Competitions, Controversies, and Computer Chess

I. Introduction

Claude Shannon was a remarkable scientist. In 1948 he developed a branch of mathematics known as information theory, which allowed for a quantification of what information is, and directly led to solutions in such varied problems as encryption, compression, and error correction. One of his lesser-known publications was *Programming a Computer for Playing Chess*, which appeared in 1950 and became the ground-breaking statement of the problem. [Shannon 1950] At the time the play of the best chess algorithm was described as "aimless." [Turing 1953] Today a computer has beaten the reigning world champion. In 1950 there were no programs that played chess, and Alan Turing, among others, tested their algorithm by working through every step by hand, with paper and pencil. The Deep Blue computer that beat the World Champion Garry Kasparov in 1997 had 256 processors, together capable of analyzing 200 million moves per second. [King 1997] There have been many incremental improvements and some significant ones, but on the whole Deep Blue remained just a much faster version of Turing's hand simulations, following the rules Shannon specified in his exposition. Deep Blue has no intelligence: it is not capable of realizing its errors and improving its own play. It is capable of billions of operations per second. Is this the only possible way for a computer to play chess? Many have disagreed, and hope to build a computer that makes "intelligent" decisions. Still, all of their attempts so far have failed. This paper will look at the approaches used for chess play in some detail, tracing their development, their weaknesses and strengths, and finally why one has prospered while the other has been

largely a failure. It will also look at a related controversy, the role of chess in science in general and artificial intelligence in particular: should researchers strive to write better chess programs through any way possible, or should they use chess to attain some higher goal.

II. Why play Chess?

When Shannon's article first appeared computer games were unheard of. Computers were meant to be serious tools for solving difficult computational problems, and not entertainment centers. So why study a game? In 1950 Computers were commonly regarded as large (back then they were very large!) calculators, useful for number crunching but with no other apparent applications. Shannon saw that while programming chess was "perhaps of no practical importance, the question is of theoretical interest." He lists several problems which could be solved by the same methods as chess, and explains that "chess is neither so simple as to be trivial nor too difficult for a satisfactory solution." [Shannon 1950] Chess also has the advantage of clear rules, a notion of move order that is easy to implement on the computer and an easily checkable goal. It would also be easy to track the progress of computer programs, because of the abundance of human players of all levels. Thus the initial goal was to learn how to solve a larger class of problems, of which chess was only an example. A final, and perhaps most important factor was that many of the mathematicians and computer scientists who wrote these programs were avid chess players, and could try out the programs and trace their progress themselves. In summary, chess presented a theoretical challenge with many possible practical results, and was just plain fun.

III Chess and Game Theory Background

Chess is a zero-sum, deterministic, complete information game. Said more simply, there is a winner and a loser, or the game is a draw (unlike the stock market), there is no randomness (unlike backgammon), and both players know the complete state of the game (unlike poker). Because chess is deterministic, all positions (including the starting one) are either winning for white, winning for black, or drawn. The longest possible chess game has only a finite number of moves because of the 50 move rule: if 50 moves go by without a pawn moved or piece taken, the game is declared a draw.

In the late 40s John von Neumann, one of the pioneers of Computer Science, developed a method of mathematically representing complete information games known as minimax, or game trees. [von Neumann 1947] Each node of the tree is either a max-node (a move by a player who is trying to maximize his own score) or a min-node, a move by his opponent, trying to minimize the score. Each edge represents a possible move. The positions are only evaluated on the lowest level using a utility (or payoff) function f. This function, when applied to a position, yields how attractive the position is for a particular side. Because chess is a zero-sum game the utility of the same position for the other side is just the negation of what it is for the first. The simplest utility function returns a 1 if the side for which you

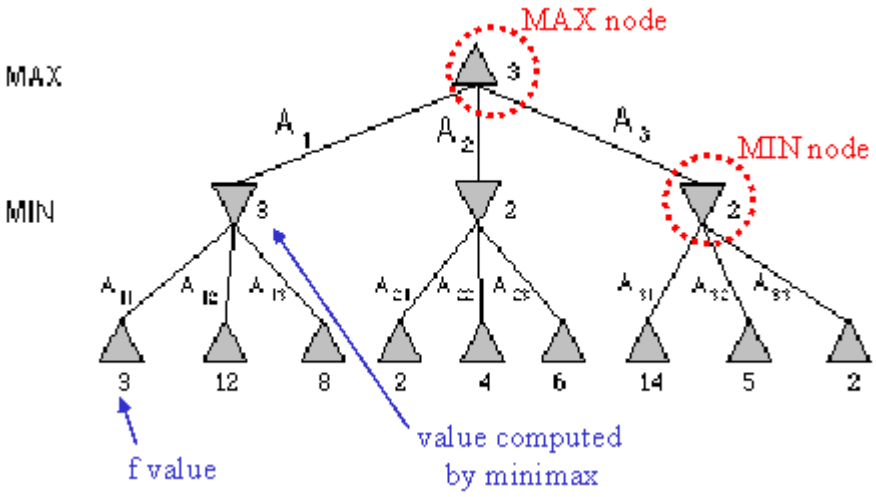


Illustration 1: A minimax tree. The maximizing player will make the move A1.

