Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress

A. L. SAMUEL

ABSTRACT.—A new signature-table technique is described together with an improved book-learning procedure which is thought to be much superior to the linear polynomial method. Full use is made of the so-called "alpha-beta" pruning and several forms of forward pruning to restrict the spread of the move tree and to permit the program to look ahead to a much greater depth than it otherwise could do. While still unable to outplay checker masters, the program's playing ability has been greatly improved.

Introduction

Limited progress has been made in the development of an improved book-learning technique and in the optimization of playing strategies as applied to the checker playing program described in an earlier paper with this same title.† Because of the sharpening in our understanding and the substantial improvements in playing ability that have resulted from these recent studies, a reporting at this time seems desirable. Unfortunately, the most basic limitation of the known machine learning techniques, as previously outlined, has not yet been overcome, nor has the program been able to outplay the best human checker players.‡

We will briefly review the earlier work. The reader who does not find this review adequate might do well to refresh his memory by referring to the earlier paper.

Two machine learning procedures were described in some detail: (1) a rote learning procedure in which a record was kept of the board

† Some Studies in Machine Learning Using the Game of Checkers, *IBM Journal* 3, 211–299 (1959). Reprinted (with minor additions and corrections) in *Computers and Thought*, edited by Feigenbaum and Feldman, McGraw-Hill, 1963.

‡ In a 1965 match with the program, the World Champion, Mr. W. F. Hellman, won all four games played by mail but was played to a draw in one hurriedly played cross-board game. Recently Mr. K. D. Hanson, the Pacific Coast Champion, has beaten current versions of the program on two separate occasions.

situation encountered in actual play together with information as to the results of the machine analyses of the situation; this record could be referenced at terminating board situations of each newly initiated tree search and thus, in effect, allow the machine to look ahead further than time would otherwise permit, and (2) a generalization learning procedure in which the program continuously re-evaluated the coefficients for the linear polynomial used to evaluate the board positions at the terminating board situations of a look-ahead tree search. In both cases, the program applied a mini-max procedure to back up scores assigned to the terminating situations and so select the best move, on the assumption that the opponent would also apply the same selection rules when it was his turn to play. The rote learning procedure was characterized by a very slow but continuous learning rate. It was most effective in the opening and end-game phases of the play. The generalization learning procedure, by way of contrast, learned at a more rapid rate but soon approached a plateau set by limitations as to the adequacy of the man-generated list of parameters used in the evaluation polynomial. It was surprisingly good at mid-game play but fared badly in the opening and end-game phases. Both learning procedures were used in cross-board play against human players and in self-play, and in spite of the absence of absolute standards were able to improve the play, thus demonstrating the usefulness of the techniques discussed.

Certain expressions were introduced which we will find useful. These are: *Ply*, defined as the number of moves ahead, where a ply of two consists of one proposed move by the machine and one anticipated reply by the opponent; *board parameter value*,† defined as the numerical value associated with some measured property or parameter of a board situation. Parameter values, when multiplied by learned coefficients, become *terms* in the learning polynomial. The value of the entire polynomial is a *score*.

The most glaring defects of the program, as earlier discussed, were (1) the absence of an effective machine procedure for generating new parameters for the evaluation procedure, (2) the incorrectness of the assumption of linearity which underlies the use of a linear polynomial, (3) the general slowness of the learning procedure, (4) the inadequacies of the heuristic procedures used to prune and to terminate the tree search, and (5) the absence of any strategy considerations for altering the machine mode of play in the light of the tactical situations as they develop during play. While no progress has been made with respect to the first of these defects, some progress has been made in overcoming the other four limitations, as will now be described.

† Example of a *board parameter* is MOB (total mobility): the number of squares to which the player can potentially move, disregarding forced jumps that might be available. Other parameters are described in *IBM Journal* 3, 211–229 (1959).

We will restrict the discussion in this paper to generalization learning schemes in which a preassigned list of board parameters is used. Many attempts have been made to improve this list, to make it both more precise and more inclusive. It still remains a man-generated list and it is subject to all the human failings, both of the programmer, who is not a very good checker player, and of the checker experts consulted, who are good players (the best in the world, in fact) but who, in general, are quite unable to express their immense knowledge of the game in words, and certainly not in words understandable to this programmer. At the present time, some twenty-seven parameters are in use, selected from the list given (*IBM Journal* 3, 211–229 (1959)) with a few additions and modifications, although a somewhat longer list was used for some of the experiments which will be described.

Two methods of combining evaluations of these parameters have been studied in considerable detail. The first, as earlier described, is the linear polynomial method in which the values for the individual parameters are multiplied by coefficients determined through the learning process and added together to obtain a score. A second, more recent procedure is to use tabulations called "signature tables" to express the observed relationship between parameters in subsets. Values read from the tables for a number of subsets are then combined for the final evaluation. We will have more to say on evaluation procedures after a digression on other matters.

The Heuristic Search for Heuristics

At the risk of some repetition, and of sounding pedantic, it might be well to say a bit about the problem of immensity as related to the game of checkers. As pointed out in the earlier paper, checkers is not deterministic in the practical sense since there exists no known algorithm which will predict the best move short of the complete exploration of every acceptable† path to the end of the game. Lacking time for such a search, we must depend upon heuristic procedures.

Attempts to see how people deal with games such as checkers or chess‡ reveal that the better players engage in behavior that seems extremely complex, even a bit irrational in that they jump from one aspect to another, without seeming to complete any one line of reasoning. In fact, from the writer's limited observation of checker players he is convinced that the better the player, the more apparent confusion

† The word "acceptable" rather than "possible" is used advisedly for reasons which relate to the so-called alpha-beta heuristic, as will be described later.

‡ See, for example, Newell, Shaw and Simon, Chess Playing Programs and the Problem of Complexity, *IBM Journal* 2, 320–335 (1958). For references to other games, see A. L. Samuel, Programming a Computer to Play Games, in *Advances in Computers*, F. Alt, Ed., Academic Press, Inc., New York, 1960.

there exists in his approach to the problem, and the more intuitive his reactions seem to be, at least as viewed by the average person not blessed with a similar proficiency. We conclude† that at our present stage of knowledge, the only practical approach, even with the help of the digital computer, will be through the development of heuristics which tend to ape human behavior. Using a computer, these heuristics will, of course, be weighted in the direction of placing greater reliance on speed than might be the case for a human player, but we assume that the complexity of the human response is dictated by the complexity of the task to be performed and is, in some way, an indication of how such problems can best be handled.

We will go a step further and maintain that the task of making decisions as to the heuristics to be used is also a problem which can only be attacked by heuristic procedures, since it is essentially an even more complicated task than is the playing itself. Furthermore, we will seldom, if ever, be able to perform a simple test to determine the effectiveness of any particular heuristic, keeping everything else the same, as any scientist generally tends to do. There are simply too many heuristics that should be tested and there is simply not enough time to embark on such a program, even if the cost of computer time were no object. But, more importantly, the heuristics to be tested are not independent of each other and they affect the other parameters which we would like to hold constant. A definitive set of experiments is virtually impossible of attainment. We are forced to make compromises, to make complicated changes in the program, varying many parameters at the same time and then, on the basis of incomplete tests, somehow conclude that our changes are or are not in the right direction.

Playing Technique

While the investigation of the learning procedures forms the essential core of the experimental work, certain improvements have been made in playing techniques which must first be described. These improvements are largely concerned with the tree searching. They involve schemes to increase the effectiveness of the alpha-beta pruning, the so-called "alpha-beta heuristic"‡ and a variety of other techniques going

† More precisely we adopt the heuristic procedure of assuming that we must so conclude.

‡ So named by Prof. John McCarthy. This procedure was extensively investigated by Prof. McCarthy and his students at M.I.T. but it has been inadequately described in the literature. It is, of course, not a heuristic at all, being a simple algorithmic procedure and actually only a special case of the more general "branch and bound" technique which has been rediscovered many times and which is currently being exploited in integer programming research. See A. H. Land and A. G. Doight, An Automatic Method of Solving Discrete Programming Problems (1957)

under the generic name of tree pruning.† These improvements enable the program to analyze further in depth than it otherwise could do, albeit with the introduction of certain hazards which will be discussed. Lacking an ideal board evaluation scheme, tree searching still occupies a central role in the checker program.

ALPHA-BETA PRUNING

Alpha-beta pruning can be explained simply as a technique for not exploring those branches of a search tree that the analysis up to any given point indicates not to be of further interest either to the player making the analysis (this is obvious) or to his opponent (and it is this that is frequently overlooked). In effect, there are always two scores, an *alpha value* which must be exceeded for a board to be considered desirable by the side about to play, and a *beta value* which must not be exceeded for the move leading to the board to have been made by the opponent. We note that if the board should not be acceptable to the side about to play, this player will usually be able to deny his opponent the opportunity of making the move leading to this board, by himself making a different earlier move. While people use this technique more or less instinctively during their look-ahead analyses, they sometimes do not understand the full implications of the principle. The saving in the required amount of tree searching which can be achieved through its use is extremely large, and as a consequence alpha-beta pruning is an almost essential ingredient in any game playing program. There are no hazards associated with this form of pruning.

