Creating a Chess Player

An Essay on Human and Computer Chess Skill

Peter W Frey
Department of Psychology
Northwestern University
Evanston IL 60201

Larry R Atkin
Health Information Services
542 Michigan Av
Evanston IL 60202

Peter Frey is currently at the Cresap Laboratory of Neuroscience and Behavior at Northwestern University. He is editor of Chess Skill in Man and Machine. *Larry Atkin is co-author of Chess 4.6, presently the world champion computer chess program.*

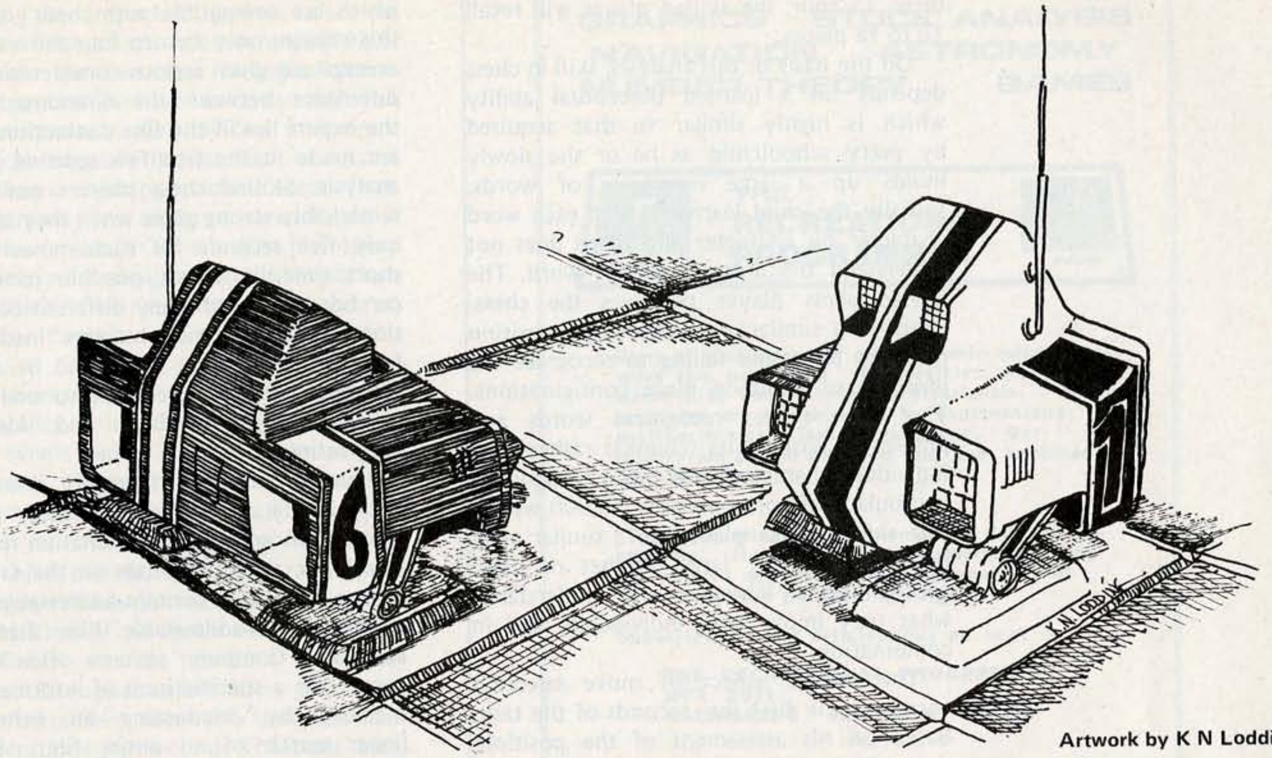
In a recent *Time* essay (see references) Robert Jastrow, director of NASA's Goddard Institute for Space Studies, predicted that history is about to witness the birth of a new intelligence, a form superior to humanity's. The pitiful human brain has "a wiring defect" that causes it to "freeze up" when faced with "several streams of information simultaneously." Jastrow suggests that "the human form is not likely to be the standard form for intelligent life" in the cosmos. Even on our own small planet, a new day is near at hand: "In the 1990s, . . . the compactness and reasoning power of an intelligence built out of silicon will begin to match that of the human brain."