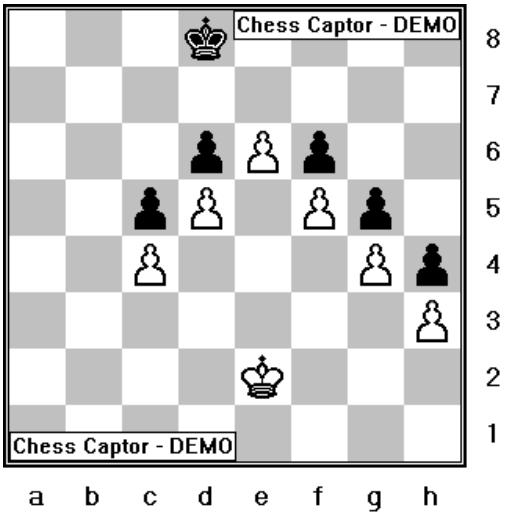
are evaluating has checkmated, a -1 if it has been checkmated, and a 0 in all other cases. Because in chess it is almost always impossible to search deep enough to find a checkmate, the utility function has to be much more complicated, taking into account material balance and positional considerations. While it is easy to assign approximate point values to pieces, it is hard to quantify their worth in a particular position: the value of pieces changes during the course of the game, with knights stronger than bishops in the middle game, and weaker in the endgame, while the value of pawns grows as fewer of them are left on the board. Positional considerations are even harder to approximate in point values. While everyone agrees that a passed pawn (one whose path to the 8th rank is not blocked by any opposing pawns) is good while an isolated one (which cannot be defended by another pawn) is bad, putting the exact point values on these aspects is impossible. It is also hard to estimate how "safe" a king is from attack and when it is time to start using it as an attacking piece. The design of a good utility function is a very hard task, and a good function is essential for the creation of a strong algorithm.

IV Shannon's two methods

Shannon's ground-breaking work not only suggested the problem, but also offered two potential ways of solving it: a thorough search of all possible variations (hereafter type A strategy), or a human-like consideration of only the better moves, played out until it is clear which side has the edge (type B strategy).

The game tree in type A strategies consist of all legal moves, each one considered in turn. For each of the moves all of the possible responses have to be considered, all of the responses to those moves, etc. The evaluation ends a certain number of half-moves, or ply into the search. An average middle-game position has about 20 legal moves. Because

with minimax trees the only positions evaluated are those on the lowest level of the tree, doing a full search at 3 ply requires the evaluation of ~8000 positions, a search at 5 ply requires ~3200000 positions, and a search at 7 ply needs ~1280000000 evaluations. This phenomenon is known as exponential explosion: the number of positions which need to be evaluated grows exponentially in the depth. In general exponential functions are considered intractable, that is computers are unable to cope with them for all but the smallest inputs. When dealing with exponential functions getting faster computers is usually of only limited help, as a computer which is 400 times faster will only be able to look 2 ply deeper. Consequently type A strategies are necessarily near-sighted: although they do not make trivial beginner blunders, they are incapable of finding a plan which takes more then a few moves to execute (see position 1). [Brudno 2000] Second the strategies suffer from the horizon effect: after carrying out a search to its set depth it may think it is winning by some margin, but in reality it is left in a vulnerable position just a move past its depth.



Position 1: White has a trivial win, but because it requires a 10 move maneuver a trivial type A strategy will not find it.

Type B strategies are much more flexible, and consequently much harder to implement. They do not consider all possible moves, but only the “plausible” ones, as a human chess player would. The immediate problem is which moves to consider “plausible”, since it is generally impossible to decide that a certain move is not worth considering without first looking at it (humans are much better at this than computers).

Consequently, the programs which follow this approach are as good as their "plausible move generators." Since an average middle game position has on average 5 plausible moves (those which a human would consider), if a program were able to choose these moves perfectly every time it would be able to search much deeper in the tree than a type A strategy. Although the number of positions still grows exponentially, it does so much more slowly. On the other hand type B strategies, just like human players, would make "blunders" by failing to consider strong responses to their moves. Shannon suggests that the computer be "taught" standard maneuvers, sacrifices, and combinations. He makes it clear, however, that such an approach would require a very deep understanding of the game, and that humans are much better than computers in generalizing from a few examples, and that "explaining" to a computer all of the possible situations is a non-trivial task.

Another important feature in type B strategies is an open-ended search depth. While type A strategies generally terminate their search a certain number of ply (half-moves) into the search, type B strategies should run until some "quiescent" position is reached. Although this term is hard to define, basically it implies no immediate danger to either side. This approach makes the strategy able to overcome the horizon effect. Type B strategies are hard to define, and hence hard to implement on a computer. Over the course of time some features of type B strategies found their way into type A programs, but all of the best programs after about 1970 employed mainly the type A approach.

V Type A Strategies: Improved Algorithms

The 1960s saw the development of two new powerful techniques, alpha-beta pruning [Brudno 1963] and the killer heuristic. Alpha-beta was a novel and powerful technique

which was not directly related to chess, but applicable to any game trees. The killer heuristic was only of limited utility in itself, but after the development of alpha-beta became a very useful tool for narrowing down the game tree and maximizing the gain made from the usage of alpha-beta.

Alpha-beta is a pruning technique which allows a program to remove from consideration large portions of the game tree if a better move has already been considered. It provably returns the same exact answer as regular minimax, but does so much faster. If the plausible move generator works well, there is a significant increase (as much as two fold) in the depth to which the program can search in an equal amount of time. At the same time alpha-beta allows for some mistakes by the plausible move generator. Even if the plausible move generator fails to suggest a good move, it will eventually be found, though this will take longer. The development of alpha-beta allowed researchers to ignore the problem of making very good plausible move generators: if a program using a plausible move generator which picked the right move four out of five times ran as a type B strategy, it would make a blunder once every five moves. The same plausible move generator but now using a full search with alpha-beta would not make any mistakes, but

