# A Very Short History of Computer Chess

Hans J. Berliner
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

## 1 Abstract

Computer chess became a reality in 1944, as computers began to exist in high priority areas, and intellects began to think about how such machines could be used to play games. Von Neumann, Shannon,and Turing all made their contributions, with Shannon laying out a plan for three different types of program. Before two years had gone by some early computer scientists had implemented each of the three types. They were interesting curios that showed the nature of the problem, and that it was very difficult to produce good chess playing behavior. It was not until Greenblatt produced his MacHack VI which was able to win games in human tournaments in 1967, that interest again peaked.

All of a sudden there were other chess programs of varying levels of ability. This activity brought about the first US Computer Chess Championship in 1970, and progress was rapid after that. The Northwestern University group of Slate & Atkin dominated the 1970's and moved the status of computer chess up to the Expert level of play. In 1980, the Belle chess machine appeared and it eventually achieved Master status. After that Hitech pushed the rating up to Senior Master, Deep Thought achieved Grandmaster performance, and later Deep Blue beat the human World Champion in a six game match. These achievements were supported by a host of technical advances. This paper attempts to provide a record of these.

We also find two insights based on the data from this history:

1. In an endeavor where enumeration of possibilities in an essential feature,the speed of the process is of fundamental importance. This is obvious.
2. However, It is now clear that just as knowledge without an adequate search engine is doomed to lay fallow, so a super search engine without enough sophisticated knowledge will be literally spinning its wheels.

## 2 Early Understanding of Search

Von Neumann posited the Maximin algorithm in his monumental "Theory of Games and Economic Behavior" [9] in 1944. Five years later Turing was doing hand simulations of a chess program, and Shannon had written about three different styles of search for such a program and had also discussed the problem of quiescence. As the alpha-beta algorithm was not yet known, there was great concern about the exponential explosion in searching for the right move. An early paper [8] had determined that the average chess position had 35 legal moves. Naively, this meant that to search to depth N it was necessary to search $35^N$ leaf positions in a tree. So Shannon [6] considered a Type A program that would look at all moves, a Type B that would prune out some sub-set at each level so as to be able to search deeper, and a Type C that would play chess in some undefined way "as humans do".

By 1955, all three approaches had been tried but none with any noteable success. These were interesting experiments done by competent researchers, and they showed that the problem was very difficult; possibly too difficult. However, in the paper by Newell, Simon, and Shaw(NSS) [4] there was a

reference to a tree pruning technique that they used in order to not investigate moves in a sub-tree that had already been refuted by another move. In a critical diagram, they illustrated the algorithm with labels containing the letters alpha and beta to point to moves in the tree. John McCarthy at MIT saw that this was an actual important algorithm, and his students produced a paper [3] that described a recursive algorithm, later called the alpha-beta heuristic that would prune large parts of the search tree, and could under optimal conditions reduce the effort to $35^{N/2}$.

In 1963 there was a match between a Soviet program named Kaissa (the chess goddess) and a program at Standford University (where McCarthy now was) called the Kotok program. Kaissa won handily running on rather inferior Soviet hardware, which was compensated for by allowing them more time. Kaissa was a Type A program, and Kotok's was a Type B. We cannot be sure whether Kaissa was using the alpha-beta algorithm. It is amazing that it escaped the attention of Von Neumann, Shannon and Turing. Arthur Samuel told me [5] that he thought it was so obvious, that he never mentioned it in his papers discussing the structure of his checkers program. I, too, being a strong game player, feel it is obvous, but it can easily escape the attention of the practitioner in the same way as when we describe what we are seeing we do not realize that there is a Fourier Transform putting together the left and right images.

The alpha-beta algorithm made all the difference in the world. The NSS effort had used the algorithm, but on such sparse trees that it made little difference. When let loose on Type A trees it could save huge amounts of effort depending on how well the moves in the tree were ordered according to goodness. Greenblatt first dealt with this problem in his program MacHack VI. This was a Type B program and had a very efficient implementation allowing it to go to depth 5 plus quiescence in all parts of the tree that had not been forward pruned. It even had a special two-ply extension for the current principle variation (PV) to make sure that the value would still be stable at deeper depth. This was really the first well-engineered program. Based upon the then state-of-the-art it had everything one could expect and a lot more. I know that my efforts to duplicate what Greenblatt had done led to considerably less success. He was a master programmer and he knew enough about chess to make his program good. MacHack VI was the first program to beat a human in an organized tournament and became an officially rated Class C player of the US Chess Federation (USCF) in 1967. This was really the impetus that the field needed. I know that I felt the whole enterprise, which at times had seemed utterly hopeless, was now doable, and I determined to have a part in it. What was little understood at this time is that Greenblatt achieved his success through a combination of providing his program with sufficient knowledge to compete at the Class C level together with enough power to search out possibilites competitively with players at that level. It was understood that a computer somehow "Did not understand strategy", but this did not seem to make too much of a difference at this level of play.

