International Journal of Computer and Information Sciences, Vol. 12, No. 6, 1983

On Game Graph Structure and Its Influence on Pathology[1]

Dana S. Nau[2]

Received February 1983; revised September 1983

Almost all game tree search procedures used in Artificial Intelligence are variants on minimaxing. Until recently, it was almost universally believed that searching deeper on game trees with such procedures would in general yield a better decision. However, recent investigations show that there are many "pathological" game trees for which searching deeper consistently *degrades* the decision.

This paper investigates one possible cause of pathology. In particular, a class of games that is normally pathological is shown to become nonpathological when the games are modified so that game positions can be reached by more than one path. This result suggests that in general, pathology is less likely when game positions can be reached by more than one path. This may be one reason why games such as chess and checkers are nonpathological. In addition, this result supports the hypothesis[9] that pathology is less likely when sibling nodes have similar minimax values.

This paper also investigates a possible cure for pathology—an alternative to minimaxing called probability estimation which has been shown to avoid pathology and thus produce more accurate decisions than minimaxing on at least one pathological game.[11] The current paper shows that depending on what evaluation function is used, probability estimation can also produce more accurate decisions than minimaxing on at least one nonpathological game. Probability estimation or other related procedures could conceivably become attractive alternatives to minimaxing if suitable tree pruning procedures could be developed for use with them.

KEY WORDS: Artificial intelligence; decision analysis; decision trees; games; game trees; minimaxing; pathology; problem solving; search.

[1] This work was supported by NSF Grant MCS-8117391 to the Laboratory for Machine Intelligence and Pattern Analysis at the University of Maryland.

[2] Computer Science Department, University of Maryland, College Park, Maryland 20742.

0091-7036/83/1200-0367$03.00/0

828/12/6-1

1. INTRODUCTION

Almost all game tree search procedures used in Artificial Intelligence are variants on the following process: the tree is searched to some arbitrary depth, a static evaluation function is used to compute approximations of the utility values of the nodes at that depth, and minimaxing is used to compute approximations of the utility values of shallower nodes.

Until recently, there was almost universal agreement that increasing the depth of the search would improve the quality of the decision. This property was dramatically illustrated in games such as chess and checkers using game playing computer programs.[4,19,21] However, recent investigations by Nau[7,8,13] demonstrated the existence of many game trees that are "pathological" in the following sense: as long as the search does not reach the end of the game tree (in which case a correct decision can be guaranteed), searching deeper consistently degrades the quality of the decision.

Pathology hs been further investigated by Beal,[2] Bratko and Gams,[5] Pearl,[17] and Nau.[9,11] The current paper extends our knowledge of pathology in two directions, as described in sections 1.1 and 1.2.

1.1. Underlying Causes of Pathology

Since the discovery of game tree pathology in 1979,[7] a major open question has been why it occurs in some games and not others. In particular, why does it not occur in games such as chess and checkers? One possible reason is the following hypothesis:

Hypothesis 1. In games such as chess and checkers, moves consist of small incremental modifications to a playing board. In a strong (or weak) position, most of the available moves are likely to lead to strong (or weak) positions; and thus the relative strength of a node in the game tree depends on the strength of its parent. Perhaps this property precludes pathology.

In a paper investigating this hypothesis,[9] pathology was shown to occur in a class of games (which we call *P*-games) in which the values of nodes depended only slightly on the values of their parents. When the games were modified to increase the amount of this kind of dependency, the resulting class of games (which we call *N*-games) was not pathological. Thus the hypothesis holds in at least one case.

A subsequent paper[11] contained theorems and statistical studies supporting the following hypothesis:

Hypothesis 2. Pathology will occur when sibling nodes in a game tree have relatively independent values, and the reason why Hypothesis I

holds is that the incremental behavior it describes causes the values of sibling nodes to be highly correlated.

If Hypothesis 2 is indeed correct, pathology ought to be unlikely under *any* conditions that cause the values of sibling nodes to be closely correlated; and incremental variations in node strength are not the only way this can occur. For example, if sibling nodes have several children in common (so that the game "tree" is actually a game *graph*), then they will have highly correlated values. To verify Hypothesis 2, it is important to investigate whether or not pathology occurs under such conditions. The current paper does this by modifying the class of *P*-games (which are known to be pathological) in such a way that sibling nodes have many children in common. The resulting class of games (which we call *G*-games) is shown to be nonpathological.

1.2. An Alternative to Minimaxing

Pearl[17] has suggested that pathology might be avoided by using an evaluation function which returns the probability that a node is a forced win, and replacing the minimax decision procedure by a procedure which treats the evaluation function values as independent probabilities. The first investigation of this proposed approach was done by Nau,[11] who modified Pearl's suggestion by using an evaluation function which returned values between 0 and 1 to approximate the probability that a node is a forced win. On *N*-games the probability of choosing a correct move using this "probability estimation" approach was almost exactly the same as it was when minimaxing was used. On *P*-games, probability estimation avoided pathology and usually gave a higher probability of correct decision than minimaxing.[3]

The current paper compares the quality of the decisions produced by probability estimation on *G*-games to the quality of decisions produced by a minimax search to the same depth, using two different evaluation functions. For one of the evaluation functions, probability estimation outperforms minimaxing, and for the other one, minimaxing outperforms probability estimation. This provides a second known case in which probability estimation outperforms minimaxing, and other approaches are currently being investigated which may do even better in many situations.[1,12,18] If tree pruning strategies similar to alpha-beta or SSS^* can be found for any of

[3] The paper also compared probability estimation to minimaxing in terms of the number of games each could win over the other. Although the results of this experiment were inconclusive, further studies[12] have shown probability estimation to win significantly more *P*-games than minimaxing does.

these decision strategies, they might become attractive alternatives to minimaxing for certain applications.

