Second Edition

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

Volume 1
A-L

Awarded

American Library Association's

Outstanding Reference Source

Association of American Publishers Award

Best New Professional and Scholarly Publication

Stuart C. Shapiro
Editor-in-Chief

Ranan B. Banerji, GAME PLAYING, 543-550

From Shapiro, Stuart C., Editor-in-Chief, Encyclopedia of Artificial Intelligence, 2nd Ed., John Wiley & Sons, Inc., NY, 1992.

GAME PLAYING

An important part of AI research lies in efforts at understanding intelligent behavior as opposed to simulating it for solving a specific problem in an applications domain. For this former area of research games remain an excellent metaphor.

A considerable body of knowledge exists in mathematics about the property of games; there has not been enough interaction between this knowledge and AI research. One reason is that it has been proven (Stockmeyer and Chandra, 1979) for games like chess, checkers, and go that no game-playing strategy can exist that remains efficient over larger and larger boards. As a result, AI research in games has been restricted to two extreme viewpoints. On the one hand, efforts are made to incorporate knowledge of specific games (eg, chess on a standard board) into the program. At the other end of the spectrum, one restricts one's attention to methods of search (qv) reduction.

There are classes of games, however, for which methods of efficient play can be developed and used. Such techniques are often applicable over a wide class of seemingly unrelated games. Study of such classes yields insight into the notion of similarity and analogy, activities of recognized value in automatic learning of problem-solving (qv) and game-playing strategies. Such studies form an important third approach to the study of games.

In what follows all of these aspects of AI research into games are discussed. The second and third approaches are discussed in some detail since a general body of knowledge exists for these. For the first approach the reader is directed to specialized treatises and papers (Frey, 1977; Berliner, 1978; Bramer, 1983). In the next section some formal definitions are made to facilitate the later discussion.

MATHEMATICAL FORMULATION

In the original, most general definition (von Neumann and Morgenstern, 1944) a game is characterized by the set of all sequence of plays possible, as made by N players, and by the payoffs to the N players corresponding to each sequence. Each play reduces the set of sequences to the subset that has that play as the initial one. The moves thus characterize a partition on the set of sequences. However, since all the players do not necessarily know what play was made (eg, in Kriegspiel or bridge), the players' knowledge restricts the set to some superpartition of this partition.

Most of the work on games in the field of AI has been in the case of two-person games with complete information and with alternating moves, although some work on bridge and poker has been reported. As a result, one obtains a considerable simplification on the structure on the partitions over the set of all play sequences. A set of nested subpartitions results as the representation of the plays, and one can analyze the games in terms of trees. Figure 1 indicates the relationship between the game tree and the partitions on the sequences represented by them. It also shows the kind of complications introduced by incomplete information that makes the game tree less useful for general N-person games.

Most of the analyses of games in AI have been in terms of game trees. In these trees each node represents a class of possible continuations of the game. However, one could also consider the node to represent the history of past moves. From the latter point of view, each arc of the tree represents a move by a player. The node also restricts how the rest of the play is allowed to continue.

Of course, two distinct histories of moves do not always restrict the possible continuations in distinct ways. For instance, in chess, the sequence P-K4, Kt-KB3, Kt-KB3 leads to the same situation as Kt-KB3, Kt-KB3, P-K4. Many authors have considered it meaningful to treat the two resulting nodes in the tree as equivalent and represented by the board configuration produced by them. This identifies two nodes of the tree as a single node, and as a result, the structure becomes a graph rather than a tree (see Fig. 2). The nodes of this graph may be considered to be represented by the configuration of pieces on the board together with a specification as to who is on move. The latter specification may or may not be uniquely determined by the node of the graph. Such subtleties need not interest us here. The interested reader will find these discussions in Banerji (1980). For our purposes we shall take the formalization where each node can be considered from the point of view of either player being on the move.

Conway (1976, 1977; Berlekamp, Conway, and Guy, 1982) abstracts the graph away by defining a game (a game node in the above terminology) to be given by two sets of games (ie, game nodes), to wit the ones that one player could reach if he was on move and the ones that the other player could reach if the other player was on move. As Conway would put it, "A game is an ordered pair of two sets of games."

The Conway approach has led to the development of a new area of nonstandard number theory and unified some known theories of impartial games with this extended number theory. However, so far a clear way has not been found to use their results to develop new winning strategies of known interesting games. This theory shall therefore not be discussed any further. However, the interested reader is urged strongly to look into the work of Conway and his colleagues for some extremely exciting and amusing, albeit occasionally strenuous reading.