We have always been fascinated by the idea of a machine that is capable of rational thought. Jastrow is neither the first nor the last person who is betting on rapid improvements in machine intelligence. His expectation that computers will rival humanity within 15 years seems optimistic to anyone who has watched half-a-dozen excited technicians flutter about for several hours trying to bring a crashed system back to life. This prophecy seems even more fanciful to those who have attempted to program machines to cope with pattern recognition, language translation or a complex game such as chess.

The chess environment, in fact, provides a particularly good example of the difficult problems which still need to be solved before silicon intelligence can become a reality. More than 20 years ago, Herbert Simon, a recognized expert in the field of artificial intelligence, predicted that within a decade, the world's chess champion would be a computer. This prognostication has not come to pass. Why was an informed scientist like Simon so wrong in his assessment of computer capabilities? A major factor is that computer scientists have often failed to appreciate the level of knowledge which is required to play master-level chess. They have also commonly underestimated the tremendous information-processing capacity of the human brain. Even though chess is a game of logic in which all legal moves can be precisely specified and in which nothing is left to chance, several centuries of intensive analysis have not exhausted the perennial challenge and novelty of the game. Psychologists have been actively studying the human brain for several decades and have discovered a fascinating mystery wrapped within an enigma. The more we learn about the brain, the more we are aware of our lamentable state of ignorance.

The Mind of the Chess Player

At a general level of knowledge, we have several provocative insights on the nature and structure of human chess skill. We know, for example, that the skilled chess player does not examine hundreds of possible continuations before selecting a move. We also know that superior chess players are not formidable "thinking machines" but in fact display a normal range of intelligence scores. Strong chess players, as a group, do not even appear to have special retention abilities such as having "photographic" memories. In most respects, top-flight chess players have the same intellectual capacities as the rest of the population and, in the technical details of move selection, seem to engage in the same type of information processing that is observed in much weaker players.



Artwork by K N Lodding.

Our knowledge in these matters is based on the early work of Binet in France and that of de Groot in Holland and on more recent investigations by other scientists in the USSR and the United States. In the late nineteenth century, Binet was surprised to discover that masters did not have a vivid image of the board when playing blindfolded chess. Instead, they seemed to remember positions in abstract terms such as by specific relations among pieces. Interviews with masters clearly indicated that a photographic memory was not a prerequisite for being able to play many simultaneous games of blindfolded chess. In the 1930s and 1940s, de Groot worked with a number of strong chess players (from Grandmasters to strong club players) and had them verbalize their thought processes while selecting a move in a complicated position. His research indicated that the Grandmasters' general approach was highly similar to that of weaker players. They analyzed a similar number of moves (about four) from the initial position, a similar number of total moves (about 35), made a similar number of fresh starts (about six), and calculated combinations to the same maximal depth (about seven plies or half-moves, where a move is defined as a play by one side and a response by the other). The only clear measurable difference was that the Grandmasters invariably chose the strongest move while the weaker players did not. Thus de Groot concluded that Grandmasters play better chess because they pick better moves. Unfortunately, this conclusion is not very informative since it is obviously circular. The fact that de Groot's extensive study did not uncover any prominent differences in the move-selection strategies used by strong and average players implies that the analysis procedure itself is not the critical factor which determines chess skill.

An important clue to the difference between skilled and unskilled players was discovered by de Groot when he displayed an unfamiliar chess position to his subjects for a few seconds and then asked them to recall the position from memory. He found that masters recalled almost all the pieces while club players remembered only about half of them. Recent work in this country by Chase and Simon at Carnegie-Mellon University has indicated that novice players recall only about a third of the pieces. Chase and Simon also added an important control procedure. They demonstrated that the differences in recall ability completely disappear if the pieces are positioned randomly. This outcome indicates that the superior memory of the chess master is chess-specific and not a general trait.

De Groot's "law" of chess is that Grandmasters play better chess simply because they pick better moves.

Simon and Gilmartin have proposed that skilled chess players learn to recognize a large number of piece combinations as perceptual chunks and perform well in the recall task because they remember four or five chunks rather than four or five pieces like the novice. If the average chunk size is three to four, the skilled player will recall 16 to 18 pieces.