1.3. Outline

Section 2 of this paper contains some preliminary definitions. Section 3 describes *P*-games and *G*-games, and the evaluation functions we use for these games. Section 4 investigates how the probability of choosing a correct move varies with search depth on *G*-games when minimaxing is used, and compares the results of this investigation with the corresponding results[9] for *P*-games. Section 5 compares probability estimation with minimaxing on *G*-games, and Section 6 contains concluding remarks.

2. PRELIMINARY DEFINITIONS

By a *game* we mean a zero sum, perfect information game between two players. The play must alternate strictly between them, and at each game position there may be only finite many possible moves among which to choose.

Let G be the game graph for such a game. Each node of G corresponds to a *game position*, which consists both of what the game board looks like and who is to move. Each arc of G corresponds to a move in the game. The root node, root(G), corresponds to the game's beginning, and each leaf node (node with no children) corresponds to one possible way the game might end. Associated with each leaf node are the payoffs the two players receive for that particular ending to the game.

By depth(g) we mean the length of the shortest path from root(G) to the node g, (i.e., the least number of moves it takes to get to the game position(g), and by depth(G) we mean

$$\max\{\text{depth}(g) \mid g \text{ is a node of } G\}.$$

By the subgraph rooted at g we mean the subgraph of G whose nodes are g and all of its descendants. The depth of this subgraph we call height(g).

If g is a node in G, the utility value $u(g)$ is the payoff which the player who moves to g would receive if both players played perfectly from g on. Since G is a zero sum game, the payoff for the player's opponent would be $-u(g)$. Utility values may be computed using the following "negamax" formula[6]:

$$\begin{aligned} u(g) &= \text{the payoff for the player who moves to } g \text{ if } g \text{ is a leaf} \\ &= -\max\{u(h) \mid h \text{ is a child of } g\} \text{ otherwise.} \end{aligned} \tag{1}$$

An evaluation function for G is any real-valued function $e(g)$ intended to return an approximation of $u(g)$. Ideally, $e(g)$ would return exactly $u(g)$, but evaluation functions are usually somewhat (and sometimes drastically) in error. For example, evaluation functions for chess are notoriously inaccurate in the endgame.[22]

Other approximations to $u(g)$ may be computed by computing $e(h)$ for all nodes h of some fixed depth d in the subgraph rooted at g and putting these approximate utility values into the negamax formula to compute values for the shallower nodes of the subgraph. The value computed for g in this way, which we call the depth d minimax value of g, is

$$m(d, g) = e(g) \qquad \text{if } d = 0$$

or

$$= -\max\{m(d-1, h) \mid h \text{ is a child of } g\} \tag{2}$$

otherwise. Pruning procedures such as alpha-beta,[6,14] SCOUT,[15] SSS*,[20] and B*,[3] have been developed to speed the computation of $m(d, g)$.

A common way to choose a move at a node g is to choose whichever child h of g has the highest depth $d-1$ minimax value $m(d-1, h)$. If more than one child of g has this value, the choice is made at random among all children of g having this value. This is called a depth d minimax search, since it involves evaluating the nodes at depth d in the subgraph of G rooted at g.

3. *P*-GAMES AND *G*-GAMES

This section describes the two classes of games (P-games and N-games) that are compared in this paper, and the evaluation functions used for these games. Since P-games have been amply described elsewhere,[9,11] their description here is rather cursory.

A P-game is played between two players. The playing board for the game is a list of 2^N elements (we use $N = 10$). Each element is either -1 or 1. The value of each element is determined before the beginning of the game by making it a 1 with some fixed probability p and a -1 otherwise, independent of the values of the other elements. We use $p = (3 - \sqrt{5})/2 \approx 0.382$, which results in each side having about the same chance of winning.[10]

To make a move in the game, the first player removes either the left half of the list (the first 2^{N-1} elements) or the right half (the last 2^{N-1} elements). His opponent then removes the left or right half of the remaining part of the list. (The rules can be generalized for branching factors greater than 2, but we are concerned only with the binary case.) Play continues in this manner

with each player selecting the left or right half of the remaining part of the list until a single element remains. If this element is a 1, then the player who made the last move wins; otherwise his opponent wins.

The game tree for a *P*-game is a full binary game tree of depth k. Thus the same player always has the last move no matter what course the game takes. We call this player Max and his opponent Min.

In games such as chess and checkers the game graph is not a tree, since several different nodes may have some of the same children. The *G*-games also have this property.

