The Anatomy of Chess Programs

T.A. Marsland

Computing Science Department
University of Alberta
Edmonton, AB
Canada T6G 2H1
Tony.Marsland@ualberta.ca

Abstract

This short paper defines the terminology used to support computer chess work, and introduces the basic concpets behind chess programs. It is intended to be of general interest, providing background information not new ideas.

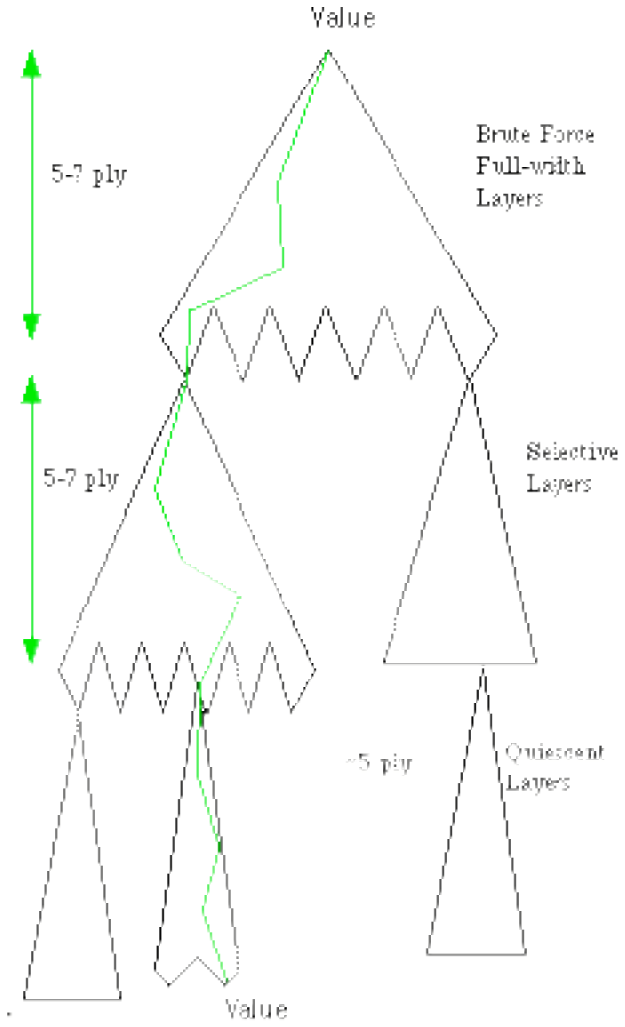
Introduction

Logically, chess is a trivial game: at every move, simply follow through each possible reply and its consequences until either a mate or a draw position is reached. In practical terms, however, this strategy is not workable, since an astronomically large number of chess positions would have to be examined. Thus both human players and computers rely on simplification to build an approximate model of the game. Human players have centuries of tradition and at least two hundred years of chess literature to draw on in building their personal model, but computer chess is less than fifty years old. Significant among the early ideas in computer chess is Claude Shannon's 1949-50 distinction between a brute force (type-A) strategy for looking at every combination of moves, and the use of chess knowledge to select and examine only a subset of the available moves (type-B strategy). Although some electro-mechanical systems to play a subset of chess had been built prior to Shannon's work, it was the programming of his ideas that led to the development of today's computer chess machines.

Current chess programs view the game as a tree search in which each position corresponds to a node in the game-tree, and each move is a branch (a transition from one node to the next). Thus the tree is made up of alternating layers or levels of moves for each side. (The term "ply" is used to denote each layer, and refers to one move by one player.) A three-stage tree model is popular with computer chess programmers. The first stage uses a brute force (Shannon type-A) approach, the second a selective (type-B) search, and the third a strategy known as a quiescence search, designed to resolve the problems and conflicts that remain. In this final stage the program evaluates sequences of capturing moves, assesses pawn promotion potentials, examines checking sequences and considers other highly constrained tactical issues. All programs use the same underlying depth-first alpha-beta search algorithm. What varies from program to program is the length (or "depth", to keep the layer analogy) of search assigned to each of these stages. Ultimately the stage length is not fixed, but varies by small amounts depending on the current sequence of moves being examined. For example, a search path may be locally lengthened because one side has attacked the King (given check), leaving the opponent with only a few alternatives to consider. There are so many options here that even programs using the same basic model can achieve a radically different style and speed of play.

