A Comparison and Evaluation of Three Machine Learning Procedures as Applied to the Game of Checkers*

Arnold K. Griffith**
The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., U.S.A

Recommended by Allen Newell

ABSTRACT
This paper presents two new machine learning procedures used to arrive at "knowledgeable" static evaluators for checker board positions. The static evaluators are compared with each other, and with the linear polynomial used by Samuel [9], using two different numerical indices reflecting the extent to which they agree with the choices of checker experts in the course of tabulated book games. The new static evaluators are found to perform about equally well, despite the relative simplicity of the second; and they perform noticably better than the linear polynomial. An indication of the significance of the absolute values of these two numerical indices is provided by a discussion of a simple, purely heuristic, static evaluator, whose performance indices lie between those of the polynomial and those of the other two static evaluators.

1. Introduction

Over the past dozen years a number of machine learning procedures have been devised and investigated. A particular procedure which has received perhaps the most attention learns to make decisions based on the values of weighted sums of factors related to that decision. The learning itself consists

* Work reported herein was supported by the Artificial Intelligence Laboratory, an MIT research program sponsored by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract No. N00014-70-A-0362-0002, and formerly under Contract Nonr-4102(01).

** Present address: Information International Inc., 12435 West Olympic Boulevard, Los Angeles, Calif. 90064, USA.

in the acquisition of values for these weights which lead to the fewest errors in the resulting decisions. Early examples are the "Pandemonium" of Selfridge [11], the "Perceptron" of Rosenblatt [8], and the static evaluation polynomial of Samuel [9] discussed in the present paper. More recent elaborations of this type of learning procedure are discussed by Nilsson [7]; and its limitations have been treated in detail by Minsky and Papert [6]. An early procedure of quite a different sort, involving an adaption of the Skinnerian "reinforcement learning" procedure [12], is presented by Friedberg [1] and Friedberg et al. [2]. Recent learning procedures which more or less depart from the strict Perceptron paradigm include those discussed by Uhr and Jordan [15], Slagle and Farrell [13], Michie and Ross [5] and Winston [16].

The present paper concerns itself with several procedures which "learn" how to evaluate the relative strengths of checker board situations from a series of paradigm examples of strong and weak positions. The first procedure, investigated at length by Samuel [9], involves the acquisition of appropriate weighting factors in a linear polynomial. The second "Signature Table" procedure developed by the present author [3], and further investigated by Samuel [10] and more recently by M. Smith [14], uses a non-linear technique. The third procedure is formally similar to the second; but very much simpler. In the final sections of this paper the performance levels of these systems, after each has learned approximately to capacity, are put into perspective by contrasting them with the performance level of a few well-chosen heuristics.

2. The Task to Be Learned

The task which is learned by the various procedures to be described is to make a reasonably accurate assessment of the strength of a checker board position on the basis of its salient features. This is known in the literature as a static evaluation procedure, and is to be contrasted with lengthy calculations using a look-ahead procedure such as the alpha-beta minimaxing technique (see [9, 10]). The procedures examine a board position, apply some set of learned information, and output a number which is high for strong or winning positions, and low for weak or losing positions.

The purpose of studying a board evaluation procedure is twofold. First, a static evaluator appears to be a necessary component of a successful program for playing a board game such as checkers or chess. Secondly, and more importantly, such procedures are interesting in their own right as objects of study in furthering our understanding of human intelligence, and furthering the art of Artificial Intelligence.

3. The Material on Which Learning Takes Place

The various learning procedures to be described collect information from tabulated games between checker experts.[1] These games are played through by a computer program which, for each position, generates all legal moves from that position, and records the positions resulting from these moves. Among these positions is the one resulting from the move actually made in the game, and these positions are collected separately in a class of "strong" positions. The remaining positions are assumed to be less strong, and are collected in a second category of "weak" positions.[2]

The book games used by the third learning procedure, and by the heuristic procedure discussed in Section 8, consisted of the main trunks and principal variations of the games appearing in Lees' guide [4], a standard early work including basic lines of play on all playable openings, and comprising about 20 000 positions. These gave rise to a total of 20 000 strong positions and about 124 000 weak positions, since the average number of legal moves per position is about 7.2. The material used in connection with both the Samuel polynomial and the signature table procedure consisted of games from various sources which are assumed to be approximately comparable to those from Lees' guide.