The playing board for a *G*-game is a list of $k+1$ elements, where $k > 0$ is an integer. The playing board is set up by randomly assigning each element the value 1 with probability r or the value -1 otherwise, for some fixed r (we use $r = 1/2$). A more (for either player) consists of removing a single element from either end of the list (see Fig. 1). As with the *P*-games, the game ends when only one element is left. If it is a 1, then Max (the player who moved last) wins; otherwise Min wins.

In both *P*-games and *G*-games, the only possible payoffs for a player are 1 (or "win") and -1 (or "loss"). Thus it is easy to prove by induction that every node g in a *P*-game or *G*-game either has $u(g) = 1$ (in which case g is a *forced win* node), or $u(g) = -1$ (in which case g is a *forced loss* node).

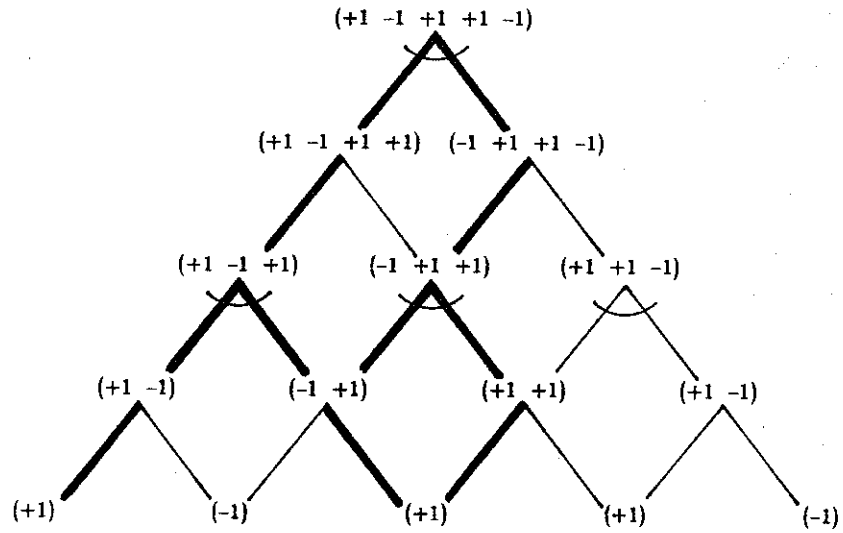

Fig. 1. A game graph for a *G*-game of depth 4. The initial playing board appears at the root of the graph. Since the depth is even, Max is the second player. Max has a forced win in this particular game graph, as indicated by the solution graph drawn in boldface.

3.1. Evaluation Functions

Let G be a game tree for a b-ary P-game or G-game, and g be a node in G. Note that the number of elements in g is $2^{\text{height}(g)}$ if G is a P-game, and $\text{height}(g)+1$ if G is a G-game. If Max has the move to g, then the more "1" elements there are in g the more likely it is that g is a forced win. If Min has the move to g, then the more "−1" elements there are in g the more likely it is that g is a forced win. Thus an obvious evaluation function for G is

$$e_1(g) = \frac{\text{the number of elements in } g \text{ having value } v(g)}{\text{the number of elements in } g}$$

where

$$v(g) = \begin{cases} 1 & \text{if Max has the move to } g \\ -1 & \text{otherwise} \end{cases}$$

As will be shown in Section 4, in G-games $u(g)$ is heavily influenced by the values of the two or three elements at the center of g. Thus a more accurate evaluation function for G-games can be created by defining

$$e_2(g) = \frac{1}{2^n} \sum_{i=0}^{n} \binom{n}{i} (1 \text{ if } x_i = v(g), \quad \text{else } 0),$$

where $x_0, x_1, \ldots, x_n$ are the elements of g. e_2 is considerably more accurate on G-games than e_1, because it gives considerably more weight to the elements near the center of the list than it does to the elements near either end of the list.

4. PROBABILITIES OF CORRECT DECISION

Let g be a node of height k having one forced win child g_1 and one forced loss child g_2, and suppose that a depth d minimax search is used to choose between g_2 and g_2. Then the correct decision is to move to g_1. Now, g_1 will always be chosen if $m(d-1, g_1) > m(d-1, g_2)$, and g_1 will be chosen half of the time if $m(d-1, g_1) = m(d-1, g_2)$. Thus the correct decision will be made with probability

$$\begin{aligned} D_d = {} & \Pr[m(d-1,\ g_1) > m(d-1,\ g_2)] \\ & + \tfrac{1}{2} \Pr[m(d-1,\ g_1) = m(d-1,\ g_2)]. \end{aligned} \tag{3}$$

Suppose a depth d minimax search is done at a node g of height k in a P-game. The probability of correct decision for this situation has been deter-

mined mathematically.[9] In particular, when k is larger than about 7 or 8 pathology occurs, in the following sense: as long as the search does not go closer than one move away from the end of the game (i.e., as long as $d < k-1$), the probability of correct decision decreases as the search depth increases.

We now examine the probability of correct decision in G-games. If g is a node in a G-game, then we let $g[i, j]$ be the descendant of g formed by removing i elements from the left end of g and j elements from the right end of g. Note that

$$(g[i, j])[u, v] = g[i+u,\ j+v]$$

The following theorem states that regardless of the value of d, the depth d minimax value of g is always determined by a certain set of two or three nodes of depth d in the subtree rooted at g.

