positional factors like pawn structure, passed pawns, and rooks on open files. Using singular extensions – focusing on promising moves – it was able to reach remarkable depth and at one point found a checkmate in 19 moves (37 plies) [23, p. 48]. In May 1988, Deep Thought demonstrated Grandmaster-level performance at the Fredkin Masters Open, and the team was awarded a $10,000 Fredkin Intermediate Prize. In November 1988, it became the first chess machine to beat a grandmaster – Bent Larsen – in a classical time control tournament game at the Software Toolworks Championship [23, p. 44]. After joining IBM in early 1989, the team continued to refine Deep Thought and laid the groundwork for a second version. Later that year, the then-current Deep Thought – running on six processors and searching more than 2 million positions per second – lost a two-game match to the world champion Garry Kasparov [23, p. 50]. The team later developed a more advanced second version, whose main innovations were medium-scale multiprocessing, enhanced evaluation hardware, improved search software, and an extended opening book. From the late 1980s to the mid-1990s, both versions of Deep Thought won five ACM North American Computer Chess Championships and the World Computer Chess Championship in 1989. By the mid-1990s, using up to 24 processors and searching 7 million positions per second, Deep Thought II was already playing at Grandmaster level and later developed into Deep Blue.

Chess engines evolved from software programs to specialized hardware-software systems. They were now capable of playing at a grandmaster level and ready to challenge a world champion.

8. The End of Rivalry (1990-2000)

Supercomputers reached unprecedented capabilities in the 1990s, allowing chess engines to search hundreds of millions of positions per second. Beating the world's best human players became just a matter of time.

Deep Blue, the successor to Deep Thought, was built in 1996 by Feng-hsiung Hsu, Murray Campbell, and Joe Hoane; Joel Benjamin served as chess consultant and opening book author. While the project's origins date to 1985, when Hsu started ChipTest, Deep Blue itself began in 1989,

when IBM hired the team to develop Deep Thought II, also known as the Deep Blue prototype. The first version of Deep Blue ran on a 36-node IBM RS/6000 SP computer and used 216 chess chips with an overall search speed of 100 million chess positions per second [24, p. 71]. It played a match against Garry Kasparov in 1996 and achieved a victory in the first game, becoming the first machine to beat the world champion in a single game under classical time control, though it ultimately lost the match 2–4. The second version of Deep Blue ran on a 30-node IBM RS/6000 SP system with 120–135 MHz P2SC processors, each node having 1 GB RAM, 4 GB disk, and 16 redesigned chess chips (480 total); the system ran AIX 4.2 over a high-speed switch, reaching 200 million positions per second [24, p. 72]. Written in C, the chess program combined a hardware evaluation function (over 8,000 features) with a hybrid search: software used selective alpha-beta pruning with transposition tables and extensions, while the chess chips performed fixed-depth null-window and quiescence searches. Deep Blue's opening book contained about 4,000 positions; an extended book drew from 700,000 grandmaster games to support play beyond the opening, while endgame tablebases for all five- and selected six-piece positions were stored on 20-GB RAID arrays. In 1997, Deep Blue won a rematch against Kasparov with a total score of 3.5–2.5, received the $100,000 Fredkin Prize, and became the first computer to defeat a reigning world champion in a match under classical time control [24, p. 70]. After twelve years of hard work on chip designs and software, driven by the goal of beating the world champion, they succeeded, made history, and ended a human-machine rivalry.

Garry Kasparov against Deep Blue, the first match, was held in Philadelphia, Pennsylvania, from February 10–17, 1996. Although computers had already defeated strong grandmasters, Kasparov was confident that no machine could beat either him or Anatoly Karpov – the two strongest players at that time – any time soon, believing they were simply on a different level. To his surprise, he lost the very first game – the search speed of 100 million positions per second had done its job, the upset sparked worldwide interest in the match, and IBM's website briefly became the most visited site on the internet [25, p. 172-173]. Despite this early setback, during the match Kasparov spotted the weaknesses of Deep Blue, sometimes played anti-computer chess – choosing moves that exploited the

machine's evaluation bugs – and eventually won the match 4–2 [25, p. 185]. After the event, there was no longer any doubt that computers had reached a level where they could genuinely challenge – and even defeat – the World Chess Champion.

