An Investigation of the Causes of Pathology in Games*

Dana S. Nau
Computer Science Department, University of Maryland, College Park, MD 20742, U.S.A.

Recommended by Judea Pearl

ABSTRACT

Game trees are a useful model of many kinds of decision-making situations, and have been the subject of considerable investigation by researchers in both artificial intelligence and decision analysis.

Until recently it was almost universally believed that searching deeper on a game tree would in general improve the quality of a decision. However, recent theoretical investigations [8–10] by this author have demonstrated the existence of an infinite class of game trees for which searching deeper consistently degrades the quality of a decision.

This paper extends the previous work in two ways. First, the existence of pathology is demonstrated in a real game (Pearl's Game) using a real evaluation function. This pathological behavior occurs despite the fact that the evaluation function function increases dramatically in accuracy toward the end of the game. Second, the similarities and differences between this game and a related nonpathological game are used as grounds for speculation on why pathology occurs in some games and not in others.

1. Introduction

Game trees are a useful model of many kinds of decision-making situations. For this reason they have been a subject of considerable investigation both in artificial intelligence and in decision analysis.

Making a decision on a game tree involves searching the tree to compute utility values for the nodes of the tree. Since the number of nodes in the tree usually grows exponentially with the depth of the tree, it is usually not feasible to do a complete search of a large game tree, even when pruning techniques such as the alpha-beta algorithm [1, 4, 5, 13] are used.

Artificial intelligence researchers have obtained good results by searching

* This work was supported in part by a General Research Board award to the author from the University of Maryland, and in part by NSF grant ENG-7822159 to the Laboratory for Pattern Analysis at the University of Maryland.

Artificial Intelligence **19** (1982) 257–278
0004-3702/82/0000–0000/$02.75

game trees to some limited depth, using a *static evaluation function* to estimate the utility values of the nodes at that depth, and then proceeding, in the usual manner,[1] to compute utility values for the shallower nodes of the tree as if the estimated values were correct [2, 3, 5, 13, 16]. Until recently there was almost universal agreement that when such a *heuristic game tree search* is done, increasing the depth of the search improves the quality of the decision. However, a recent investigation by this author [8–10] showed that, given a certain theoretical model of the errors made by an evaluation function, there exists an infinite class of game trees which are *pathological* in the sense that as long as the search does not reach the end of the tree (in which case a correct decision could be guaranteed), searching deeper will not increase the probability that a decision is correct, but will instead cause the decision to become increasingly random.

This paper extends the previous work in two ways.

First, a game called Pearl's game is analyzed mathematically, and is shown to be pathological when the obvious evaluation function for that game is used. It is notable that this game was not invented for the purpose of demonstrating pathology, but was developed by Judea Pearl [15] for the purpose of analyzing the efficiency of game tree search procedures. Furthermore, pathology occurs in this game despite the fact that the evaluation function used for the game increases dramatically in accuracy as the end of the game approaches.

Second, a game closely related to Pearl's game is devised which is shown not to be pathological. The differences between this game and Pearl's game provide grounds for speculation on why pathology occurs in Pearl's game but not in games such as chess or checkers.

Section 2 describes Pearl's game. In Section 3, the quality of play in Pearl's game is analyzed as a function of the search depth when a minimax search is used. Section 4 uses the equations derived in Section 3 to provide numerical results which demonstrate the existence of pathology and describes a Monte Carlo simulation which corroborates the numerical results. Section 5 describes the related nonpathological game, and demonstrates its lack of pathology by means of a similar Monte Carlo simulation. Section 6 contains conclusions and speculations.

2. Pearl's Game

Judea Pearl [15] has described a class of two-player zero-sum games which he developed for use in analyzing the efficiency of various game tree search procedures [14]. This section describes the original class of games, generalizes it slightly, and discusses an appropriate evaluation function. *Pearl's game*

[1] In artificial intelligence research, 'minimaxing' is normally used, and the utility values are called 'minimax values'. This corresponds to the maximin criterion of decision analysis [6, 21].

refers to the entire class of games; each member of the class is called a *Pearl game.*

2.1. The original game

As originally described, Pearl's game is played on a chessboard measuring 2^c by 2^c rather than 8 by 8, where c is a positive integer. The initial playing board for a game in this class is constructed by randomly assigning each square of the board the value 1 with probability p, or the value 0 with probability $1-p$, where p is a constant and $0 \leq p \leq 1$.

The two players move in strict alternation. A move for the first player consists of dividing the board in half vertically, and discarding one half. A move for the second player consists of dividing what is left of the board in half horizontally, and discarding one half. The play continues in this manner until only one square is left. If the square has value 1, the player who made the last move wins. If it has value 0, his opponent wins.