4. Measures of Relative Performance

Two numerical indices were used to assess the relative performance of the various static evaluators. These indices not only provide evidence of the degree to which proficiency increases as more information is assimilated during the learning process, but also provide a means of assessing the relative knowledge of the various procedures after learning to capacity. For all but the first procedure, data of both indices are available. In addition, the heuristic method discussed in the analysis section lends itself to evaluation in terms of exactly these indices, allowing a direct comparison of its "knowledgeability" with that obtained by the learning procedures.

Both performance indices reflect the extent to which the assessment of the strength evaluation procedure, after some amount of learning, agrees with the experts' moves in tabulated games. A number of these games, usually not those involved in the learning process, are "played through", and for each position occurring in the course of the game, all legal moves are generated, and the resulting positions are rated by the evaluation procedure.

[1] The procedure here described, developed by Samuel [9], was used by the present author as well.

[2] Positions sometimes occur in which there are two or more "best" moves. Evidently, this would result in putting a certain number of strong positions in the weak category. However, the effects of this were minor since much stronger use is made of the information that a position is strong than that it is weak.

Among these alternatives is the "book position", i.e. the one actually played into by the checker expert; and a perfect strength evaluation procedure would assign the highest score to this position.[3] The performance indices measure the extent to which this ideal performance is achieved.

The first index, devised by Samuel [9], is a quantity similar to a correlation coefficient. For each set of alternative positions, the number of positions which were rated equal to or higher than the book position (not including the position itself) was noted, together with the number rated lower. The sum H of the first of these quantities from a large number of positions was computed, together with the sum L of the second of these quantities. The performance index C is given by the formula

$$C = (L-H)/(L+H).$$

In the case of a strength evaluation procedure which was in exact agreement with the experts, H would be zero, resulting in a value of 1 for C. If the strength evaluation procedure always gave the book position the lowest score, the value of L would be zero resulting in a value of -1 for C. Finally, if the evaluation procedure simply guessed, then L and H would be equal and C would be zero.

The second index D, suggested by the present author, is simply the fractional number of times the evaluation procedure assigned a higher score to the book position than to all the alternatives. In the case of a perfect evaluation procedure, D would have the value 1, and for a very poor one, which never gave the book position a relatively high score, D would have the value 0. In the case of guessing,[4] D would have a value of around 0.15.

5. The First Learning Procedure: The Samuel Polynomial

The checker board static evaluator investigated by Samuel in [9] consists of a weighted sum of the values of parameters, or numerical descriptors, of a checker board. An example of a parameter used by Samuel might be the number of central squares occupied by a player's pieces. If a board position is denoted by B and the value of the ith parameter for board B is denoted by $P_i(B)$, then the static evaluator function $S_1(B)$ is given by

$$S_1(B) = A_{1,j}P_1(B) + \ldots + A_{n,j}P_n(B).$$

The second subscript on the coefficients ranges from 1 to 5 and indicates the "temporal phase" of the game at the time the move occurred.

The various learning procedures studied by Samuel consisted in various

[3] Exceptions would occur either when the expert made a bad move, a possibility which will be ignored; or when there are two "best" moves, in which case both should be given equal values.

[4] Since the branching factor in checkers is around 7, guessing would give the highest score to the book position about 1 time in 7, or about 0.15 of the time.

methods for arriving at a good set of values for the $A_{i,j}$'s. In earlier studies [9], a so-called "non-parametric" approach was used, in which the coefficients were successively adjusted to make $S_1(B)$ a good measure of the strength of a position. In later studies [10], a "parametric" procedure was used, which involved calculating the coefficients from a sort of correlation coefficient relating the value of the parameter with strength and weakness of positions as defined in Section 3.

Samuel reports in [10] that on a particular sample, which shall be here referred to as "Sample 1", a value of 0.26 was obtained for the performance index C for this learning procedure. In [3], the present author reports a C value of 0.27 on another sample of one thousand positions, which will hereafter be referred to as "Sample 2".