A move tree of the type that results when alpha-beta pruning is effective is shown in Fig. 1, it being assumed that the moves are investigated from left to right. Those paths that are shown in dashed lines need never be considered, as can be verified by assigning any arbitrary scores to the terminals of the dashed paths and by mini-maxing in the usual way. Admittedly the example chosen is quite special but it does illustrate the possible savings that can result. To realize the maximum saving in computational effort as shown in this example one must investigate the moves in an ideal order, this being the order which would result were each side to always consider its best possible move

reported in bibliography *Linear Programming and Extensions*, George Dantzig, Princeton University Press, 1963; M. J. Rossman and R. J. Twery, Combinatorial Programming, abstract K7, *Operations Research* 6, 634 (1958); John D. Little, Katta P. Murty, Dura W. Sweeney and Caroline Karel, An Algorithm for the Traveling Salesman Problem, *Operations Research* 11, 972–989 (1963).

† It is interesting to speculate on the fact that human learning is involved in making improvements in the tree pruning techniques. It would be nice if we could assign this learning task to the computer but no practical way of doing this has yet been devised.

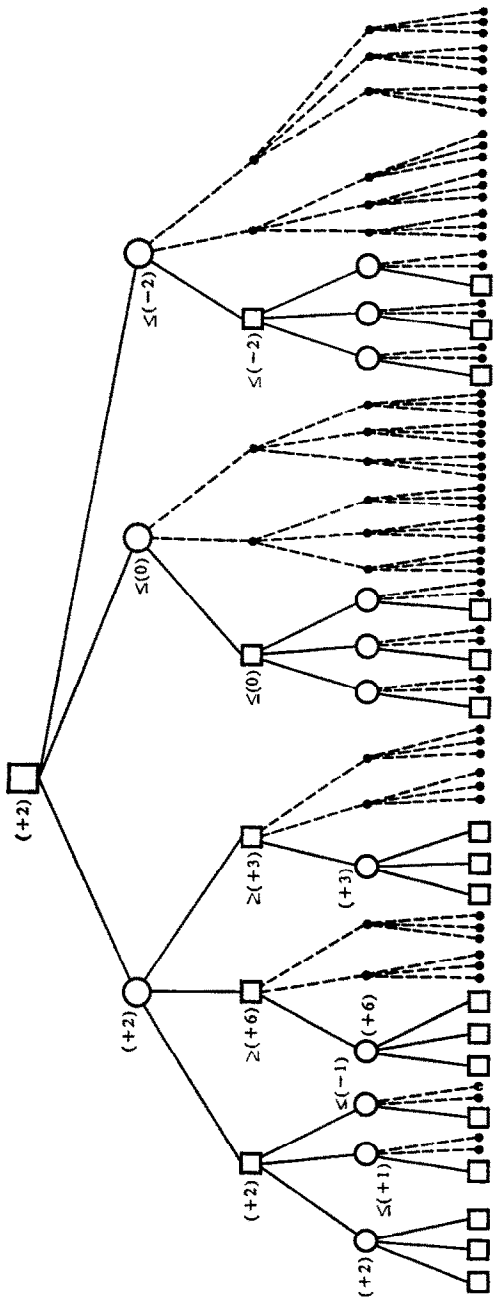

FIG. 1. A (look-ahead) move tree in which alpha-beta pruning is fully effective if the tree is explored from left to right. Board positions for a look-ahead move by the first player are shown by squares, while board positions for the second player are shown by circles. The branches shown by dashed lines can be left unexplored without in any way influencing the final move choice.

first. A great deal of thought and effort has gone into devising techniques which increase the probability that the moves will be investigated in something approaching this order.

The way in which two limiting values (McCarthy's alpha and beta) are used in pruning can be seen by referring to Fig. 2, where the tree of Fig. 1 has been redrawn with the uninvestigated branches deleted. For reasons of symmetry all boards during the look-ahead are scored as viewed by the side whose turn it then is to move. This means that mini-maxing is actually done by changing the sign of a score, once for each ply on backing up the tree, and then always maximizing. Furthermore, only one set of values (alpha values) need be considered. Alpha values are assigned to all boards in the tree (except for the terminating boards) as these boards are generated. These values reflect the score which must be exceeded before the branch leading to this board will be entered by the player whose turn it is to play. When the look-ahead is terminated and the terminal board evaluated (say at board *e* in Fig. 2), then the value which currently is assigned the board two levels up the tree (in this case at board *c*) is used as the alpha value, and unless the terminal board score exceeds this alpha value, the player at board *c* would be ill advised to consider entering the branch leading to this terminal board. Similarly if the negative of the terminal board score does not exceed the alpha value associated with the board immediately above in the tree (in this case at board *d*), then the player at board *d* will not consider this to be a desirable move. An alternative way of stating this second condition, in keeping with McCarthy's usage, is to say that the negative of the alpha value associated with the board one level up the tree (in this case board *d*) is the beta value which must not be exceeded by the score associated with the board in question (in this case board *e*). A single set of alpha values assigned to the boards in the tree thus performs a dual role, that of McCarthy's alpha as referenced by boards two levels down in the tree and, when negated, that of McCarthy's beta as referenced by boards one level down in the tree.

Returning to the analysis of Fig. 2, we note that during the initial look-ahead (leading to board *e*) nothing is known as to the value of the boards, consequently the assigned alpha values are all set at minus infinity (actually within the computer only at a very large negative number). When board *e* is evaluated, its score (+2) is compared with the alpha at *c* ($-\infty$), and found to be larger. The negative of the score (−2) is then compared with the alpha at *d* ($-\infty$) and, being larger, it is used to replace it. The alpha at *d* is now −2 and it is unaffected by the subsequent consideration of terminal boards *f* and *g*. When all paths from board *d* have been considered, the final alpha value at *d* is compared with the current alpha value at board *b* ($-\infty$); it is larger, so the negative of alpha at *d* (now +2) is compared with the current

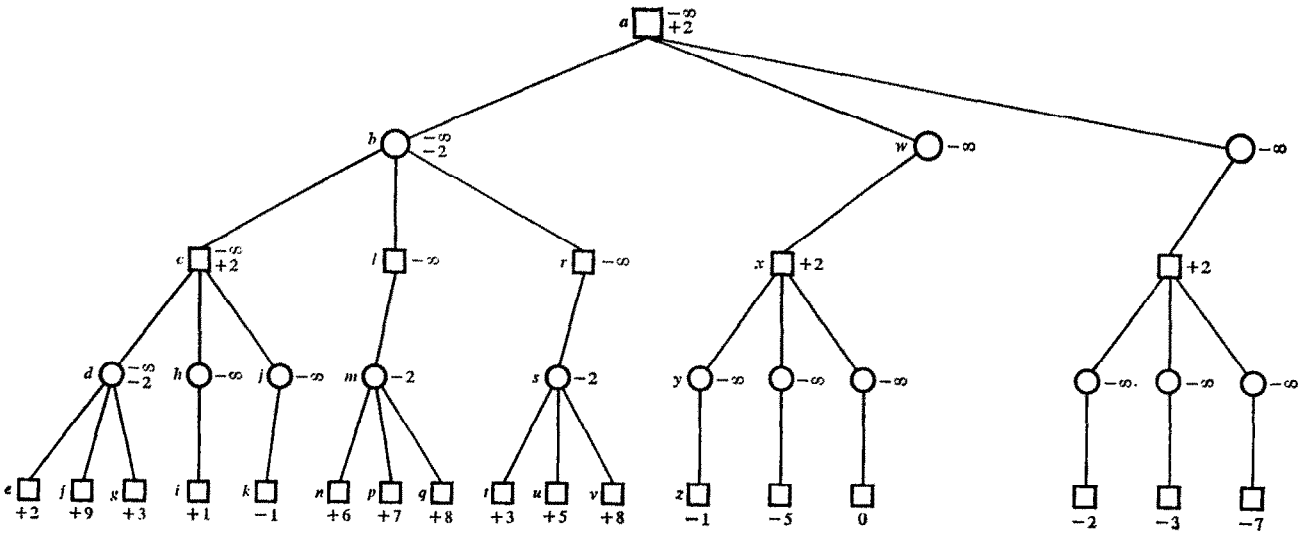

FIG. 2. The move tree of Fig. 1 redrawn to illustrate the detailed method used to keep track of the comparison values. Board positions are lettered in the order that they are investigated and the numbers are the successive alpha values that are assigned to the boards as the investigation proceeds.

alpha value at c ($-\infty$) and, being larger, it is used to replace the c value, and a new move from board c is investigated leading to board h and then board i. As we go down the tree we must assign an alpha value to board h. We cannot use the alpha value at board c since we are now interested in the minimum that the other side will accept. We can, however, advance the alpha value from board b, which in this case is still at its initial value of $-\infty$. Now when board i is evaluated at $+1$ this value is compared with the alpha at board c ($+2$). The comparison being unfavorable, it is quite unnecessary to consider any other moves originating at board h and we go immediately to a consideration of boards j and k, where a similar situation exists. This process is simply repeated throughout the tree. On going forward the alpha values are advanced each time from two levels above and, on backing up, two comparisons are always made. When the tree is completely explored, the final alpha value on the initial board is the score, and the correct move is along the path from which this alpha was derived.

The saving that results from alpha-beta pruning can be expressed either as a reduction in the apparent amount of branching at each node or as an increase in the maximum ply to which the search may be extended in a fixed time interval. With optimum ordering, the apparent branching factor is reduced very nearly to the square root of its original value or, to put it another way, for a given investment in computer time, the maximum ply is very nearly doubled. With moderately complex trees the savings can be astronomical. For example, consider a situation with a branching factor of 8. With ideal alpha-beta pruning this factor is reduced to approximately 2.83. If time permits the evaluation of 66,000 boards (about 5 minutes for checkers), one can look ahead approximately 10 ply with alpha-beta pruning. Without alpha-beta this depth would require the evaluation of 8^{10} or approximately 10^9 board positions and would require over 1,000 hours of computation! Such savings are of course dependent upon perfect ordering of the moves. Actual savings are not as great but alpha-beta pruning can easily reduce the work by factors of a thousand or more in real game situations.