2.2. A generalization

Every game in the class described above always takes $2c$ moves to play no matter what moves the players make. We can augment the class to contain games taking d moves to play, where d may be either even or odd, by letting the playing board measures $d^{\lfloor d/2 \rfloor}$ squares by $2^{\lceil d/2 \rceil}$ squares rather than 2^c by 2^c. If d is odd, the first player will always have the last move; if d is even, the second player will always have the last move. This 'last player' we call Max, and his opponent we call Min.

For every game in this class, the corresponding game tree is a complete binary game tree of depth d. Game tree nodes where it is Max's (or Min's) move we call max (or min) nodes, respectively.

This class can be augmented to include games having complete b-ary game trees for arbitrary b, by allowing the playing board to measure $b^{\lfloor d/2 \rfloor}$ by $b^{\lceil d/2 \rceil}$, and redefining a move to consist of dividing the board into b pieces (rather than just 2) and retaining only one of them. Each such game we call a *b-ary Pearl game* of depth d, and we call b the *branching factor* of the game.

The playing board and game tree for a Pearl game with $b = 2$ and $d = 4$ are shown in Fig. 1.

2.3. A restriction on p

Let G be the game tree for a b-ary Pearl game of depth d. We denote the root of G by root(G). It is easy to see that each node of G is either a forced win or a forced loss for Max if both players play perfectly. Such nodes we call *win nodes* and *loss nodes*, respectively. For example, a terminal node of value '1' is a win node and a terminal node of value '0' is a loss node.

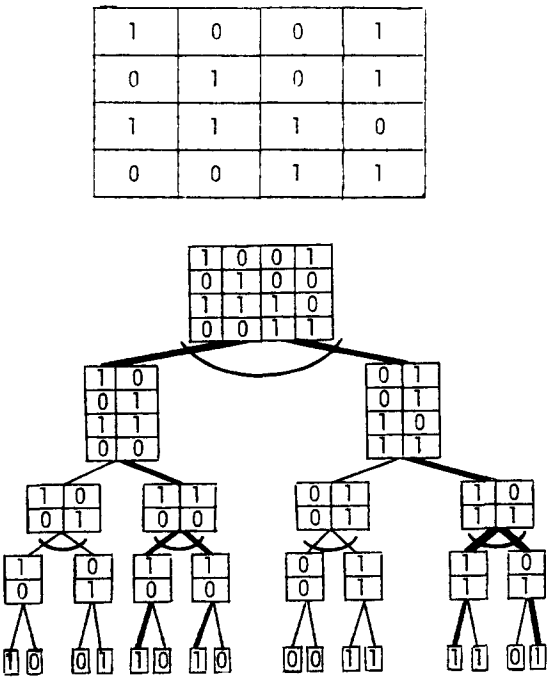

FIG. 1. A playing board for a 2-ary Pearl game of depth 4, and the corresponding game tree. Each Min node is indicated by an arc through the edges leaving the node. Note that Max has a forced win, as indicated by the solution tree drawn in boldface.

Let $p_{b,d}$ be the probability that root(G) is a win node. If p is the probability that a terminal node of G has the value '1', then according to the Last Player Theorem [11],

(1) if $p > w_b$, then $\lim_{d\to\infty} p_{b,d} = 1$;
(2) if $p < w_b$, then $\lim_{d\to\infty} p_{b,d} = 0$;
(3) if $p = w_b$, then

$$p_{b,d} = \begin{cases} 1 - w_b & \text{if } d \text{ is odd,} \\ w_b & \text{if } d \text{ is even,} \end{cases}$$

where w_b is the unique solution in the interval (0, 1) of the equation $(1 - x)^b = x$. Thus the probability that Max has a forced win converges to 1 or 0 when the probability that a terminal node is a win is above or below w_b, respectively. According to [11] this convergence is generally quite rapid.

Since we are interested in games where both players have a reasonable chance of winning, we will only consider games for which $p = w_b$. For $b = 2$, $w_b = \frac{1}{2}(3 - \sqrt{5}) \approx 0.382$. Other values of w_b are given in Table 1.[2]

[2] The importance of w_b in game tree searching has also been investigated by Pearl [14] and Baudet [1].

