Mary H. Livermore Library
Pembroke State University
Pembroke, N. C. 28372

PRESS
CARD
HERE

The Design of a Heuristic and Learning Chess Program

Psychology 399
Independent Study
Jerry Wayne Gilmer
Spring, 1976

SAMPSON - LIVERMORE LIBRARY
UNC PEMBROKE
PEMBROKE, NC 28372-1510

Case
AS
36
.N6
P45
1976
no.4
c.2

The purpose of this independent study project was to design a chess-playing program for a computer in PL/I programming language. The project was successful as the accompanying computer printout shows.

In recent years, chess programs have become a favorite form of research for computer scientists interested in artificial intelligence. There are several reasons for this. The nature of the chess program leads to development of new techniques; chess-playing is an activity generally regarded as intelligent behavior; the performance of chess programs can be directly compared to the performance of human chess players.

A layman might think that a computer would be unbeatable at chess. With its super-fast rate of operation, it would appear that a computer need only generate all possible moves and all possible replies, etc. ad infinitum. However such a strategy rapidly falls prey to geometric progression. One estimate for the possible number of plays for chess is 10^{120}.

Thus the computer can only search to a certain depth. Most of the better programs search to a preset depth of six or seven ply. If a position is "alive" at that depth - contains captures or checks - then those moves are searched until a "dead" position is reached. This program involves a search of only two ply - the computer's move and its opponent's reply.

Almost all chess programs involve an evaluation function. At the maximum depth, the positions are evaluated according to certain characteristics. The

177944

nine variables in the evaluation function of this program are as follows: space controlled, number and strength of pieces, center control; castling of king, pawn position in front of castled king, passed pawns, doubled pawns, rook on the seventh rank, possession of the two bishop pair.

Most programs involve plausibility ordering of possible moves. Not all moves are investigated - only the more plausible ones. This program has such ordering based on space control. The relative number of squares attacked by each side is used to order the moves according to their desirability. Only the most desirable are investigated -- ten at level one and seven at level two. Thus the program demonstrates tapered forward pruning. A quick calculation shows that seventy possible positions are evaluated at level two.

The evaluation of positions at ply two are backed up to ply I and ply 0 (the read-in board position) by use of Mini - Maxing. Basically the human is expected to choose the minimum of the seven evaluated positions at ply 2 (as this would be his most desirable move). In turn the computer chooses the maximum of the ten backed-up minimum values at ply I (as this is its most desirable move.). Mini - maxing of evaluation functions are omnipresent in game-playing chess programs.

A good chess program needs not only heuristic (or search) techniques, it also needs algorithmic (how-to-do-it) techniques. This could take the form of a book of openings (to save computational time and avoid traps) or of a list of procedures to be followed in choosing moves. In the attached chess program there

opening, midgame, or endgame, based on the number of pieces on the borard. It keeps a list of different parameters for each stage of the game. A passed pawn is much more valuable in the endgame than in the opening, for example.

The Pemborke computation facilities do not possess on-line capabilities. Therefore this program does not actually "play" a game of chess. Instead it reads in a given position and chooses its best move.

Certain things are not implemented in this chess program. Among them are pawn promotion and capturing en passant. The former might so overwhelm by means of its four to twelve moves the simple plausibility ordering mechanism as to preclude investigation of other possibly better moves. The latter is relevant to a game situation as its legality is dependent on a previous move. It is not really relevant to a single position situation which this program handles. The same shortcoming could cause this program to make an illegal castling move. If the involved king or rook had previously moved but then returned to its original position, this program would regard castling as legal. Such problems are minor however and present no particular programming difficulty if a more complex program were practical in terms of time.

This is basically a very simple program. It contains only about 1000 PL/I statements. A program implemented at Duke, called DUCHESS, contained about 5000 PL/I statements. Furthermore it searched to much greater depth than two ply. DUCHESS also employed alpha-beta search techniques. This sophisticated

procedure, which reduces the number of positions which must be evaluated was not included in the accompanying program because of the two ply depth limit. DUCHESS achieved a United States Chess Federation rating of over 1300. For purposes of comparison, this established DUCHESS as the third highest ranked "female" tournament player in the state of North Carolina. The attached program is much too limited to achieve such results. It would not even recognize a mate-in-one by its opponent at ply two. However all the basic parts of a good chess program are included.

Will the descendants of DUCHESS and MORPHY battle for the world chess championship? It appears doubtful for the next couple of decades anyway. The best existing chess programs such as Greenblatt's MAC HAC VI have a rating of about 1550. Bobby Fischer's rating is approximately 2800.

The problem does not seem to be one of speed. Even if computer speed were to explode exponentially due to some engineering breakthroughs, it would not enable the computer to search much deeper as positions explode exponentially also.

The problem seems to be one of software. More algorithmic knowledge must somehow be programmed into the computer. Assuming computer memory storage is made large enough, a grandmaster - level of knowledge of the openings seems plausible. Play in the midgame could be improved by specific algorithmic strategies and by refining of the evaluation function through various learning techniques. It is in the endgame that present programs are weakest. Human selective

search works well in this stage; machine brute force search is poor is in this stage. It is difficult to give the computer the goals it needs in this stage in order to plan its strategy.

More likely than an early world chess championship for a machine is joint authorship of a chess manual by a grandmaster and a computer. The computer's speed and depth of search combined with the grandmaster's algorithmic knowledge could be a big boost in chess research. Chess knowledge at present depends today in great part on published grandmaster games. A man-machine partnership could open a whole new facet of chess knowledge.

In the spirit of such a partnership the accompanying program is christened MORPHY after the first American world chess champion.

References

Bell, A.G. Games Playing With Computers. London: George Allen and Unwin Ltd., 1972.

Jackson, Philip C. Introduction To Artificial Intelligence. New York: Mason and Lipscomb, 1974.

Kozdrowicki, Edward W. and Dennis W. Cooper. Algorithms for a minimal chess player: a blitz player. International Journal of Man-Machine Studies (1971) 3, 141-165.

Kozdrowicke, Edward W. and Dennis W. Cooper. Coko III: the Cooper-Kozdrowicki chess program. International Journal of Man-Machine Studies (1974) 6, 627-699.

Nilsson, Nils J. Problem-Solving Methods in Artificial Intelligence. New York: McGraw-Hill, 1971.

177944