Theorem 1. Let g be a node of height k in a G-game. For $0 \leqslant d \leqslant k$,

$$m(d, g) = e(g) \text{ if } d = 0$$

$$= -\max\left\{e\left(g\left[\frac{d+1}{2}, \frac{d-1}{2}\right]\right),\ e\left(g\left[\frac{d-1}{2}, \frac{d+1}{2}\right]\right)\right\} \quad \text{if } d > 0 \text{ is odd}$$

$$= \max\left\{e\left(d\left[\frac{d}{2}, \frac{d}{2}\right]\right),\ \min\left\{e\left(g\left[\frac{d+2}{2}, \frac{d-2}{2}\right]\right),\ e\left(g\left[\frac{d-2}{2}, \frac{d+2}{2}\right]\right)\right\}\right\} \quad \text{if } d > 0 \text{ is even}$$

Proof. See the Appendix. □

Corollary 1. Let g be a node of height k in a G-game. Then for $0 < d \leqslant k$,

$$u(g) = -\max\left\{u\left(g\left[\frac{d+1}{2}, \frac{d-1}{2}\right]\right),\ u\left(g\left[\frac{d-1}{2}, \frac{d+1}{2}\right]\right)\right\} \quad \text{if } d > 0 \text{ is odd}$$

$$= \max\left\{i\left(g\left[\frac{d}{2}, \frac{d}{2}\right]\right),\ \min\left\{u\left(g\left[\frac{d+2}{2}, \frac{d-2}{2}\right]\right),\ u\left(g\left[\frac{d-2}{2}, \frac{d+2}{2}\right]\right)\right\}\right\} \quad \text{if } d > 0 \text{ is even}$$

Proof. Recall that $e(g)$ is an approximation to $u(g)$. Theorem 1 holds even when this approximation is exact; i.e., when $e(g) = u(g)$. □

Suppose we are searching to some depth d. If d is even, then from Eq. (3) and Theorem 1 we get

$$D_d = \Pr[-\max\{e(g_{d,1}), e(g_{d,2})\} > -\max\{e(g_{d,3}), e(g_{d,4})\}] + \tfrac{1}{2}\Pr[-\max\{e(g_{d,1}), e(g_{d,2})\} = -\max\{e(g_{d,3}), e(g_{d,4})\}] \quad (4)$$

where $g_{d,1} = g_1[d/2, d/2-1]$, $g_{d,2} = g_1[d/2-1, d/2]$, $g_{d,3} = g_2[d/2, d/2-1]$, and $g_{d,4} = g_2[d/2-1, d/2]$. But from Corollary 1 it follows that $u(g_{d,1}) = u(g_{d,2}) = -1$ and $u(g_{d,3}) = u(g_{d,4}) = 1$. Thus if i and j are even and e is more accurate at depth i than j in the subtree rooted at g, then

$$-\max\{e(g_{i,1}), e(g_{i,2})\} > -\max\{e(g_{j,1}), e(g_{j,2})\}$$

and

$$-\max\{e(g_{i,3}), e(g_{i,4})\} < -\max\{e(g_{j,3}), e(g_{j,4})\}$$

whence $D_i > D_j$. A similar conclusion can be obtained when i and j are odd.

The point of the above is that the quality of a decision in a G-game using minimaxing depends solely on the accuracy of the evaluation function. If the evaluation function is more accurate at large depths than small ones, then a deeper search will help; if it is less accurate at large depths, then a deeper search will hurt.

The evaluation functions $e_1(g)$ and $e_2(g)$ basically compute weighted averages of the values of all of the elements of g. But from Theorem 2, the utility value of g does not depend on all of these elements, but only on the two or three elements in the center of g. If the height of g is small, then there will not be many elements other than these two or three, and so $e_1(g)$ and $e_2(g)$ will be quite accurate. However, if the height of g is large, then there will be many elements other than the two or three relevant ones, so the two evaluation functions will not be so accurate. They become monotonically more accurate as the search depth increases, so pathology should not occur in G-games when these evaluation functions are used.

The above statement can be verified by direct measurement. For each k, the number of distinct G-game nodes of height k is exactly 2^{k+1} (because each node of height k contains $k+1$ elements, each of which may be a 1 or -1). For some of these nodes, either both children are forced losses or both children are forced wins—and at these nodes it does not matter what move is chosen. Each remaining node g has one forced win child g_1 and one forced loss child g_2. For these nodes we compute $m(d-1, g_1)$ and $m(d-1, g_2)$. If out of a set of n nodes g we have $m(d-1, g_1) > m(d-1, g_2)$ for r of them

Table I. Probability of correct decision D as a function of search depth d using minimaxing with the evaluation function e_1, for nodes of height k in binary G-games.[a]

	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
$d=1$	0.750	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
$d=2$	1.00	0.750	0.625	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
$d=3$	1.000	1.000	0.750	0.563	0.531	0.500	0.500	0.500	0.500	0.500	0.500
$d=4$		1.000	1.000	0.750	0.625	0.500	0.500	0.500	0.500	0.500	0.500
$d=5$			1.000	1.000	0.750	0.563	0.531	0.500	0.500	0.500	0.500
$d=6$				1.000	1.000	0.750	0.625	0.500	0.500	0.500	0.500
$d=7$					1.000	1.000	0.750	0.653	0.531	0.500	0.500
$d=8$						1.000	1.000	0.750	0.625	0.500	0.500
$d=9$							1.000	1.000	0.750	0.563	0.531
$d=10$								1.000	1.000	0.750	0.625
$d=11$									1.000	1.000	0.750
$d=12$										1.000	1.000
$d=13$											1.000

[a] The results come from examining every possible node of height k.

and $m(d-1, g_2)$ for s of them, then the probability of correct decision is $D_d = (r + s/2)/n$.