TABLE 1. Approximate values of w_b for $b = 1, 2, \ldots, 50$

b	w_b	b	w_b	b	w_b	b	w_b	b	w_b
1	0.50000	11	0.15560	21	0.10271	31	0.07872	41	0.06463
2	0.38197	12	0.14745	22	0.09955	32	0.07700	42	0.06352
3	0.31767	13	0.14024	23	0.09662	33	0.07536	43	0.06246
4	0.27551	14	0.13382	24	0.09387	34	0.07380	44	0.06144
5	0.24512	15	0.12805	25	0.09130	35	0.07231	45	0.06045
6	0.22191	16	0.12283	26	0.08889	36	0.07088	46	0.05950
7	0.20346	17	0.11809	27	0.08662	37	0.06952	47	0.05858
8	0.18835	18	0.11375	28	0.08448	38	0.06822	48	0.05770
9	0.17570	19	0.10977	29	0.08245	39	0.06697	49	0.05684
10	0.16492	20	0.10610	30	0.08054	40	0.06577	50	0.05601

2.4. An evaluation function for Pearl's game

As is widely known, it is unfeasible to do a minimax search of a game tree unless the tree is very small. Hence, minimaxing requires searching to some arbitrary depth of the tree and using a heuristic evaluation function to estimate the minimax values of the nodes at this depth. If the value returned by the function is high, it should mean that the node is probably a forced win, and if the value is low, it should mean that the node is probably a forced loss.

Let g be a node in a Pearl game, and let g_1 and g_2 be two of g's children. Both g_1 and g_2 are rectangular boards containing 2^k squares for some k, and each square has a random value 1 or 0.

If g_1 has more '1' squares than g_2, it would seem more likely that g_1 is a forced win then that g_2 is a forced win. It is easy to prove that this is true using induction. As a special case, if the fraction of '1' squares in g_1 exceeds w_b and the fraction of '1' squares in g_2 is less than w_b, then (as we discussed earlier) it is quite likely that g_1 is a forced win and g_2 is a forced loss. Thus the number of '1' squares in a game position is a reasonable evaluation function for a node in a Pearl game. We denote the number of '1' squares in a node g by $e(g)$.

3. A Mathematical Analysis of Minimaxing in Pearl Games

Let g be a node in a game tree G, and suppose the subtree rooted at g has depth d or greater, for some d. We define the depth d minimax value of g by

$$e_d(g) = \begin{cases} e(g) & \text{if } d = 0, \\ \min\{e_{d-1}(g') \mid g' \text{ is a child of } g\} & \text{if } g \text{ is a min node}, \\ \max\{e_{d-1}(g') \mid g' \text{ is a child of } g\} & \text{if } g \text{ is a max node}. \end{cases}$$

Choosing a move at g using a depth d minimax search means choosing that

child of g which has the best depth $d-1$ minimax value (so that the nodes at depth d relative to g determine the decision). By the 'best' value we mean the highest value if Max is to move, or the lowest value if Min is to move. If several nodes (say, J of them) receive this same value, then the player must choose among them at random. If I of them are forced win nodes, then the probability of making a 'correct' move (i.e., a move to a position from which the player can force a win) is thus I/J. Hence the probability that a decision is correct depends on the accuracy of the depth $d-1$ minimax values of g's children.

In this section we discuss how to compute the probability of correct decision in Pearl games when minimaxing is used. Eqs. (1) through (16) apply to b-ary Pearl games for arbitrary b, but because of computational difficulties with the general case, (17) through (32) apply only to binary Pearl games.

3.1. Computation of probability distributions for minimax values

Let g be a node in a Pearl game G, and let $h(g)$ be the height of g in G, i.e., the number of moves from g to the end of the game. Then the number of squares in the node g is $\mathrm{sq}(g) = b^{h(g)}$. This is the maximum possible value of $e(g)$.

Since each square in g independently receives value 1 with probability p and 0 with probability $1-p$, we have

$$\Pr[e(g) = i \mid h(g) = k] = \binom{\mathrm{sq}(g)}{i} w_b^i (1 - w_b)^{\mathrm{sq}(g)-i}. \tag{1}$$

We denote the events that g is a win or loss node by $\mathrm{win}(g)$ and $\mathrm{loss}(g)$, respectively. From the definition of conditional probability,

$$\begin{aligned}
&\Pr[e(g) = i \mid h(g) = k, \mathrm{win}(g)] = \\
&\quad = \frac{\Pr[\mathrm{win}(g), e(g) = i \mid h(g) = k]}{\Pr[\mathrm{win}(g) \mid h(g) = k]} \\
&\quad = \frac{\Pr[\mathrm{win}(g) \mid e(g) = i, h(g) = k] \Pr[e(g) = i \mid h(g) = k]}{\Pr[\mathrm{win}(g) \mid h(g) = k]}.
\end{aligned} \tag{2}$$

$\Pr[e(g) = i \mid h(g) = k]$ is given by (1).

According to the Last Player Theorem,

$$\Pr[\mathrm{win}(g) \mid h(g) = k] = \begin{cases} w_b & \text{if } k \text{ is even}, \\ 1 - w_b & \text{if } k \text{ is odd}. \end{cases} \tag{3}$$

The subtree of G rooted at g is a b-ary Pearl game of depth $h(g)$. Since its terminal node values are independent, each possible configuration of i '1' values and $\mathrm{sq}(g) - i$ '0' values is equally likely. Thus

$$\Pr[\text{win}(g) \mid e(g) = i, h(g) = k] = W_b(i, k) \Big/ \binom{\text{sq}(g)}{i}, \tag{4}$$

where $W_b(i, k)$ is number of b-ary Pearl games G' of depth k for which $e(\text{root}(G')) = i$ and $\text{win}(\text{root}(G'))$.