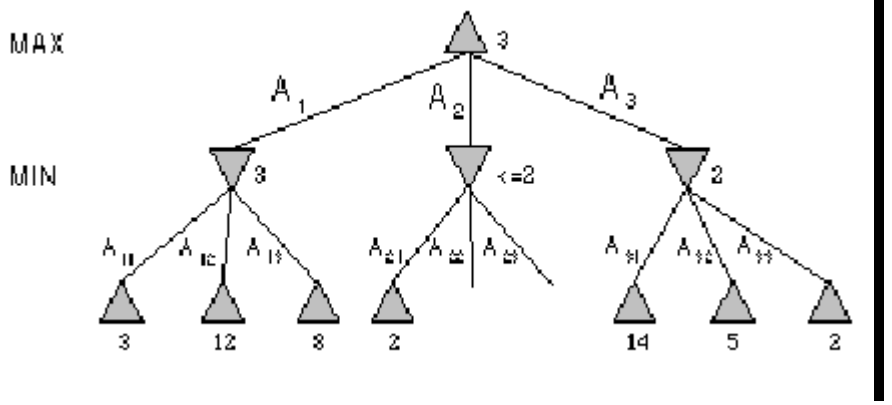


Illustration 2: Alpha-beta pruning for the minimax tree from illustration 1. In our analysis it is not necessary to consider moves A_{22} and A_{23} because the result of node A2 is guaranteed to be worse then the already-calculated result of node A1.

would take longer once every five moves. Once this increase in time usage is divided over the other four moves as well it is not very significant. Thus alpha-beta, while allowing researchers to improve other aspects of their programs, stopped development of forward pruning techniques (methods where some moves are not looked at all, as in a standard type B strategy). At the same time the game tree resulting from even the most efficient alpha-beta, where every possible elimination is made, still grows exponentially, although the exponent is halved. This means that it is still impossible to search very deeply.

Alpha-beta can be used very effectively with the killer heuristic. If during the search of the minimax tree it is discovered that a certain move that you opponent can play is very powerful (for instance it delivers a checkmate), that is the first move that your algorithm should examine in all other branches of the game tree, because if the move that the program is examining fails to stop the checkmate, it is irrelevant what other options the opponent has. By remembering what “killer” moves have been already seen and having alpha-beta analyze these moves first, it is possible to attain close to the optimal number of cutoffs.

Some researchers feel that “Brute-force alpha-beta may have done as much damage to the progress and credibility of computer-chess research as it has given chess programs high ratings.” [Donskoy 1989] The availability of such a powerful technique which is not directly chess-related has pushed the emphasis away from chess-based solution to more generic answers. Many of the key advances made between 1970 and now were based on improvements in computers and not due to any improved understanding of chess.

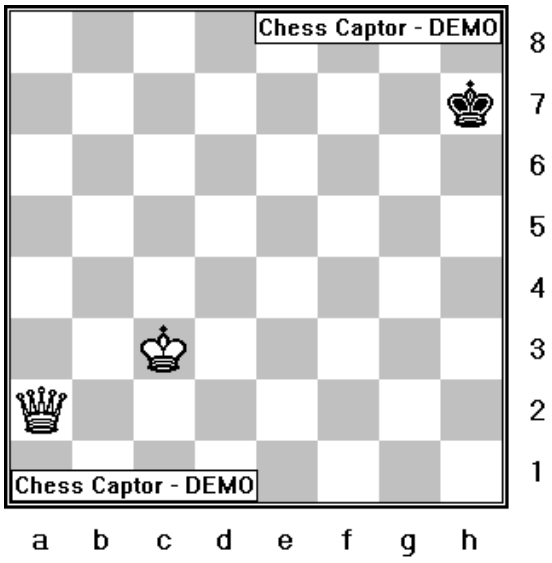
VI Type A Strategies: Improved Computers

The history of improvements in Computers during the time period is largely beyond the scope of this essay, but several trends need to be illustrated in order to explain why the development of chess-playing programs progressed the way it did. The first is what is commonly referred to as Moore's law, the fact that the fastest computer chips available double in speed every eighteen months, while their price decreases. Although a chip that is twice as fast is not really able to examine any deeper into the tree, over time the increases become substantive: Even if no other improvements are made in the computer program, every 7.5 years an old program will be able to search one ply deeper (given our assumption of 20 continuations) in the same amount of time. But it is the other phenomenon of Moore's law that has lead to the most significant increase in the strength of chess programs. Because processors get cheaper with every new generation, it has become possible to harness many of them together in order to vastly improve the performance overall.

Chess is a problem that is easily parallelized: you can give each chip its own variation to consider, wait for it to return its answer, and then just check which chip found the best line. Although this approach can not lead to very large gains because of the exponential nature of the problem, harnessing together many processors has turned out to be a useful technique. Versions of the alpha-beta algorithm for parallel systems were developed throughout the 80s [Hsu 1990], and by the end of the decade using multiple processors within a single program became common. Another consequence of the decreased cost of hardware was the ability to develop specialized hardware for chess. Because a computer chip is limited in its size and the amount of current that it can draw,

it can only support a small set of basic operations. If the programmer wants the chip to do something more complicated, he needs to do this through repeated calls to some of the basic functions. The basic operations are generally very fast, while those that need to be programmed are much slower. A generic computer chip has no notion of a chess position, the position is represented as a portion of memory containing bytes which represent pieces. By making a chess position something basic to the computer, that is by implementing it "in hardware," it is possible to speed up operations such as move generation and utility calculation. [Hsu 1987]