Some improvement results from the use of alpha-beta pruning even without any attempt to optimize the search order. However, the number of branches which are pruned is then highly variable depending upon the accidental ordering of the moves. The problem is further complicated in the case of checkers because of the variable nature of the branching. Using alpha-beta alone the apparent branching factor is reduced from something in the vicinity of 6 (reduced from the value of 8 used above because of forced jump moves) to about 4, and with the best selection of ordering practiced to date, the apparent branching is reduced to 2.6. This leads to a very substantial increase in the depth to which the search can be carried.

Although the principal use of the alpha and beta values is to prune useless branches from the move tree, one can also avoid a certain amount of inconsequential work whenever the difference between the current alpha value and the current beta value becomes small. This means that the two sides have nearly agreed as to the optimum score and that little advantage to either one side or the other can be found by further exploration along the paths under investigation. It is therefor possible to back-up along the tree until a part of the tree is found at which this alpha-beta margin is no longer small. Not finding such a situation, one may terminate the search. The added savings achieved in this way, while not as spectacular as the savings from the initial use of alpha-beta, are quite significant, frequently reducing the work by an additional factor of 2 or more.

Plausibility analysis

In order for the alpha-beta pruning to be truly effective, it is necessary, as already mentioned, to introduce some technique for increasing the probability that the better paths are explored first. Several ways of doing this have been tried. By far the most useful seems to be to conduct a preliminary plausibility survey for any given board situation by looking ahead a fixed amount, and then to list the available moves in their apparent order of goodness on the basis of this information and to specify this as the order to be followed in the subsequent analysis. A compromise is required as to the depth to which this plausibility survey is to be conducted; too short a look-ahead renders it of doubtful value, while too long a look-ahead takes so much time that the depth of the final analysis must be curtailed. There is also a question as to whether or not this plausibility analysis should be applied at all ply levels during the main look-ahead or only for the first few levels. At one time the program used a plausibility survey for only the first two ply levels of the main look-ahead with the plausibility analysis itself being carried to a minimum ply of 2. More recently the plausibility analysis has been applied at all stages during the main look-ahead and it has been carried to a minimum ply of 3 during certain portions of the look-ahead and under certain conditions, as will be explained later.

We pause to note that the alpha-beta pruning as described might be called a backward pruning technique in that it enables branches to be pruned at that time when the program is ready to back up and is making mini-max comparisons. It assumes that the analyses of all branches are otherwise carried to a fixed ply and that all board evaluations are made at this fixed ply level. As mentioned earlier, the rigorous application of alpha-beta technique introduces no opportunities for erroneous pruning. The results in terms of the final moves chosen are

always exactly as they would have been without the pruning. To this extent the procedure is not a heuristic although the plausibility analysis technique which makes it effective is certainly a heuristic.

While the simple use of the plausibility analysis has been found to be quite effective in increasing the amount of alpha-beta pruning, it suffers from two defects. In the first place the actual amount of pruning varies greatly from move to move, depending upon random variations in the average correctness of the plausibility predictions. Secondly, within even the best move trees a wrong prediction at any one point in the search tree causes the program to follow a less than optimum path, even when it should have been possible to detect the fact that a poor prediction had been made before doing an excessive amount of useless work.

A MULTIPLE-PATH ENHANCED-PLAUSIBILITY PROCEDURE

In studying procedures used by the better checker players one is struck with the fact that evaluations are being made continuously at all levels of look-ahead. Sometimes unpromising lines of play are discarded completely after only a cursory examination. More often less promising lines are put aside briefly and several competing lines of play may be under study simultaneously with attention switching from one to another as the relative goodness of the lines of play appears to change with increasing depth of the tree search. This action is undoubtedly prompted by a desire to improve the alpha-beta pruning effectiveness, although I have yet to find a checker master who explains it in these terms. We are well advised to copy this behavior.

Fortunately, the plausibility analysis provides the necessary information for making the desired comparisons at a fairly modest increase in data storage requirements and with a relatively small amount of reprogramming of the tree search. The procedure used is as follows. At the beginning of each move, all possible moves are considered and a plausibility search is made for the opponent's replies to each of these plays. These moves are sorted in their apparent order of goodness. Each branch is then carried to a ply of 3—that is, making the machine's first move, the opponent's first reply and the machine's counter move. In each case the moves made are based on a plausibility analysis which is also carried to a minimum depth of 3 ply. The path yielding the highest score to the machine at this level is then chosen for investigation and followed forward for two moves only (that is, making the opponent's indicated best reply and the machine's best counter reply, always based on a plausibility analysis). At this point the score found for this path is compared with the score for the second best path as saved earlier. If the path under investigation is now found to be less

good than an alternative path, it is stored and the alternative path is picked up and is extended in depth by two moves. A new comparison is made and the process is repeated. Alternately, if the original path under investigation is still found to be the best it is continued for two more moves. The analysis continues in this way until a limiting depth as set by other considerations has been reached. At this point the flitting from path to path is discontinued and the normal mini-maxing procedure is instituted. Hopefully, however, the probability of having found the optimum path has been increased by this procedure and the alpha-beta pruning should work with greater effectiveness. The net effect of all of this is to increase the amount of alpha-beta pruning, to decrease the playing time, and to decrease the spread in playing time from move to move.

This enhanced plausibility analysis does not in any way affect the hazard-free nature of the alpha-beta pruning. The plausibility scores used during the look-ahead procedure are used only to determine the order of the analyses and they are all replaced by properly mini-maxed scores as the analysis proceeds.

One minor point may require explanation. In order for all of the saved scores to be directly comparable they are all related to the same side (actually to the machine's side) and as described they are compared only when it is the opponent's turn to move—that is, comparisons are made only on every alternate play. It would, in principle, be possible to make comparisons after every move, but little is gained by so doing and serious complications arise which are thought to offset any possible advantage.

A move tree as recorded by the computer during actual play is shown in Fig. 3. This is simply a listing of the moves, in the order in which they were considered, but arranged on the page to reveal the tree structure. Asterisks are used to indicate alternative moves at branch points and the principal branches are identified by serial numbers. In the interest of clarity, the moves made during each individual plausibility search are not shown, but one such search was associated with each recorded move. While the tree of Fig. 3 exhibits the combined effect of several forms of pruning, some yet to be explained, the flitting from path to path is clearly visible at the start. In this case there were 9 possible initial moves which were surveyed at the start and listed in the initially expected best order as identified by the serial numbers. Each of these branches was carried to a depth of 3 ply and the apparent best branch was then found to be the one identified by serial number 9, as may be verified by reference to the scores at the far right (which are expressed in terms of the side which made the last recorded move on the line in question). Branch 9 was then investigated for four more moves, only to be put aside for an investigation of the branch identified by the serial

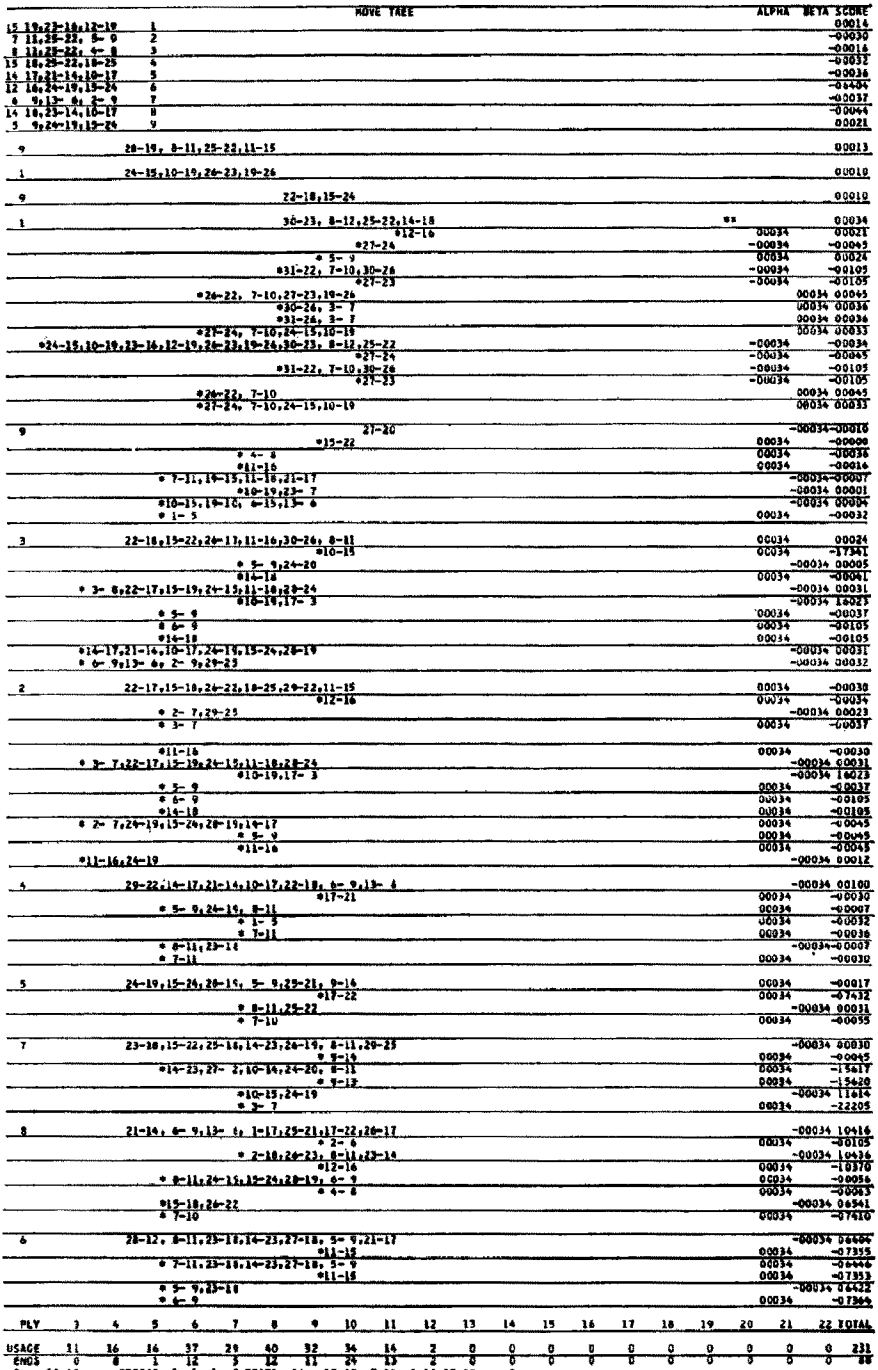

FIG. 3. An actual look-ahead move tree as printed by the computer during play.