MacHack VI gave exhibitions and played in human tournaments. The interest of computer programers world-wide had been aroused. In 1970, the First US Computer Chess Championship was organized under the auspices of the Association for Computing Machinery (ACM) in New York City. The play was not exceptional, and there were many human Grandmasters there who laughed along with the lesser human players. However, when the smoke had cleared in a field of six competitors (not including Greenblatt who had declined to participate), a Type B program named CHESS3.0 by David Slate and Larry Atkin had won all its games and the tournament. It played very solidly, but just how solidly was not possible to tell because it really had no competition. It is unfortunate that Greenblatt did not participate. His program was designed to defeat humans, and he felt that brute-force (Type A) programs on faster

machines might be able to beat MacHack. This was a reasonable conjecture. However, CHESS3.0 was a Type B program also, and the showdown between these two would have been historically important. As it was, a Type A program authored by Daly, running on an on-site micro-processor with no quiescence searching finished second; a possible aberation of the pairing system in such a short (3 round) tournament. This last result gives some indication of the soundness of the participating programs, each of which had some good knowledge in them, but was hardly well engineered to search effectively.

## 3 The Age of Competitive Chess

After the first US Championship, tournaments proliferated rapidly. There was a second US Championship and then it became the North American Championship. In the meantime, the International Computer Chess Association was founded and a Computer World Championship was held in Linz, Austria in 1971. That tournament was won by the program Kaissa, but some evidence revealed since the end of the Cold War casts a great deal of doubt on its victory. However, the World Championship and the North American Championship persevered, and during the 1970's it was clear that Slate & Atkin's CHESS4.X (as it kept getting upgraded) was the best around.

This was a time of great progress in computer chess. Slate & Atkin's program was originally a Type B program, as when one reads Shannon's famous paper [6] no doubt is left that this **must** be the right way. However, Slate and Atkin were experimental geniuses. They had worked hard on pruning algorithms to make valid decisions about when a move can effectively be forward-pruned. It turned out that there were so many exceptions to knowledge-oriented pruning, that they decided to see if it was not possible to make a workable Type A program.

It turned out that it was! The trick was to combine this with several other techniques. In the early days, programmers were very worried about their program over-stepping the time control. The set of moves that each program made were timed, and there was a certain amount of time to make a certain number of moves. In the beginning, programs would guess how deeply they could search in the time they were willing to allot to the next move. However, sometimes these estimates were far wrong, and programs would use up **all** their remaining time and lose on time.[1] To combat this, someone suggested doing the search by iteratively increasing the depth. In this way, small amounts of time were allocated before large amounts. At the end of an iteration, not only would the best move to that point be known, but it would be possible for the program to make a sensible decision as to whether to invest more time in further searching. This seemed like a waste of time resources, but it could be useful for better time management. However, Slate and Atkin found a way to have their cake and eat it too. By incorporating a hash-table to store a position and the values achieved for the deepest search conducted from that position, and also the best move, it was possible to save large amounts of effort. Not only did iterative deepening (as this technique came to be known) pay for itself, but it produced significant savings. These savings were due to two factors:

1. A position that had already been searched to a sufficient depth did not have to be searched again, and
2. By knowing the best move from the most recent iteration it was possible to introduce excellent move ordering that helped alpha-beta be much more efficient.

[1] In those days the techniques of programming were not what they are today, and the real-time interrupt was not available on all machines or not familiar to many chess programmers.

The move-ordering efficiency of alpha-beta was shown to be very nearly its maximum, thus reducing the effective branching factor to $35^{1/2}$ = ~6. In addition, the ability to avoid searching positions that had already been searched, reduced this number still further. Branching factors in the range of 4 - 5 were now the expectation, thus further speeding up the program. Thus, was born CHESS4.0, the first Type A program with brains.