Tree Searching

While the human method of analyzing alternatives seems to involve selecting a few promising lines of play and exploring them, computers are necessarily exhaustive rather than selective, so refinement techniques have been developed. In a technique called "iterative deepening," instead of embarking on a single search of a certain ply (which might not be completed in the given time) the computer performs a series of increasingly deeper searches (N-ply, then N+1, then N+2, etc.) until the allotted time runs out. Thus it is able to produce the best move that the time constraint allows--a computer-chess situation that has many parallels in real-time applications. The computer can combine iterative deepening with various memory functions, particularly refutation and transposition tables, to reorder moves, so that at the next iteration its selected "principal variation" (best sequence of moves found during the previous iteration) is explored first. Another move-reordering technique is to keep a short list of "killer" moves, which are tried first. Killer moves are those that have successfully "cut off" or pruned the search elsewhere. Often these killer moves are captures, so a simplification involves considering capture moves before all others. This technique is nicely generalized in the "history heuristic table" that many programs use. In its most elementary form a history table has 64x64 entries, each containing a



value that measures the frequency with which the corresponding possible move has recently pruned the search.

Move-reordering mechanisms enhance the efficiency of the depth-first alpha-beta search algorithm. Three other improvements--Pearl's Scout algorithm and the related NegaScout and Principal Variation Search (PVS) methods--share a common theme: once a principal variation has been found it is sufficient to show that each alternative is inferior. Any that is not inferior must be re-searched, since it now constitutes the preferred path. Another technique for curtailing the search is called aspiration alpha-beta search. In this approach the value of the tree from the current position is estimated and a narrow search window (customarily plus and minus the value of half a pawn around that estimate) is used. Aspiration searching is a popular and better understood alternative to the Principal Variation Search method, although not as efficient.

It is difficult to be precise about the advantages that more searching provides. The size of the chess tree for any position is highly variable. In many endgames there are only about 8 moves for each side, while in complex middle game positions each side might have close to 80 moves. With today's technology, programs search 9 to 12 ply in the middle game, while at least one programmer claims to extend searches selectively to 40 ply! Selective extensions are based on heuristics devised by individual programmers to explore the sphere of influence associated with a key move: to examine the moves that might defend against a mate threat, or that might provide a counter attack and thus indirectly avoid some imminent loss. Selective extensions are not to be confused with singular extensions. The latter technique re-examines any move that looks singularly good relative to the others. The search depth is increased to determine whether the singular move remains best. In some sense this is a way of extending the principal variation in the small. It is a potentially costly but interesting method.

More popular and more widely used is the null move heuristic, where one side provisionally makes two successive moves. If the value of the position remains poor even with the benefit of two moves in a row, then the line of play is abandoned. This is one way to identify situations where an inevitable loss is otherwise being pushed out of sight beyond the search horizon. While many forward pruning methods fail too often to be useful, null move forward pruning is usually beneficial.

Transposition Table

A transposition table serves as a cache memory and is used to store information about positions that have been visited before, usually during an earlier part of an iterative deepening search. It is so called because it can be used to recognize transpositions in the order of moves. Stored in the entry associated with a position are important items like the "value" of the position, the best move from there, and the length of the previous search. "Value" is computed by applying an evaluation function at the terminal nodes of the tree (the nodes on the horizon where the search is stopping). This evaluation function often includes a quiescent search to help resolve existing capture sequences and other uncertainties in the position, such as pending pawn promotions. Transposition tables are also invaluable as a means of extending search in the endgame, where only a few new moves emerge at each node, the others leading through transposition to positions that have been seen before. These tables do not increase program size or complexity, since the total space allocated to them is simply a matter of cost. Each transposition-table entry requires about 10 bytes of memory, and most programs have tables in the range from 32,000 to 1 million entries, though in 1993 one Supercomputer program boasted a table with a 1,000 million entries! This wide range simply

reflects the memory available to the programmer.