number 1 which in turn was displaced by 9, then finally back to 1. At this point the normal mini-maxing was initiated. The amount of flitting from move to move is, of course, critically dependent upon the exact board configuration being studied. A fairly simple situation is portrayed by this illustration. It will be noted that on the completion of the investigation of branch 1, the program went back to branch 9, then to branch 3, followed by branch 2, and so on until all branches were investigated. As a matter of general interest this tree is for the fifth move of a game following a 9–14, 22–17, 11–15 opening, after an opponent's move of 17–13, and move 15–19 (branch 1) was finally chosen. The 7094 computer took 1 minute and 3 seconds to make the move and to record the tree. This game was one of a set of 4 games being played simultaneously by the machine and the length of the tree search had been arbitrarily reduced to speed up the play. The alpha and beta values listed in the columns to the right are both expressed in terms of the side making the last move, and hence a score to be considered must be larger than alpha and smaller than beta. For clarity of presentation deletions have been made of most large negative values when they should appear in the alpha column and of most large positive values when such values should appear in the beta column.

Forward pruning

In addition to the hazardless alpha-beta pruning, as just described, there exist several forms of forward pruning which can be used to reduce the size of the search tree. There is always a risk associated with forward pruning since there can be no absolute assurance that the scores that would be obtained by a deeper analysis might not be quite different from those computed at the earlier ply. Indeed, if this were not so, there would never be any reasons for looking ahead. Still, it seems reasonable to assume that some net improvement should result from the judicious use of these procedures. Two simple forms of forward pruning were found to be useful after a variety of more complicated procedures, based on an initial imperfect understanding of the problem, had been tried with great effort and little success.

To apply the first form it is only necessary to limit the number of moves saved for future analysis at each point in the tree, with provisions for saving all moves when the ply is small and gradually restricting the number saved as the ply becomes greater until finally when the maximum feasible ply is being approached only two or three moves are saved. (The decision as to which are saved is, of course, based on the plausibility analysis.)

In the second form of forward pruning one compares the apparent scores as measured by the plausibility analysis with the current values

of alpha and beta that are being carried forward, and terminates the look-ahead if this comparison is unfavorable. Rather than to apply this comparison in an unvarying way it seems reasonable to set margins which vary with the ply so that the amount of pruning increases with increasing ply. At low plies only the most unlikely paths can then be pruned, while fairly severe pruning can be caused to occur as the effective ply limit is approached. If the margins are set too high, then only negligible pruning will result, while if they are low or nonexistent, the pruning will be extreme and the risks of unwise pruning correspondingly large.

There are, then, several factors which may be experimentally studied, these being the magnitudes of the several forms of pruning and the way in which these magnitudes are caused to vary with the ply. The problem is even more complicated than it might at first appear since the various kinds of forward pruning are not independent. It seems reasonable to assume that the rate at which the margins are reduced in the last described form of forward pruning and the rate at which the number pruning is increased in the earlier described form should both depend upon the position in the plausibility listings of earlier boards along the branch under investigation. It is quite impractical to make a detailed study of these interdependencies because the range of possible combinations is extremely large and a whole series of games would have to be played for each combination before valid conclusions could be drawn. Only a very few arrangements have, in fact, been tried and the final scheme adopted is based more on the apparent reasonableness of the arrangement than upon any real data.

The problem of "pitch" moves

In both of the above forms of forward pruning serious difficulties arise with respect to the proper consideration of so-called "pitch moves", that is, of moves in which a piece is sacrificed in return for a positional advantage which eventually leads at least to an equalizing capture, if not to an actual winning position. In principle, one should be able to assign the proper relative weights to positional and material advantages so as to assess such moves correctly, but these situations generally appear to be so detail-specific that it is impossible to evaluate them directly in any way other than by look-ahead. Troubles are encountered because of the limited look-ahead distance to which the plausibility analysis can be extended; the equalizing moves may not be found and as a consequence a good pitch move may be pruned. A two-ply plausibility search in which the analysis is terminated only on a non-jump situation will correctly evaluate move sequences of the type P, J, J, where P stands for pitch and J for jump (with N used later for non-jump

moves which are not forcing), but it is powerless to evaluate sequences of the P, J, P, J, J type or of the P, J, N, P, J type. Both of these occur quite frequently in normal play. A three-ply search will handle the first of these situations but will still not handle the second case. Unsatisfactory as it is, the best practical compromise which has been achieved to date seems to be to employ a two-ply plausibility search for the normal non-jump situation and to extend the search to three-ply whenever the first or the second move of the plausibility search is a jump. As noted earlier, a three-ply search is customarily employed during the preliminary multi-path phase of the analysis.

Several more complicated methods of handling this problem have been considered, but all of the methods tried to date have proved to be very expensive in terms of computing time and all have been discarded. One of these methods which seemed to be marginally effective consisted of a procedure for keeping a separate account of all pitch moves encountered during the plausibility search, defined in this case as sequences in which the first move in the search *is not* a jump and the second move *is* a jump. These pitch moves were sorted on the basis of their relative scores and a record was kept of the four best pitch moves. Of course, some of these moves might have been also rated as good moves quite independently of their pitch status, either because most or all of the available moves were of this type or because the return capture was not delayed beyond the ply depth of the search. After the normal number of unpruned moves at any branch point had been explored, the best remaining pitch move (eliminating any already considered) was then followed up. Since most of the apparent pitch moves may in fact be sheer giveaway moves, it was quite impractical to consider more than a single pitch move, but hopefully that apparent pitch which led to the highest positional score should have been the most likely move to investigate. This procedure causes a two-ply plausibility search to salvage one likely candidate per move which could be of the P, J, N, J, J type and it increases the power of the three-ply plausibility search correspondingly. Unfortunately a rather high percentage of the additional moves so considered were found to be of no value and the bookkeeping costs of this procedure also seemed to be excessive.

As a further extension of this general method of handling pitch moves, it is possible to cause pitch sequences of the P, J, N, P, J type to be investigated using a two-ply plausibility search. One need only specify that the main tree should not be terminated when there is a jump move pending. While the cost of this addition might seem to be small, in practice it leads to the exploration in depth of extended giveaway sequences, and as a consequence it is of very questionable value.

LOOK-AHEAD TERMINATION

Regardless of the form or amount of forward pruning the time arrives along each path when it is necessary to terminate the look-ahead and evaluate the last board position. It is rather instructive to consider the termination as simply the end of the pruning process in which the pruning is complete. The use of a fixed depth for this final act of pruning, as previously assumed, is, of course, not at all reasonable and in fact it has never been used. In the earlier work much attention was given to the wisdom of terminating the look-ahead at so-called "dead" positions. With the current use made of the plausibility analysis this becomes a restriction mainly applicable to the plausibility analysis and it is of but little value in terminating the main tree itself. A limit is, of course, set by the amount of storage assigned for the tree, but since the tree storage requirements are not excessive this should normally not be allowed to operate. If the plausibility analysis is at all effective one should be able to ration the computing time to various branches on the basis of their relative probability of being the best. For example, the initial path which survives the swapping routine during the initial look-ahead procedure should certainly be carried quite far along as compared with a path resulting from investigating, say, the fourth choice as found by the plausibility analysis when this is again followed by a fourth choice, etc., all the way through the tree.

The procedure found most effective has been that of defining a parameter called the *branching count* which is assigned a value for each board encountered during the tree search. To insure that all of the possible initial moves are given adequate consideration, identical values are given to the counts for the resulting boards after these initial moves. As each move originating with one of these boards is made, the branching count for the originating board is reduced by one unit and the resulting board after the move is assigned this new value as well. This process is repeated at each branch point down the tree until the branching count reaches zero, whereupon the search down this path is terminated (more correctly steps are taken to initiate termination unless other factors call for a further extension of the search, as will be explained later). Along the preferred branch, the branching count will thus be reduced by one unit for each ply level. For the second choice at any branch point a two-unit reduction occurs, for the third choice a three-unit, etc. The net result is that the less likely paths are terminated sooner than the most likely paths and in direct proportion to their decreasing likelihood.