Tables I and II contain the results of such measurements using e_1 and e_2, respectively. As expected, pathology does not occur on any of the G-games tested.

Table II. Probability of correct decision D as a function of search depth d using minimaxing with the evaluation function e_2, for nodes of height k in binary G-games.[a]

	$k=3$	$k=4$	$k=5$	$k=6$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
$d=1$	1.000	1.000	1.000	0.969	0.938	0.875	0.902	0.844	0.883	0.812	0.834
$d=2$	1.000	1.000	1.000	0.969	0.938	0.875	0.883	0.848	0.871	0.814	0.838
$d=3$	1.000	1.000	1.000	1.000	0.969	0.953	0.906	0.875	0.881	0.843	0.857
$d=4$		1.000	1.000	1.000	1.000	0.969	0.938	0.875	0.883	0.848	0.871
$d=5$			1.000	1.000	1.000	1.000	0.969	0.953	0.906	0.875	0.881
$d=6$				1.000	1.000	1.000	1.000	0.969	0.928	0.875	0.883
$d=7$					1.000	1.000	1.000	1.000	0.969	0.953	0.906
$d=8$						1.000	1.000	1.000	1.000	0.969	0.938
$d=9$							1.000	1.000	1.000	1.000	0.969
$d=10$								1.000	1.000	1.000	1.000
$d=11$									1.000	1.000	1.000
$d=12$										1.000	1.000
$d=13$											1.000

[a] The results come from examining every possible node of height k.

5. THE PROBABILISTIC DECISION PROCEDURE

So far, we have only discussed minimax decision procedures. Pearl[16,17] has proposed the following alternative to minimaxing. Suppose we have a special evaluation function $e^*(g)$ which returns the probability that a node g is a forced win given some of its measurable features. Suppose further that the probabilities of sibling nodes being forced wins are always independent, and that we have evaluated $e^*(g_1)$ for every node g_1 at some depth d in the subtree rooted at g. Then the probability that g is a forced win can be computed by applying the formula

$$\Pr[g_1 \text{ is a win node}] = \prod \{1 - \Pr[g_2 \text{ is a win model}] \mid g_2 \text{ is a child of } g_1\} \quad (5)$$

to successively shallower nodes g_1 in the subtree.

This "probability estimation" procedure can be used even when $e(g)$ only approximates the actual probability that g is a forced win given the features examined. In such a case, it works as follows: use an evaluation function $e(g)$ which returns values between 0 and 1. Search the game tree to some depth d, evaluating the nodes at this depth. Instead of using minimaxing to compute values for the shallower nodes of the tree, compute values for these nodes as if the computed values for their children were

Table III. Probability of correct decision D as a function of search depth d using probability estimation with the evaluation function e_1, for nodes of height k in binary G-games.[a]

$d=1$	0.750	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
$d=2$	1.000	0.875	0.688	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
$d=3$	1.000	1.000	0.938	0.719	0.609	0.500	0.500	0.500	0.500	0.500	0.500
$d=4$		1.000	1.000	0.875	0.828	0.641	0.570	0.500	0.500	0.500	0.500
$d=5$			1.000	1.000	0.891	0.750	0.711	0.580	0.540	0.500	0.500
$d=6$				1.000	1.000	0.828	0.801	0.672	0.650	0.564	0.532
$d=7$					1.000	1.000	0.891	0.742	0.725	0.617	0.592
$d=8$						1.000	1.000	0.840	0.812	0.672	0.646
$d=9$							1.000	1.000	0.885	0.749	0.733
$d=10$								1.000	1.0000	0.805	0.802
$d=11$									1.000	1.000	0.836
$d=12$										1.000	1.000
$d=13$											1.000

[a] The results come from examining every possible node of height k.

independent probabilities of the occurrence of forced wins. This amounts to replacing the "negamax formula"

$$m(d, g) = \begin{cases} e(g) & \text{if } d = 0 \\ -\max\{m(d-1, g_1) \mid g_1 \text{ is a child of } g\} & \text{otherwise} \end{cases} \quad (6)$$

by the formula

$$p(d, g) = \begin{cases} e(g) & \text{if } d = 0 \\ \prod \{1 - p(d-1, g_1) \mid g_1 \text{ is a child of } g\} & \text{otherwise} \end{cases} \quad (7)$$

We call $p(d, g)$ the depth d probability estimate for g.