Other things were also introduced:

- Certain moves were not counted as a ply toward max-depth. Responses to check were usually a very small set, and it was found that not counting these as a ply improved play, since it dealt better with forcing variations.
- Hash tables for pawn structure could be very useful since evaluating a pawn structure was expensive computationally, but there were very few different pawn structures in any search. So the results could usually be looked up.

By 1973 there were lots of computer chess tournaments, and lots of competitors. However, CHESS4.x won most of them, and there was no program that could mount a consistent challenge to them. There was not much computer vs. human competition so it was not possible to tell whether computers were **really** getting better on the human scale, or whether their programmers were just coming to grips with how to beat other computers.

In the summer of 1976 something very interesting happened. Chess4.5 had been invited to play in a tournament sponsored by a vintner. The tournament was held in the vineyard, and free samples were available to all participants. The tourney was organized in classes, so that players of the same caliber competed against each other. Since CHESS4.5 was then a class B program, it decided to put it into the Class A tourney so that it would not cause any problems among the aspiring prize winners. To everyone's amazement, CHESS4.5 scored 5-0 in the Class A tourney, and the players all complained that it was too strong for its section. I read these reports and could only think that the humans had been affected by the beverages being served. However, I should have known that chess players are a very dedicated lot, especially when in the hunt for prizes. Thus, it was unlikely this was the explanation. The real explanation was that CHESS4.5 was running on a new CDC machine, the CYBER 176, which was capable of running 10 times faster than the machine it used to run on, a CDC 6800. This allowed CHESS4.5 to search about 1.5 ply deeper than it had before.

This proved to be the dawn of a new era, brought on by the realization that speed, translated into depth of search, could make a tremendous difference.

## 4 The Age of Speed

In retrospect it seems ridiculous that it took most of us so long to realize this. MacHack VI was the only program around that could go to depth 5 in real time, and it was the first to achieve good results. People thought it was due to a combination of good knowledge and thorough search. CHESS3.0 also searched to depth 5, and it was never quite clear just how deep CHESS4.0 searched in the average mid-game position but it must have been 6-ply; certainly, once the CYBER 176 got into action. In the meantime it was also clear to all participants that faster was better. However, time saved by doing efficient searches could be used to do better evaluation, couldn't it? Yes, but deeper searches were worth something too.

This was the motivation for Ken Thompson of Bell Labs who had been competing on a PDP-11 with

some home-made hardware hanging from it. Ken thought that speed was the ticket, and so in 1979 he and Joe Condon constructed the new Belle, a dedicated chess computer. Belle [2] was a chess move generator with built-n evaluation hardware. It was driven by a general purpose computer whose only job was to supervise the whole process by making decisions when an iteration was finished, and doing the time management. The new Belle was based on the CHESS4.5 paper [7], and simplified many things in order to make the machine more efficient. However, in the process Belle became considerably dumber than CHESS4.5 was. However, Belle immediately became one of the top chess programs and soon became the dominant chess computer. It had a terrific outing at the US Open (for humans) in 1982. However, on certain occasions it exhibited strange behavior as when it won CHESS4.7's queen for Bishop and Knight and then proceeded to lose the game because it did not have the knowledge to invest a little of its material advantage in order to increase its positional advantage. Belle and Thompson showed decisively that speed mattered, and it mattered a lot. However, the issue of how far knowledge can be left to lag behind was one that had yet to be answered.

Inspired by Thompson, Robert Hyatt and associates moved a previously rather inept program called Blitz onto a Cray supercomputer and achieved instant success. This program became somewhat faster than Belle, and in 1983 beat out Belle at the very strong World Computer Championships, in which CHESS5.0 also participated. However, Blitz's understanding of chess was always quite suspect, and in matches held under the auspices of the American Association for Artificial Intelligence (AAAI) during 1979-1985 both Belle and Blitz showed themselves to be unable to deal effectively with well prepared humans of equal chess rating, while programs such as CHESS4.X and Hitech coped well under these conditions. The latter, Hitech, was a newcomer in 1985. It was constructed in the computer science department at Carnegie Mellon University (CMU) in late 1984 by Carl Ebeling who had designed and built a VLSI move generator, and a group of graduate students under my direction. The move generator was built so that each square on the board had one chip dedicated to it, and each chip would be initialized so that it would take account of the location of the square on the board. This design allowed asking many questions that were not possible in Belle's scheme. Within one micro-second it was possible to find all the legal moves on the board and order them according to their likelihood to be effective. It took 1 micro-second to evaluate a position, and the whole cycle of generate moves, setup new position, evaluate that, enter information in tables, manage the tree, and retract back to original position took 7 micro-seconds.