Actually, a slightly more complicated procedure is used in that the branching count is set at a higher initial value and it is reduced by one unit when the move under consideration is a jump move and by four

units when it is a normal move. This procedure causes the search to be extended further along those paths involving piece exchanges than along those that do not. Also the search is not permitted to terminate automatically when the branching count reaches zero if the indicated score for the move under consideration implies that this is in fact a preferred path. In this case the search is extended until the same depth has been reached along this path as had been reached along the previously indicated preferred path.

Tree pruning results

It has been found singularly difficult to assess the relative value of the various tree pruning techniques in terms of their effect on the goodness of play. Special situations can always be found for which the various forward pruning procedures are either very effective or quite inadequate. Short of very extensive tests indeed, there seems to be no very good way to determine the relative frequency with which these different situations occur during normal play. About all that has been done has been to observe the resulting game trees and to depend upon the opinions of checker masters as to the goodness of the resulting moves and as to the reasonableness in appearance of the trees.

As mentioned earlier, for each move that is tabulated in Fig. 3 there was actually an auxiliary plausibility move analysis to a ply of 2 or more which is not shown at all for reasons of clarity. One can think of this as a fine brush of moves emanating from each recorded move. Examples of all types of pruning can be noted in this tree, although additional information is needed for their unambiguous identification. Checker experts all agree that such trees as these are much denser than they probably should be. Attempts to make them less dense by stronger pruning always seem to result in occasional examples of conspicuously poor play. It may well be that denser trees should be used for machine play than for human play, to compensate for deficiencies in the board evaluation methods.

Evaluation Procedures and Learning

Having covered the major improvements in playing techniques as they relate to tree searching, we can now consider improvements in evaluation procedures, with particular reference to learning. We will first discuss the older linear polynomial scheme and then go on to consider the signature-table procedure.

Linear polynomial evaluations

While it is possible to allow for parameters interaction—for example, by using binary connective terms as described in *IBM Journal* 3, 211–229 (1959)—the number of such interactions is large, and it seems necessary to consider more than pair-wise interactions. This makes it quite difficult to depart very much from the linear case. Some improvement in performance resulted when the overall game was split, initially, into 3 phases (opening, mid-game, and end-game) and more recently into 6 phases with a different set of coefficients determined for each phase. Various procedures for defining the phase of the game were tested; the simple one of making the determination solely in terms of the total number of pieces on the board seemed as good as any tried, and there were indications that little was to be gained by going to more than 6 phases.

The total number of parameters used at any one time has been varied from a very few to as many as 40. It has been customary to use all of the currently assessed successful parameters during the learning phase. A number of attempts have been made to speed up actual play by limiting the number of parameters to 5, 10, 15, or 20, selecting those with the larger magnitude coefficients. Five terms in the learning polynomial proved definitely inadequate, an improvement in going from 10 to 15 terms appeared to be barely discernible, and no evidence could be found for improvements in using more than 20 terms. In fact, there seemed to be some indication that a fortuitous combination of many ineffectual parameters with correspondingly low coefficients could, on occasion, override a more effective term and cause the program to play less well than it would with the ineffectual parameters omitted. In a series of 6 games played against R. W. Nealey (the U.S. blind checker champion) using 15 terms, the machine achieved 5 draws with one loss. The six poorest moves in these games as selected by L. W. Taylor, a checker analyst, were replayed, using 20 terms with no improvements and then using only 10 terms with a distinct improvement in two cases. There is, of course, no reason to believe that the program with the fewer number of terms might not have made other and more grievous errors for other untested board situations. Twenty terms were used during the games with W. F. Hellman. No further work has been done on the linear polynomial scheme in view of the demonstrated superiority of the "signature-table" procedure, which will now be described.

Signature-table evaluations

The impracticality of considering all inter-parameter effects and the obvious importance of such interactions has led to the consideration of

a number of different compromise proposals. The first successful compromise solution was proposed and tested on the Project Mac computer by Arnold Griffith, a graduate student at M.I.T. In one early modification of this scheme, 8 subsets of 5 parameters each were used, initially selected from 31 different parameters with some redundancy between subsets. Each subset was designated as a signature type and was characterized by an argument computed in terms of the values measured for the parameters within the subset for any particular board situation. The arguments for each signature type thus specify particular combinations of the parameters within the subset and serve as addresses for entering signature tables where the tabulated values are meant to reflect the relative worth to the computer's side of these particular combinations. In the initial Griffith scheme the values read from the 8 different signature tables were simply added together to obtain the final board evaluation. Parameters which are thought to be somehow related were grouped together in the individual subsets. While it would have been desirable to consider all possible values for each parameter and all possible interrelations between them, this quickly becomes unmanageable. Accordingly, the range of parameter values was restricted to but three values, $+1$, 0 and -1—that is, the two sides could be equal or one or the other could be ahead in terms of the board property in question. Many of the board properties were already of this type. With each parameter limited to 3 values and with 5 parameters in a subset, a total of 3^5 or 243 entries in a signature table completely characterizes all possible interactions between the parameters. Actually since checkers is a "zero sum" game and since all parameters are defined symmetrically, it should be possible to reduce the size of the table roughly by two (122 entries instead of 243) by listing values for positive arguments only and taking values with a reversal of sign when negative arguments are evaluated. Allowing for 48 signature tables, 8 signature types for each of the 6 different phases, we arrive at a memory space requirement for 5856 table entries. Actually two words per table entry are used during the learning phase, as explained later, so the total memory requirement for the learning data is 11,712 words.

An example will make this procedure clear. Consider one signature type which might comprise the following 5 parameters: ANGLE, CENTER, DOREO, GUARD and KCENT, which will not be explained now but which all have to do with the control of the king row and the center of the board. Now consider the GUARD parameter. This can be assigned a value of 0 if both or neither of the sides have complete control of their back rows, a value of $+1$ is the side in question controls his back row while the opponent does not, and a value of -1 if the conditions are reversed. The other 4 parameters can be similarly valued, giving a ternary number consisting of a string of 5 digits

selected from the set −, 0, and + (where − is used for −1, etc.), e.g., "+ − 0 − −" characterizes one particular combination of these five different parameters. This argument can be associated with some function value, a large positive value if it is a desirable combination, a near zero function value if the advantages to the two sides are about even, and a large negative value if it is a disadvantageous combination. Both the arguments and functions are symmetric—that is, the argument and function for the other side would be that gotten by reversing all signs. (In the −, 0, + ternary system the first symbol in the list gives the sign and the processes of complementing and sign reversal are synonymous.) The argument for the other side would thus be − + 0 + +, a negative number which would not be tabulated, but the function value would be the negative of the value listed under + − 0 − −, as it of course must be for the sum of the functions for the two sides to be zero.

The results obtained with this relatively simple method of handling parameter interactions were quite encouraging and as a result a series of more elaborate studies has been made using signature procedures of varying degrees of complexity. In particular, efforts were made (1) to decrease the total number of parameters by eliminating those found to be of marginal utility, (2) to increase the range of values permitted for each parameter, initially increasing the range for certain parameters to permit 7 values (−3, −2, −1, 0, +1, +2, +3) and more recently dividing the parameters into two equal groups, one group being restricted in range to 5 values, and (3) to introduce a hierarchical structure of signature tables where the outputs from the first level signature tables are combined in groups and used as inputs to a set of second level tables, etc.

Most of the experimental work has been restricted to a consideration of the two arrangements shown in Figs. 4 and 5. These are both three-level arrangements. They differ in the degree of the correlation between parameters which is recognized and in the range of values permitted the individual parameters. Both are compromises.

Obviously, the optimum arrangement depends upon the actual number of parameters that must be used, the degree to which these parameters are interrelated and the extent to which these individual parameters can be safely represented by a limited range of integers. In the case of checkers, the desired number of parameters seems to lie in the range of 20 to 30. Constraints on the range of values required to define the parameters can be easily determined, but substantially nothing is known concerning the interdependencies between the parameters. A series of quite inconclusive experiments was performed in an effort to measure these interdependencies. About all that can be said is that the constraints imposed upon the permissible distribution of pieces on the board in any actual game, as set by the rules of the game and as dictated

by good playing procedures, seem to produce an apparent average correlation between all parameters which is quite independent of the specific character of these parameters. The problem is further complicated by the fact that two quite opposing lines of argument can be advanced—the one to suggest that closely related terms be placed in the same subsets to allow for their interdependencies and the second to

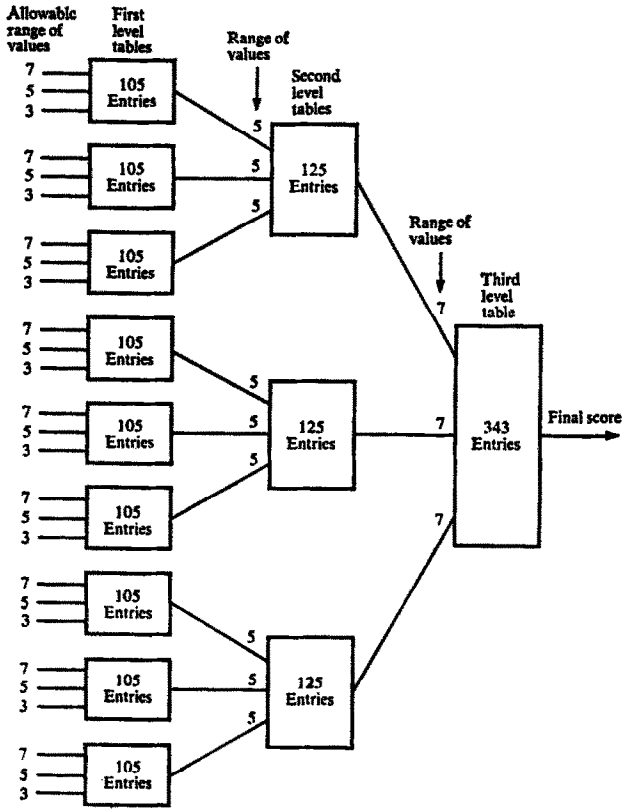

FIG. 4. A 3-level signature-table arrangement with 27 terms.

suggest that such terms be scattered among groups. The second suggestion can be made to look reasonable by considering the situation in which two parameters are unknowingly so closely related as to actually measure the same property. Placing these two terms in the same subset would accomplish nothing, while placing them in different subgroups permits a direct trade-off evaluation to be made between this property in question and the properties measured by the other parameters in both subgroups.