On the basis of this analysis, skill in chess depends on a learned perceptual ability which is highly similar to that acquired by every schoolchild as he or she slowly builds up a large repertoire of words. Initially the child learns to read each word character by character and often does not understand the meaning of the word. The novice chess player perceives the chessboard in a similar way, assessing a position piece by piece and failing to recognize the *meaning* of common piece configurations. The adult reader recognizes words and phrases as basic units (chunks) rather than individual characters and has a recognition vocabulary of approximately 50,000 words. The skilled chess player, in a similar vein, recognizes a very large number of piece configurations (chunks) and understands what they imply both individually and in combination.

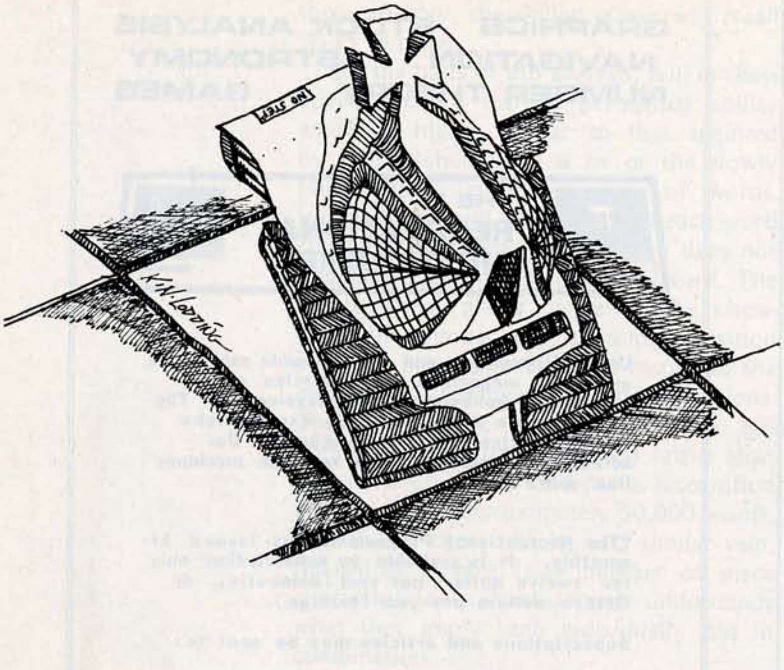
The critical aspect of move selection occurs in the first few seconds of the task. Based on his assessment of the position, the skilled player immediately recognizes appropriate long-term and short-term goals and has a good feel for the specific moves which are compatible with these goals. For this reason, only two to four moves on the average are given serious consideration. The difference between the Grandmaster and the expert lies in the fine distinctions which are made in the first few seconds of their analysis. Skilled chess players can play a remarkably strong game when they are given only five seconds for each move. In this short time, it is not possible to make a careful analysis of many different continuations. The player must have an "instinctive" feel for the correct move and be able to recognize key features and to understand both their immediate and long-term implications.

Human chess skill, therefore, is based on two highly refined capacities, pattern recognition and rapid information retrieval. The latter ability depends on the fact that human memory is content-addressable rather than location-addressable like that of a computer. Computer systems often have to search for a specific item of information in memory by conducting an exhaustive, linear search of an entire file. Human memory however is organized in an amazingly complex fashion such that most of us can easily recall a specific fact on the basis of a completely novel retrieval cue. For example, name a flower that rhymes with *nose*. In this case, your quick response demonstrates that words are grouped together on the basis of their phonetic similarity (ie: sound). Your ability to quickly recall words which are similar in meaning to the word *fat* (such as obese, chubby, rotund, flabby, plump and stout) demonstrates that human memory is also organized by semantic similarity (ie: meaning). When a person is given a retrieval cue which does not elicit an immediate response, he or she can usually find the correct information after a brief search of related ideas or concepts. This facility contrasts sharply with the extremely limited linear searches which are generally conducted with large computer based storage systems. Even sophisticated computer retrieval strategies which arrange the data base in multilinked lists with elaborate tree structures presently lack the large system efficiency displayed by their biological counterparts.