Another improvement in computers which has lead to improved chess programs was the larger amounts of memory available on more modern machines: the M20 computer which was used by the ITEP program (described lower) had 24 thousand bytes of memory. The computer on which I am typing this essay has 64 million. This improvement allowed programs to remember positions that it has already evaluated, so that if it encounters them again it can just see what evaluation it previously gave the position and not bother recalculating. This is referred to as a transposition table, and because transpositions commonly occur in chess play these tables allow for a remarkable increase in the possible depth of the search. This tool proves especially useful in the endgame, where the total number of positions is small, while the number of ways to get to any of them is large. This method was developed from an earlier method of backward enumeration. Using backward enumeration it was first determined that the "untouchable



Position 2 “The Untouchable King”: White to mate the black king without moving its own king. First chess problem solved by a computer before it was solved by a human.

king” problem (position 2) is a win for white [Brudno 1969]. In order to solve this problem every possible position which could occur from the movement of the two “free” pieces was given a ranking: all of the positions with the black king checkmated were given a rank of 0, all of the position from which it is possible to get to the winning position in one move are given rank one, and so on, with positions of rank N being those which are wins, if both sides play properly, in N ply. If a certain position is never enumerated then it is not winning. In addition to solving the untouchable king problem, the method of backward enumeration was used to study many simple endings, with some very novel results: for instance two bishops versus one knight endings were thought to be drawn in some cases and won in others, but a backward enumeration approach showed that the ending was always a win for the stronger side, although in some cases the win might take longer then the 50 moves allowed by the rules.

Finally programmers have used the availability of cheap disk storage to store large amounts of known opening theory into the computer. Because chess has been so widely studied it is well known that certain opening moves are much better then others. If a computer is “taught” all of these moves, it does not have to use precious time in the opening, and also does not fall into simple opening traps even if it does not really understand why a particular move is bad (because, for instance, of the horizon effect). We

will see how the computer exploited its opening book in order to defeat a very strong chess player.

Transposition tables, opening books, specialized hardware, and parallel algorithms are the only significant improvements implemented in chess programs since the early 1970s, and the three ideas arise not out of a better understanding of chess, but out of the capabilities of the computer.

VII Type B strategies

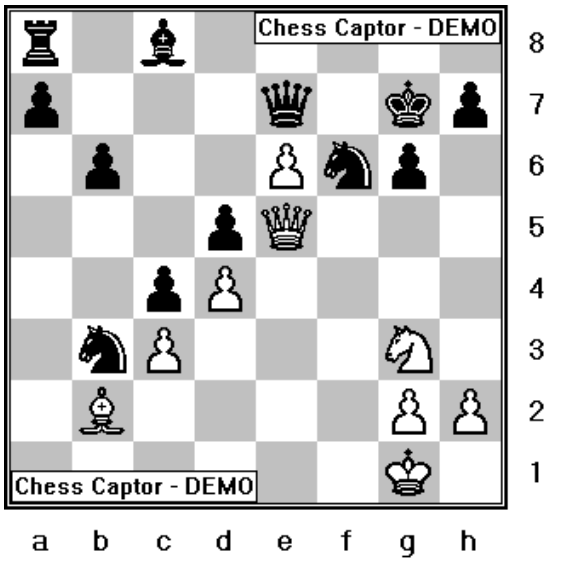
Although type A strategies are much simpler to implement, type B strategies are much more pleasing to humans. There is something aesthetically pleasing in the emphasis on cognition, rather than brute force, and hence it should not be surprising that many researchers have tried to implement a type B strategy. What is surprising is their uniform failure. Although many algorithms for type B programs have been presented in papers and at conferences, they have been failures when implemented, mainly because they make horrible blunders when actually playing. This section will concentrate on the work of the most famous proponent of type B strategies the late Soviet World Chess Champion Mikhail Botvinnik and on machine learning methods..

A. Mikhail Botvinnik's Algorithm

Botvinnik seemed to be the perfect person to develop chess programs: he was the best chess player of his generation, who held on to the world championship for almost 15 years. His education was in Electrical Engineering, a profession that gave him a deep understanding of the internal workings of the computer. Finally he was very good in not only using, but also imparting chess knowledge: he trained both Karpov and Kasparov,

two men who would go on to become world champions. It is ironic that some of these advantages turned against him in his quest to develop a chess program.

Botvinnik's interest in Computer Chess started very early. In 1968 he published his main treatise on the subject, *An Algorithm for Chess*, in which he suggested a method for finding the "weaknesses" in the opponent's position and organizing attacks against these. He spectacularly demonstrated his approach on some very hard tactical problems.



Position 3: Botvinnik-Capablanca 1938. By playing 1.Ba3!! White starts a remarkable combination which gives it a victory 12 moves later. This position was used by Botvinnik as a test case for his chess algorithms.

In position 3, for instance, white sacrifices both its bishop and its knight in order to attain a winning attack. Finding the winning sequence of moves is difficult, and Fritz II one of the strongest chess programs from the mid-90s, was unable to do so (actually finding the moves is left as an exercise for the reader). Botvinnik's algorithm succeeded in this difficult position. [Botvinnik 1968] The problem was that it failed in many simple ones. Although good at finding complicated themes and formulating powerful attacks, the algorithm was helpless when faced with a trivial situation: Botvinnik's co-worker recalled his frustration when the program would follow Botvinnik's algorithm and find the weakness in the opponent's position and attack it, while missing a much simpler checkmate in two moves. When this was remedied by making a special case in the program, it would miss taking an undefended piece. Once that was remedied another problem would come up. [Brudno

2000] Botvinnik was used to explaining the intricacies of chess to very good human players, to whom something as trivial as seeing a mate in two did not need to be explained. Consequently while Botvinnik's algorithm played well on certain hard examples, it never played a consistently strong game, and in actual matches would lose due to bad blunders long before it had any opportunity to show its brilliant tactical abilities.

