CURRENT CHESS PROGRAMS: A SUMMARY OF THEIR POTENTIAL AND LIMITATIONS*

P.G. RUSHTON AND T.A. MARSLAND

Computing Science Department, University of Alberta, Edmonton, Alberta

ABSTRACT

The purpose of this paper is to discuss ideas used in current chess playing programs. A short history of events leading to the present state of the art is given and a survey made of present day programs. The Newell, Shaw, and Simon program of 1958 is included since it embodies useful ideas that other programs appear not to employ. The possible performance limits for current techniques will be considered, including reasons for these beliefs. A summary of the major ideas contained in these programs is then presented and suggestions made for the improvement and development of future chess-playing programs.

RÉSUMÉ

Le but de cet article est de discuter certaines idées utilisées dans les programmes pour le jeu d'échecs. On présente un bref historique des évènements qui ont abouti à l'état actuel. On fait la revue des programmes actuels. Le programme de Newell, Shaw, et Simon, écrit en 1958, est inclus. En effet, il incorpore certaines idées utiles qui ne semblent pas être encore exploitées dans les programmes actuels. Les limitations des méthodes courantes ainsi que leurs causes sont considérées.

Finalement, on résume les idées principales contenues dans ces programmes et on présente des suggestions pour l'amélioration et le développement des programmes qui jouent aux échecs.

INTRODUCTION

Interest in designing machines to play games has only developed seriously over the past two decades. Many reasons justify the research spent on games. Most games present a closed environment with well-defined rules and goals. Chess has been used extensively for the above reasons and also because devising a high calibre chess program is a difficult task. It is believed that the techniques used in a master class program would be relevant to many other problem areas – natural language processing and theorem proving for example. More reasons for using games, and chess in particular, can be found in References (3) and (10) and in the book *Human Problem Solving* by Newell and Simon.

A basic set of principles for chess playing programs was set down by Shannon.[12] The ideas he suggested involved the following: limiting the size of the game tree; using a polynomial scoring function and employing the forward pruning technique, where the deeper one gets into the tree the fewer variations are considered; using the minimax procedure to search the game tree for the best move; applying the evaluation function only in quiescent positions (one where no move exists which will drastically alter the value of the scoring function); and separation of the phases of the game.

*Received 12 June 1972

Many of Shannon's ideas were applied to the game of checkers by Samuel.[11] This work is mentioned since many of the techniques in Samuel's checker playing program have been applied in current chess programs. The alpha-beta algorithm in conjunction with plausibility ordering (possible moves from a given position are ordered by a crude estimate of their likely worth) and forward pruning are the methods used to search the game tree. A technique that has not been applied to chess is the use of "signature tables" which allow relations between terms in the scoring function to be considered. This evaluation method is non-linear and has enhanced play, but the program still cannot beat the best human checker players.

This paper is divided into two parts. In the first section emphasis will be placed on programs which employ new ideas, are of noteworthy playing ability, and for which reasonably complete documentation was available. Many other programs exist and are described in more general terms. The second section deals with ideas for unwritten or incomplete programs.

Existing Programs

Newell, Shaw, Simon[8]

A goal directed approach to the game was adopted by Newell, Shaw, and Simon.[8] Features of the game were associated with goals, each of which required a move generator, a static evaluator, and an analysis procedure. For ease of modification, each goal consisted of a separate procedure.

In order to select which goals were relevant to a position, a preliminary analysis routine was invoked and the corresponding goals chosen were ordered by their suspected importance. This static set of goals controls the move generation process, including selection of variations, evaluation, and final choice.

Move generators for goals were responsible for proposing moves relevant to a particular goal and finding positive reasons for making these. The generators did not suggest continuations. A proposed move had then to be valued by an analysis procedure which was concerned with acceptability of that move. Before a position can be assigned a value it must be dead with respect to all goals. If a position is not dead for a particular goal, moves are suggested by the corresponding move generator and the resultant positions are checked to see if they are quiescent for all goals. If not, the above procedure is repeated, constituting an analysis of variations, until a position is reached which can be evaluated. The final choice depends on an acceptance value and if a move receives a value greater than this threshold it is played, otherwise the best move found by the alpha-beta backing up procedure will be made.