Pattern recognition and rapid information retrieval are not only key capacities for chess, but are also essential for a wide range of important human problem solving skills. Whether your field is medicine, engineering, plumbing or computer programming, you would be a complete failure at your job without these essential abilities. Jastrow's claim that machine intelligence will soon equal man's intelligence seems to overlook the important points made in BYTE by Ernest Kent (see references). Kent emphasizes the fact that biological information processors have a vastly different architecture than their silicon imitations. In fact, he suggests that our lack of success in building a thinking machine stems from our attempts "to make a wrench do a screwdriver's job." Our modern high-speed computers were designed to do important tasks which men are not very good at, such as complex mathematical calculations.

The human brain evolved, in contrast, on its ability to identify important environmental events and to quickly recognize their significance. Natural selection has never placed much emphasis on our ability to multiply or our ability to compute the inverse of a matrix. Kent also reminds us that organic evolution worked with a very different kind of hardware than that which is available to the modern computer engineer. Biological information processors have an incredibly slow cycle time, less than 100 operations per second. The basic unit, the neuron, operates in milliseconds rather than in nanoseconds. The brain, however, makes up in quantity and in structural complexity what it lacks in speed. Computers, on the other hand, have many fewer components and a much simpler gating architecture, but are orders of magnitude faster.

It may be that present machine hardware configurations are simply inappropriate for efficient pattern recognition or semantic recall. An analysis of the history of computer chess is instructive. Although there have been numerous advocates for chess programs which imitate human playing methods, only a few have been attempted, and none of these have played reasonable chess. The eariliest paper on machine chess, written by Claude Shannon in 1950 (see references), proposed a mechanical algorithm which was not modeled on human chess play. Shannon suggested a workable procedure for representing the board and piece locations, specified simple mathematical algorithms for generating the legal moves of each piece and gave an example of a straightforward technique for evaluating a position (see *Chess Skill in Man and Machine*, chapter 3). The key feature of Shannon's proposal was the adoption of the *minimax* technique as described by von Neuman and Morgenstern in 1944. The basic idea of the minimax technique is to assume that the player whose turn it is to play will always choose the move which minimizes his opponent's maximum potential gain. Hence, the name minimax.



The Type B Strategy

One of the difficulties of this approach is that a complete analysis of all possible continuations (type A strategy) very rapidly leads to an overwhelming number of potential positions. The look-ahead tree grows at an exponential rate and with an average, according to de Groot, of 38 legal moves at each position, a search involving three moves (three half-moves for each player) produces over 3 billion (38^6) terminal positions. You may recall that de Groot's research indicated that human players regularly searched a tree to seven plies and sometimes much deeper. Because of this, Shannon concluded that it would not be possible for the machine to consider all possible legal continuations at each *node* of the game tree. Instead, he proposed a type B strategy in which only reasonable (ie: plausible) moves are pursued at each branching point. If the program considered only five continuations at each node instead of all 38, a 6 ply look-ahead would involve only 15,625 (5^6) terminal positions.

The superior memory of the chess master is chess-specific and not a general trait.

The attractiveness of the type B approach seems overwhelming when the number of terminal positions increases exponentially with depth. The fact that skilled human players explore only a limited number of continuations at each choice point is additional evidence which favors the adoption of this strategy. It is not surprising, therefore, that most programmers have used Shannon's type B strategy in designing a chess program.

Sometimes our understanding of the real world, however, is not always as accurate as we presume. In selecting a type B strategy in preference to a type A strategy, the programmer does not necessarily simplify the problem. This approach was competently implemented in 1967 by Greenblatt at MIT. His program played reasonable, and at that time, fairly impressive chess. The major design problem in a selective search is the possibility that the look-ahead process will exclude a key move at a low level in the game tree. The failure to consider an important move can lead to a very serious miscalculation. A chess game can be lost by a single weak move. For this reason, it is of critical importance that a necessary move not be missed. The type B programs place a critical dependence on the accuracy of their plausible move generator. Chess is an extremely complex game and in many situations a move which at a superficial level seems unlikely, is, in fact, the best one. Grandmasters find these moves while lesser players, including machines, fail to see them. For a decade, several dozen individuals have tried to create a plausible move generator that is superior to Greenblatt's. The evidence is fairly clear, however, that type B programs have improved very little since 1967.