A few comments are in order at this time as to the supposedly

symmetrical nature of the parameter data. While it is true that checkers is a zero-sum game and while it is true that the parameters are all defined in a symmetrical way, that is, as far as black vs. white is concerned, the value of a board situation as defined by these parameters is actually dependent upon whose turn it is to play. A small but real bias normally exists for most parameters in favor of the side whose turn it is to move, although for certain parameters the reverse is true. The linear

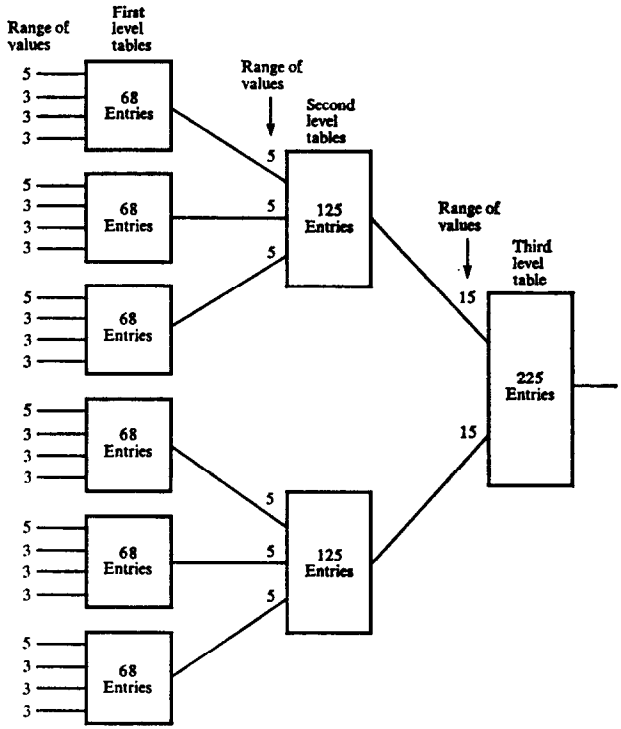

FIG. 5. Revised 3-level signature-table scheme with 24 terms.

polynomial method of scoring is unfortunately not sensitive to these peculiarities of the different parameters since the partial scores for all types are simply added together. The signature-table procedure should be able to take the added complication into account. Of course, the distinctions will be lost if the data are incorrectly stored or if they are incorrectly acquired. By storing the data in the uncompressed form one can evaluate this effect. More will be said about this matter later.

In the arrangement shown in Fig. 4 there were 27 parameters divided into 9 groups of three each, with each group being made up of one 3-valued parameter, one 5-valued parameter and one 7-valued parameter. Each first level signature table thus had 105 entries. The output

values from each of these tables were quantized into 5 values and second level signature tables were employed to combine these in sets of three. These second level tables thus had 125 entries each. These outputs are further quantized into 7 levels and a third level signature table with 343 entries was used to combine the outputs from the three second level tables into a final output which was used as the final board evaluation. Obviously, the parameters used to enter the first level tables were grouped together on the basis of their assumed (and in some cases measured) interdependencies while the resulting signature types were again grouped together as well as possible, consistent with their assumed interdependencies. As always, there was a complete set of these tables for each of the 6 game phases. The tables were stored in full, without making use of the zero-sum characteristic to halve their size, and occupied 20,956 cells in memory. Outputs from the first level tables were quantized into 5 levels and the outputs from the second level tables into 7 levels.

THE LATEST SIGNATURE-TABLE PROCEDURE

The arrangement shown in Fig. 5 used 24 parameters which were divided into 6 subgroups of 4 parameters each, with each subgroup containing one 5-valued parameter and three 3-valued parameters. In this case the first level tables were compacted by taking advantage of the assumed symmetrical character of the data, although this is a dubious procedure, as already noted. It was justified in this case because of the added parameter interactions which this made possible and because of a very large inverse effect of table size on speed of learning. This reduced the size of the first level tables to 63 words each. The outputs from the first level tables were quantized into 5 levels as before and the outputs from the second level tables were quantized into 15 levels. The second and third level tables were not compacted, in an attempt to preserve some non-symmetrical features. The total memory requirement for the tables as thus constituted was 10,136 words.

Before we can discuss the results obtained with the signature-table scheme it will be necessary to turn our attention to the various book-learning procedures.

Book Learning

While book learning was mentioned briefly in *IBM Journal* 3, 211–229 (1959), we will describe it in some detail as it was used throughout the studies now to be reported. Just as books speed up human learning, one might expect that a substantial increase in machine learning speed might result if some use could be made of book information, in this case, the

existing library of master play. To this end a reasonable sample (approximately 250,000 board situations) of this master play has been key punched and transcribed to magnetic tape. These are mostly draw games; in those cases where a win was achieved, data are used only from the moves made by the winning side. The program has been arranged to play through these recorded games considering one side, then the other, much as a person might do, analyzing the situation in terms of the existing evaluation procedures and listing the preferred move. This move is then compared with the book-recommended move and a suitable adjustment made in the evaluation procedure. This, of course, assumes that the book-recommended move is the only correct move, which it may not be, either because of a plurality of good moves or in some cases because of an actual error. However, if enough book moves are used, if the books are usually correct and if the adjustments per move are of the proper size, the process should converge toward an optimum evaluation procedure, subject always to a basic limitation as to the appropriateness and completeness of the parameter list that is used.

While it still takes a substantial amount of machine time to play through the necessary book games, the learning process is very much faster than for learning from actual play. In the first place, the game paths followed are from the start representative of the very best play since the program is forced always to make the recommended book move before proceeding to consider the next move. Secondly, it is possible to assign values to be associated with the moves in a very direct fashion without depending upon the unreliable techniques which were earlier described. Finally the analysis of each move can be extremely limited, with little or no mini-maxing, since the only use made of the overall scores is that of measuring the learning, whereas in the earlier procedures these scores were needed to determine credit assignments to the parameters. The net effect of these factors is to make it possible to consider many more moves, at the rate of 300 to 600 moves per minute rather than at the roughly one move per minute rate which is typical for actual games.

We will first explain how learning is achieved in terms of coefficients in a linear polynomial and then go on to the signature-table case.

During the learning process, use must be made of the previously determined coefficients to perform the evaluation of all board situations either right after the initial moves or, if jump situations are encountered, at some terminating ply depth with the scores backed up by the mini-maxing procedure. During this mini-maxing, it is also necessary to back up the values of the parameter values themselves (i.e., the terms without coefficients), associated with the selected terminating board situations corresponding to the optimized path leading from each of the possible first moves. If there are 9 possible moves, a 9×27 table will

be produced in which the rows correspond to the 9 different moves and the columns correspond to the 27 different parameters. On the basis of the book information, one row is indicated as being the best move.

The program must analyze the data within the table and accumulate totals which on the average indicate the relative worth of the different parameters in predicting the book move, and it must alter the coefficients to reflect the cumulative learning indicated by these totals. A variety of different procedures has been tested for accumulating totals; one of the simplest and, surprisingly, the most effective, seems to be to simply count the number of moves, for each parameter separately, for which the parameter value is larger than the value associated with the book move and the number of moves for which the parameter value is smaller than the value associated with the book move. If these cumulated counts over all board situations examined to date are designated H and L, then one measure of the goodness of the parameter in predicting the book move is given by

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

This has the dimensions of a correlation coefficient. It would have a value of $+1$ if the parameter in question always predicted the book move, a value of -1 if it never made a correct prediction, and a value of 0 if there was no correlation between the machine indications and the book. The best procedure found to date is simply to use the values of the C's so obtained as the coefficients in the evaluation polynomial, although arguments can be advanced for the use of the values of the C's raised to some power greater than 1 to overcome the effect of several inconsequential terms overriding a valuable indication from some other term as mentioned earlier.

Typical coefficients as tabulated by the computer are shown in Table 1 based on roughly 150,000 board situations and using 31 functions during the learning process. The 19 terms per phase having the largest magnitude coefficients are listed. The play against Hellman mentioned earlier used this particular set of terms.

