A Chess Playing Program for the IBM 704

A. BERNSTEIN
NONMEMBER AIEE

M. DE V. ROBERTS
NONMEMER AIEE

T. ARBUCKLE
NONMEMBER AIEE

M. A. BELSKY
NONMEMBER AIEE

THIS paper describes the program which enables a human being to play a game of chess with the International Business Machines Corporation (IBM) *704* computer. The machine may play either white or black, and is capable of playing a complete game of chess, including such moves as castling, promoting, and capturing en passant.

The program is divided into five parts: (1). Input-output, (2). Table generation, (3). Evaluation, (4). Decision, and (5). Tree

Input-Output

The input-output section of the program allows the human player to state his moves to the machine in the normal English notation on punched cards, for example: P-K4 or B(Q4) × P(N7). The program then translates this statement into the machine notation for moves, which gives the "to" square and the "from" square with a code. A move in the machine notation may also be entered from the keys on the console.

The squares of the board are numbered octally from 00 in the upper right-hand corner of the board to 77 in the lower left. For a regular move, the tag field contains a 1, the decrement contains the "from square" co-ordinates, the address contains the "to square" co-ordinates, and the prefix contains a minus sign if the move is a capture. For a castling move, the tag field contains a 2, the decrement contains the "to square" of the king, the address contains 77 if castling up, 0 if castling down, and the prefix is 0. (Castling upwards means that the final position square of the king has a higher number than its initial square.) For an en passant capture, the tag field is 3, the decrement contains the "from square" of the capturing pawn, the address contains the "to square" of the capturing pawn, and the prefix contains a minus sign. For a promotion, the tag field contains a 4, the decrement contains the "from square" of the pawn that promotes, the address contains the "to square" on which promotion occurs, and the prefix contains a minus sign if the move is a capture.

At the start of a game the machine's pieces always reside on the squares 00 to 07, 10 to 17; the opponent's on the squares 60 to 67, 70 to 77.

If the machine were to play white and make P-K4 as its first move, in machine notation this would be the octal word, 000014100034.

When the machine has made a move, or accepted an opponent's move, it prints that move in the English notation, together with a picture of the board position, and makes a record of the move on tape. At the end of the game the full score is printed.

Table Generation

The table generating section accepts as its input a table of the board position which will be referred to as *TA*1. *TA*1 at the beginning of the game is arranged so that it represents the starting position; but any board position may be assembled into *TA*1 and entered as the starting position for the program.

*TA*1 is 64 words long, one word per square. The first word of *TA*1 refers to square 77, the last word to square 00. If the square is empty then the word is all zero. The word is negative if the square is occupied by a machine piece. The address contains a number indicating whether the piece is a king (5), queen (0), rook (1), knight (4), bishop (2), or pawn (3), and the decrement contains an indexing quantity to link *TA*1 with Table 2, which will be called *TA*2.

*TA*2 is now generated from *TA*1 in the following form. It is 32 words long, one per piece. The order is shown in Table I.

A word in *TA*2 is of the following kind: It is zero if the piece has been captured. Otherwise, the address contains the value of the piece, and the decrement contains the co-ordinate of the square where the piece is located. It can be seen that the decrement of *TA*2 is the indexing quantity of *TA*1.

Table 31 and Table 32, called *TA*31 and *TA*32 are also generated from *TA*1. *TA*31 refers to the machine and *TA*32 to the opponent. *TA*31 and *TA*32 are similar in nature and function. Thus it is sufficient to describe *TA*31.

*TA*31 is 512-words long, eight words per square. The first eight words refer to square 77 and the last eight words refer to square 00. Consider the eight words referring to square (*xy*). If the square *xy* is occupied, the eighth word is negative, and contains a bit in the first position if it is occupied by the machine but not if occupied by the opponent. The address of the eighth word contains:

1. The number of machine pieces which may move into square *xy* if *xy* is empty.
2. The number of machine pieces which may immediately capture an opponent's piece situated on *xy*.
3. The number of machine pieces immediately defending a machine piece situated on *xy*.