STRATEGIES

For the rest of this overview the important problem of concern is, given a node in a given game, how does the player on move assure a win if possible? A number of

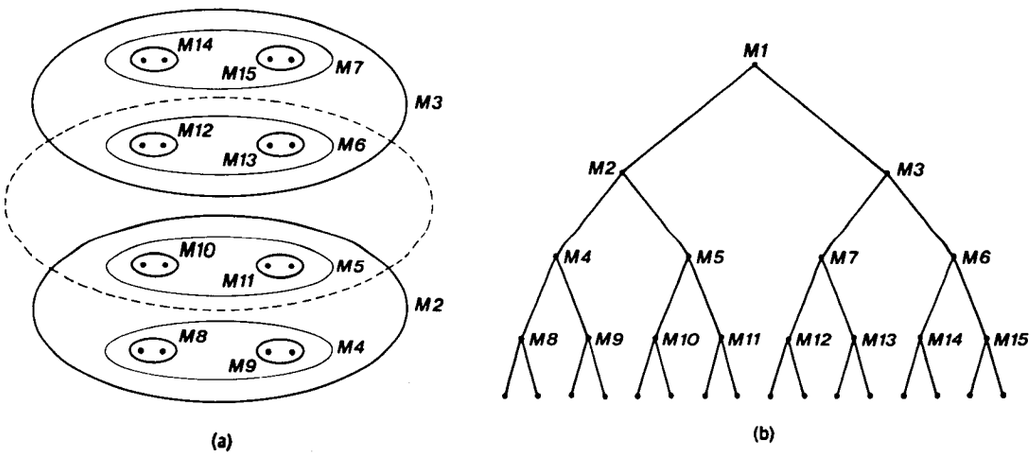

Figure 1. Game with 16 possible plays. (*a*) Partition representation of von Neumann and Morgenstern (1944). The game being with complete information, two persons, and alternate moves, the tree representation (*b*) is also possible. If the second player is not allowed to know the first player's first move, after his move he would not know where the play is and would merely have the play localized in either the set enclosed by the dotted line in (*a*) or its complement. If the player on move is determined by rules other than alternation, more subsets would appear in (*a*) and the simplicity of the tree representation would be lost.

considerations arise in determining how such an assurance can be obtained. First, the only decision available to the player is the choice of a move available at the node. The opponent's move leads the game from the resulting node to a node where the player has to make another choice. This choice cannot be made beforehand: until the opponent has played, the player does not know what the resulting node will be. The player's decision cannot be with respect to a single move or even a sequence of moves. He or she has to decide, not on a move, but on a method by which, given a node, a move can be chosen. In mathematical parlance, one needs a function mapping nodes to moves. Such a function is called a *strategy*. A winning strategy is one whose repeated application, one at each node where a choice is needed, leads to a win whenever a win is possible.

This concept of strategy in games has a very close relationship with the same concept in game theory as studied in economics (see MINIMAX PROCEDURE). Suffice it to say here that in a two-person game with complete information and

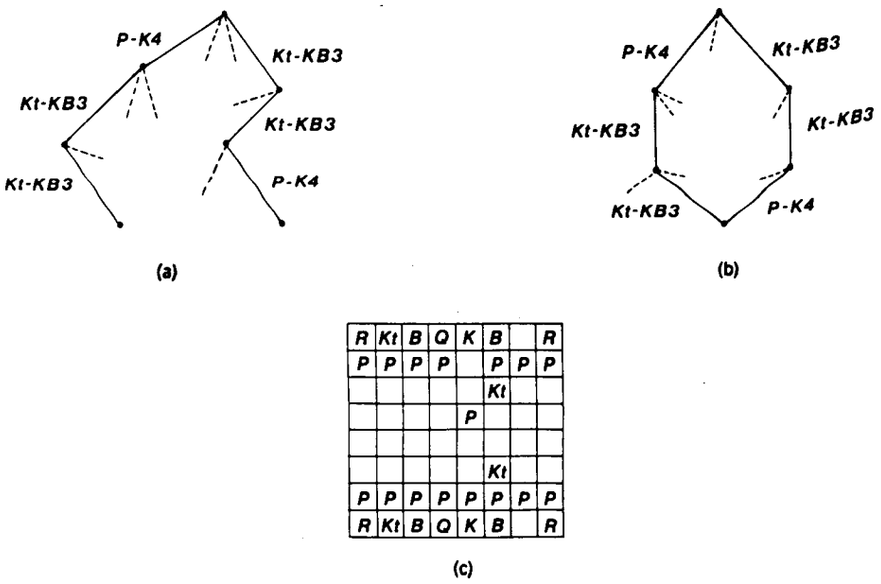

Figure 2. (*a*) Two distinct move sequences can determine the same constraint on possible continuations and are, in that sense, "equivalent." By identifying the equivalent nodes of the game tree, one obtains the game graph (*b*). In the case of most games these equivalent nodes do indeed represent identical "game board configurations" (*c*), giving concrete meaning to the nodes of the graph.

alternating moves the calculation of strategy (albeit still inefficient; see above) becomes much simpler than in the general case. The general method applied to the case of Figure 1 would lead to the calculation of a 32 × 1024 matrix. Only the simple special case is discussed here.