The aim of this machine was to be faster than Belle (this was achieved by about 50%) and much more knowledgeable than any program before that. The latter was achieved by having 14 pattern recognizers. These special hardware units had programs to load specially designed chess patterns into them. These patterns were such that they tended to be of the type usually referred to as "chunks" when exhibited by humans. Chunks could be for recognizing that a bishop was "bad", or that this type of position was a known draw. Pattern recognizers also dealt with king safety, pawn structure, and notions such as who is ahead in development and by how much. The definitive document on Hitech is found in [1]. From 1985 until mid 1988, Hitech was the dominant force in computer chess. It was the highest rated program of all time when it won the 1985 North American Computer Championship, and after that went on to reach a USCF rating in excess of 2400. En route it won the Pennsylvania Chess Championship three years in a row in 1987, 88, and 89. On each occasion it was denied the title through manipulations of the tournament director in the first instance and by rule changes on the last two occasions. Hitech was also victimized in the 1986 World Computer Championships when one of the programs that tied for first place with Hitech was found to have cheated, but this did not result in any corrective action by the ICCA who

was staging the event. For two years after this, Hitech did not compete any more in computer events.

Hitech's success was the product of speed and knowledge, although the latter was not always in the best form due to well intentioned but inadequately tested new knowledge introduced by yours truly. The stage was now set for a new advance in speed. The success of Hitech inspired F. H. Hsu, a hardware design student at CMU to see if he could not improve on Hitech's VLSI speed by going to a higher level of integrated circuit. He found a clever way to implement Belle's move generator on a single VLSI chip. Soon a new program called Deep Thought (DT) existed. Since DT and Hitech were in the same department, it was not unnatural for them to play practice games. I remember well how the initial DT was tactically considerably better than Hitech (because of its greater speed) but would always lack the understanding to drive home its advantage. However, over time a lot of this was overcome, and in late 1988 DT made its debut in tournament chess by beating Grandmaster Bent Larsen, once one of the prime candidates for the World title. It played some scintilating games at that American Open in Santa Monica, and became an instant celebrity.

The move-generator on DT ran at 500,000 pos/sec, and later versions of DT used two to four such processors in parallel. There were many versions of DT that competed, and while the hardware was being upgraded so was the knowledge it used. DT was the first program that used learning techniques to learn the coefficients of its PC/SQ table, the device upon which almost all evaluation was based. Learning was based upon using a set of 1000 Master and Grandmaster games as a model of what is correct. As the learning became more effective, DT improved, and soon achieved a performance rating of USCF 2500 which entitled it to the Fredkin prize for achieving International Master status.

But Hsu was still not satisfied. He soon had a design for a move generator that could do 2,000,000 pos/sec. He and Murray Campbell and Thomas Anantharaman completed their degrees at CMU and moved to IBM to continue the chess work. From there they ran various versions of DT and soon had a new machine/program named Deep Blue (DB). However, DB did not always do too well in tournament style games against well prepared opposition. In 1996 it lost a match to World Champion Garry Kasparov 4-2 even though it won its first game (the first time a Chess World Champion had been beaten in a tournament style game). The general impression of top-notch players was that DB was not really prepared to give Kasparov a good fight. It's opening book was not adequate for this level of competition, and it did not seem to know what to do when there were no pressures or no advanatages to be exploited. Kasparov discoved in the last two games, that he could win as he pleased by just not pressing too hard. This DB ran on an IBM machine with 16 fast processors, 15 of which had 16 2-meg move generators hanging off it. This gave it a processing power of about 30,000,000 pos/sec after accounting for losses due to the fact that parallel alpha-beta is not efficient.

Most people did not believe things would be very different when these two opponents met again in 1997. The basic machine was said to be two times faster and otherwise improved, but this seemed rather ephemoral untilsome actual games could be seen.However, from the start DB2 played excellent quality chess, even though it lost the first game. There was little doubt that this was a much stronger program.