Program Performance and Rating

Despite the underlying similarity in methods there is wide variation in performance among the programs, even in machines using identical hardware. In some cases this merely reflects the effort put into the program's development. For example, although every program has an opening book, there is no basic book for them to use. Each team develops its own. At present these books vary in size from about 10,000 chess positions to about 500,000 positions, although one experimental program has 1.7 million book entries. Conversely, only a few people use Ken Thompson's CD-ROM database of 5 and 6-piece endgames. This is partly for technical reasons related to relatively slow access to the database, but also because most games finish before reaching these known endings. Perhaps programmers are just being realistic about how to spend their time!

When it comes to speed of execution, contemporary programs examine between 3,000 and 500,000 positions per second on a single processor. Big differences in speed exist even for programs using identical machines. There are many explanations. Those who program in assembler tend to have faster programs, but even for the same programming language, not all compilers (translators) produce equally fast executable code. Much depends too on the relative sizes of the brute force, the selective and the quiescent search stages. Extra time is required in the selective stage to assess and identify which moves will be examined. The extent of this slow, knowledge-based process accounts for much of the speed difference. One other factor that influences the speed and strength of a program is the size of its transposition table.

Although many chess programs are similar to each other, their relative playing strength can still differ greatly. Determining that strength is no easy matter, since programs can be tuned to perform well on any standard test suite. For this reason the group who produce the Swedish Rating List use a more traditional approach. All commercially available programs continually and automatically play games against each other, leading to hundreds of statistically valid results. From these data an ELO rating is computed, much like the rating system used for chess-players in America and elsewhere. In the US the average player has a rating over 1500, while experts are in the range 2000-2200 and masters are rated 2200-2400. Above that come the super elite players called Grand Masters, of whom about 100 are active worldwide. At the Eighth World Computer Chess Championships most programs have an ELO rating in the range 2100-2500. The current Swedish rating list is published in each issue of the International Computer Chess Association Journal.

The Future

These days the top chess machines are challenging the Grandmasters, especially in rapid play where the stand-alone PC-based machines have an advantage over multiprocessor-based systems. Stand-alone machines are especially fast, because they don't need the services of a computer network to transmit their moves. Multiprocessor machines using 10 to 100 processors are often better at the standard competition rate of play of 40 moves in 2 hours. Soon systems with 1000 processors, each as powerful as a high-performance PC, will be with us. Even if their efficiency is only at the 50% level, they will be able to search 2 or 3 ply deeper in a typical middle-game position than any single-processor system. By then computers will be clearly out-searching humans. Whether this will be enough to compensate for the human's proven strength in long-term planning remains to be seen. Human chess players are especially skilled at simplifying complex situations and identifying the fundamental issues. They are also adept at provoking long-term weaknesses in their opponent's position, until it becomes indefensible. Despite these advantages, each year we draw closer to realizing the perennial prediction that computers will beat the best humans at chess within 5 years. It could certainly take another decade to achieve that, but the inevitability is clear.

References

These are general sources about Chess Programs and Programming Techniques.

Levy, D. ed. 1988. *Computer Chess Compendium*, New York: Springer-Verlag.

Marsland, T.A. 1992. Computer Chess and Search. In *Encyclopedia of Artificial Intelligence*, S. Shapiro (editor), 2nd edition, 224-241. New York: J. Wiley & Sons.

Herik, van den H.J. ed., since 1983. *International Computer Chess Association Journal*, Universitiet Maastricht, The Netherlands.

Marsland, T.A. and Schaeffer, J. eds. 1990. *Computers, Chess, and Cognition*, New York: Springer-Verlag.

Various editors, 1997-1994. *Advances in Computer Chess*, Volumes 1 to 7, various publishers.