Garry Kasparov against Deep Blue, the rematch, took place in New York City from May 3–11, 1997. Unlike the quieter first match, the rematch attracted significantly greater attention – more than two hundred journalists attended the press conference, major broadcasters covered the event in full, and it attracted public interest far beyond the chess community [25, p. 217]. Although Kasparov was aware of Deep Blue's improvements and the one game he lost previously, he remained confident in his abilities and started the rematch by winning the first game. The second game became a turning point, where the machine played an unexpected move, after which Kasparov accused the Deep Blue team of human intervention – 20 years later, he admitted he had been wrong [26, p. 188] – and stated that that game decided the match because he couldn't recover. With the score tied before the final game, Kasparov decided to play a risky line in the opening, not expecting Deep Blue to have a piece sacrifice in its opening book, but the machine responded precisely, and in under an hour, the match and chess history were decided [25, p. 257].

The quest for the holy grail was finally over: man as toolmaker triumphed over man as performer [25, p. 258].

9. Multicore Processors (2000-2010)

While there were several strong engines in this decade, such as six-time World Computer Chess Championship winner **Deep Junior** and three-time winner **Shredder**, most of them were commercial, and their algorithmic details were kept secret. But one of the most influential was an open-source project, **Fruit**, that didn't win major events but became an inspiration for many other top engine developers.

Fruit was written in C by a French computer chess programmer, Fabien Letouzey, in 2003 [27]. Fabien wrote clean code that was carefully debugged and easy to understand, making it highly maintainable. Fruit was an open-source project until version 2.1 in 2005, and many of its innovative ideas – such as late move reduction and tapered evaluation – became widely adopted in other engines, significantly influencing computer chess

development. ***Late move reductions*** is a search technique that reduces the depth of moves appearing later in the move list after sorting, under the assumption they are less likely to cause cutoffs, and selectively re-searches them at full depth if the reduced search fails high, thereby significantly reducing the effective branching factor. ***Tapered evaluation*** is a technique that ensures a smooth transition between game phases by weighting opening and endgame evaluations based on the material remaining on the board, helping to avoid sudden score shifts (evaluation discontinuities) that can occur after exchanges. Fruit was the runner-up at the 2005 World Computer Chess Championship, surpassing the then-dominant engines Junior and Shredder. It remains one of the most influential chess engines that helped computer chess make a rapid and significant leap in playing strength.

Personal computers and the internet made it easier for people to connect and share ideas. By this time, anyone with the right knowledge could develop their own chess engine, driving innovation and new possibilities in computer chess.

10. Neural Networks (2010-2020)

As the World Computer Chess Championship became obsolete, the ***Top Chess Engine Championship*** emerged as the most prestigious tournament in the field. While one of the first attempts to use neural networks as an evaluation function dates back to 1994 with **NeuroChess** by Sebastian Thrun [28, p. 1069], the real groundbreaking progress came with AlphaZero.

AlphaZero was developed by Google DeepMind in 2017, led by David Silver, Thomas Hubert, Julian Schrittwieser, and Matthew Lai. Two years before working on chess, the team developed a Go algorithm that, with its larger board and more complex permutations, became the first to defeat a world champion. AlphaZero replaced the handcrafted evaluation function and alpha-beta pruning with a deep neural network and Monte Carlo tree search [29, p. 16]. ***The neural network*** had an input layer encoding the chess position, 19 residual blocks, and two heads: a policy head for a distribution over 4,672 moves and a value head estimating the position's evaluation. ***The Monte Carlo tree search*** prioritizes high-probability moves from the policy head and evaluates positions using the value head, following the steps of selection, expansion, simulation, and

backpropagation. AlphaZero was trained in 700,000 steps (mini-batches of size 4,096), using 5,000 first-generation tensor processing units (TPUs) for self-play games and 64 second-generation TPUs for neural network training [29, p. 4]. While typical chess programs used handcrafted features, pruning strategies, and heuristics, AlphaZero relied solely on the game rules and exceeded top engines after only four hours of self-play training. Running on a single machine with 4 first-generation TPUs and evaluating 80,000 positions per move (compared to Stockfish's 70 million), AlphaZero won its first 100-game match 64–36 (28 wins, 72 draws, 0 losses) [29, p. 27]. While recognizing the achievement, some critics raised concerns about Stockfish's setup: outdated version, limited hash size, fixed time per move, and absence of an opening book. Responding to criticism, the team configured Stockfish with 44 threads on 44 cores, a 32 GiB transposition table, and 6-men Syzygy bases, and under a 3-hour plus 15-second increment time control, the updated AlphaZero won the follow-up 1,000-game match 574.5–425.5 (155 wins, 839 draws, 6 losses) [29, p. 19-20]. This innovative approach, especially using neural networks for evaluation, has been widely adopted by top engines.