The failure of Botvinnik's programs were closely paralleled by several others. In general, type B strategies, because of their narrower game trees, were able to look deeper into the position. The unsolved problem has been making them look at the strongest moves and ignore the weak ones.

B. Machine Learning

Another approach taken by researchers trying to develop a type B strategy is machine learning. Machine learning is a very active research area in artificial intelligence, and one of the most successful. Using various learning algorithms (neural networks, decision trees, support vector machines, and others) researchers have been able to design "intelligent" systems like auto-pilots for airplanes, cars which drive themselves, and programs which pick jokes for your taste. All of this success, however, came in one of the two sub-fields of machine learning, directed learning. The other, undirected learning, has been largely a failure. [Russell 1994]

In directed learning the human programmer tells the computer which parts of the data are relevant and what features to look for in the data. The algorithm is responsible for figuring out the relative importance of each of the features and whether some of the features work in groups (i.e. any single feature is insufficient, but three specific ones

together are) in order to classify the data into one of the pre-determined groups (in cases of games the positions could be classified based on how attractive they are for a specific side). In particular, the neural networks approach has been very successful in playing backgammon, where TD-Gammon achieved master-level strength just by playing against itself [Tesauro 1994]. Similar efforts in chess, however, have been failures. Chess positions are much more complicated then backgammon ones, and all of the features that are responsible for contributing to the strength and/or weakness of the position are hard to enumerate. Because directed learning techniques rely on a complete set of the relevant feature being provided by the programmer. Consequently learning chess has turned out to be a very hard problem, with no satisfactory solution to date.

History of Computer Chess

The next few sections summarize the most important matches and games played by computers over the course of the last half-century. These are of interest not only for historical reasons, but because the successes and failures in these matches affected the development of the subsequent chess programs. The competitions also attracted the attention of the press and the general public. Finally these competitions were useful for deciding which approaches to programming chess were most successful, the proof being the standings table at the end of the tournament.

VIII Cold War on an 8x8 Board

Round 1: In between 1950, when Shannon's paper was first published, and 1966 only three chess programs were developed. By 1970 six programs (none from the initial three) participated in the first US Computer Chess Championship. The first World

Championship in 1974 had 13 participants. This remarkable growth was largely spurred by a well publicized match between the Kotok/McCarthy program developed at MIT and Stanford University and a program developed at the Institute for Theoretical and Experimental Physics (ITEP) in Moscow. This match was a “first” in many ways: it was the first match between two computer programs. It was the first match where a type A strategy faced a type B strategy. But most importantly it was a challenge in the Cold War scientific race. Just as putting a man into space, it was of no practical value, but it had similar psychological implications.

Alan Kotok developed his program while an undergraduate at MIT in collaboration with several other students and under the direction of John McCarthy. His program implemented a type B strategy, considering 4 moves at the first ply, 3 moves on second, 2 on levels three and four and 1 on five through eight. The payoff function considered such elements as material (the main component), development, control of the center, and pawn structure. It also did not use several of the latest algorithmic improvements made between its initial creation (1962) and the match, most notably alpha-beta and the killer heuristic. It had a weak plausible move generator, causing Botvinnik to remark that “the rule for rejecting moves was so constituted that the machine threw out the baby with the bath water.” [Botvinnik 1967]

The Soviet program was implemented by Georgiy Adelson-Velskiy, Vladimir Arlazarov, Alexander Bitman, Anatoly Uskov, and Alexander Zhivotovsky, working in Alexander Kronrod’s Laboratory. It implemented Shannon’s type A strategy, with the search depth set as a parameter. In games 1 and 2 the machine looked ahead 3 ply, while in games 3 and 4 a depth of 5 ply was used. The payoff function was similar to the one in

the Kotok/McCarthy program, but an emphasis was placed on gaining a spatial advantage.

The match was played by telegraph, starting on November 22 1967 and continuing for a year. All games were agreed drawn if no mate was delivered or announced in 40 moves, as both programs showed complete incompetence in the endgame. In games one and two, against the weaker version of the ITEP program Kotok/McCarthy drew twice through the 40 move rule, although it was slightly worse in one of the games and much worse in the other. It was thoroughly beaten in both of the games against the stronger version, losing game three in 19 moves and game four in 41. In all of the matches the ITEP program was playing slightly better positional chess: because of the emphasis on space advantage the ITEP program was better at pushing pawns forward. It won, however, not because of any superiority in positional play, but by taking advantage of blunders on the part of the American program. [Newborn 1975] Because there were possible moves that were much better then the moves the Kotok/McCarthy program actually made, it was clear that the program failed to consider them at all, indicating a weakness in the plausible move generator. Thus the first round was won by the Russians and by the type A approach, and although the Soviet dominance in Computer chess was short-lived, the dominance of type A approaches continues today.

This match has a very sad postscript: Alexander Kronrod, the head of the Computational lab at ITEP, was a highly principled person who, among with many other mathematicians, signed a letter in defense of Esenin-Volpin, a mathematician who was placed in an insane asylum for anti-Communist views. For his signature of the letter Kronrod was reprimanded by the Communist Party. The physicists at ITEP, who were

irritated because computer time was “wasted” on game playing instead of their problems used the reprimand as an excuse to oust Kronrod from his position. At the same time Kronrod was fired from his professorship at the Moscow Pedagogical Institute. These actions effectively ended the career of this brilliant mathematician. [Landis 2000]

Round 2: The second round of the east-west battle was held in 1974, at the Computer World Championship at Stockholm. Although the tournament featured programs from eight countries, the real competition would be between the Soviet *Kaissa*[1], an improvement of the old ITEP program by Michael Donskoy, and American programs, of which David Slate’s *Chess 4.0* from Northwestern University was considered the strongest. *Kaissa* won the tournament by winning all 4 matches, but avoided a match with *Chess 4.0* when that program lost in the second round. An exhibition game held at the end of the tournament ended in a draw. *Chess 4.0,* like *Kaissa,* relied on an exhaustive search. Both of the programs added a variable cutoff depth, through which they were able to look at the more interesting variations in depth. The first World Championship was also the last won by a Soviet program. *Kaissa* finished second in 1977 and seventh in 1980. The main reason was that it was not further improved (beyond some minor modifications) during the time period. After the first World Championship the Soviet government decided that the programmer’s time was better spent working on practical projects, and Arlazarov’s group was transferred to a different institute where they concentrated on database programming, developing the Russian equivalent of Oracle. The loss, however, was an impetus for better funding for chess research in the US, and the new American programs quickly surpassed *Kaissa*.