However, there were a number of strange happenings at this match which included Kasparov resigning in a position he could have drawn, and playing a known losing line in the final game, presumably on the theory that computers would not know how to play such positions. However, the final outcome of 3.5‘- 2.5 in favor of DB speaks for itself. Most people feel that this demonstrated the long sought result of a

computer being better than the World Chess Champion. Although more play may be needed to demonstrate this beyond a shadow of a doubt, I felt the this was an adequate demonstration for all intents and purposes.

## 5 What have we learned

### 5.1 There is Always a Better Way to do the Search

In the beginning, with a known branching factor of 35 moves for chess, and no relief in sight, it appeared that a Shannon Type A (brute-force) program was hopeless. But over time, with the discovery of alpha-beta and hash-tables, things looked much rosier. Machines were getting faster all the time, and this added to the performance level of programs. Then the age of special purpose hardware brought on even faster searching.

However, even with all these important technological advances, there is another aspect of search that has permeated the history. This is the continuing advance of small efficiencies that produce speed or sensitivity to what should be searched. This, in my mind, is one of the truly marvelous aspects of the whole enterprise.

It starts with Slate & Atkin not counting responses to being in check, which creates sensitivity to attacks on the king by following lines of such attacks more deeply. In 1983, Thompson modified Belle so that it would not count captures as a ply of depth. This did not work out, but Hitech improved on this by having a recapture window, and if a capture resulted in the value of a position being inside this window (based on the expected value of the position) then it must have been a meaningful recapture and was not counted. In the meantime, many programmers of micro chess programs had found that it was possible to discount the depth contibution of a move toward max-depth based upon what kind of a move it was. Thus, certain kinds of captures could be counted as less than a full ply, and certain kinds of attacks likewise. The details of each of these schemes were closely guarded secrets, but the general idea was subject to learning and could result in a very efficient discounting that would concentrate the search in the most important parts of the tree. This method became known as "partial depths", and was extremely effective in getting the micros to play good chess. It was also adopted in other programs once the scheme became known.

Another kind of help was offered by "null-move rejection" in which a move that was about to be searched to depth N, was first searched to depth N-2. If, given an extra "null move" at the start, it still failed to achieve a certain value, it was assumed to not be worth a full depth N search. This method was pioneered by Don Beal and later tried on Hitech by Gordon Goetsch. It is still not clear whether this method, which definitely speeds up the search has any negative aspects, but for most purposes it seems to gain some time in the computation.

Another "freebee" that appeared early on was the realization that it was possible to find a position in the hash-table that had been searched very deeply because it resulted from some ineffective move that was tried early in the search. However, deeper in the tree this position might be the result of forced play and now the fact that only X ply are left in the search would be over-shadowed by the fact that this position had already been searched to M>X ply earlier. Thus, the new knowledge of the position would be based upon an M+N ply search, whereas without this it might be based only upon an N+2 ply search.

Thus, a much deeper understanding of the position is reached. This kind of thing allowed solving positions in which the solution was known to extend over 30 ply, and thus estimated to take centuries to solve. But, because there were only a few thousand legal positions that could occur, and these would all have been searched at some point to a deep ply level, it was possible to construct a solution in which the result of one 16-ply search was inserted near the end of another and thus achieved the necessary depth. This effect, which has never been named, occurred occasionally in ordinary positions with amazing effect.

### 5.2 Speed Matters

In the beginning, with a mind-set that deep searching was impossible, effort was focussed on how to impart the right level of smarts to a program. There were a few exceptions here: The Ulam program at Los Alamos, the Daly program that finished 2nd in the first US Computer Championship in 1970, and most notably the program Kaissa, from the Soviet Institute for Experimental Physics that was headed by Adelson-Velsky. These programs all showed that quite a bit could be accomplished by brute-force searching. This was mainly in the area of avoiding gross errors that could cost a game. However, it is possible to look at chess as a contest in which it is important to avoid making errors, and the side that loses is the one that makes the last mistake. This is not the view that is promulgated in the chess literature, but there is a certain validity to it anyway.