6. The Second Learning Procedure: Signature Tables

The signature table static evaluator employs a particular non-linear function of the board parameters used by the "Samuel polynomial" just described. These were modified to output only 3 values, 0, 1 and 2, corresponding to high positive, near zero and low negative values of the parameter. The modified parameters were grouped into sets of five, called "signature types". For each signature type, 5 tables were constructed corresponding to the five phases of the game as used by Samuel in the linear polynomial procedure. Each table contains 243 (3^5) entries corresponding to all the possible configurations of values of the five parameters comprising that signature type. Thus a board position gives rise to a series of 5-tuples, called "signatures" of values (which can be 0, 1 or 2) which in turn correspond uniquely to a series of entries in the set of signature tables for the temporal phase of the board position. As before, denoting a board position by B and the modified parameters by $P_i'(B)$, then the strength function may be written as

$$S_2(B) = F_{1,j}(P_1'(B), \ldots, P_5'(B)) + \ldots + F_{n,j}(P_{5n-4}'(B), \ldots, P_{5n}'(B)),$$

where the functions $F_{i,j}$ are explicitly tabulated.

The process of learning consists in the acquisition of the values of the tabulated functions. Each table entry is calculated as the quotient of the number of strong board positions to which the entry corresponds, divided by the total number (strong and weak) to which it corresponds. The process of learning is simply the acquisition of this data from the sets of strong and weak positions derived from the book games. After learning, a board position's strength is rated by accessing from the various tables the entries to which the position corresponds, and summing them.

The material upon which learning took place was divided into a series of sections, and the learning procedure was applied to each section in turn, retaining the information "learned" from preceding sections. The performance index C was calculated after learning was complete on each section, using a set of positions not among those used in the learning process. As would be expected, there was a general upward trend in this index as successively more positions were involved in the learning process. In addition, the rate of increase slowly dropped toward zero, and the index appeared to asymptotically approach some limiting value. A "learning curve", obtained by plotting these successive values, is illustrated in Fig. 1. The lowest curve,