If one unfolds the game graph into a tree, one obtains a method of calculating the winning strategy by a method that, with some modifications (see below), has remained the only error-proof method known. The method can best be described in terms of a recursive definition:

If the node is a leaf (end of game), the game is already won or lost, and its value is the value of the node. No move needs to be made.

If the node is the player's move, find the value of each node to which one can move. Make the move that maximizes this value. The value of the node is this maximum value.

If the node is the opponent's move, the value of the node is the minimum value among the nodes that can be reached by a move.

Figure 3 shows the value of a node. The optimum player's move from a node and for all the nodes reached from it by the optimal move of each player follow the leftmost branch sequence.

This method of evaluation and strategy construction is called the *minimax method.* The trouble with this method of finding the strategy is the amount of calculation involved. Early estimates about chess revealed that such a calculation made from the starting position of chess would involve the generation of about 7^{32} nodes. The most optimistic guess about the speed of calculation and space used by memory still yields the fact that such a calculation would take millenia to calculate on an impossibly large machine.

Thus, playing chess "optimally" is not feasible using such a naive search technique. People do play chess well; nobody knows if anybody has ever played optimal chess. See Frey (1977) and Berliner (1978) for a discussion of how machines can be made to play acceptable chess and how it compares with human play. Some of the principles on which optimal and suboptimal game playing can be based are described below.

SEARCH REDUCTION: KERNELS AND EVALUATIONS

Any method of game playing has to be based on some principle that allows one to reduce the amount of search involved in the calculation of strategy. Such methods have been developed both in AI and in the mathematical theories of games. The latter are described first, since the former are somewhat more difficult to justify in any precise manner.

The set of all nodes in the game graph whose minimax value is 1 are called the winning nodes and the others the losing nodes (for the time being, draws are not considered). The minimax principle can be stated in terms of these sets of nodes: a node is winning if there is at least one node connected to it that is a losing node; a node is losing if all the nodes connected to it are winning nodes; and terminal nodes are winning if their value is 1.

Consider the property of being winning. Of course, every leaf node of the game tree (terminal node of the game graph) has this property if its value is 1. Also, if a node lacks this property, every node connected to it has this property. Again, each node having this property is connected to at least one node lacking this property.

It can be seen that if a property of nodes, which is easy to calculate, is shared by all leaf nodes with value 1 and has the above characteristics, the nodes having the property are precisely the winning nodes. Having minimax

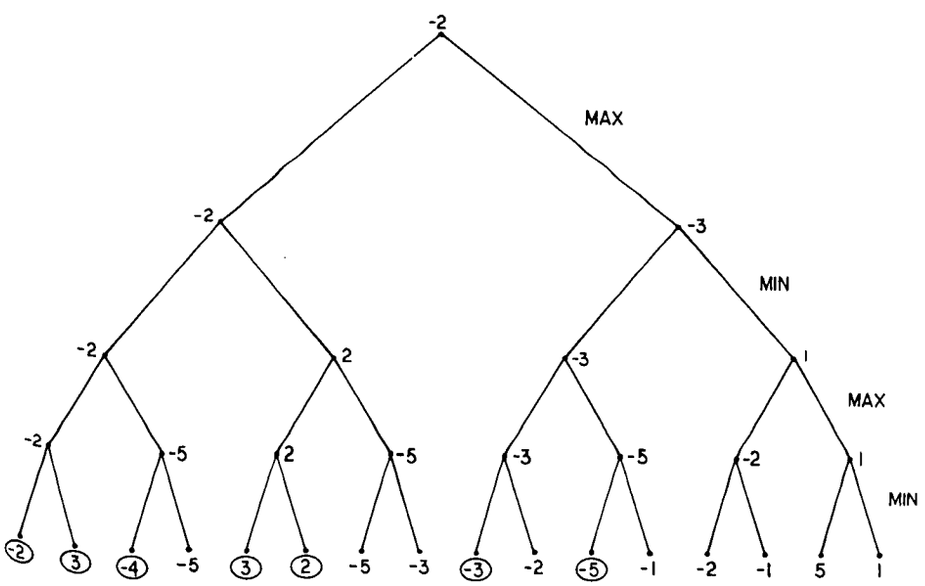

Figure 3. Minimax value of nodes in a game tree. The leaves are evaluated by some intermediate evaluation function. The leaves whose values are enclosed in circles are the only ones that need to be evaluated by an alpha-beta algorithm.

value 1 at a player's move is one such property. However, to check this property, one has to search the whole game tree. In many special games, however, these properties can be calculated in terms of the board configuration in the node itself or just of a few nearby nodes.