Book learning using signature tables

Extending this book-learning technique to the signature-table case is relatively easy. All that need be done is to back up the signatures corresponding to the signature types being used in a way quite analogous to the handling of parameters in the linear polynomial case. Taking the example used earlier, one signature corresponding to one possible move might be $+ - 0 - -$ (actually stored in the machine in binary form). Each signature type for each possible move is similarly characterized. Two totals (called D and A) are accumulated for each

TABLE 1. LINEAR POLYNOMIAL TERMS (PARAMETER NAMES AND LEARNED COEFFICIENTS) AS USED IN THE GAMES WITH W. F. HELLMAN. (THESE COEFFICIENTS RESULTED FROM AN ANALYSIS OF APPROXIMATELY 150,000 BOOK MOVES)

Phase 1—Terms and coefficients									
GUARD	QUART	DIAGL	EDGES	FRONT	ANGLE	CENTR	NODES	DCHOL	ADVAN
0.33	0.29	−0.21	−0.20	−0.19	−0.18	0.14	0.13	0.11	−0.08
PINS	DYKSQ	FREE	EXCHS	THRET	STARS	PRESS	UNCEN	LINES	
0.07	0.07	0.06	−0.05	0.04	0.04	−0.04	0.03	0.02	
Phase 2—Terms and coefficients									
SPIKE	GUARD	EDGES	QUART	CENTR	ANGLE	FRONT	ADVAN	SHOVE	THRET
0.85	0.36	−0.24	0.23	0.21	−0.21	−0.19	−0.18	0.16	0.14
NODES	PINS	DCHOL	STARS	OFSET	HOLES	DIAGL	UNCEN	MOBIL	
0.13	0.11	−0.10	−0.09	0.09	0.09	−0.09	0.08	0.05	
Phase 3—Terms and coefficients									
SPIKE	KCENT	PANTS	GUARD	FRONT	CRAMP	ADVAN	EDGES	CENTR	STARS
0.88	0.48	0.42	0.37	−0.23	0.23	−0.23	−0.22	0.20	−0.19
QUART	ANGLE	THRET	DCHOL	PINS	SHOVE	NODES	UNCEN	OFSET	
0.19	−0.19	0.15	0.14	0.13	0.10	0.10	0.09	0.08	
Phase 4—Terms and coefficients									
SPIKE	GUARD	PANTS	KCENT	STARS	ADVAN	FRONT	THRET	ANGLE	EDGES
0.86	0.62	0.61	0.56	−0.30	−0.30	−0.27	0.26	−0.23	−0.22
DIAGL	CENTR	SHOVE	QUART	PINS	UNCEN	OFSET	DENYS	UNDEN	
0.22	0.20	0.18	0.16	0.12	0.11	0.09	0.09	−0.07	
Phase 5—Terms and coefficients									
GUARD	SPIKE	PANTS	KCENT	THRET	DIAGL	ADVAN	UNCEN	ANGLE	SHOVE
0.81	0.68	0.62	0.55	0.36	0.33	−0.32	0.27	−0.26	0.25
UNDEN	FRONT	DENYS	PINS	CENTR	EDGES	DYKSQ	QUART	DEUCE	
−0.22	−0.22	0.20	0.19	0.18	−0.16	−0.16	0.15	0.06	
Phase 6—Terms and coefficients									
PRESS	KCENT	UNCEN	UNDEN	DYKSQ	DENYS	SHOVE	DIAGL	SPIKE	THRET
−0.54	0.54	0.45	−0.41	−0.40	0.40	0.39	0.39	0.37	0.36
EXCHS	OFSET	ADVAN	PINS	ANGLE	FRONT	DEUCE	FREE	QUART	
−0.34	−0.26	−0.24	0.23	−0.23	−0.32	−0.16	−0.11	0.08	

of the possible signature types. Additions of 1 each are made to the D totals for each signature for the moves that were not identified as the preferred book move and an addition of n, where n is the number of nonbook moves, is made to the A totals for the signatures identified with the recommended book move. The reason for adding n to the book move A totals is, of course, to give greater positive weight to the book recommended move than is the negative weight given to moves that do not happen to correspond to the currently found book recommendation (there may be more than one good move and some other authority might recommend one of the other moves). This procedure has the incidental effect of maintaining equality between the grand totals of the A's and D's accumulated separately for all signatures in each table, and so of preserving a zero-sum character for the data.

When enough data have been accumulated for many different board situations, additions will have been made in the A and D columns against most of the signature arguments. The program then computes correlation coefficients for each signature defined in an analogous fashion to the earlier usage as

$$C = (A - D)/(A + D).$$

In the case of the third level table these values are used directly as board evaluations. For the other two levels in the signature-table hierarchy, the actual values to be entered must be quantized so as to restrict the range of the tabulated values. This quantization has normally been done by first separating out all zero values and entering them into the tables as such. The nonzero values are then quantized by ranking the positive values and negative values separately into the desired number of equisized groups. The table entries are then made in terms of the small positive and negative integer numbers used to specify the relative ranking order of these groups.

This process of updating the signature tables themselves is done at intervals as determined by the rate at which significant data accumulate. During the intervals between updating, additions are, of course, continually being made to the tables of A's and D's.

There are several problems associated with this newer learning scheme. Reference has already been made to the space and time limitations which restrict the number of parameters to be combined in each signature type and restrict the range allowed for each parameter. The program has been written so that these numbers may be easily varied but this facility is of little use because of the very rapid rate at which the performance and the storage requirements vary with the values chosen. Values less than those indicated lead to performance but little different from that exhibited by the older linear polynomial experiments, while larger values greatly increase the memory require-

ments and slow down the learning rate. A great deal of juggling is required in order to make even the simplest change if the operating times are to be kept within a reasonable range, and this still further complicates the problem of considering meaningful experiments.

This inverse effect of the table size on the learning rate comes about because of the need to accumulate data in the A and D columns for each signature-table entry. The effect is, of course, compounded by the hierarchical nature of the table complex. At the start of a new learning run there will be no entries in any of the tables, the computed C's must all be set to zero and the program will have no basis for the minimaxing procedure. Depending upon the particular selection of the book games used there may, in fact, be a relatively long period of time before a significant fraction of signatures will have been encountered, and as a consequence, statistically unreliable data will persist in the "C" table. Not only will the individual function values be suspect but the quantizing levels will perforce be based on insufficient data as well. The magnitude of this effect will, of course, depend upon the size of the tables that the program is generating.

Palliative measures can be adopted to smooth the C tables in order to compensate for the blank entries and for entries based on insufficient data. Four of the more effective smoothing techniques have been found to be (1) smoothing by inversion, (2) smoothing from adjacent phases, (3) smoothing by interpolation and (4) smoothing by extrapolation. Smoothing is, of course, most needed during the early stages of the learning process, but it also must be used during play even after a rather extensive learning run.

As a matter of fact, certain signatures are so improbable during book play (some may in fact be impossible) that voids are still found to exist in the signature tables, even after playing 100,000 book game board situations. There is the reassuring thought that signatures not found during the learning process are also unlikely to be found during play. However, because of the very many board situations explored during the look-ahead process and presumably because of the consequences of making decisions on the basis of statistically unreliable entries, the quality of the play using unsmoothed data was found to be somewhat erratic until a fairly large amount of learning had been achieved.

It should be pointed out that the smoothing techniques are employed as temporary expedients. All previous smoothed results are discarded and completely new calculations of values of C are made periodically during learning from the accumulated and uncorrupted A and D data. The effects of smoothing do persist, however, since the entries in the second and third level tables, and hence the locations at which the A and D data are stored, are influenced by it.

Smoothing by inversion is done by averaging positive and negative entries (with compensating sign inversions), and it is partially justified by the zero-sum symmetrical characteristic of the data.

Smoothing from adjacent phases is done by transferring data between phases. This is possible because of the random way in which data accumulate for the different phases, and it is reasonably valid because the values associated with a given signature vary but little between adjacent phases. This form of smoothing has been found to be of but limited utility since the same reasons which account for the absence of specific data for one phase often operate to prevent corresponding data from being generated for adjacent phases.

Smoothing by interpolation is based on the assumption that a missing correlation for a signature which contains one or more zeros in its argument can be approximated by averaging the values appearing for the related signatures where the zeros are individually replaced by a + and then by a −. In order for this to be effective there must be data available for both the + and − cases for at least one of the zero-valued parameters. This form of smoothing assumes a linear relationship for the effect of the parameter to which the interpolation is applied. It is, therefore, no better as far as this one parameter is concerned than the older linear polynomial procedure. This form of smoothing is quite ineffectual since all too often balanced pairs of entries cannot be found.

Smoothing by extrapolation may take two forms, the simplest being when entries are found for the zero value of some particular function and for either the + or the − case and a void for the remaining case is to be filled. All too often, however, the more recalcitrant cases are those in which the zero entry only for some one parameter is found and substitute data are sought for both the + and the − case. Here we have recourse to the fact that it is possible to compute the apparent effect of the missing parameter from all of the pertinent data in the signature table, on the assumption of linearity. The program therefore computes a correlation coefficient for this parameter alone and uses this with the found signature data. Admittedly this is a very dangerous form of extrapolation since it completely ignores all nonlinear effects, but it is often the only recourse.

SIGNATURE-TABLE LEARNING RESULTS

The results of the best signature-table learning run made to date are shown in Table 2. This particular run was arranged to yield comparable figures for both the newer signature-table procedure and the older linear polynomial procedure. Because of the great amount of machine time required (approximately 10 hours per run) it has not yet been

possible to optimize (1) the choice of parameters to be used, (2) the range of values to be assigned to these parameters, (3) the specific assignments of parameters to signature types, (4) the detailed hier-

TABLE 2. CORRELATION COEFFICIENTS MEASURING THE EFFECTS OF LEARNING FOR THE SIGNATURE-TABLE PROCEDURE AND FOR THE LINEAR POLYNOMIAL PROCEDURE AS A FUNCTION OF THE TOTAL NUMBER OF BOOK MOVES ANALYZED. (THESE TESTS USED 27 PARAMETERS WHICH FOR THE SIGNATURE-TABLE SCORE WERE GROUPED IN THE CONFIGURATION SHOWN IN FIG. 4)

Total number of book moves analyzed	Correlation coefficient, C	
	Signature-table case	Polynomial case
336	−0.08	−0.18
826	+0.06	−0.13
1,272	0.13	+0.06
1,769	0.18	0.10
2,705	0.27	0.15
3,487	0.31	0.16
4,680	0.34	0.15
5,446	0.36	0.16
8,933	0.38	0.19
10,762	0.39	0.20
14,240	0.40	0.21
17,527	0.41	0.22
21,302	0.41	0.23
23,666	0.42	0.23
30,173	0.43	0.24
40,082	0.43	0.25
50,294	0.43	0.26
55,165	0.44	0.26
66,663	0.45	0.26
70,083	0.45	0.26
90,093	0.46	0.26
106,477	0.46	0.26
120,247	0.47	0.26
145,021	0.47	0.26
173,091	0.48	0.26
183,877	0.48	0.26

archical structure of the signature tables, (5) the table sizes and (6) the various smoothing techniques which must be used during the early learning phases.