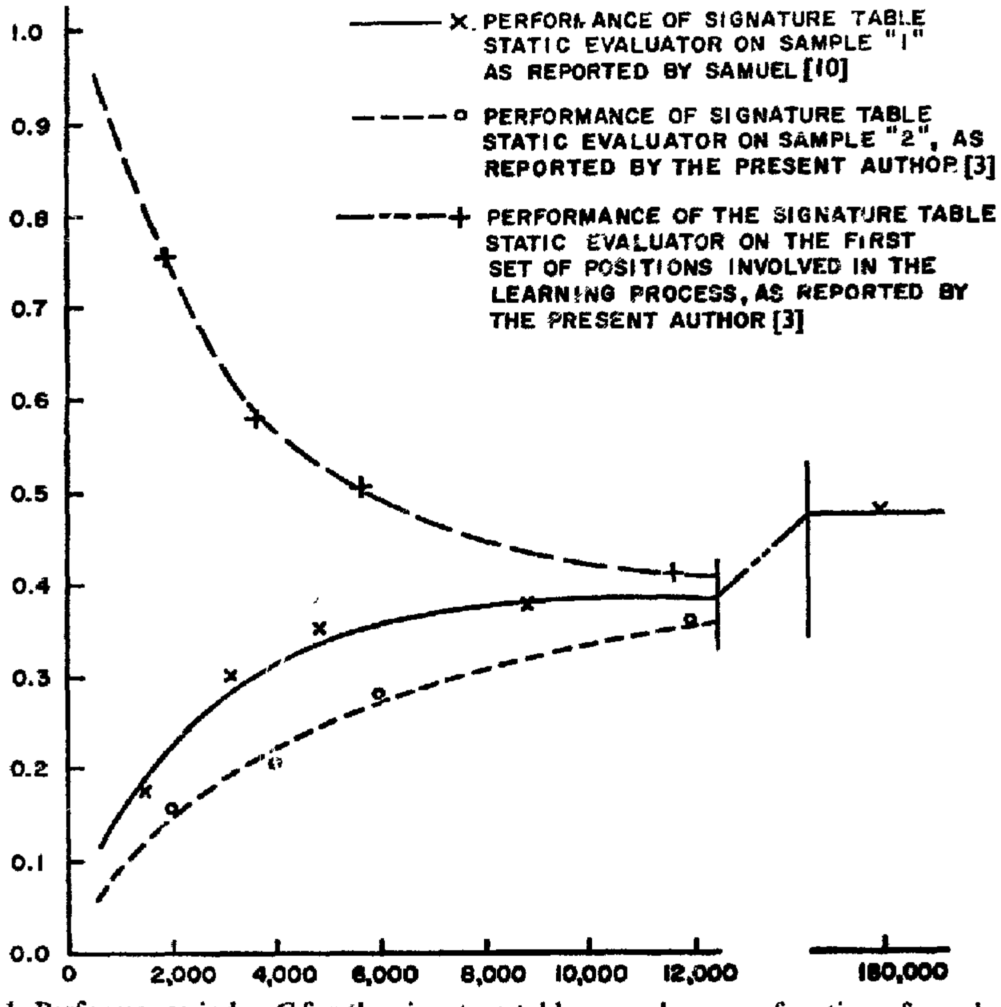

FIG. 1. Performance index C for the signature table procedure as a function of number of positions involved in learning.

results of the present author, involve Sample 2. Results on Sample 1 as reported by Samuel comprise the other of the two lower curves. The upper curve in the figure is the result of computing values of C on the first section of the data upon which learning took place. The highest data point on this curve represents the results of both learning and testing on this one small data sample. The very high value of the performance index C is a result of the fact that this sample was largely learned by rote. This is because the tables are so sparsely filled in at this point that most entries contain information about a single position from the original data. As more information is assimilated, a sort of "generalization" takes place, and the performance index drops in value, indicating that this learning scheme is better at rote learning than at generalized learning. It would seem that if the test sample used to derive the upper curve were exactly as difficult, in some sense, as the test sample used to derive the lower curves, then the asymptotic value to which the upper curve tends would give an upper bound on values of the lower curves.

It may be seen from the curves in Fig. 1 that for Sample 1, the index C has a value of at least 0.48, and for Sample 2, it has a value of at least 0.34.

In [10], Samuel reports additional data obtained by testing the signature table procedure on a "test lot of 895 representative moves", which we shall here refer to as "Sample 3". The data consists in the fractional number of times the signature table procedure rated no position higher than the book position, the fractional number of times one position was rated higher, and so forth. Evidently, the first of these is the performance index D, and its value is 0.38 in this case.

7. The Third Learning Procedure: Move-Phase Tables

The third learning procedure is similar to the second in that it employs functions whose values are tabulated. This time the game is divided into 6 phases.[5] For each phase, there is a table with 98 entries corresponding to each possible legal move in the game.[6]

The assignment of a value to a position consists in identifying both the temporal phase of the position, and the immediately preceding move which brought about the position. The strength value is simply the value of the corresponding entry in the table for that phase.

The process of learning consists, as before, in the calculation of the values to be placed in the tables. This involves a process similar to that used for the signature table technique. A large number of book games are played through and for the strong and weak positions, as defined in Section 3, the temporal phase is noted, together with the immediately preceding move which brought about the position. For a particular table entry, the number of strong

[5] The phases are: 1st ten moves, 2nd ten moves, 3rd ten moves, 4th ten moves, 5th ten moves, and 50th move onward.

[6] Since each piece can move in 4 directions at most, and only 32 squares are played on, there are a maximum of 32×4 possible moves. In fact, the number is smaller (98) since edge squares allow fewer moves. A move is indicated by the square numbers corresponding to the origin and destination of the piece moved. The squares played on are numbered from 1 through 32, with the squares on the first row of the "black" side being numbered 1 through 4 from right to left, and so forth.

positions corresponding to it is calculated. Again, this is simply the number of strong positions in the given phase arrived at by making the move corresponding to that table entry. Similarly, the number of weak positions corresponding to that entry is calculated. The table entry itself is simply the quotient of the first quantity with the sum of both. The table entries corresponding to the 49 possible forward moves[7] are given in Table 1. As an example of

TABLE 1. The move-phase table, for forward moves, generated by the third learning procedure.