Leela Chess Zero is an open-source project initiated by Gary Linscott in 2018 as an adaptation of Gian-Carlo Pascutto's Leela Zero Go for chess, inspired by the breakthrough of DeepMind's AlphaZero and employing a similar methodology that combines deep neural network with Monte Carlo Tree Search. Originally using convolutional neural networks, it trained T networks through self-play and J networks via supervised learning on T networks' games and – with the help of thousands of volunteers – reached grandmaster level within months. In 2022, Leela switched from ***convolutional*** networks, mainly used in image recognition, to ***transformer*** architectures, typical of language models, improving the handling of long-range dependencies and significantly increasing strength. Its strongest models encode chess-specific topologies – like square pairs connected by piece moves – into attention via trainable biasing, helping prioritize strategically relevant squares [30, p. 1-2]. Each of the 64 square tokens contains a linear projection of eight one-hot vectors (each 12-dimensional), encoding piece history, en passant and castling rights, the 50-move rule and repetition data, and a learnable positional embedding [30, p. 6]. Additional gains came from ***attention biasing***, ***the Smolgen module*** for

adaptive attention, ***shallower FFNs and heads***, and ***a flattened input layer*** encoding the full board state. Leela Chess Zero has won two Top Chess Engine Championships and played in 13 superfinals, becoming the main rival of Stockfish. Leela's success comes from its open-source nature and the community whose contributions made all the difference and laid the groundwork for hybrid evaluation functions.

With rapid technological advances, computer chess continued to be a proving ground for new ideas. Cloud computing, Big Data, and Deep Learning enabled the rise of new methods. In this decade, neural networks changed the game, showcasing their dominance over the handcrafted evaluation functions that had been the standard since the beginning of computer chess development.

11. Evaluation Evolution (2020-2025)

Neural networks, designed to simulate the human brain, revolutionized computer chess by surpassing traditional evaluation methods. Now, engines play both more human-like and much stronger than ever before.

Stockfish is an open-source chess engine written in C++ by Tord Romstad, Marco Costalba, and Joona Kiiski in 2008, with Gary Linscott joining later; it is now developed and maintained by the Stockfish community [31]. Techniques that made it one of the strongest in its early years were ***relaxed singular extensions, logarithmic late move reductions, complex king safety evaluation, fine-tuned evaluation tables and constants, speed-optimized code, and aggressive late move pruning at low depths***. After implementing the testing framework Fishtest in 2013, Stockfish advanced rapidly, topping rating lists, winning most major tournaments, and remaining dominant since. ***Fishtest*** is a web application written mainly in Python where developers submit patches with new ideas and improvements, contributors volunteer processing power to run tests, and statistically significant ones are accepted for new engine versions. Another major boost came in 2020 with a hybrid evaluation adding ***efficiently updatable neural networks (NNUE)*** – originally created by Yu Nasu for Shogi and later adopted by Stockfish – to its traditional alpha-beta search algorithm, enabling efficient CPU-based evaluation, unlike AlphaZero and Leela Chess Zero, which require TPUs and GPUs. It had a compact architecture with an

82,048-bit Half-King-Piece input, two parallel 256-node shared-weight layers, two fully connected 32-node layers, and a clipped ReLU output. NNUE in Stockfish has been refined over time, now using eight subnetworks indexed by piece count – ***SmallNet*** for large material imbalances and ***BigNet*** for detailed evaluation – with larger layers and improved training. NNUE updates its accumulator incrementally after each move, avoiding full reevaluation; trained with supervised and reinforcement learning on billions of positions – including data from the open-source Leela Chess Zero collaboration – it significantly improves accuracy and efficiency. In 2023, Stockfish removed the handcrafted evaluation function, which had been a core method since the very beginning of computer chess, as it was no longer beneficial to engine performance. Stockfish has won 18 Top Chess Engine Championships, making it one of the most successful engines in computer chess history. Stockfish's key to success is its community: hundreds of developers and thousands of testers and contributors worldwide who have volunteered nearly 20,000 years of CPU time to play about 10 billion games.

The decade is not over yet, but it brought a major shift from handcrafted to neural network evaluation. What the future holds for computer chess remains to be seen.