As strange as it may seem, recent progress in computer chess has come by abandoning the type B strategy. Shannon's logical analysis was made in a "stone-age" hardware environment and without knowledge of several important algorithms. Today, the type A strategy is not as ridiculous as it seemed in 1950. In addition, very few individuals anticipated the immense difficulty involved in constructing a competent plausible move generator. To become a chess master, a man has to study chess intensively (20 hrs or more a week) for at least 5 years. During this time he acquires an immense amount of detailed knowledge about the game of chess. Subtle features of a particular position are recognized immediately and suggest both short-term and long-term goals as well as specific moves. This kind of knowledge is sufficiently abstract that most players find it impossible to verbalize the relevant thought processes. The one factor which stands out clearly, however, is that the chess master has acquired a tremendous library of factual information which can be retrieved quickly and applied in apparently novel situations. No chess program has been able to duplicate this facility and, without it, the creation of a workable plausible move generator is next to impossible.

When a type A strategy is employed, however, this problem can be bypassed. By making all the moves *plausible*, the program never overlooks a subtle but important one. In fact, by reverting to a brute force search of all possible continuations, the program often finds interesting combinations that are commonly missed even by strong human players. It seems ironic that the brute force approach (full width searching) produces many more brilliant moves than the smart approach (selective searching). This important discovery was made independently by Slate and Atkin at Northwestern (the authors of the current world champion chess program, Chess 4.6) and by the Russian KAISSA team.

Minimax and the Alpha-Beta Algorithm

Slate and Atkin's work has demonstrated that a full width search can be conducted considerably more efficiently than anyone had previously suspected (including Slate and Atkin; see references). There are a number of important developments which are responsible for this reassessment. The most important discovery was made in the late 1950s by Newell, Shaw and Simon as well as by Samuels. Because of the basic logic underlying a minimax search, it is not necessary to search the entire look-ahead tree before selecting the best move. Consider a simple 2 ply search (one move for you and one for your opponent). First you examine one of your possible moves and the 38 or so terminal positions which result from each of your opponent's legal replies. You select the one reply which is best, according to your evaluation function, for your opponent (ie: the one which minimizes your own maximum potential gain). Next, you consider a second move for yourself and the 38 or so replies that your opponent can make to this move. In considering these moves, you discover that the third reply you examine would give your opponent a better outcome than his best reply to your first candidate. Immediately you realize that it is a complete waste of time for you to analyze any more of his replies to your second candidate. Since you are already guaranteed a worse position after the second move than after the first, it is reasonable to reject the second one and turn to your third candidate. This decision eliminates the need for evaluating 35 of the potential replies to your second candidate. A very tidy savings.

Historically, the score for the best move so far for White has been designated as α and the score for the best move so far for Black has been called β. Thus the name alpha-beta (α–β) algorithm. When the tree is both wide and deep, this algorithm can reduce the number of terminal nodes to a small fraction of the number which would be examined by a complete minimax search. The beauty of this procedure is that it always produces the same result as the full minimax search.

An important factor in determining the efficiency of the alpha-beta algorithm is the order in which the moves are examined. If White's best moves and Black's best replies are considered first at each choice point, the search of the uniform game tree of height h (number of plies deep) and width d (number of successors at each node) will involve approximately $2 \cdot d^{h/2}$ terminal positions instead of d^h (see references, Knuth and Moore). The potential magnitude of this saving can be appreciated by considering our previous example with a 6 ply search: 38^6 is more than 3 billion while 2×38^3 is about 110,000. Shannon might have given more consideration to the type A strategy if he had been aware of the alpha-beta algorithm and some of the other technical improvements which were to follow.

General Strategy

To maximize the benefit of the alpha-beta procedure, it is necessary to devise an efficient strategy for generating the moves at each node in an order which is likely to produce a cut-off, such that searching can be terminated at that node. There are several general heuristics which have proven their value time and time again. One is extremely simple and powerful: try capturing moves first. Because a full width search includes many ridiculous moves, a reply which involves a capture will often remove a piece which was "stupidly" placed *en prise* (ie: attacked and insufficiently defended).