A standard example is the following simple take-away game played in elementary school mathematics classes. There is a pile of sticks on the table. Each player in his turn removes at least one and no more than three sticks from the pile. The first player who cannot move (ie, faces an empty table) loses. It is clear that a player can win if he is on move and there are three or less (ie, less than four) sticks on the table. Such positions can be considered leaf nodes with value 1. If there are four sticks on the table, the player on move has to leave less than four and at least one stick. Hence the node with four sticks is a losing node. Induction readily shows that any node with a multiple of four sticks is losing and the rest are winning.

There is thus no need to do a minimax calculation when playing this game: if the player on the move does not have a multiple of four sticks on the table, he or she can reduce the number of sticks to a multiple of four, and the opponent cannot but return him or her to a node where the number of sticks is not a multiple of four.

Kernels

There are games where the winning nodes and winning strategies can be identified without search. The class of such games is not a trivial one: various rather complicated (but efficient) techniques are known for calculation of the values of nodes in such games. See Banerji (1980) and Berlekamp and co-workers, 1982 for many examples. In what follows, the technical term *kernel* calculation means the calculation of winning and losing nodes. (In precise terms, the losing nodes are said to form the kernel of the game graph.)

Such techniques, often called knowledge-based techniques in AI, have also been developed (albeit with lesser success and precision) for chess, yielding methods independent of minimax. See Pitrat (1977) and Bratko (1984) for examples.

The minimax technique "beats" this class of methods in one way, of course. Minimax works for all games. These game-specific methods work only when someone is clever enough to describe the winning nodes. What one needs is some method whereby the computer, given the description of the game, can develop the description of the set of winning nodes by itself. Samuel's checkers-playing program (qv) (1959) succeeded in doing this to an extent, and other programs have achieved similar successes (see below).

There are times when one can approximate the calculation of winning nodes. The method for one such calculation was developed by Koffman (1968) and by Citrenbaum (1972) for another wide class of games which, for want of any better name, is called *positional*. These include such trivial games as tic-tac-toe and more difficult games like Hex and Go-Moku. The game $4 \times 4 \times 4$ tic-tac-toe ("Qubic," as it is often called) is another nontrivial member of this class.

The major strategy for this class of games is the formation of forks. Indeed, in some sense any winning node in any game is a fork, but in the positional class of games the fork has a clear visual significance that can be very efficiently represented in a computer. The most elementary fork, of course, is when one encounters a board position like that in Figure 4. There are two lines of squares each of which has two empty squares, one of which is common to the two lines. In his turn, X can play at this common intersection and produce two potentially winning lines, and the opponent cannot block both.

The concept of a force can be pushed back much further; 14 or 15 move deep forces of this nature can be recognized on a go-moku board just by the configuration of pieces alone (1967). Without going into the depths of the discussion of the data representation needed to do this (Banerji, 1980), the reader is asked to convince himself that the empty board in $3 \times 3 \times 3$ tic-tac-toe is a forcing configuration: the player on move can assure himself of a win just by playing at the center.

The trouble with the Koffman–Citrenbaum technique (1968, 1972) for recognizing forces was that some deep forces could be upset if one of the defensive moves of the opponent posed a direct threat to the attacking player. The calculation of the winning nodes is thus only approximate in their method. A node recognized to be only three moves away from a win may be further away or even not be a winning move. Thus, the calculation has to be backed up by some search (albeit not a minimax search) of the game tree.

Evaluations

Many approximate measures of the "goodness" of a position (with "goodness" not necessarily defined in terms of kernels as above) have been suggested for various games. In most cases this has been done with the purpose of simplifying the minimax search for games. The arguments that have led to such efforts at simplification are as follows.

Consider the case where there is a method for finding whether a node is winning just by looking at the board configuration. In such a case, if a node is given the value 1 if it is winning and 0 if it is losing, this "static" value will be the same as its minimax value. On the other hand, if nothing is known as to whether a node is winning or los-

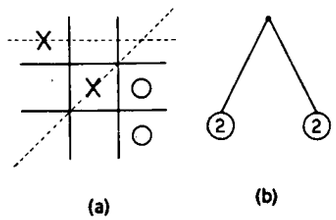

Figure 4. *(a)* Simple fork in tic-tac-toe. The two lines shown dotted each have two empty cells, and they intersect in an empty square where the forcing move is to be made. Graph *(b)* expresses the same fact. The two circles stand for the two lines (the numbers indicating the number of empty cells). The empty intersection is the solid node. Graph *(b)* describes not just the figure *(a)* but many other strategically equivalent positions.

ing, a minimax search done to the end of the game also leads to the determination as to whether a node is winning. This seems to indicate that if one has a very good evaluation for a node, very shallow minimax is needed to determine whether a node is winning; conversely, if there is a very bad evaluation available for a node (eg, no evaluation at all), a very deep minimax is needed. One can make a very unsophisticated interpolation from this that seems to indicate that it is possible to get a good idea as to whether a node is winning even if there is an imperfect evaluation function but the minimax on it is deep enough.