	SCORES FORWARD MOVES					
MOVE	PH. 1	PH. 2	PH. 3	PH. 4	PH. 5	PH. 6
(1 5)	.07	.09	.12	.06	.05	.05
(1 6)	.11	.18	.15	.06	.07	.02
(2 6)	.07	.11	.12	.10	.07	.12
(2 7)	.11	.11	.15	.11	.04	.08
(3 7)	.17	.17	.17	.09	.01	.01
(3 8)	.05	.06	.07	.06	.04	.11
(4 8)	.30	.40	.38	.18	.02	.09
(5 9)	.24	.22	.21	.16	.12	.09
(6 9)	.12	.23	.22	.27	.10	.52
(6 10)	.21	.22	.36	.27	.39	.33
(7 10)	.24	.34	.30	.27	.30	.07
(7 11)	.11	.12	.27	.30	.22	.41
(8 11)	.46	.42	.41	.26	.25	.22
(8 12)	.22	.31	.37	.27	.31	.50
(9 13)	.32	.24	.33	.32	.19	.37
(9 14)	.38	.35	.38	.31	.34	.47
(10 14)	.14	.13	.24	.35	.43	.27
(10 15)	.20	.19	.36	.40	.38	.32
(11 15)	.36	.42	.37	.34	.41	.34
(11 16)	.28	.23	.35	.37	.28	.20
(12 16)	.11	.10	.21	.18	.12	.08
(13 17)	.11	.16	.26	.17	.10	.22
(14 17)	.30	.29	.20	.38	.31	.20
(14 18)	.22	.12	.31	.33	.33	.24
(15 18)	.17	.18	.30	.41	.28	.29
(15 19)	.05	.34	.29	.39	.28	.41
(16 19)	.37	.41	.32	.42	.25	.27
(16 20)	.32	.21	.33	.30	.21	.28
(17 21)	.18	.21	.25	.26	.22	.14
(17 22)	.00	.27	.89	.58	.45	.24
(18 22)	.15	.28	.42	.44	.29	.18
(18 23)	.00	.42	.40	.49	.24	.30
(19 23)	.00	.20	.72	.46	.29	.26
(19 24)	.00	.20	.33	.27	.27	.14
(20 24)	.00	.05	.24	.24	.15	.15
(21 25)	.00	.00	.77	.69	.36	.30
(22 25)	.00	1.00	.90	.30	.21	.05
(22 26)	.00	1.00	.55	.50	.34	.25
(23 26)	.00	.00	.75	.50	.18	.22
(23 27)	.00	.00	.33	.35	.16	.06
(24 27)	.00	.00	1.00	.63	.52	.30
(24 28)	.00	.36	.25	.08	.08	.09
(25 29)	.00	.00	.50	.15	.14	.15
(25 30)	.00	.00	.40	.97	.36	.18
(26 30)	.00	.00	.50	.43	.21	.17
(26 31)	.00	1.00	.62	.29	.22	.25
(27 31)	.00	.00	.33	.58	.31	.21
(27 32)	.00	.00	.75	.18	.10	.05
(28 32)	.00	.00	1.00	.47	.51	.37

[7] By far the most common moves, since until a piece is "crowned" by moving to the opposite side of the board, it can only move forward.

the significance of these numbers, the value 0.40 beside (4 8) in the phase 2 column indicates that 40% of the time this move[8] was available it leads to a strong position, i.e. it is a "good move" independent of other considerations!