The probability of correct decision obtainable using probability estimation on G-games can be measured in the same way that it was measured for minimaxing. For each node of height k having one forced win child g_1 and one forced loss child g_2, we compute $p(d-1, g_1)$ and $p(d-1, g_2)$. If out of a set of n nodes g we have $p(d-1, g_1) > p(d-1, g_2)$ for r of them and $p(d-1, g_1) = p(d-1, g_2)$ for s of them, then the probability of correct decision is $D_d = (r + s/2)/n$. The results of measurements using the evaluation functions e_1 and e_2 are given in Tables III and IV, respectively.

Table IV. Probability of correct decision D as a function of search depth d using probability estimation with the evaluation function e_2, for nodes of height k in binary G-games.[a]

	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
$d=1$	1.000	1.000	1.000	0.969	0.938	0.875	0.902	0.844	0.883	0.812	0.834
$d=2$	1.000	1.000	1.000	0.969	0.938	0.875	0.902	0.844	0.883	0.812	0.834
$d=3$	1.000	1.000	1.000	1.000	0.953	0.898	0.902	0.844	0.883	0.813	0.834
$d=4$		1.000	1.000	1.000	0.969	0.906	0.906	0.844	0.883	0.813	0.835
$d=5$			1.000	1.000	0.969	0.906	0.906	0.844	0.883	0.813	0.835
$d=6$				1.000	1.000	0.922	0.906	0.844	0.891	0.813	0.825
$d=7$					1.000	1.000	0.906	0.844	0.887	0.813	0.837
$d=8$						1.000	1.000	0.844	0.889	0.816	0.837
$d=9$							1.000	1.000	0.989	0.813	0.842
$d=10$								1.000	1.000	0.832	0.845
$d=11$									1.000	1.000	0.861
$d=12$										1.000	1.000
$d=13$											1.000

[a] The results come from examining every possible node of height k.

Table V. Percentage difference $100\% \times (D_{III} - D_I)/D_{III}$, where D_I and D_{III} are the values of D given in Tables I and III, respectively.

	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
$d=1$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$d=2$	0.0%	16.7%	10.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$d=3$	0.0%	0.0%	25.1%	27.7%	14.7%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$d=4$		0.0%	0.0%	16.7%	32.5%	28.2%	14.0%	0.0%	0.0%	0.0%	0.0%
$d=5$			0.0%	0.0%	18.8%	33.2%	33.9%	16.0%	8.0%	0.0%	0.0%
$d=6$				0.0%	0.0%	10.4%	28.2%	34.4%	30.0%	12.8%	6.4%
$d=7$					0.0%	0.0%	18.8%	31.8%	36.5%	23.4%	18.4%
$d=8$						0.0%	0.0%	12.0%	29.9%	34.4%	29.2%
$d=9$							0.0%	0.0%	18.0%	33.0%	38.0%
$d=10$								0.0%	0.0%	7.3%	28.3%
$d=11$									0.0%	0.0%	11.5%
$d=12$										0.0%	0.0%
$d=13$											0.0%

Probability estimation and minimaxing can now be compared to each other by computing the relative differences between corresponding entries in Tables I and III and the relative differences between corresponding entries in Tables II and IV. This is done in Tables V and VI, respectively. As can be seen, probability estimation does better than minimaxing when e_1 is used, and worse than minimaxing when e_2 is used.

Table VI. Percentage difference $100\% \times (D_{III} - D_I)/D_{III}$, where D_I are D_{III} are the values of D given in Table I and III, respectively.

	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
$d=1$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$d=2$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	2.2%	−0.5%	1.4%	−0.2%	−0.5%
$d=3$	0.0%	0.0%	0.0%	0.0%	−1.7%	−5.8%	−0.4%	−3.5%	0.2%	−3.6%	−2.7%
$d=4$		0.0%	0.0%	0.0%	−3.1%	−6.5%	−3.4%	−3.5%	0.0%	−4.1%	−4.1%
$d=5$			0.0%	0.0%	−3.1%	−9.4%	−6.5%	−11.4%	−2.5%	−7.1%	−5.2%
$d=6$				0.0%	0.0%	−7.8%	−9.4%	−12.9%	−5.0%	−7.1%	−5.4%
$d=7$					0.0%	0.0%	−9.4%	−15.6%	−8.5%	−14.7%	−7.6%
$d=8$						0.0%	0.0%	−15.6%	−11.1%	−15.8%	−10.8%
$d=9$							0.0%	0.0%	−10.2%	−18.7%	−13.1%
$d=10$								0.0%	0.0%	−16.8%	−15.5%
$d=11$									0.0%	0.0%	−13.9%
$d=12$										0.0%	0.0%
$d=13$											0.0%

6. CONCLUSION

This paper has investigated some of the conditions responsible for the occurrence or absence of pathology in game trees, and has also examined an alternative to minimaxing. We now discuss the results of these investigations.

6.1. Conditions which Cause Pathology

Since the discovery of game tree pathology in 1979,[7] a major open question has been why it occurs in some games and not others.

It was previously hypothesized[11] that pathology should be less likely when the values of sibling nodes are closely correlated. This hypothesis was shown to hold for certain games[9,11] using one way of achieving correlation among sibling nodes: incremental variation in the strength of nodes as a game progresses. This kind of behavior occurs in games such as chess and checkers, and thus may be one explanation for why those games are not pathological.