The location of the pieces described in the address of the eighth word are in the decrements of the words 8 through 1. The address of the seventh word stands for the number of doubled machine pieces bearing on square *xy*, e.g. a rook behind a rook bearing on *xy*, or a queen behind a bishop, etc. The location of these doubled pieces are to be found in the addresses of the words 6 through 1.

It will be seen that the location of the pieces bearing on *xy* is the indexing quantity needed to find that piece in *TA*1, while each square *xy* in *TA*3 is eight times the indexing quantity of *TA*1.

All possible moves are now listed in *TA*3, for each square has in it the number and locations of all pieces which may move into it.

The routine which generates *TA*3 also examines the position for any possible pieces pinned to the king, so that no illegal moves are listed in *TA*3.

*TA*2 and *TA*3 are generated in an average time of 89 milliseconds.

Evaluation

The evaluation routines calculate a score for the machine and the opponent based upon the following criteria:

1. Mobility
2. Area control
3. King defense
4. Material

A. BERNSTEIN and M. DEV. ROBERTS are with International Business Machines Corporation, New York, N. Y. T. ARBUCKLE and M. A. BELSKY are with The Service Bureau Corporation, New York, N. Y.

The program counts the number of available moves for each side; the number of squares completely controlled by each side; the number of controlled squares around each king; and the material count of each side.

These criteria are parametrized for easy variation, so that different values may be given to center squares, or to the squares around the king. At present criteria 1, 2, and 3 are added together while 4 is multiplied by a large factor before adding it to the total. This prevents the machine from sacrificing material, and encourages it to exchange when it is ahead, as it considers the ratio of its own and opponent's scores.

Decision Routines

The decision routines serve the function of producing a set of moves to be considered for further analysis. The idea behind these routines is to select a small number of moves which are potentially good (for strategic reasons) rather than to eliminate from all possible moves the much larger number which can be determined to be bad.

The decision routines examine the position and determine whether certain states exist; if the answer is yes, certain moves are generated. The questions asked are:

1. Is the king in check?
2. (*a*). Can material be gained?
   (*b*). Can material be lost?
   (*c*). Can material be exchanged?
3. Is castling possible?
4. Can minor pieces be developed?
5. Can key squares be occupied? (Key squares are those squares which are controlled by diagonally connected pawns.)
6. Can open files be occupied?
7. Can any pawns be moved?
8. Can any piece be moved?

If the answer to question 1 is yes, the machine asks itself whether it is checked by one or two pieces; this information is obtained by observing the square *xy* on which the king is located in *TA*32, the address of word 8 will be 2 if a double check exists and 1 if a single check. If it is a double check the only possible answer is a king move; the machine examines the empty squares around the king for legality of moves and if any of these moves are possible, the move is constructed and entered in the Plausible Move Table. The routine then returns control of the tree. If instead the king is in check by a single piece, interpositions of pieces, and the capture of the checking piece are considered, and if possible these moves as well as the king moves are entered in the Plausible Move Table. Control then returns to the tree.

Table I

1	Machine King
2	Machine Queen
3	Machine Bishop
4	Machine Bishop
5	Machine Knight
6	Machine Knight
7	Machine Rook
8	Machine Rook
9	Machine Pawn
10	Machine Pawn
11	Machine Pawn
12	Machine Pawn
13	Machine Pawn
14	Machine Pawn
15	Machine Pawn
16	Machine Pawn
17	Opponent's King
18	Opponent's Queen
19	Opponent's Bishop
20	Opponent's Bishop
21	Opponent's Knight
22	Opponent's Knight
23	Opponent's Rook
24	Opponent's Rook
25	Opponent's Pawn
26	Opponent's Pawn
27	Opponent's Pawn
28	Opponent's Pawn
29	Opponent's Pawn
30	Opponent's Pawn
31	Opponent's Pawn
32	Opponent's Pawn