This program did not realize the state of development and use that the more recent programs have; however, hand simulation gave an indication, in the openings at least, that the program would make moves for reasons similar to those of chess players.

Greenblatt, Eastlake III, Crocker[4]

The program (Mac Hack) written by this group implemented many of

The tree structure is built and searched in a straightforward fashion. All legal moves are obtained for the starting position and ordered by their scores as assigned by the chess heuristics. This ordering is performed only once per move at the root node for the tree. The game tree is now produced by brute force to a set depth (which depends on the time available), the terminal positions evaluated (difference in material is the only consideration), and the move selected using the alpha-beta algorithm.

TECH has played in a number of tournaments and presently has a USCF rating of approximately 1300, which puts it at the strength of a weak amateur with little tournament experience.

Slate, Atkin, Gorlen[13]

Chess 3.5 (as Northwestern's program is called) is relatively young and is capable of learning in a rather limited sense. Its learning ability is restricted since it involves memorization rather than generalization.

A depth-first, forward pruned tree is built, and the alpha-beta algorithm is used to select the move. In order to choose moves to be analysed, a number of chess heuristics are employed. The main one of these chooses a number of moves, dependent on depth, whose scores were the highest. For each position to be evaluated, tables of information are built which, in turn, are used while analysing capture sequences, square control, and pawn structure. This evaluation results in a single number which is the score for the position. This means of scoring may be avoided in the opening since it is also possible to look up moves in a library.

The learning process is not very advanced and is of limited use. One form of learning involves memorization of positions and the moves associated with them. A seemingly better technique involves continual monitoring of the game and when the value for a position is sufficiently different from what was expected by an earlier look ahead analysis, credit or blame is assigned to the previous four moves. Usage of these methods tends to cause the library to grow rapidly.

Chess 3.5 has performed quite well, and indeed, has won the first, second, and third U.S. Computer Chess Championships. The learning ability is an interesting development but appears to be of use only in those positions which are likely to match ones in the library (for example during the opening and endgame). Future work along these lines will be interesting to observe.

Others[9]

The remaining programs, including those by Boos, Raymond, Scott, and Smith, have not introduced new concepts. The approach used has been to build a small tree using a sophisticated scoring function. WITA, written by T.A. Marsland[6] embodies the essential features of this group and is described as follows.

WITA generates every legal move and automatically considers all checks and captures. The remaining moves are selectively added to the same list, depending on the results of a preliminary scoring function. If possible, twenty moves are always retained and in general the original move list is trimmed to

Shannon's ideas. Its ability over and above that of its contemporaries was particularly noteworthy.

For any position a list of legal moves is made and ordered by an assigned plausibility score which is computed using a number of specific chess heuristics. A subset of the possible positions is selected using the plausibility score and these are used as root nodes for another tree. Thus a tree structure is built to a fixed depth and then the evaluation function is applied to obtain a score for the terminal positions. The evaluation function is linear and is comprised of terms reflecting material balance, piece ratio (useful when considering exchanges), pawn structure, king safety, and centre control. The alpha-beta algorithm is used to search the game tree. Use of a hash coding technique avoids unnecessary computation of duplicated positions and also aids in detecting draws by repetition.

As a member of the USCF (United States Chess Federation) Mac Hack has a rating of about 1500. It has improved in play over the years but this was due to an evolutionary process of adding more chess heuristics rather than a learning mechanism within the program.

Kozdrowicki, Cooper[5]

The COKO III program, as it is called by its authors, was based on the assumption that an opponent would resign upon the loss of material. As they noted, this could lead to problems whereby a player could sacrifice material for a mating attack.

The program is organized around the concept of a minimal game. In a minimal game, the move to be made is determined from the application of chess heuristics to information collected from the position under consideration. No look ahead is used. A preliminary analysis routine gathers the information and chess heuristics are separate subroutines.

In order for COKO III to look ahead, a tree of minimal games is constructed. The size of the tree is adjusted dynamically, dependent on the present size of the tree. The sprouting procedure ensures that the next path searched is the one calculated (on the basis of chess heuristics) as most likely to succeed. Thus, this technique is similar to that of Samuel's checker playing program and the methods people use.