Table 2 reports the apparent goodness of play based upon a correlation factor defined as

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

where L is the accumulated count of all available moves which the program rates lower than its rating for the book recommended move and H is the accumulated count of all available moves which the program rates higher than or equal to its rating for the book recommended move. During this learning run the program looked ahead only a single ply except in those cases where jumps were pending. The observed correlation coefficients are fairly good measures of the goodness of the evaluation procedures without mini-maxing. Coefficients were computed during the run both by using the signature-table procedure and by the older linear polynomial procedure. These figures are tabulated in the second and third columns against the total number of moves in the first column. It will be observed that the coefficient for the polynomial procedure appears to stabilize at a figure of 0.26 after about 50,000 moves, while the coefficient for the signature-table procedure continues to rise and finally after perhaps 175,000 moves reaches a limit of 0.48. Interestingly enough, the signature-table coefficient was always larger than the polynomial coefficient even during the very early stage, although a detailed analysis on a move-by-move basis, which cannot be easily reproduced here, did show that the signature-table method was the more erratic of the two during this stage.

It should be noted that these linear polynomial results are not directly comparable with the coefficients for individual terms as reported in Table 1, since for Table 1 the H values used in computing the C's did not include those moves rated equal to the book move while in Table 2 equals are included, and the computed coefficients are correspondingly lower. The discrepancy is particularly marked with respect to those parameters which are usually zero for most moves but which may be extremely valuable for their differentiating ability when they do depart from zero. Most of the terms with high coefficients in Table 1 have this characteristic. Furthermore, when mini-maxing was required during the two tests it was based on different criteria, for Table 1 on the linear polynomial and for Table 2 on signature tables.

The results of Table 2 seem to indicate that the signature-table procedure is superior to the linear polynomial procedure even in its presently unoptimized form. It would be nice if one could measure this improvement in some more precise way, making a correct allowance for the difference in the computation times.

Perhaps a better way to assess the goodness of the play using signature tables is to list the fraction of the time that the program rates 0, then 1, 2, 3, etc., moves as equal to or higher than its rating of the book

recommended move. Typical figures are tabulated below, measured for a test lot of 895 representative moves after the program had learned by analyzing 173,989 book moves:

moves higher or equal	0	1	2	3	4	5	6
fractional times found	0.38	0.26	0.16	0.10	0.06	0.03	0.01

In view of the high probability of occurrence of two equally acceptable moves, the sum of the figures in the first two columns, namely 0.64, is a reasonable estimate of the fraction of time that the program would make an acceptable move without look-ahead and mini-maxing. Look-ahead greatly improves the play and accounts for the difference between this prediction and the observed fact that the playing program tends to follow book recommended moves a much higher fraction of the time.

The Introduction of Strategies

The chief defect of the program in the recent past, according to several checker masters, seems to have been its failure to maintain any fixed strategy during play. The good player during his own play will note that a given board situation is favorable to him in some one respect and perhaps unfavorable in some second respect, and he will follow some fairly consistent policy for several moves in a row. In general he will try to maintain his advantage and at the same time to overcome the unfavorable aspect. In doing this he may more or less ignore other secondary properties which, under different circumstances, might themselves be dominant. The program, as described, treats each board situation as a new problem. It is true that this procedure does not allow the program to exploit those human failings of the opponent that might have been revealed by the earlier play or to conduct a war of nerves intended to trick the opponent. Such actions have little place in games of complete information and can well be ignored.†

† This statement can be questioned and, in fact, has been questioned by an anonymous reviewer who quite rightly pointed out that it would be desirable for the program to be able to define what is called "deep objectives", and, more importantly, to be able to detect such "deep objectives" on the part of a human opponent. The reviewer went on to say in part "—the good player will sometimes define a 'deep objective' and maneuver toward that point. He is always on the lookout for possibilities which will help him to get the better of the opponent. The opponent, unaware of his true objective until too late, does not defend adequately and loses.—It is most helpful to him to know that his opponent is not also playing a similar 'deep game.' I believe that the 'practical indeterminacy' of checkers makes the technique of 'deep' objectives by good players quite feasible. Indeed, I don't doubt the technique is part of the basic equipment of any champion player, however inarticulately he may describe it. This is perhaps the reason Hellman did better in the games by mail. He had time to study out appropriately 'deep' objectives and then to realize them. This is

What may certainly be questioned is the failure to take account of the initial board situation in setting the goals to be considered during the look-ahead process. Were the program able to do this, then it could adopt a strategy for any particular move. If the program finally made a move that was consistent with this strategy, and if the opponent were unable to vitiate this strategy, then the program would, on the next move, again tend to adopt the same strategy. Of course, if the program had been unable to maintain an advantage by following its initial strategy, it might now find that a different strategy was indicated and it would therefore change its strategy. Nevertheless, on the average, the program might follow a given strategy for several moves in a row and so exhibit playing characteristics that would give the impression of long-range planning.

A possible mechanism for introducing this kind of strategic planning is provided by the signature-table procedure and by the plausibility analysis. It is only necessary to view the different signature types as different strategic elements and to alter the relative weights assigned to the different signature types as a result of the plausibility analysis of the initial board situation. For this to be effective, some care must be given to the groupings of the parameters into the signature types so that these signature types tend to correspond to recognizable strategic concepts. Fortunately, the same initial-level grouping of parameters that is indicated by interdependency considerations seems to be reasonable in terms of strategies. We conclude that it is quite feasible to introduce the concept of strategy in this restricted way.

For reasons of symmetry, it seems desirable to pick two signature types for emphasis, that one yielding the highest positive value and that one yielding the most negative value for the most plausible move found during the initial plausibility analysis. This procedure recognizes the fact that to the opponent, the signs are reversed and his strongest signature type will be the first player's weakest one and vice versa. The simplest way to emphasize a particular strategy is to multiply the resulting values found for the two selected signature types by some arbitrary constant before entering a subsequent stage of the analysis. A factor of 2 (with a limit on the maximum resulting value so as not to exceed the table range) seemed reasonable and this has been used for most of the experiments to date.

---

also what checker masters have in mind when they criticize the program's failure to maintain any fixed strategy during play."

This point of view finds support in the observation that those master players who have defeated the computer have all asked searching questions regarding the program, while good players who fail to win usually seem to hold the program in awe and generally fail to make any attempt to understand it.

This opens up what may be a fruitful line for additional research.

The results to date have been disappointing, presumably because of the ineffectual arrangement of terms into usable strategic groups, and as a consequence, this method of introducing strategies has been temporarily abandoned.

Conclusions

While the goal of getting the program to generate its own parameters remains as far in the future as it seemed to be in 1959, we can conclude that techniques are now in hand for dealing with many of the tree pruning and parameter interaction problems which were certainly much less well understood at the time of the earlier paper. Perhaps with these newer tools we may be able to apply machine learning techniques to many problems of economic importance without waiting for the long-sought ultimate solution.

Acknowledgements

These studies were largely carried out while the writer was at the Thomas J. Watson Research Laboratories of the IBM Corporation, and while he was a Visiting Professor at M.I.T. More recently, the work has been supported in part by Stanford University and by the Advance Research Projects Agency of the Office of the Secretary of Defense (SD-183). The IBM Corporation has continued to aid the work by supplying time on an IBM 7094 computer at their San Jose Development Laboratories. Many individuals have contributed to these studies, and in particular, Arnold Griffith of M.I.T. deserves commendation for suggesting the initial form of the signature-table procedure. The continuing interest and cooperation of the officers and player-members of the American Checker Federation has been most helpful.

Appendix

A two-game match between an IBM 7094 and a CDC 6600. The 7094 program was the latest signature-table version while the 6600 program was of the older linear polynomial type as extensively modified and reprogrammed for the 6600 by James R. Slagle and his associates at the University of California, Lawrence Radiation Laboratory at Livermore. Game 2 was marred and finally terminated by a curious 7094 programming bug which had never previously caused any trouble.

Game 1 *with the* 6600 *playing Black*

9–14	5–9	11–15	10–17	3–8	9–14
22–19	22–17	26–22	21–14	32–28	6–1
11–15	9–13	16–20	12–16	8–12	Black
18–11	30–25	31–26	19–10	29–25	resigns.
7–16	13–22	4–8	2–6	1–5	
24–19	25–9	23–18	26–23	10–6	
8–11	6–13	8–11	6–15	5–9	
25–22	28–14	18–14	14–10	25–21	

Game 2 *with the* 7094 *playing Black*

9–14	8–11	7–11	12–19	14–18	Game
22–17	25–22	29–25	23–16	8–3	terminated
11–15	11–15	11–15	4–8	18–23	because of
24–19	17–13	22–17	16–12	26–19	erratic
15–24	15–24	3–7	8–11	15–24	Black
28–18	27–20	20–16	12–8	25–22	behavior.