[1] Kaissa is the legendary goddess of chess

Although the rivalry between Soviet and American programs ended soon after the first World Championship, another rivalry quickly took its place. Throughout the 60s and early 70s it was impossible to talk about computers challenging any serious chess players, but as *Kaissa* faded from the scene as a serious contender, a different rivalry was established, one that pitted the silicon-based computer against carbon-based life forms.

IX Man v. Machine

When people talk about *artificial intelligence* they think of human intelligence as a model. So it was only logical that in order to test the success of the chess programs, they would be matched up not only among themselves, but also against human players. Because human players tend to last for more years than an average computer, they provide a more consistent scale to track the improvement of chess programs. A program capable of beating a strong human player has been called intelligent by some, and the quest to create a program capable of beating the best human players has affected the development of chess programs. The history of man-machine matches is best divided into three periods, based on the strength of the best algorithms: the first is 1950-1972, during which the machine improved from the level of a weak beginner to about a class C ranking: an average high school player. The second period is 1972-1988, during which the computer rose to grandmaster level chess, and the third is 1988-1997, a period during which started with the computer defeating a series of Grandmasters and ended with the fall of the World Champion. Although throughout the time-period the improvements in the programs were incremental, these periods correspond to improvements in computing power: the programs until the early seventies ran on very weak machines, most of which were not capable of doing a full search of a tree of more than five ply. After 1970 the

improvements in algorithms and hardware allowed for effective brute force solutions which were immediately implemented in Chess 4.0 and *Kaissa*. Finally in 1988 the final step in computer advancements was made with VLSI (Very Large Scale Integration) technology, which allowed for large scale, specialized chess computers to be made.

A. The Beginnings

The first recorded game between a human and a chess algorithm was played without a computer: Alan Turing simulated his algorithm by hand. The opponent was a weak beginner. The result was an easy win for the human. The "program" had only a search-depth of one ply and could not see the trivial combination (such as forks) of its opponent. Turing resigned for his program on the 30th move. Although Turing himself described his algorithm's play as aimless, it was the first proof that there existed an algorithm for chess, one that could (and would) be improved in the later programs.

The first chess program that played with any success against human competitors was Mac Hack Six. It used a type-B strategy and was running on a very modern PDP-10 computer. In the several tournaments in which it participated it accumulated three wins, twelve losses and three draws, and a ranking of about 1400 on the USCF scale, commensurate with a strong beginner.

The early 70s saw the introduction of the first powerful full-tree search programs, *Kaissa* (1972) and Chess 4.0 (1973). The use of faster machines and more efficient algorithms pushed the strength of the programs to about 1650, the level of a strong high school player. In 1972 *Kaissa* played a match against the readers of Komsomol'skaya Pravda, a popular Russian newspaper. The previous year Boris Spassky, then the world champion also played the readers of the paper and finished with only a half a point out of

two, losing one game and drawing the other. The match was played with *Kaissa's* move posted in the paper every Sunday and the readers mailing in their response in the next few days. It surprised many when *Kaissa* matched Spassky's result. One advantage that Kaissa had in a correspondence game is that it could spend hours pondering a single move, impossible in over-the-board situations. Unlike a human, the computer is not fatigued by hours of thinking. Overall *Kaissa* played very impressively, foreshadowing its success at the first World Computer Chess Championship.

B. Target: David Levy

Throughout the 1970s and eighties the computer chess community seemed to concentrate on beating a particular human opponent: David Levy. In 1968 Levy, a master chess player and an expert on computer chess agreed to a wager of 500 pounds with John McCarthy and several others that no computer would beat him in the next ten years. Over the next few years the size of the bet increased, as more people wanted to join in on the computer's side. It should be noted that although David Levy was a very strong chess player, he was even stronger when facing a computer. He had probably more experience in playing against computers then any other strong player. He was the Tournament Director for every World and North American Computer Chess Championship through the eighties. Levy easily beat his computer opposition throughout the time period of the bet. His strategy of "do nothing, but do it well" would result in positions in which computers had the least hope of finding good moves. Consequently the computer would beat itself by making aimless moves while Levy would strengthen his position and prepare for the killer blow. During the 1978 payoff match, Levy handily defeated Chess

4.7 with a score of 3.5/5. It was apparent, however, that Levy could beat the program at will, the only game he lost it was clear that he was experimenting.

After winning his bet, Levy, together with OMNI magazine offered a $5000 prize to the first program to beat Levy. In 1984 Levy was challenged by Cray Blitz, a program based on the Cray supercomputer which had already beat several strong masters. Levy's experience at playing computers gave him a definite advantage, and Cray Blitz was shut out 4-0.