COKO III has a broad background of over the board play and has undergone many programming improvements. Its level of play is comparable to that of other chess programs.

Gillogly[2]

Gillogly's program (TECH) was written specifically to provide a baseline for other programs. The ideas are not novel, since TECH uses brute force to examine the game tree.

While developing the program it was discovered that before anything tactical happens (recognizable by material changes) the program had a hopeless position. To rectify this difficulty about 5 per cent of the processing time was devoted to chess specific heuristics.

about two-thirds of its initial length. A separate and more detailed non-linear evaluation function is used to assess the positions. The coefficients of its individual terms vary with depth but otherwise remain unchanged. This second score is used to restrict the number of variations analysed from any position to a user specified limit. The actual tree is forward pruned and duplicate positions eliminated. The move played is the first one whose score, backed-up by the alpha-beta algorithm, exceeds some acceptance value. In order to save computation time, the tree is salvaged from move to move.

WITA has played a number of opponents and its results have been checkered. In general, all of the programs have evolved by the additional programming of chess heuristics and the level of play is in class D (1200–1400 USCF scale).

Unwritten or Incomplete Programs

Good[3]

Good gave a plan for the creation of a chess playing program in five years. However, no one has followed it up in detail. A brief outline of the plan follows.

The first phase involves writing a program to play legal chess, compiling a list of principles, and expressing them as clearly and unambiguously as possible. One should now review the literature of learning theory, concept formation, etc. to see if it contains relevant information. The next step is to devise a special purpose chess language in which to express the principles of chess and chess planning. This second phase involves program organization and division for ease of construction, and then writing it using the special language. Thirdly, one should develop the theory for evaluation techniques, optimal tree search, and generation of goals. One ought to formalize the concept of "progressive deepening" and apply this to the chess program. The final phase consists of trying the program, improving it as seems necessary, and conducting an investigation concerning how one can take advantage of the psychology of a particular opponent.

Botvinnik[1]

A novel idea has been proposed by Botvinnik. He believes it is important to know which pieces are able to reach a certain square or sector of the board in a set number of half-moves. In this manner it is possible to determine the pieces that one should be concerned with when planning a move and it establishes what Botvinnik calls an "horizon." Thus one could vary the horizon by changing the amount of time pieces are allowed to take in arriving at a given area. At the present time some of his ideas have been programmed, but the successful completion of a program based on Botvinnik's ideas has not yet been announced.

Critique

Essentially two approaches to the game of chess can be distinguished among the programs that use scoring functions. TECH does not prune the move tree

and uses a simple evaluation function, whereas most other programs employ a complex scoring function to trim the tree. This trade-off has not been investigated sufficiently but obviously merits attention since, in the second U.S. Computer Chess Championship, TECH placed second, ahead of others which used a sophisticated scoring function.

With the exception of the Newell *et al.* program, all use an evaluation function to limit the size of the tree. Strictly speaking COKO III does not, but its techniques for limiting the size of the tree are sufficiently similar, in our opinion, for it to be grouped along with the others. This approach will not produce a program capable of playing master strength chess. The reason for this belief can be shown by the following analogy. Suppose a small girl is lost in a corn field and is moving this way in an attempt to escape, but is making little progress. If she could raise herself above the corn it would be possible for her to get an idea of the direction in which she should move and eventually find her way out. It is felt that using an evaluation function to determine the proper move results in action similar to the lost girl trying blindly to find a way to escape in that the program has no real idea of the direction to take. This cannot be obtained by examining the corn stalks (the approach of programs using evaluation functions). The move tree should thus be examined with a particular goal in mind. A program using a goal directed approach would behave like the girl who could raise herself above the corn stalks.

This, then, is similar to the fault with current chess playing machines. An evaluation function in no way allows one to get an idea of the direction of the game and there will be a limit to the success of programs written along these lines. Note that when specialized routines are written to handle certain endgames, they are likely to be successful. Such an approach is equivalent to a special form of a goal-directed search.