Part of becoming a strong player is learning to avoid making errors that cost material or lead to being checkmated. Only after such errors have been overcome is it meaningful to play strategically. If one can avoid making errors that can cause an immediate loss, then strategy could carry the day. There is no doubt that the success of MacHack VI was due in large part to being able to avoid many such errors while maintaining a view toward making meaningful positional moves. The fact that CHESSX.X was so good positionally obscured the fact that as it increased its speed of play it got better; that is, until that fateful week in which it ran on a CDC Cyber 176 for the first time. In the following weeks it captured the Michigan State Open Championship thus making clear that something had changed. After that special purpose machines took the lead in producing speed. However, the fact that certain micros were able to beat machines such as Deep Thought and Deep Blue even though they were being outsearched by three or more orders of magnitude should have alerted people that **knowledge matters also**.

### 5.3 Knowledge versus Speed

Now, in mid-1997 things seem to have finally reached a point where some things can be stated with definitive clarity. Certain programs have achieved success against other programs even though they were somewhat knowledge deficient according to the then prevailing state-of-the-art. However, when these programs played against well prepared humans, these programs did not do well -- a clear indication that the humans had found some knowledge deficiency to exploit. Yet knowledgeable programs were frequently beaten by programs that outsearched them by a ply or more. The main example here is CHESS4.X being overtaken by Belle, and Hitech being overtaken by DT.

From time to time there would be a spurt in speed that would carry the banner forward some significant amount. The first such instance was Belle, but that was short lived as humans learned to adapt to its weaknesses. Then came Deep Thought which was able to search to depth 10 usually and thus outsearch all except the most advanced human players. I was able to watch DT improve as it was playing practice matches against Hitech all the time, and over the period of a few months it got significantly better as it acquired meaningful chess knowledge. The lastest example was Deep Blue

which led a life of secrecy until the 1996 match against Kasparov. There were some games against Grandmasters, but these unpublished games usually showed a clear lack of understanding on the part of DB. Also, in the 1996 match against Kasparov, DB showed a significant lack of understanding. However, it is also clear that since the 1996 Kasparov match, DB was subject to vast improvements in its knowledge.

I subjected the games of the 1996 and 1997 matches to a great deal of analysis and came up with some interesting conclusions. It is generally agreed among top chess experts that DB's opening book was vastly improved for the 1997 match. Further, I noticed that there appeared to be a new level of understanding in the 1997 DB with respect to pins. One simple definition of a pin is a situation where a piece of greater value would be attacked by one of lesser value except for the intervention of its own piece of lesser value than the attacker (something that could be detected in the process of move generation on a chip). Using this definition there were 7 instances in the 1996 match where DB was pinning a Kasparov piece after having made its move, and 10 instances in which Kasparov was pinning DB. In the 1997 match there were 40 instances where DB had a pin, and only 11 by Kasparov. This difference could be accounted for my other random factors, but is most likely due to a new understanding of the value of pins.

DB also on at least two occasions showed that it had a new understanding of the value of bishops than it nor any of its predecessors had ever evidenced before. DB also avoided certain moves that computers almost always make to their own detriment, which is almost certainly due to some pattern information that was imparted to it, since it seems unlikely that even a 14-ply search could have discovered the strategic difficulty. DB also seemed to be more likely to pursue an active course; something that we have found can be encouraged by counting the number of pieces that are being attacked and defended by each side. All these factors point to a vast improvement in the knowledge in DB, and the DB team has also said that its knowledge had been improved.

## 6 The Final Understanding

However, to add much more meat to the bones of the knowledge versus speed discussion we present a new and very important insight gained by plotting performance of important programs in Figure 1. We have used this basic type of figure before to illustrate the trade-off between Search and Speed. There are a set of hyberbolic isobars that show the trade-off for different performance levels. To show the validity of this idea it is only necessary to note that it is possible to solve chess by either knowing the right move in every position, or by searching to a depth that is sufficient to determine the best move in any position. Practically speaking, both these alternatives are impossible, but there is no doubt that a perfect searcher or a perfect knower would play perfect chess.

It is also clear that perfection can be achieved through some combination where a little knowledge makes it unneccessary to determine certain things by search, and a little search makes it unneccessary to know certain things. A wonderful illustration is offerred by something first done by Slate & Atkin. In the late 1960's and early 1970's people had programs that searched to very shallow depths, and it was not even clear that a program could win in a position of King and Rook versus King (for instance, Greenblatt's program could not always do it). Slate & Atkin devised a special PC/SQ table that would be invoked only for this ending. This table was pre-computed before each search once this ending was reached. It rewarded

1. Decentralizing the losing king,
2. Keeping the winning king near the losing king, and
3. Keeping the rook near the winning king.