This learning procedure, devised and implemented by the present author, was applied to the positions from Lees' guide mentioned in Section 3. Learning took place on all but the first 3 000 or so of these positions, which were set aside for testing performance, and which will be referred to as "Sample 4". The lower curve in Fig. 2 is a "learning curve", similar to the lower curve in Fig. 1, which gives values of the C index on this reserved sample, as a function of the number of positions out of the remaining 20 000 that were involved in the learning process. The upper curve in Fig. 2, similar to the upper curve in Fig. 1, gives values of the C index obtained by testing

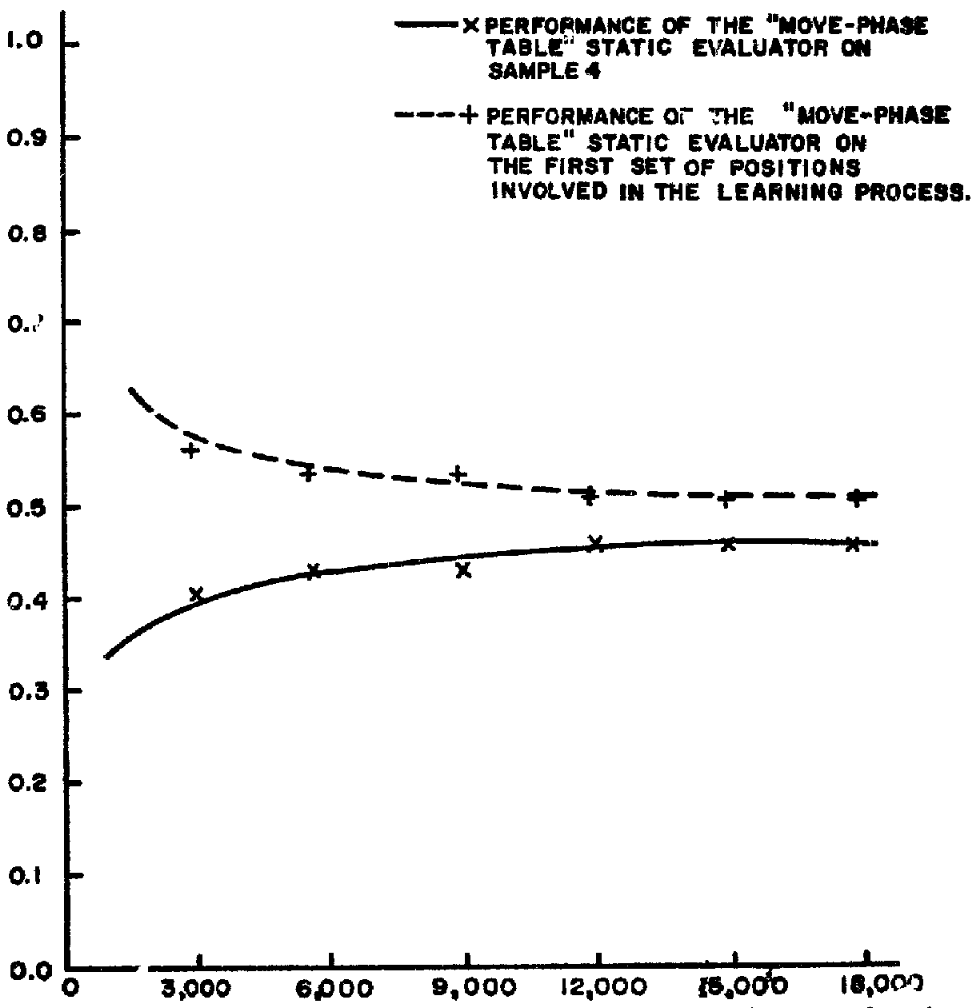

FIG. 2. Performance index C for the "Move-Phase Table" procedure as a function of the number of positions involved in learning.

[8] Along the main diagonal starting at the lower left corner.

the procedure on the first two thousand or so positions upon which learning took place. This curve starts out at a relatively high value, again suggesting that this procedure, when learning on some small sample, essentially rote memorizes it. Again, the value of C represented by this curve gives an approximate upper bound on the possible value of C for this procedure.

From Fig. 2 it may be seen that on Sample 4 the value of the index C is at least 0.49. On this sample the index D was computed as well, and its value was 0.35.

8. A Heuristic Static Evaluator

The checker board position static evaluators involving the learning procedures just described were compared with a static evaluator using only four simple heuristics. These heuristics are:

(1) To give highest priority to moving a king.

(2) To give second highest priority to moves along the main diagonal and into the central two squares.

(3) To give lowest priority to moves leading to jumps and moves out of the third from the left and second from the right squares of the first row.