If the above hypothesis is correct, then it should hold no matter how the correlation in values of sibling nodes is achieved. In this paper, we have shown that the hypothesis holds for some games using a different way of achieving correlation among sibling nodes: by giving them common descendants. Since games such as chess and checkers exhibit this kind of behavior too, this may be another reason why those games are not pathological.

It should be noted that none of these studies deal with what happens when a game tree search reaches a leaf node. Thus they do not conflict with Pearl's conjecture[17] that another reason for the absence of pathology in chess, checkers, and similar games is the occurrence of leaf nodes at all levels of the game tree. There may very well be a number of factors whose presence can preclude pathology.

6.2. An Alternative to Minimaxing

The other topic discussed in this paper is a game tree decision procedure called "probability estimation" which is distinctly different from minimaxing. This decision procedure was compared with minimaxing on G-games. On G-games, it performs better than minimaxing if the evaluation function e_1 is used and worse than minimaxing if e_2 is used.

Purdom[12] has pointed out that minimaxing is the best way to combine values at a node if those values are exact. The values obtained using e_2 are certainly not exact, but they are more accurate than those obtained using e_1.

This may explain why minimaxing did better in relation to probability estimation when e_2 was used than it did when e_1 was used.

This paper shows that if an appropriate evaluation function is used, probability estimation does significantly better than minimaxing on G-games. Probability estimation can do better than minimaxing on P-games as well,[11,12] and recent results show that there are also other decision procedures which outperform minimaxing under various conditions.[1,12,18] One large drawback of all of these procedures is that tree pruning procedures such as alpha-beta cannot be used with them. One or more of these approaches might become attractive alternatives to minimaxing if suitable tree pruning procedures can be found.

APPENDIX

Proof of Theorem 1 (by induction on d). If $d=0$, then $m(d, g)=e(g)$ by Eq. (2). Let $0 \leqslant i < k$, and suppose the theorem holds for $d = i$. We must show that it holds for $d = i + 1$. From Eq. (2),

$$m(i+1, g) = -\max\{m(i, g[1,0]), \quad m(i, g[0,1])\}$$

Case 1. If $i+1$ is even, then i is odd. So from the induction assumption,

$$\begin{aligned}
m(i+1, g) = -\max \Bigg\{ & -\max \left\{ e\left(g[1,0]\left[\frac{i+1}{2}, \frac{i-1}{2}\right]\right), \right. \\
& \left. e\left(g[1,0]\left[\frac{i-1}{2}, \frac{i+1}{2}\right]\right) \right\} \\
& -\max \left\{ e\left(g[0,1]\left[\frac{i+1}{2}, \frac{i-1}{2}\right)\right], \right. \\
& \left. e\left(g[0,1]\left[\frac{i-1}{2}, \frac{i+1}{2}\right]\right) \right\} \Bigg\} \\
= -\max \Bigg\{ & -\max \left\{ e\left(g\left[\frac{i+3}{2}, \frac{i-1}{2}\right]\right), \quad e\left(g\left[\frac{i+1}{2}, \frac{i+1}{2}\right]\right) \right\} \\
& -\max \left\{ e\left(g\left[\frac{i+1}{2}, \frac{i+1}{2}\right]\right), \quad e\left(g\left[\frac{i-1}{2}, \frac{i+3}{2}\right]\right) \right\} \Bigg\} \\
= \min \Bigg\{ & \max \left\{ e\left(g\left[\frac{i+3}{2}, \frac{i-1}{2}\right]\right), \quad e\left(g\left[\frac{i+1}{2}, \frac{i+1}{2}\right]\right) \right\} \\
& \max \left\{ e\left(g\left[\frac{i+1}{2}, \frac{i+1}{2}\right]\right), \quad e\left(g\left[\frac{i-1}{2}, \frac{i+3}{2}\right]\right) \right\} \Bigg\}
\end{aligned}$$

$$= \max\left\{e\left(g\left[\frac{i+1}{2}, \frac{i+1}{2}\right]\right),\right.$$
$$\left.\min\left\{e\left(g\left[\frac{i+3}{2}, \frac{i-1}{2}\right]\right), e\left(g\left[\frac{i-1}{2}, \frac{i+3}{2}\right]\right)\right\}\right\},$$

so the theorem holds for $d = i + 1$, if $i + 1$ is even.