This has led to some extensive experimentation with game-playing programs along the following lines. Given a game to play with a computer, the researcher uses his own intuition and the literature on the specific game to decide a method for attaching a value to the nodes in such a way the most winning nodes have a higher value than most losing nodes. One then chooses the "best" move at a node as follows: one considers all the moves one can make and then all the moves an opponent can make from each of the resulting nodes. A part of the game tree is produced this way by alternating these "expansions" of nodes to a certain depth. The leaves of the resulting tree are then evaluated by calculating the static evaluation function. These values are then propagated "up the tree" by minimax to obtain the best move at the node.

Such evaluations have been found useful in constructing game-playing programs in some cases, although there is very little known as to why this is so (see below). However, the general experience has been that the technique is useful only when the depth of the tree is chosen carefully. It may even be that certain parts of the search tree should be explored deeper than some other parts. For a thorough discussion of the strengths and weaknesses of such procedures, see Berliner (1978), Jackson (1974), and Rich (1983). Only one such technique is given here.

This technique deals with the concept of "stability" of a position and is relevant to games like chess and checkers. The basic idea is that one should not evaluate a board position in the middle of a major skirmish, where pieces are being exchanged by a sequence of kills. In a part of the game tree where this is happening, it is better to explore the tree until the end of the skirmish, so the evaluation function does not oscillate violently. At the resulting position, the evaluation can be expected to be stable.

This technique is not foolproof since an apparently stable position can be hiding a threat that can be pushed back due to irrelevant but powerful moves demanding answers, so the threat (on the part of either player) can remain invisible. For a discussion of this so-called *horizon effect,* see Berliner (1978) (see also HORIZON EFFECT).

Many of the game-playing programs augment the minimax evaluation with a static evaluation of moves with respect to their relevance to a position. For instance, one may not want to consider moving a rook's pawn while one is busily engaged building up one's forces at the center of the board. Pruning away the branches of the game tree makes the tree grow at a smaller rate, so that greater depths can be explored.

Three distinct heuristics (qv) are being suggested at this point: one for choosing the intermediate evaluation, one for choosing the depth at which the evaluation is to be made, and one for pruning the search. How the accuracies of the three different heuristics co-operate to increase the accuracy of the final evaluation, indeed what one means by "accuracy" in the case of depth limitation, is not known. All one knows is that if the pruning heuristic is exact, it alone suffices to determine a winning strategy.

Meanwhile, there is always an effort to make the minimax search as deep as resources permit, the conventional wisdom being the argument given a few paragraphs earlier. We shall have occasion for discussing this conventional wisdom again later. However, we should mention another game-independent technique that allows one to increase the depth of search by avoiding the evaluation of every "leaf" at the end of the specified depth of search. This method, known as the alpha-beta search in the literature (Rich, 1983; Knuth and Moore, 1975) was informally suggested by Simon, Newell, and Shaw (1958) and placed on a formal basis by McCarthy (see ALPHA-BETA PRUNING). In Figure 3 the circled leaves of the tree are the only ones that would be evaluated by an alpha-beta procedure. In recent years a number of improvements on the alpha-beta pruning technique have been suggested (Baudet, 1978; Stockman, 1979; Pearl, 1980).

The popular line of attack on games has been on improving intermediate static evaluation of positions and efforts made at deepening the minimax by the use of some ad hoc pruning strategies and by some formal pruning techniques like alpha-beta. It has not been terribly popular to ask questions like, "Why is it better to minimax on static evaluations than to use the evaluation directly in choosing a move?" or "How can one judge the effectiveness of a static evaluation?" Nevertheless, efforts have been made to face these questions.

A DISTURBING RESULT: THE IMPORTANCE OF EVALUATION AND LEARNING

Results obtained by Nau (1983a) seem to confirm (as hinted by all that has gone above) that success at game playing is guaranteed only if one can find automatic methods for calculating kernels (ie, finding good static evaluation). Efforts at finding more efficient methods of minimaxing may be the wrong approach to writing strong game-playing programs.

Nau's result can be paraphrased by saying that if one has an imperfect evaluation function, in a large class of games the quality of the evaluation function deteriorates rather than improves with minimaxing, and its use leads to lower probability of winning than would have if the evaluation function was used directly.