In 1989 Levy faced another challenge, this time from Deep Thought, a VLSI specialized Chess super-computer, capable of analyzing 700,000 positions per second. Deep Thought had already beaten several grandmasters, but its effortless defeat of Levy, winning all 4 games and claiming the Levy/OMNI prize showed that an era in computer chess had ended. The computer was playing grandmaster level chess, and Levy, the standard benchmark for the past 20 years was surpassed. Only one target remained at that point -- the world champion.

C. Better than the Best Human

The world champion at that point was Gary Kasparov of the USSR (now Russia). Although currently it is unclear who the World Champion is because politics have fragmented the chess throne, Kasparov remains one of the best, if not the best player. In 1989 the predictions for when Kasparov would lose varied: Kasparov himself set the time as between five and ten years, while Levy, who in 1990 wrote a piece called "How Will Chess Programs Beat Kasparov" predicted twenty to twenty five. And although in 1989 just before the Deep Thought- Levy match, the same computer also faced Kasparov in a 2

game exhibition match, and was trounced 2-0 the question was when Kasparov would fall, and not whether.

In 1996 and 1997 there were two matches between Kasparov and Deep Blue, an improvement of Deep Thought sponsored by IBM, with the name being a combination of Deep Thought and Big Blue, IBM's nickname. Both matches received wide-spread coverage in the specialized and the main-stream press. During the first match Deep Blue ran on a machine capable of analyzing 100 million positions per second. In the second, 200 million. The program was also improved based on the lessons learned from the first match. By comparison, Kasparov analyzes about three positions per second, but can change his playing style at any point of the game, not just between matches. He can also adopt to the style of his opponent, something that Deep Blue cannot do.

The first match started with a surprise: Kasparov lost the first game, the first time a computer beat the reigning champion in tournament conditions. He quickly recovered, winning game 2, and after two draws he won games five and six, winning the match 4-2. The second match started with Kasparov winning the first game and Deep Blue pulling even in the second. Three more draws set up a game six showdown. In that game Kasparov chose to use the conservative Caro-Kann defense, well suited to counter the computer's playing style. The disadvantage of using that particular opening is that Kasparov did not use it on a regular basis in tournaments, and was not an expert on it. During the opening Kasparov mistakenly interchanged the move order, setting up a brilliant finish: the computer sacrificed a piece and through very strong tactical play forced a resignation on only the 19th move. The world of chess was shocked. Kasparov was not only beaten, he was humiliated. Could it be that the computer had finally become

better then humans in chess? It has been widely argued that it was not the computer that beat Kasparov in the last game. The strategic sacrifice that Deep Thought made had been known to chess players for many years, and was entered into the computer's opening book. It would not have found the move on its own. Furthermore, the sacrifice was allowed by a blunder from Kasparov, something which happens rarely. Perhaps what this match showed best was that Kasparov was human. Regardless, he was beaten, and although it is unclear how Deep Blue would do in a rematch with Kasparov or in a match against another top-flight grandmaster, the psychological implications of the match on the chess community were devastating. There was a wide-spread feeling that Deep Blue, a large piece of silicon, was in some way more intelligent than humans could ever be. In fact Deep Blue had no intelligence. It just out-calculated Kasparov, not a very hard feat when you evaluate 100 million positions for every one that your opponent looks at. [King 1997]

X The Role of Chess in Science

Alexander Kronrod, when attacked for wasting computer time on mere games, remarked that "Chess is the *Drosophila* of Artificial Intelligence," referring to the fruit fly, the study of which led to many discoveries in genetics. This, in fact, was the emphasis of chess research during the sixties, with improvements like alpha-beta pruning and the killer heuristic developed for making strong chess programs but useful in a much wider range of problems. This emphasis was lost in the 70s. John McCarthy feels that before 1970 chess was mainly a research area within artificial intelligence, but became largely a sport afterwards. Following the analogy created by Kronrod, McCarthy suggests that "computer chess has developed much as genetics might have if the geneticists had

concentrated their efforts starting in 1910 on breeding racing Drosophila. We would have some science, but mainly we would have very fast fruit flies." He thinks that the quest to win championships and defeat players has created an atmosphere where the immediate goal of higher ratings has supplanted the larger one, that of doing good research. [McCarthy 2000]

Michael Donskoy and Jonathan Schaeffer suggest that the sporting atmosphere surrounding the chess tournaments has been detrimental for another reason: "Since the benefits of winning are high, there is little incentive for quickly disseminating new research results. Many chess programmers hold back on new ideas to ensure that their programs maintain a competitive edge. Consequently, each team of chess programmers may have to discover ideas that are well known to other teams. This can only slow down progress in computer chess." [Donskoy 1989] At the same time the competitions provide motivation to improve the programs from year to year and provide publicity for the science.

Others, like the authors of the Deep Blue program point out that chess has continued to contribute to computer science, but instead of contributing to artificial intelligence it has been useful for the study of parallel algorithms and specialized hardware. When asked when the Deep Blue project would end, Chung-Jen Tan, the manager of the Deep Blue team, said "This is really part of the overall research to understand how to use parallel processing's computational capability to solve complex problems. We have many activities going on, and chess is one of them. When we get to a point where we think we understand enough from chess to derive benefit from it for improving our understanding of parallel processing, we will stop." [Tan 1996] In fact the

parallel algorithms developed initially for playing chess are now being used to simulate protein folding, a very hard computational problem an effective solution to which would greatly facilitate drug design. [Altman 2000]

Even if one looks at the contributions of chess just to artificial intelligence, they have not been small: alpha-beta pruning and transposition tables are both basic algorithms which originated in chess research, but have been used in other settings as well. Chess has also helped to create interest in artificial intelligence in the general public. While possibly not as useful to artificial intelligence as the *Drosophila* was to genetics, it occupies a level at least equivalent to Mendel's peas, which were used to find the most fundamental laws of genetics, but abandoned soon thereafter.