With this combination of knowledge, it was possible to mate using only 3-ply searches; something that depth 5 searches could not do without the knowledge.

In Figure 1, the isobar labelled 1400 shows that a chess rating of USCF 1400 can be achieved by a tremendous amount of knowledge and little search, or a tremendous amount of search and little knowledge, or by any of many combinations of search and knowledge that are more balanced. It is clear that when a program is operating with extreme amounts of speed, it is more difficult to cross isobars by continuing to provide it with more speed. Likewise, if it has extreme amounts of knowledge (which never has happened) it would be very difficult to cross isobars by providing more knowledge. Thus, the sensible course is to balance these factors which leads to easier progress by moving upward diagonally in the graph.

What we have done in Figure 1 is to plot the established USCF rating of the most outstanding programs against their speed in thousands of nodes/second at the time they achieved this result. This makes it possible to read out an estimate of the value of the knowledge in the program on the left-hand axis. The scale of this knowledge is unknown, but it clearly is monotonic. Knowledge includes both good things that help the program, and bad knowledge and bugs that impede its performance. However, it is a reliable estimate of how fast a program would have to run with the knowledge at hand in order to achieve the performance level that is shown on the graph.

Looking at Figure 1, we see that MacHack VI which was very slow by today's standards, must have had a certain level of knowledge to achieve its performance. CHESSX.X, running at a greater speed, nevertheless had to have more knowledge to achieve its greater performance. The CHESSX.X curve also shows the effect of the introduction of new hardware which made possible a very beneficial effect due to speed. Belle, which was limited to a unique piece of hardware could not achieve any gain in speed, so all of its performance gains were due to improvements in its understanding of chess. The same was true of Hitech where almost all the effort after the machine was built went into improving its understanding. Deep Thought went through several changes which added additional parallel hardware and thus more speed. It also improved its understanding which is shown by the diagonal showing gains in both dimensions. It was difficult to estimate the performance of Deep Blue since it played in no tournaments from which ratings could be derived. However, there has been a general concensus among strong players that the 1996 version played at best like a weak Grandmaster, while the 1997 version played like a very strong Grandmaster. Since there was at most a speed-up of a factor of two, the major change in performance must have been due to a significant increase in its knowledge, as already indicated above.

## 7 Conclusions

So there it is. It is clear that progress in a domain such as computer chess must be achieved by a balanced increase between knowledge and speed of search. We see that in this domain, and with the practitioners who have been successful, speed has almost always led the way. However, until knowledge followed, there was no fulfillment of speed's promise. We conjecture that the same relationship will hold

true in other domains that can be addressed by a combination of knowledge and speed of search.

# References


[1] Berliner, H. and Ebeling, C.
The SUPREM Architecture: A New Intelligent Paradigm.
*Artificial Intelligence* 28(1), February, 1986.

[2] Condon, J. H., and Thompson, K.
Belle Chess Hardware.
In M. R. B. Clarke (editor), *Advances in Computer Chess 3*. Pergamon Press, 1982.

[3] Edwards, D. J. and Hart, T. P.
*The Alpha-Beta Heuristic*.
Technical Report 30, MIT Artificial Intelligence Project, October, 1963.

[4] Newell, A., Simon, H., and Shaw, C.
Chess Playing Programs and the Problem of Complexity.
In E.A. Feigenbaum and J. Feldman (editor), *Computers and Thought*. McGraw-Hill, 1963.

[5] Samuel, A. L.
Personal Communication.
*August* , 1983.

[6] Shannon, C.E.
Programming a Computer to Play Chess.
*Philosophy Magazine* 41(314), 1950.

[7] Slate, D. J., & Atkin, L. R.
CHESS 4.5 -- The Northwestern University Chess Program.
In P. Frey (editor), *Chess Skill in Man and Machine*. Springer Verlag, 1977.

[8] Slater, E.
Statistics for the Chess Computer and the Factor of Mobility.
In *Symposium on Information Theory*, pages 150-152. Ministry of Supply, London, 1950.

[9] Von Neumann, J., and Morgenstern, O.
*Theory of Games and Economic Behavior.*
Princeton University Press, 1944.

# Table of Contents