Since the result is counterintuitive and since many game-playing programs (especially in the case of chess) seem to yield greater strength the deeper the minimaxing goes, there is need to understand what Nau's class of games consist of and why chess does not seem to belong to this class. A number of efforts (Nau, 1983b; Pearl, 1980) have shed light on this question.

In those cases where Nau's results apply, however, it seems futile to use approximate evaluation functions and

deep search: the only hope seems to be in the use of exact determination of the kernel. Unfortunately, no one knows any general method to find them. The fact that intuitive methods are inadequate is already obvious. The case where mathematically solid methods have succeeded, on the other hand, have not been very general or exciting. Also, the realization has remained that these methods were by themselves the product of hard and sustained human analysis. The automation of such analysis, then, becomes the real challenge. No easy shortcut around this challenge is available, as any consideration of past history would show.

Another difficult question that arises is one on the nature of approximate strategies. After all, human game playing has always been based on such approximations. Nau's results seem to indicate that minimaxing does not guarantee improved approximations. It is not known, given an approximate (in some sense) evaluation, to what extent games played on their basis approximate wins. More precisely, if an evaluation function predicts winning positions with 80% correctness, would a person using a strategy based on it win 80% of the games or would he win 10% of the games?

Pearl (Nau, 1983b) has suggested that perhaps one can skirt the question raised by Nau's results by bypassing minimaxing as a method of move choice. Instead, he suggests that one consider the evaluation as a measure of the probability of a win. So if $p_1, p_2, \ldots$ are the probabilities of win of the nodes reachable from a given node, the probability of win of the node itself, instead of being the maximum of $p_1, p_2, \ldots$ would be

$$1 - (1 - p_1)(1 - p_2), \ldots$$

The consequence of this suggestion on the previous questions apparently has not been analyzed.

Meanwhile, in relation to the search for better evaluation functions, there have been some successful experimentations done with the automatic development of evaluation functions.

Learning

So far as present intuition goes, there are two possible ways one can go about developing kernel descriptions or evaluations on the basis of the description of a game. One can use deduction. One can, if one knows that in go-moku one can win by making five in a row, figure out that two simultaneous four in a rows would be impossible to beat. Such deductive techniques have not been tried in the field so far. Alternatively, one could develop the description of good strategies by making generalization from experience. A technique that was tried in the very early days of AI was for the computer to "remember" the positions that lead to a win for either player. In so far as such a large set can be remembered, this is a perfectly reasonable way of going about the business. Good performance can be obtained if one can develop an encoding method that would enable the storing of a large list of positions. The method becomes inadequate against strong players, however, even in such comparatively simple games like three-dimensional tic-tac-toe.

What is needed here is not a method of encoding individual nodes so that they can be listed and accessed easily, but a method by which one can describe sets of nodes in implicit form, that is, be developing descriptions for them in some language. The importance of language to facilitate description has been discussed at some depth in pattern recognition (qv) literature (occasionally under the presently popular term "learning") (Banerji, 1980; Mitchell, 1983). Learning (qv) techniques of various kinds are also known. Two techniques by which strategies have been learned on the basis of game experience are described below.

It may be significant that in both these techniques the basis for the learning has not been pure experience but a reliance on the formal definition of a kernel. It will be recalled that one of the important properties of a winning position is that its minimax value and its static value are the same. The basis of Samuel's checker-playing program was a learning technique where the description was modified any time the minimax and the static value were not close enough. The description language chosen was one that gained great popularity at the time through the independently described perceptron (qv) (Rosenblatt, 1959; Minsky and Papert, 1969).

Given a node in checkers, certain measurements were made on the board configuration to yield numerical values for such concepts as mobility, center control, material balance, and so on. A linear combination of these numbers was taken as the value of the node. The learning done by Samuel's program consisted of modifying the coefficients of this linear combination so that the static and minimax values of nodes came to be close to one another. The evaluation so obtained would satisfy one property of an evaluation function leading to a kernel. The other property, that is, that the value should be high for winning positions and low for losing positions, was not confirmed. Also, there was no assurance that the technique used for modifying the coefficients would lead to convergence; as a matter of fact, there was a good bit of oscillation in the values found. Also, it was not clear that the kernel would be a linearly separable function of the measurements performed.

The fact remains, however, that the program played checkers very well after a period of learning. Efforts were later made (Samuel, 1985) to improve the learning performance by the use of a technique that had been used previously in the field of pattern recognition under the name of learning logical descriptions (Banerji, 1985). This technique has been discussed quite a bit in the literature in recent times (Valiant, 1984). It seems that the expressive ability of the descriptions learned is somewhat easy to control: one can trade expressive power for efficiency of learning. However, if the basic measurements used in constructing descriptions is well suited, both efficiency and expressive power can be obtained.