(4) To give third priority to all other types of moves.

The performance indices C and D corresponding to this static evaluator may be arrived at as follows: A set of positions resulting from making all possible legal moves from the preceding position are given scores according to the priorities assigned to the moves leading to the positions. Positions resulting from moves of highest priority are given scores randomly chosen from the range 0.9 to 1.0; those resulting from moves of second priority are assigned randomly chosen scores in the range 0.8 to 0.9; and so forth.[9]

The C and D indices were thus computed for Sample 4, mentioned in Section 7, and were 0.39 and 0.29, respectively.

9. Summary and Discussion

We have presented two new machine learning techniques by which a computer has learned to distinguish strong checker positions from weak ones. These have been shown to be significantly superior, by two different numerical criteria, to an older, more traditional machine learning approach. In addition, we have presented a purely heuristic static evaluator whose performance is about half way between the polynomial and the other two. The performance indices for all evaluators are summarized in Figs. 3 and 4.

A major purpose of introducing the heuristic evaluator is that it provides a measure of the absolute, as opposed to relative, level of performance

[9] The randomization is desirable in computing the C index since its value is biased if positions are assigned equal scores.

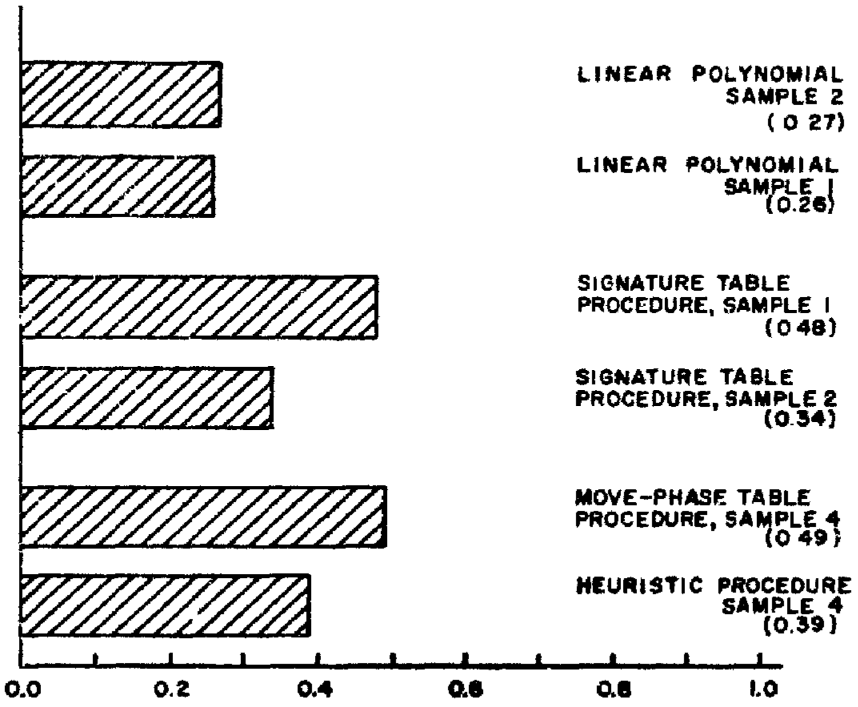

FIG. 3. *C* indices for various static evaluators.

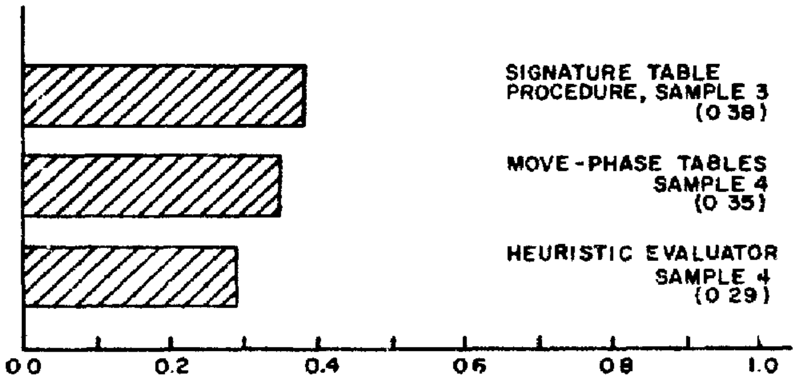

FIG. 4. *D* indices for various static evaluators.