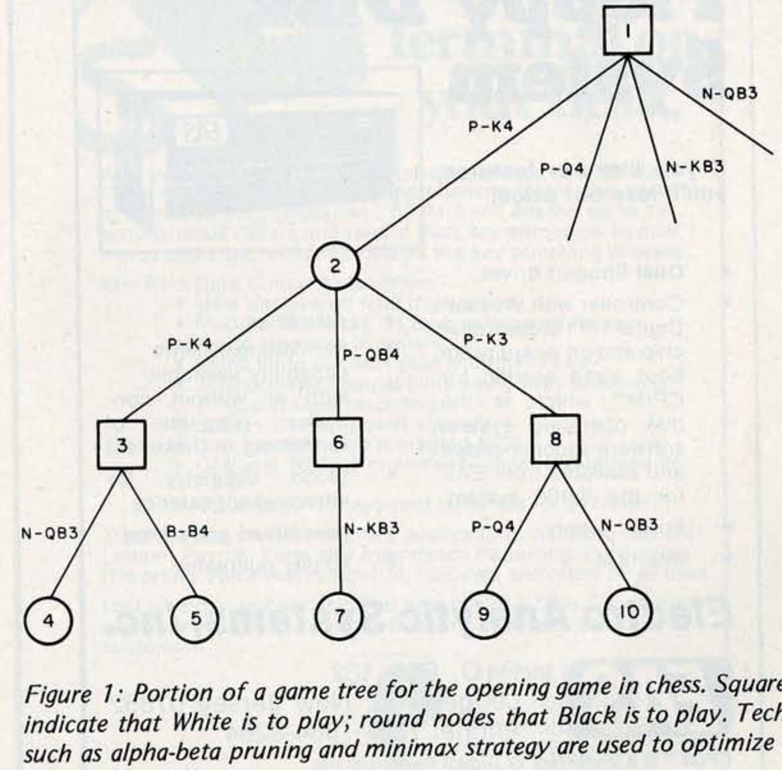


Figure 1: Portion of a game tree for the opening game in chess. Square nodes indicate that White is to play; round nodes that Black is to play. Techniques such as alpha-beta pruning and minimax strategy are used to optimize the use of trees like this.

Captures also have the beneficial effect of reducing the number of potential offspring. An additional important characteristic of a capturing move is that it will generally have to be examined sooner or later in order to insure the quiescence of the terminal position. Because of this, every capture that is examined early generally reduces the amount of work which will have to be done later. In practice, investigators have reported a speed-up in search time of as much as 2 to 1 by simply putting all the captures at the beginning of the move list.

In addition to captures, there is another class of moves which is also effective for producing cut-offs. These are called *killers* because they are moves which have produced cut-offs in the immediate past and have been specifically remembered for that reason. A short list of killers is maintained by the program and whenever the legal capturing moves fail to produce a cut-off, each of the killers (if legal in the given position) is then examined. This *killer heuristic* is quite effective in producing a move order which enhances the probability of a quick cut-off.

The general features of the alpha-beta algorithm and its important servants, the capture and killer heuristics, were reasonably well-known late in the 1960s. In recent years, several important refinements have been added to this list. One of the most important is the staged or iterative alpha-beta search. For example, instead of conducting a 5 ply search all at once the search is done in stages, first a 2 ply search, then a 3 ply search, then a 4 ply search, and finally a 5 ply search. Superficially this might appear to be wasteful since the staged search requires the full 5 ply search eventually anyway. This is not at all the case. As each search is completed, the principal variation (best moves for each side at each depth) is used as the base for the next (1 ply deeper) search. The 3 ply search therefore starts with a move at ply 1 and a reply at ply 2 which has already been proven to be reasonable (from the machine's limited perspective). The 4 ply search starts with reasonable moves at its first three plies. The 5 ply search has the benefit of reasonable moves at its first four plies. Because the efficiency of the alpha-beta algorithm is tremendously sensitive to move ordering, the spill-over in information from one iteration to the next has a surprisingly powerful effect. A single 1 stage 5 ply search might require 120 seconds of processor time. The last segment of the staged 5 ply search might require only half as much time (ie: 60). Since each iteration requires about five times as much processor time as its predecessor (the exponential character of the look-ahead tree is diminished somewhat by the alpha-beta algorithm), the staged 4 ply search would take about 12 seconds, the staged 3 ply search about 3 seconds, and the 2 ply search about 1 second. The total time for the iterative search would be approximately 76 seconds (1 + 3 + 12 + 60) rather than 120 seconds.