If the answer to question 1 is no, the program goes down to question 2 and if the answer to 2(*a*) is yes, lists those moves which gain material in the Plausible Move Table; if 2(*b*) is yes, finds which moves will take the attacked piece to safety, enters them in the Plausible Move Table, and if 2(*c*) is yes, will enter those exchanges in the Plausible Move Table. Whenever the Plausible Move Table is filled, control goes back to the tree for further examination. At the end of question 2 if the Plausible Move Table is not filled, question 3 is asked. If yes, the castling move is constructed and entered in the Plausible Move Table. If the answer to question 3 was yes, then control goes back to the tree. If no, control goes to question 4, etc.

The reason for stopping the decision routine, if castling is possible, is that the castling move does very little to enhance the score of a position, but is nevertheless a very important element in bringing the king to safety. Therefore, whenever castling is possible no other alternatives except for material exchanges are given, and eventually, when there are no exchanges or pieces to be gotten out of attack, the program is forced to castle.

The ordering of these routines is important. At the beginning of the game questions 1, 2, and 3 are not applicable and questions 4 and 7 are the only ones that produce moves. In the middle game, questions 2, 5, and 6 bear the brunt of the work. In the end game, questions 5, 6, 7, and 8 are most employed. This ordering is arbitrary and will probably require adjustment as experience and more knowledge of the way the program plays is gained.

The Plausible Move Table is limited by time and coding considerations to seven.

The Tree

The tree is the master routine. It operates in the following manner: Having received a position, it asks for a score of the position, and for the set of plausible moves; executes the first move; gets a score and asks the decision routine for a set of plausible replies. It performs the first of these and repeats the procedure until it reaches the fourth level, where it performs each of the opponent's moves, obtaining a score for each. It now takes the maximum score of this last set of plausible moves and brings it down as the score of the move made on the third level, undoes that move on the third level, makes the second move of that set and requests plausible replies for the opponent. It performs each of these, gets their score, and brings back the maximum opponent score (its own minimum score) for the second move.

In this fashion it examines the full tree, and determines which move will maximize its score. With a plausible move set of seven, and depth of four half-moves, this examination takes an average of 8 minutes.

The tree has a selective mechanism which can cut some branches to be examined. This mechanism is based upon the relative scores of each of the plausible moves, and will function efficiently only after the scoring parameters are well adjusted. When this selective method is in operation only moves which are greater or equal in score to the position score before the move was made are allowed to be examined further (except when no moves are better, in which case it takes only the two highest scored moves).

At present this makes the total time for producing a move 2 minutes on the average. The program plays more conservatively in this mode, but does not allow certain moves which are definitely good, and which are otherwise considered and sometimes made.

Conclusions

The machine plays a passable amateur game at present during the opening and

middle game. It is weak toward the end, but the authors believe that additions to the decision routine will remedy the situation. There are several other routines that could be expanded, such as defending pieces rather than moving them away when attacked. All of these are, however, demonstrable, and add little to the entire scope of the program.

Self-adjustment routines are being considered, to change the value of parameters after the loss of a game, and to change the ordering in the decision routines, but these are still in the talking stage. While it is true that the machine makes the same move given in the same position, there are so many positions, that its play is not predictable in a new situation.

Applications of Digital Computers to Problems in the Study of Vehicular Traffic

WALTER HOFFMAN
NONMEMBER AIEE

RICHARD PAVLEY
NONMEMBER AIEE

THE study of vehicular traffic on a scientific basis is still in its infancy. This is due to the enormity of the job. Currently, research in this area is carried on along two distinct lines. First, there is the approach which attempts to study traffic conditions in the small and gradually builds up to larger systems. This is an approach which may accomplish the desired goals, but from all current indications success in this area is distant.