The Newell *et al.* program[8] used a goal oriented approach but, since work was not completed on the program, it is not known how successful it may have been. One difficulty with this kind of approach is recognizing which goals are to be attained in a position. Newell, Shaw, and Simon do not describe their preliminary analysis routine for selecting goals, so until it is known it would be unfair to criticize this part of their work. We are told that a goal selector rountine is used each time the program is to move, which indicates that there is no planning continuity. The analysis routines, too, seem not to be the best. These procedures essentially tried to make the position dead with respect to their corresponding goal and proposed moves to accomplish this. It would seem a better approach is for planning routines to suggest variations towards the achievement of a goal. These would likely be the areas for possible failure of a program using this approach.

Botvinnik's ideas are important but he has not indicated the significance of goals. His method, used by itself, does not give an indication of the direction of the game and thus will fail. Given a means for determining the relevant goals, these ideas will most certainly assume an important role and must be accounted for in future chess programs (especially since Botvinnik was the world chess champion from 1948 to 1960, except for a few years).

Good's suggestion of formalizing the concept of "progressive deepening" and using a special purpose language in which to express chess programs has not been investigated fully, but some attention should be devoted to these ideas as they will prove relevant if a program is to model human behaviour.

A Look to the Future

If a program could learn to improve its chess play, then most of the problems would be solved. It is reasonable to assume that progressive deepening plays an important role in this respect.

We are working on these ideas at the present and have conceived of a set of pattern matching routines which identify goals worthy of achievement and a set of analysis routines capable of developing and evaluating plans.[7],[10] A learning program will then have the responsibility of changing the preceding sets of routines. In particular, it will make changes depending on the difference between present assessment of a position and that of a much superior chess player. Exact methods for this adaptation have not yet been developed, but it is felt that in order for learning to take place a program must be able to change itself.

Conclusions

Current chess programs will not be of master calibre as long as an evaluation function is used to limit the size of the tree. In keeping with the progressive deepening idea, a better means is needed for selecting and evaluating relevant plans in a given position. Learning is essential for effective play since otherwise every time the program fails, a new change will have to be introduced by hand. This does not seem to be any better than current approaches to the problem. In order to make significant further progress in the area of chess playing, current practices should be examined for something better.

References

(1) M.M. Botvinnik, *Computers, Chess and Long Range Planning*, Springer-Verlag, 1970.
(2) J.J. Gillogly, "The Technology Chess Program," Nov. 1971, Rep., CMU-CS-17-109, Department of Computer Science, Carnegie Mellon University.
(3) I.J. Good, "A Five Year Plan for Automatic Chess," Machine Intelligence, vol. 2, 1967, 89.
(4) R.D. Greenblatt, D.E. Eastlake III, and S.D. Crocker, "The Greenblatt Chess Program" AFIPS Conf. Proc. vol. 31, 1967, 801.
(5) E.W. Kozdrowicki and D.W. Cooper, "Algorithms for a Minimal Chess Player: A Blitz Player" *Int. J. Man-Machine Studies*, vol. 3, 1971, 141.
(6) T.A. Marsland, Unpublished notes, 1970.
(7) T.A. Marsland and P. Rushton, "A Study of Techniques for Game-Playing Programs," *Proceedings of the World Organization of General Systems and Cybernetics*, Oxford University, August 1972. To be published by Gordon and Breach 1973.
(8) A. Newell, J.C. Shaw, and H.A. Simon, "Chess Playing Programs and the Problems of Complexity," IBM J. Res. Develop., Oct. 1958, 320.
(9) "Computer Chess Programs (Panel)," *Proc. of the ACM Annual Conf.*, Aug. 1971.
(10) P. Rushton, "A Critique of Programming Techniques for Playing Chess," M.Sc. thesis, University of Alberta, 1972.

(11) A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers. II-Recent Progress" IBM J. Res. Develop., Nov. 1967, 601.
(12) C.E. Shannon, "Programming a Digital Computer for Playing Chess," Phil. Mag. vol. 41, March 1950, 356.
(13) D.J. Slate, L.R. Atkin, and K.E. Gorlen, CHESS 3.5 User Guide 1971, Northwestern University.