An added benefit of the iterative search, and, incidentally, the reason for its discovery in the first place, is that it provides a useful mechanism for time control. In tournaments, a move must be calculated within a fixed time limit such as 90 to 120 seconds. If one decides to do a 5 ply search in a single stage, it is possible to find oneself tied up in calculation after 120 seconds with no idea of how much more time will be needed to complete the search, and without a move to make until the search is completed. In some complex situations the search might take as long as 10 minutes — a disaster for time control. An iterative search allows one to predict the probable duration of the next iteration and to make a decision whether it is cost effective to initiate the next one. If this decision is a go and the search, for some reason, fails to terminate in the anticipated time, the machine can abort and play the move selected by the last iteration. This provides relatively neat and tidy time control. The iterative search was first mentioned by Scott in 1969 and was apparently discovered independently several years later by Jim Gillogly at Carnegie-Mellon, by Slate and Atkin at Northwestern and by the Russian KAISSA team.

Refinements to the Type A Strategy

Several other refinements have also made the type A strategy more manageable. One of the time intensive activities involved in tree searching is move generation. This can be minimized by generating only one move at a time and seeing if it produces a cut-off before generating the next move. If a cut-off occurs and the node is abandoned, one can avoid generating a large number of potential moves. With the n-best approach, it is customary to generate all moves at each node and then invest time attempting to decide which ones are worthy of further consideration. Thus the smaller tree, obtained by selective searching, has to be partially paid for by an additional time investment in plausibility analysis.

Another time-intensive activity in the tree search is the repeated use of the evaluation function. Since many thousands of terminal nodes have to be evaluated in each move selection, any refinement that reduces the work of the evaluation function will pay rich dividends. There are three important

techniques which fall in this category. One of these is called *incremental updating*. In order to make an evaluation of a node, it is necessary to have certain key facts available, such as which squares are attacked by each piece, which pieces are present, etc. This information can be newly calculated at each terminal node or can be incrementally maintained by updating the appropriate tables as the tree is generated during the search. This latter procedure is more complex to program but tremendously more efficient in terms of computing time because neighboring terminal positions are highly similar. They usually differ in respect to only a single piece, and therefore the updating procedure requires about 10 percent of the computations that would be expended if the evaluation data base were recalculated from scratch for each evaluation.

A second refinement in this category is the use of serial organization in the evaluation function. In order to assess the relative merit of a chess position, most programs place heavy emphasis on the material balance (ie: the relative number of pieces for each side). This tradition is founded on the idea that winning or losing is strongly correlated with being ahead or behind in material. An additional rationale is that this information is readily available and easily updated.

In most programs material factors are so dominant that the other evaluation terms, such as mobility, pawn structure, King safety, area control, etc, taken together almost never account for more than two pawns. Because of this, it makes sense to compute the material balance factor first and then determine if the result is within two pawns of the target value. If not, there is no need to assess the other factors, because the final decision will be independent of their value.

This simple idea encourages one to organize the evaluation function in strict serial order such that influential (heavily weighted) terms are analyzed first and the result examined to see if a decision is possible based on this initial information. If not, the next most influential term(s) are examined and another determination is made. This process is repeated until an escape condition occurs or until all terms have been examined. In most cases, the evaluation will be terminated long before the list of potential terms has been exhausted. This technical refinement can save a significant amount of time.

A third procedure for speeding the evaluation process is to remember past evaluations. For instance, one should avoid reassessing the same position two or more times. In chess, there are many pathways by which one can reach identical positions. In a 3 ply sequence in which the middle move remains constant, for example, the first and third moves can be interchanged and the resulting position will be the same. Transpositions such as this occur frequently in the end game where the King may have literally hundreds of 4 move pathways that end on the same square. Rooks, Bishops and Queens also have a special facility for reaching a particular destination square in multiple moves rather than in one or two.

A full width search (ie: type A strategy) greatly accentuates this foolishness. By creating a large table of past positions which have been already evaluated, and using a hashing procedure to check if the present position is in the table, the programmer can completely eliminate a portion of the evaluation effort. In most middle game positions, this technique will produce a 10 to 50 percent saving. In certain end game positions, however, the transposition table can eliminate more than 80 percent of the evaluation effort. This idea seems to have been implemented first by Greenblatt in 1967.