Case 2. If $i+1$ is odd, then i is even. So from the induction hypothesis,

$$\begin{aligned}
m(i+1, g) = -\max &\left\{e\left(g[1,0]\left[\frac{i}{2}, \frac{i}{2}\right]\right)\right. \\
&\min\left\{e\left(g[1,0]\left[\frac{i+2}{2}, \frac{i-2}{2}\right]\right), \quad e\left(g[1,0]\left[\frac{i-2}{2}, \frac{i+2}{2}\right]\right)\right\} \\
&e\left(g[0,1]\left[\frac{i}{2}, \frac{i}{2}\right]\right) \\
&\left.\min\left\{e\left(g[0,1]\left[\frac{i+2}{2}, \frac{i-2}{2}\right]\right), \quad e\left(g[0,1]\left[\frac{i-2}{2}, \frac{i+2}{2}\right]\right)\right\}\right\} \\
= -\max &\left\{e\left(g\left[\frac{i+2}{2}, \frac{i}{2}\right]\right)\right. \\
&\min\left\{e\left(g\left[\frac{i+4}{2}, \frac{i-2}{2}\right]\right), \quad e\left(g\left[\frac{i}{2}, \frac{i+2}{2}\right]\right)\right\} \\
&e\left(g\left[\frac{i}{2}, \frac{i+2}{2}\right]\right) \\
&\left.\min\left\{e\left(g\left[\frac{i+2}{2}, \frac{i}{2}\right]\right), \quad r\left(g\left[\frac{i-2}{2}, \frac{i+4}{2}\right]\right)\right\}\right\} \\
= -\max &\left\{e\left(g\left[\frac{i+2}{2}, \frac{i}{2}\right]\right), \quad e\left(g\left[\frac{i}{2}, \frac{i+2}{2}\right]\right)\right\}
\end{aligned}$$

so the theorem holds for $d = i + 1$, if $i + 1$ is odd. Therefore, by mathematical induction, the theorem holds for $0 \leqslant d \leqslant k$. □

REFERENCES

1. B. W. Ballard, Non-Minimax Search Strategies for Minimax Trees: Theoretical Foundations and Empirical Studies, Technical Report, Duke University, Durham, North Carolina (July 1983).

2. D. Beal, An Analysis of Minimax, in *Advances is Computer Chess 2*, M. R. B. Clarke (ed.) University Press, Edinburgh (1980).
3. H. Berliner, The B* Tree Search Algorithm: A Best-First Proof Procedure, *Artificial Intelligence* **(12)**:23–40 (1979).
4. A. W. Biermann, Theoretical Issues Related to Computer Game Playing Programs, *Personal Computing*, pp. 86–88 (September 1978).
5. I. Bratko and M. Gams, Error Analysis of the Minimax Principle, in *Advances in Computer Chess 3*, M. R. B. Clarke (ed.) Pergamon Press, London (1982).
6. D. E. Knuth and R. W. Moore, An Analysis of Alpha-Beta Pruning, *Artificial Intelligence* **(6)**:293–326 (1975).
7. D. S. Nau, *Quality of Decision Versus Depth of Search on Game Trees*, Ph.D. Dissertation, Duke University, Durham, North Carolina (August 1979).
8. D. S. Nau, Pathology on Game Trees: A Summary of Results, Proc. First Annual National Conference on Artificial Intelligence, Stanford, California, pp. 102–104 (1980).
9. D. S. Nau, An Investigation of the Causes of Pathology in Games, *Artificial Intelligence* **(19)**:257–278 (1982). (An abstract of this paper will appear in *Zentralblatt fuer Mathematik* in 1983.)
10. D. S. Nau, The Last Player Theorem, *Artificial Intelligence* **(18)**:53–65 (1982). (An early version is available as Technical Report TR-865, Computer Science Department, University of Maryland, February 1980).
11. D. S. Nau, Pathology on Game Trees Revisited, and an alternative to Minimaxing, *Artificial Intelligence* **(21)**1, 2, 221–244 (March 1983). (Also available as Technical Report TR-1187, Computer Science Department, University of Maryland, July 1982.)
12. D. S. Nau, P. W. Purdom, and C. H. Tzeng, Experiments on Alternatives to Minimax (inpreparation) (1983).
13. D. S. Nau, Decision Quality as a Function of Search Depth on Game Trees, *Journal of the ACM* (October 1983) (to appear) (An early version is available as Technical Report TR-866, Computer Science Department, University of Maryland, February 1980.)
14. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto (1980).
15. J. Pearl, Asymptotic Properties of Minimax Trees and Game-Tree Searching Procedures, *Artificial Intelligence* **(14)**:113–138 (1980).
16. J. Pearl, Heuristic Search Theory: Survey of Recent Results, Proc. Seventh Internat. Joint Conf. Artif. Intel., Vancouver, Canada, pp. 554–562 (August 1981).
17. J. Pearl, On the Nature of Pathology in Game Searching, *Artificial Intelligence* **(20)**:427–453 (1983). (An early version is available as Technical Report UCLA-ENG-CSL-82-17, School of Engineering and Applied Science, University of California, Los Angles.)
18. A. L. Reibman and B. W. Ballard, Non-Minimax Search Strategies for Use against Fallible Opponents, National Conference on Artificial Intelligence, Washington, DC, pp. 338–342 (1983).
19. A. L. Robinson, Tournament Competion Fuels Computer Chess, *Science* **(204)**:1396–1398 (1979).
20. G. C. Stockman, A Minimax Algorithm Better than Alpha-Beta?, *Artificial Intelligence* **(12)**:179–196 (1979).
21. T. R. Truscott, Minimum Variance Tree Searching, Proc. First Internat. Symposium on Policy Analysis and Information Systems, Durham, North Carolina, pp. 203–209 (June 1979).
22. T. R. Truscott, Personal communication (January 1981).