12. Elo Progress (1950-2025)

It is not easy to estimate chess engines' strength across different times. The earliest engines did not participate in tournaments. Also, some engines, such as Deep Blue and AlphaZero, were built to play one match only. Furthermore, various rating lists use distinct methods to evaluate engine strength.

We used multiple data sources to build a graph of engines' approximate strength: Hans Moravec's "Robot" [32, p. 71] – for Bernstein, MacHack, Chess, Cray Blitz, Belle, HiTech, ChipTest, Deep Thought, and Deep Blue; L. Stephen Coles' "*Computer Chess*" [33] – for NSS; Alan Kotok's thesis [14, p. 16-17] – for Kotok-McCarthy; performance rating in its matches against Stockfish – for AlphaZero; and the Computer Chess Rating Lists (CCRL) [34] – for Leela Chess Zero and Stockfish. Computer chess engine progress is shown in Figure 2.

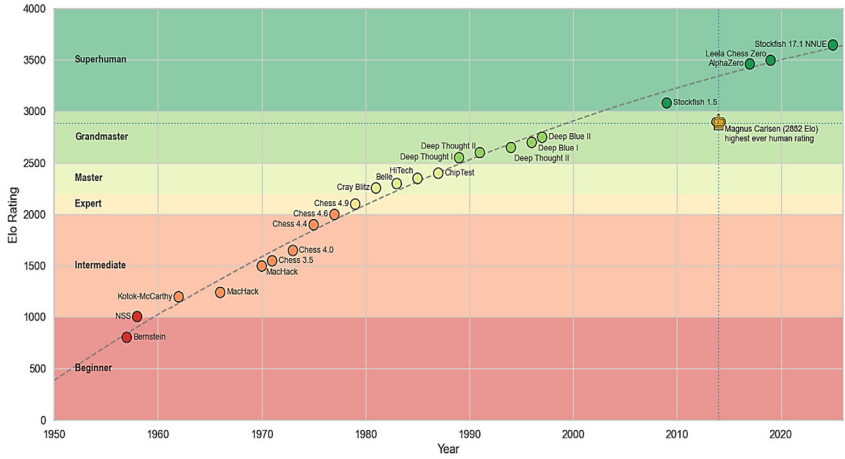

Figure 2. Computer chess engines progress

Chess engines' strength increased consistently through the second half of the 20th century, but the pace of improvement has gradually declined since achieving superhuman level.

13. Conclusion

In this study we presented a structured, technical overview of major chess engines. Both technology in general and chess in particular helped each other to push the boundaries. While hardware advancements provided more power and speed, chess engines leveraged this to perform deeper searches and ultimately improve play. At the same time, the new methods that were invented while working on chess were later used in many other fields far beyond chess, game theory, or the technical world. This report helps build the bridge between chess and technology by presenting a clear summary that tracks the evolution of computer chess.

The main takeaways from the research:

– Chess was and still is a good tool to test innovative methods.

– Technology and computer chess evolved simultaneously.

– Hardware progress boosted chess engines' capabilities.

– Software techniques evolved together with developments in computer science.

– Many ideas, implemented in chess, became useful outside the game.

Computer chess development can sometimes lead to breakthroughs in other fields. From Alan Turing and Claude Shannon, who worked across many domains, to the co-creator of Unix, Ken Thompson. From the Nobel Prize winner in Economics in 1978, Herbert Simon, to the Nobel Prize winner in Chemistry in 2024, Demis Hassabis. Many researchers who worked on chess have contributed significantly to other disciplines. The hot topic today is neural networks that are designed to simulate the human brain. They have already become the main method to evaluate positions on the chessboard, replacing handcrafted evaluation functions. With many powerful big data tools, one of the ways chess engines might evolve is by modeling human thinking. One example is Maia – a neural network chess model that captures human style. Studying computer chess might be a very fruitful experience for anyone. Not only is it engaging, but it can also lead to life-changing ideas for humanity. Chess was one of the best tools for this decades ago – and nothing has changed since.

References:

1. Standage, T. (2002). *The Turk: The life and times of the famous eighteenth-century chess-playing machine*. Walker & Co.

2. Mitchell, S. W. (1857). Last of a veteran chess player. *Chess Monthly: An American Chess Serial, 1.*

3. Hooper, C. A. (1885). *The Adventures of Ajeeb: The Wonderful Chess Automaton.* Chas. H. Woeltje & Company, Printers.

4. Gumpel, C. G. (1889). *"Mephisto", the marvellous automaton, exhibited at the International Theatre, Exposition Universelle, Paris, 1889*. T. Pettitt & Company.