An extension of this idea is to use the table to store likely moves as well as evaluations. By remembering a move which previously produced a cut-off, the table can facilitate move ordering decisions. In addition, the use of the same reply at a familiar position may have the added benefit of increasing the number of transpositions which will be encountered at later nodes. Additional details on the use of a transposition table are discussed in chapter 4 of *Chess Skill in Man and Machine*.

One of the most difficult challenges for a chess program is the end game. A machine which calculates a move for each position has difficulty competing with humans who "know" the correct move on the basis of their own or someone else's past experience. There are a huge number of end game situations in which a specific and highly technical strategy is required. Strong chess players study these intricacies at great length and use this knowledge at the chessboard to avoid unnecessary calculations. For example, a King and a pawn against a lone King is a win in some positions, and a draw otherwise. The same is true for a King and two pawns against a King and a pawn. If a Rook or minor piece is added to each side, the situation changes dramatically. Unfortunately our present day programs are oblivious to these subtleties. For this reason they can find the correct move only by engaging in prodigious calculations. Their human counterpart, on the other hand, "knows" the correct move after a cursory glance at the position.

Newborn (see references) has introduced a useful technique for reducing this knowledge gap. The main idea is to categorize familiar end game positions as wins or draws. Many games end with a King and a pawn fighting a lone King. Skilled players usually terminate the contest before it runs its inevitable course because the outcome is not in doubt. Newborn has shown that it is feasible, taking advantage of the symmetries of the chessboard, to make a bit map that indicates either a win (1) or a draw (0) for each potential square on which the lone King might reside for each of the potential locations of the opposing King and pawn. This knowledge can be encoded in approximately 300 bit boards of 64 bits each (see chapter 5 of *Chess Skill in Man and Machine*).

Although a tremendous amount of work and chess knowledge is required to complete this task, the end result is well worth the effort. When a position involving two Kings and a pawn is encountered anywhere in the look-ahead tree, it can be immediately scored with 100 percent accuracy as a win or a draw. This extends the look-ahead horizon of the program by as much as 12 to 15 plies for these specific situations, and eliminates all the tree searching effort which would normally be required. Furthermore, it permits accurate evaluations at the end points of a deep search, which allows the program to select a continuation which leads to a favorable end game. If this approach were extended to a wider range of situations, the machine's present knowledge deficit with respect to the end game would be greatly reduced.

These programming refinements, together with rapid hardware advances, have made the Shannon type A strategy feasible if not particularly elegant. For this reason it is possible to program a machine to play a game of chess which is free of gross blunders and which sometimes even contains an innovative move or two. Although this approach is clearly not a final solution, it does provide a solid base which can be used as a reliable starting point for future developments.■

REFERENCES

Charness, N, "Human Chess Skill," *Chess Skill in Man and Machine,* Frey, P W (editor), New York, Springer-Verlag, 1977.

Frey, P W, "An Introduction to Computer Chess," *Chess Skill in Man and Machine,* Frey, P W (editor), New York, Springer-Verlag, 1977.

Jastrow, R, "Toward an Intelligence Beyond Man's," *Time,* February 20 1978, page 59.

Kent, Ernest W, "The Brains of Men and Machines" (4 part series): January 1978 BYTE, page 11; February 1978 BYTE, page 84; May 1978 BYTE, page 74; and April 1978 BYTE, page 66.

Knuth, D E and Moore, R, "An Analysis of Alpha-Beta Pruning," *Artificial Intelligence,* volume 6, 1975, pages 293 thru 326.

Newborn, M, "PEASANT: An Endgame Program for Kings and Pawns," *Chess Skill and Man and Machine,* Frey, P W (editor), New York, Springer-Verlag, 1977.

Shannon, C E, "Programming a Computer For Playing Chess," *Philosophical Magazine,* volume 41, 1950, pages 256 thru 275.

Slate, D J and Atkin, L R, "CHESS 4.5 – The Northwestern University Chess Program," *Chess Skill in Man and Machine,* Frey, P W (editor), New York, Springer-Verlag, 1977.