The second approach is a study of traffic problems in the large, meaning a major metropolitan area. Clearly, the methods which are being used today are relatively crude; but constant research for new methods and improvement of the old ones had led to results which have been verified in actual traffic situations, and the predictions made have agreed remarkably well with the observed data.

The research reported in this paper is based on data obtained by the Detroit Metropolitan Area Traffic Study in 1953. At that time a survey of existing traffic conditions was made under the direction of Dr. J. Douglas Carroll at a cost of one million dollars. The survey consisted of three parts: First, a suitably large sample of the population of the metropolitan area was interviewed at home and information was obtained as to the origins and destinations, nature, and frequency of the trips which were made. Second, the major truck and taxi companies were consulted and the same kind of information was obtained from them. Third, roadside interviewing stations were set up and vehicles were stopped and drivers were asked the origin and destination and nature of their trips.

The metropolitan area which was studied included the city of Detroit and its suburbs, covering three counties, and extended far enough beyond them to include those areas which could reasonably be expected to play any part in the metropolitan Detroit area traffic picture at the end of the forecast period, which was set at 1980. The area was divided into 265 zones, and the survey data were expanded to give the total number of trips between any pair of zones for an average 24-hour period. These results were then checked by comparing actual traffic counts across screen lines against computed traffic volumes. It was found that the data obtained from the traffic survey were uniformly about 10% low, and suitable adjustments were made. Thus, an accurate picture of the existing traffic flow had been obtained and the data were presented in the form of a matrix of order 265.

The assumption was made that any trip which originated in the metropolitan area would retrace itself within a 24-hour period so that the matrix mentioned was symmetric with nonnegative integers as entries. Thus, the element a_{ij} of the matrix represents the number of vehicles which make a trip between zones i and j.

The first problem which presented itself was the prediction of this matrix for some future date. It is well known that the trip generating characteristics of the given area depend on land use and on the population of the area. Thus, it was a relatively simple matter to predict the growth of the number of trips for each of the 265 zones. The difficult problem is the distribution of the total number of trips originating or terminating in a given zone over the remaining zones. There are a number of methods in use which attempt to cope with this problem, all of which are imperfect. The problem can be stated mathematically as follows: Given a real symmetric matrix with nonnegative integral entries, find a matrix of the same kind whose row sums are prescribed. Unfortunately, this mathematical problem does not have a unique solution. Its formulation thus must be revised to take into account the characteristics of traffic flow. All of the methods currently used operate in a similar fashion.

If the predicted trip volume between zones i and j is denoted by a'_{ij}, then this quantity must be influenced by the growth factors g_i and g_j of both zones which are involved. Thus a_{ij} must be multiplied by some mean value of the appropriate growth factors. In practice, a_{ij} is multiplied by both growth factors, after which an adjustment is made by dividing the product, which is thus formed by some normalization factor, K. The various methods employed differ in the choice of K. Regardless of how K is chosen, the new matrix obtained by this process will not have the correct row sums. To obtain a matrix of the proper kind the process is repeated, and after a few iterations convergence is obtained. The methods used vary in the number of iterations required for convergence. The method used at this laboratory was developed by Dr. Carroll's staff and has become known as the Detroit Method. The number K in this method is obtained by dividing the sum of all matrix elements at the end of any iteration into the predetermined sum of the elements of the final matrix. This method has been improved upon by choosing K as that factor which will give the total matrix sum as prescribed at the end of any iteration. It is clear that this will accelerate convergence.

Regardless of which method was used, this explains how the traffic picture for the entire metropolitan area is predicted.

A new problem is the presentation of this large number of individual data elements in an easily comprehensible form. The accepted means of doing this is the so-called trip-desire map. This is a map of the area divided into areas of equal traffic density by means of isolines. Production of such a map involves a great deal of labor. The usual procedure can

WALTER HOFFMAN and RICHARD PAVLEY are with Wayne University, Detroit, Mich.