XI Conclusion

If the goal of chess research is considered to be the writing of the strongest chess program, the research of the last fifty years has to be judged a great success: the world champion has been beaten, while desktop computers can compete in master-level play. If instead we define our goal to be the creation of a program which "understands" something about the position at which it is looking and is capable of formulating a "plan of action" as a human player would, the research has been largely a failure, as most of the successful programs rely on brute force to find the correct move. To some extent by selecting a criteria for success or failure one presupposes the answer. Because of this it is probably useful to look back at the original source of the problem to find an unbiased criterion.

In his ground-breaking article, Claude Shannon made it clear that "It is not being suggested that we should design the strategy in our own image. Rather it should be

matched to the strengths and weaknesses of the computer." [Shannon 1950] If one considers this to be a valid benchmark, computers have progressed substantially, taking advantage of the advances in computer science: alpha-beta pruning allows programs to search the best variations, while the worst ones can be quickly eliminated. Transposition tables have allowed computers to play endgames on a strong level, and have solved several open chess problems. Specialized hardware has allowed chess programs to improve to play on the level of the best humans. Chess programs have adapted themselves extremely well to the advances made in computer science throughout the time period. Although some people find it aesthetically displeasing that the initial problem has been solved by brute force calculation rather than through an efficient and beautiful algorithm, it should be noted that in this regard chess is an exception. Most hard problems in computer science have not been solved at all.

Bibliography

English Books: (including translations)

Adel'son-Vel'skii, G. M. et al. *Algorithms for games.* Translated by Arthur Brown, Springer-Verlag, 1988.

Botvinnik, M. M. *Computers, Chess and Long-range Planning*. Translated by Arthur Brown. Springer-Verlag, 1970.

Frey, Peter Ed. *Chess Skill in Man and Machine.* Springer-Verlag, 1978

King, Daniel. *Kasparov v Deeper Blue.* Batsford, 1997.

Levy, David *Chess and Computers.* Computer Science Press, Inc, 1976

Levy, David Ed. *Computer Chess Compendium*. Springer-Verlag, 1988

Levy, David *How Computers Play Chess.* Computer Science Press, 1991.

Marsland, T. Anthony and Jonathan Schaeffer, Ed. *Computers, Chess, and Cognition.* Springer, 1990.

Newborn, Monroe *Computer Chess.* Academic Press 1975

Russell S. and P. Norvig *Artificial Intelligence, A Modern Approach* Prentice Hall 1994

Von Neumann, J. and O. Morgenstern *Application to Chess.* in *Theory of Games and Economic Behavior*, Princeton University Press, 1947

Journal Articles and Shorter works:

Adelson-Velskiy, G.M., et al. *Programming a Computer to Play Chess*. Russian Math Surveys, 1970, 25, 221-262.

Brudno A.L. *Bounds and Valuations for Abridging the Search of Estimates.* Problems of Cybernetics vol. 10 pp. 225-241

Donskoy, M and J. Schaeffer *Perspectives on Falling from Grace.* Journal of the International Computer Chess Association, 12.3 pp. 155-163.

Hsu, F-h. *A Two Million moves/s CMOS Single Chip Move Generator.* IEEE Journal on Solid State Circuits 22.5 pp. 841-846 1987.

Hsu F-h. *Large Scale Parallelization of Alpha-Beta Search.* PhD thesis Carnegie-Mellon University, Dept. of Computer Science 1990.

Kotok, Alan *A Chess playing Program for the IBM 7090.* MIT Undergraduate Thesis, June 1962.

Landis E.M. and I.M. Yaglom *Remembering A. S. Kronrod.* Stanford Center for Computational Mathematics Technical Report 1/2000. Available at http://www-sccm.stanford.edu/pub/sccm/sccm00-01.ps.gz

Levy, David *How Will Chess Programs Beat Kasparov.* In Marshland & Schaeffer *Computer, Chess and Cognition* (see above).

Michie, D. *Brute Force in Chess and Science.* Journal of the International Computer Chess Association, 12.3 pp 127-143.

McCarthy, John *Chess as the Drosophila of AI.* In Marshland & Schaeffer *Computer, Chess and Cognition* (see above).

Shannon, C. E. *Programming a Computer to Play Chess*. Philosophical Magazine, 1950, 41, 256-275.

G. Tesauro, *TD-Gammon, A Self-Teaching Backgammon Program Achieves Master-level Play.* Neural Computation, vol. 6, no. 2, p. 215-19, March 1994

Turing, Alan *Chess.* First appeared in Bowden, B.V. ed. *Faster Than Thought,* London, Pitman 1953 pp. 286-295

Russian:

Adelson-Velskiy, G.M., et al *Programmirovanie igr* Nauka, 1978

Arlazarov, V.L. and A.R. Bitman *Obygraet li mashina cheloveka?* Shakhmaty v SSSR 1968-2 pp. 9-11

Botvinnik, M. *Algoritm igry v shakhmaty* Nauka 1968

Brudno A.L. and I.Ya. Landau *Neprikasaemyi Korol'.* Shakhmaty 1969

Donskoy, M.V. *O programme igrayuschei v shahmaty* Problemy Kibernetiki, 1974 29.169-200

Interviews:

Altman, Russ (Stanford University, personal interview by author 2000)

Brudno, Alexander L. (Israel, several phone interviews by author in 2000)

Donskoy, Michael (Moscow, communication by electronic mail by author 2000)

McCarthy, John (Stanford University, personal interview by author 2000)

Tan, Chung-Jen (IBM Research, interview by Scientific American available at **Error! Bookmark not defined.**, 1996)