5. Vigneron, H. (1914). *Les Automates [Robots].* La Nature.

6. von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. Mathematische Annalen [*On the Theory of Games of Strategy*], *100*(1), 295–320. https://doi.org/10.1007/bf01448847

7. Turing, A. M. (1953). *Faster than thought: A symposium on digital computing machines* (B. V. Bowden, Ed.). Pitman.

8. Turing, A. M. (2004). *The essential Turing, seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life plus the secrets of enigma* (B. J. Copeland, Ed.). Oxford University Press.

9. Shannon, C. E. (1950). XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 41*(314), 256–275. https://doi.org/10.1080/14786445008521796

10. Kister, J., Stein, P., Ulam, S., Walden, W., & Wells, M. (1957). Experiments in Chess. *Journal of the ACM, 4*(2), 174–177. https://doi.org/10.1145/320868.320877

11. Bernstein, A., & Roberts, M. de V. (1958). Computer v. chess-player. *Scientific American, 198*(6), 96–106. https://doi.org/10.1038/scientificamerican0658-96

12. Newell, A., Shaw, J. C., & Simon, H. A. (1958). Chess-Playing programs and the problem of complexity. *IBM Journal of Research and Development, 2*(4), 320–335. https://doi.org/10.1147/rd.24.0320

13. Edwards, D. J., & Hart, T. P. (1961). The Alpha-Beta Heuristic. http://hdl.handle.net/1721.1/6098

14. Kotok, A. (1962). *A chess playing program for the IBM 7090 computer* [Thesis, Massachusetts Institute of Technology]. http://hdl.handle.net/1721.1/17406

15. Greenblatt, R. D., Eastlake, D. E., & Crocker, S. D. (1967). The Greenblatt chess program. In *The November 14-16, 1967, fall joint computer conference*. ACM Press. https://doi.org/10.1145/1465611.1465715

16. Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, *38*(8).

17. Slate, D. J., & Atkin, L. R. (1977). CHESS 4.5 – The Northwestern University chess program. In *Chess skill in man and machine* (pp. 82–118). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-06239-5_4

18. Condon, J. H., & Thompson, K. (1982). Belle chess hardware. In *Advances in computer chess* (pp. 45–54). Elsevier. https://doi.org/10.1016/b978-0-08-026898-9.50007-3

19. Hyatt, R. M. (1983). *Cray Blitz: A computer chess playing program* (Doctoral dissertation, University of Southern Mississippi).

20. Berliner, H. J. (1985). Hitech wins North American computer-chess championship. *ICGA Journal, 8*(4), 246–247. https://doi.org/10.3233/icg-1985-8412

21. Berliner, H. J., & McConnell, C. (1996). B∗ probability based search. *Artificial Intelligence*, *86*(1), 97–156. https://doi.org/10.1016/0004-3702(95)00092-5

22. Hsu, F.-H. (1987). A two-million moves/sec CMOS single chip chess move generator. *1987 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, XXX*, 278-279.

23. Hsu, F.-H., Anantharaman, T., Campbell, M., & Nowatzyk, A. (1990). A Grandmaster Chess Machine. *Scientific American*, *263*(4), 44–51. https://www.jstor.org/stable/24997060

24. Hsu, F.-H. (1999). IBM's Deep Blue chess grandmaster chips. IEEE Micro, 19(2), 70–81. https://doi.org/10.1109/40.755469

25. Hsu, F.-H. (2002). Behind Deep Blue: Building the computer that defeated the world chess champion. Princeton University Press.

26. Kasparov, G., & Greengard, M. (2017). *Deep Thinking: Where Machine Intelligence Ends and Human Creativity Begins*. PublicAffairs.

27. Fruit – pure playing strength. (n.d.). http://www.fruitchess.com/

28. Thrun, S. (1994). Learning to play the game of chess. *Advances in neural information processing systems*, *7*.

29. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

30. Monroe, D., & Chalmers, P. A. (2024). Mastering chess with a transformer model. *arXiv preprint arXiv:2409.12272*.

31. Stockfish. (n.d.). Stockfish. https://stockfishchess.org/

32. Moravec, H. (1998). *Robot: Mere Machine to Transcendent Mind*. Oxford University Press.

33. Coles L. H. (1994). *Computer Chess: The Drosophila of AI*. AI Expert, vol. 9, no. 4.

34. CCRL – Index. (n.d.). https://www.computerchess.org.uk/ccrl/4040/