represented by values of the two numerical indices.[10] The evaluator is patently exactly this intelligent: it knows that it is very important to keep kings in motion, that it is important to move into the center, into the kingrow and along the main diagonal, that it is bad to make a move leading to a capture, or to evacuate two of the four squares on the first row. We know that this much "knowledge" is equivalent to a *C* index of 0.39, and to a *D*

[10] Heuristic programs are studied primarily as models, and hence theories, of intelligence. In the present case a heuristic program is used not merely as a model of intelligence, but as a measure of the absolute quantity of knowledge required to perform a particular task at a particular level of proficiency.

index of 0.29. Conversely, we know that a good set of weights in the linear polynomial represents significantly less knowledge than this and that the signature tables and move-phase tables contain significantly more.

ACKNOWLEDGMENTS

The author would like to acknowledge financial support for the present work, which was supplied by parts of the following: An I.B.M. research assistantship and a Project MAC research assistantship while the author was a graduate student at M.I.T., and a D.S.R. staff appointment which the author held at the Artificial Intelligence Laboratory subsequent to completing graduate studies. In addition, the author would like to thank Professors Minsky and Papert for guidance, and Information International Inc., for providing the facilities for the production of this paper. A particular debt of gratitude is owed to Arthur Samuel for supervising the development of the Signature Table procedure, and for making available certain of his checker playing and learning programs and book game data.

REFERENCES

1. Friedberg, R. A learning machine, part I. *IBM J. Res. Develop.* **2** (1) (January 1958), 2–13.
2. Friedberg, R., Dunham, R., and North, J. A learning machine, part II. *IBM J. Res. Develop.* 3 (3) (July 1959), 282–287.
3. Griffith, A. A new machine learning technique applied to the game of checkers. Massachusetts Inst. of Technol. Project MAC Artificial Intelligence Memo 94 (March 1966).
4. Lees, J. *Lees' Guide to the Game of Draughts or Checkers.* McKay, Philadelphia, Pa. (1931).
5. Michie, D., and Ross, R. Experiments with the adaptive graph traverser. *Machine Intelligence* **5**, American Elsevier, New York (1970), 301–318.
6. Minsky, M., and Papert, S. *Perceptrons.* M.I.T. Press, Cambridge, Mass. (1969).
7. Nilsson, N. *Learning Machines.* McGraw-Hill, New York (1965).
8. Rosenblatt, F. *Principles of Neurodynamics.* Spartan Books, New York (1962).
9. Samuel, A. Some studies in machine learning using the game of checkers. *IBM J. Res. Develop.* 3 (3) (July 1959), 211–229.
10. Samuel, A. Some studies in machine learning using the game of checkers, II—recent progress. *IBM J. Res. Develop.* **11** (6) (November 1967), 601–617.
11. Selfridge, O. Pandemonium: a paradigm for learning. *Proc. Symp. on Mechanization of Thought Processes,* National Physical Laboratory, H.M. Stationery Office, London (1959), 511–529.
12. Skinner, B. F. *The Behavior of Organisms. An Experimental Analysis.* Appleton-Century Crofts, New York (1938).
13. Slagle, J., and Farrell, C. Experiments in automatic learning for a multipurpose heuristic program. *Commun. ACM* **14** (2) (February 1971), 91–99.
14. Smith, M. H. A learning program which plays partnership dominoes. *Commun. ACM* **16** (8) (August 1973), 462–467.
15. Uhr, L., and Jordan, S. The learning of parameters for generating compound characterizers for pattern recognition. *Proc. Intern. Joint Conf. on Artificial Intelligence,* D. Walker and L. Norton (eds.), Washington, D.C. (May 7–9, 1969).
16. Winston, P. Learning structural descriptions from examples. Massachusetts Inst. of Technol. Project MAC Techn. Rept. MAC-TR-76 (September 1970).

Received 13 November 1973; accepted 13 February 1974