The effect of a good choice of language was demonstrated very well in the late sixties in the work of Koffman (1968), whose program learned approximations to the kernels of positional games in stages. Also, the nature of the approximation was clearly understood: any winning node would satisfy the approximate description, but not all nodes satisfying the approximate description would be a winning node. However, the reason for this discrepancy

was well understood so that once a node satisfied the approximate description, one could determine with a very limited search of the game tree as to whether the node was winning.

The nature of the descriptions has been indicated above in connection with positional games. The basic measurements of the language consisted of looking at the winning paths on the board (eg, row, columns, and diagonals at every plane of Qubic) and noting which of these were unobstructed by the opponent and among these the number of empty squares on each and which of the paths had empty intersections. Figure 5 indicates a seven deep force in the language and two of its interpretations on a plane of the Qubic board. It will be noticed that the two positions cannot be obtained one from the other by any symmetry of the board. The basic measurements have yielded a language of considerable power.

The design of the language in the sixties could not be automatic. The problem of learning, whether in games or any other activity, lies with discovering the basic measurements. Until very recently, no method was known for the automatic discovery for such measurements. Some recent work on problem solving (qv) (Ernst and Goldstein, 1982; Mitchell, 1983) has thrown some light on learning. The following were developed in the study of problem-solving: a class of nodes exists on problem graphs that has a clear analogy with the winning nodes of game graphs. Languages have been automatically developed for writing easy descriptions for these nodes. A program developed by Ernst and Goldstein (1982) has been effective also in discovering the similarity between a given game and games with known winning strategies. The interested reader is referred to the literature on problem solving and learning for details.

SUMMARY

In this article, a number of concepts that are of importance in research on game-playing programs have been elucidated. Concepts of game graphs and game trees have been introduced as well as the idea of evaluating a position by complete search of game trees. Due to the prohibitive amount of computation involved in such evaluation, one is forced to introduce the idea of shallow search and intermediate evaluations. Precise discussions have been included to explain when such an evaluation can be considered useful. The difficulties in the way of improving a bad evaluation have been indicated. Programs have been described that in the past could develop such evaluations from experience. These learning have been heavily dependent on the quality of the language used in these evaluations. Recent work on automatic modification of languages by definition have been mentioned.

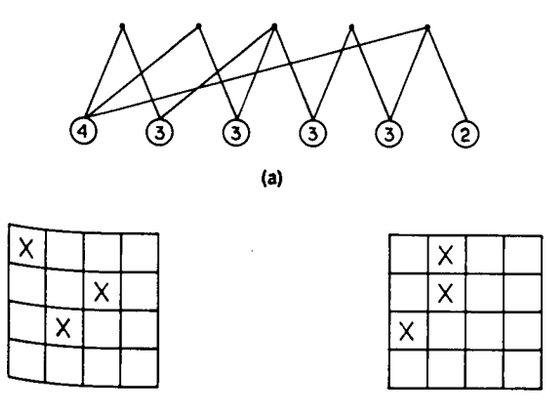

Figure 5. *(a)* Description of a five deep force in positional games; the language is identical to the one used in Figure 4a. *(b)* Two planes on a qubic (three-dimensional tic-tac-toe) board which obey the description. The reader is encouraged to work through the force. Notice that the two positions are not symmetrical with one another.

BIBLIOGRAPHY

R. B. Banerji, *Artificial Intelligence: A Theoretical Approach,* North Holland, Amsterdam, 1980.

R. B. Banerji, *The Logic of Learning: A Basis for Pattern Recognition and Improvement of Performance,* Progress in Computers, No. 24, Academic Press, New York, 1985.

G. M. Baudet, "On the Branching Factor of the Alpha-Beta Pruning Algorithm," *Artif. Intell.* **10,** 173 (1978).

H. J. Berliner, A Chronology of Computer Chess and its Literature, *Artif. Intell.* **10,** 201 (1978).

E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways,* Academic Press, New York, 1982.

M. A. Bramer, *Computer Game-Playing: Theory and Practice,* Ellis Horwood Series, Wiley, New York, 1983.

I. Bratko, "Advice and Planning in Chess Endgames," in A. Elithorn and R. B. Banerji, eds., *Artificial & Human Intelligence,* North Holland, Amsterdam, 1984.

R. L. Citrenbaum, "Strategic Pattern Generation: A Solution Technique for a Class of Games," *Patt. Recog.* **4,** 317 (1972).

J. H. Conway, *On Numbers and Games,* Academic Press, New York, 1976.

J. H. Conway, "All Games Bright and Beautiful," *Am. Math. Mon.* **84,** 417 (1977).

E. W. Elcock and A. M. Murray, "Experiments With a Learning Component in a Go-Moku Playing Program," in *Machine Intelligence,* Vol. 1, Oliver & Boyd, Edinburgh, UK, 1967.

G. W. Ernst and M. Goldstein, "Mechanical Discovery of Classes of Problem Solving Strategies," *J. Assoc. Comp. Mach.* **29** (1982).

P. Frey, ed., *Chess Skill in Man and Machine,* Springer-Verlag, 1977.

P. C. Jackson, *Introduction to Artificial Intelligence,* Petrocelli, Princeton, N.J., 1974.

D. E. Knuth and R. W. Moore, "An Analysis of Alpha-Beta Pruning," *Artif. Intell.* **6,** 293 (1975).

E. B. Koffman, "Learning Through Pattern Recognition Applied to a Class of Games," *IEEE Trans. Sys. Sci. Cybern.* **SSC-4,** March 1968.

M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry,* MIT Press, Cambridge, Mass., 1969.

T. M. Mitchell, "Learning and Problem Solving," *Proceedings of the International Joint Conference on Artificial Intelligence,* Karlsruhe, FRG, 1983, p. 1139.

D. S. Nau, "Decision Quality as a Function of Search Depth on Game Trees," *J. Assoc. Comp. Mach.* **30,** 687 (1983).

D. S. Nau, "Pathology on Game Trees Revisited, and an Alternative to Min-Maxing," *Artif. Intell.* **21,** 222 (1983).

A. Newell, J. C. Shaw, and H. A. Simon, "Chess Programs and the Problem of Complexity," *IBM J. Res. Dev.* **2,** 320 (1958).

J. Pearl, "Asymptotic Properties of Minimax Trees and Game Searching Procedures," *Artif. Intell.* **14,** 113 (1980).

J. Pitrat, "A Chess Combination Program which Uses Plans," *Artif. Intell.* **8,** 275 (1977).

E. Rich, *Artificial Intelligence,* McGraw-Hill, New York, 1983.

F. Rosenblatt, "Two Theorems on Statistical Separability in the Perceptron," *Proceedings of the Symposium on the Mechanization of Thought Processes,* Her Majesty's Stationery Office, London, 1959.

A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM J. Res. Devel.* **3,** 210 (1959).

M. Samuel, "Some Studies in Machine Learning Using the Game of Checkers II," *IBM J. Res. Dev.* **11,** 601 (1985).

G. A. Stockman, "A Minmax Algorithm Better than the Alpha-Beta?" *Artif. Intell.* **12,** 179 (1979).

L. J. Stockmeyer and A. K. Chandra, "Intrinsically Difficult Games," *Scientif. Am.* **240**(5), 140 (1979).

L. G. Valiant, "A Theory of the Learnable," *Proceedings of the Sixteenth Annual Symposium on Theory of Computing,* Washington, D.C., 1984, p. 436.

J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior,* Princeton University Press, Princeton, N.J., 1944.

R. BANERJI
St. Joseph's University

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

This extensively revised and expanded *Second Edition* of the *Encyclopedia of Artificial Intelligence* defines the discipline by bringing together the core of knowledge from all fields encompassed by AI. It covers the latest developments in current AI topics such as neural networks, fuzzy logic, machine vision, natural language generation, and many more. Includes:

- Over 450 articles—all entries written expressly for the *Encyclopedia*
- Over 5,000 literature references; 454 illustrations and color photographs
- Over 50% new and revised material
- Exemplary indexing and cross-referencing for easy, complete information access to all topics

Praise for the *First Edition*...

"The *Encyclopedia* is a wonder of clarity and scope: surprisingly easy to read...the clarity is an especially pleasant surprise, considering the articles were all written by AI experts...It's a treasure house of easily accessible knowledge."
—*Language Technology*

"Excellent bibliographies are attached to most of the articles, and diagrams and sketches are clear and helpful. The indexing and cross-indexing are exemplary. As the editor points out, the reader will be led by the extensive cross-references to almost every other article..." —*Artificial Intelligence Reporter*

"The *Encyclopedia* is a first-class piece of work that will be an indispensable part of any AI library." —*Computing Reviews*

"...A tour de force...a truly fantastic encyclopedia which no one in the field of artificial intelligence can afford to be without."
—*Systems Research & Information*

WILEY-INTERSCIENCE
John Wiley & Sons, Inc.
Professional, Reference and Trade Group
605 Third Avenue, New York, N.Y. 10158-0012
New York • Chichester • Brisbane • Toronto • Singapore
ISBN 0 471-50307-X (Two-Volume Set)
ISBN 0 471-50305-3 (Vol. 1)
ISBN 0 471-50306-1 (Vol. 2)

ISBN 0-471-50305-3

9 780471 503057 90000