In a manner similar to the above, we obtain

$$\begin{aligned}
&\Pr[e(g) = i \mid h(g) = k, \text{loss}(g)] = \\
&= \frac{\Pr[\text{loss}(g), e(g) = i \mid h(g) = k]}{\Pr[\text{loss}(g) \mid h(g) = k]} \\
&= \frac{\Pr[\text{loss}(g) \mid e(g) = i, h(g) = k] \Pr[e(g) = i \mid h(g) = k]}{\Pr[\text{loss}(g) \mid h(g) = k]};
\end{aligned} \tag{5}$$

$$\Pr[\text{loss}(g) \mid h(g) = k = \begin{cases} 1 - w_b & \text{if } k \text{ is even}, \\ w_b & \text{if } k \text{ is odd}; \end{cases} \tag{6}$$

and

$$\Pr[\text{loss}(g) \mid e(g) = i, h(g) = k] = L_b(i, k) \Big/ \binom{\text{sq}(g)}{i}; \tag{7}$$

where $L_b(i, k)$ is the number of b-ary Pearl games G' of depth k for which $e(\text{root}(G')) = i$ and $\text{loss}(\text{root}(G'))$.

Note that

$$W_b(i, k) + L_b(i, k) = \binom{\text{sq}(g)}{i}. \tag{8}$$

Substituting into (2) and (5) yields

$$\begin{aligned}
&\Pr[e(g) = i \mid h(g) = k, \text{win}(g)] = \\
&= \frac{W_b(i, k)}{\binom{\text{sq}(g)}{i}} \frac{\binom{\text{sq}(g)}{i} w_b^i (1 - w_b)^{\text{sq}(g) - i}}{(w_b \text{ if } k \text{ is even, else } 1 - w_b)} \\
&= \frac{W_b(i, k) w_b^i (1 - w_b)^{bk - i}}{(w_b \text{ if } k \text{ is even, else } 1 - w_b)};
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\Pr[e(g) = i \mid h(g) = k, \text{loss}(g)] = \\
&= \frac{L_b(i, k)}{\binom{\text{sq}(g)}{i}} \frac{\binom{\text{sq}(g)}{i} w_b^i (1 - w_b)^{\text{sq}(g) - i}}{(1 - w_b \text{ if } k \text{ is even, else } w_b)} \\
&= \frac{L_b(i, k) w_b^i (1 - w_b)^{bk - i}}{(1 - w_b \text{ if } k \text{ is even, else } w_b)}.
\end{aligned} \tag{10}$$

3.2. Computation of W_b and L_b

$W_b(i, k)$ and $L_b(i, k)$ can be computed recursively. For $k = 0$ the game tree rooted at g consists of a single node, whence

$$W(0, 0) = 0 \quad \text{and} \quad L(0, 0) = 1\,; \tag{11}$$

$$W(1, 0) = 1 \quad \text{and} \quad L(1, 0) = 0\,. \tag{12}$$

Every Pearl game G' of even depth has a min node as its root. Such a node can be a win only if all b of its children are wins. Thus if k is even,

$$W_b(i, k) = \sum_{\substack{u_1+u_2+\cdots+u_b=i \\ \forall_j 0 \leq u_j \leq b^{k-1}}} \prod_{j=1}^{b} W_b(u_j, k-1) \tag{13}$$

and

$$L_b(i, k) = \binom{b^k}{i} - W_b(i, k)\,, \tag{14}$$

where u_j is the number of '1' squares in the jth child of root(G').

Every Pearl game G' of odd depth has a max node as its root. Such a node can be a loss only if all b of its children are losses. Thus if k is odd,

$$L_b(i, k) = \sum_{\substack{u_1+\cdots+u_b=i \\ \forall_j 0 \leq u_j \leq b^{k-1}}} \prod_{j=1}^{b} L_b(u_j, k-1) \tag{15}$$

and

$$W_b(i, k) = \binom{b^k}{i} - L_b(i, k)\,, \tag{16}$$

where the u_j are as before.

The computation of $W_b(i, k)$ and $L_b(i, k)$ as specified by (15) and (16) is rather complicated. This can be simplified in the case $b = 2$. In this case, $W_2(i, k)$ can be computed independently of $L_2(i, k)$ in the following manner.

If k is even (13) reduces to

$$W_2(i, k) = \sum_{u=r_1}^{r_2} W_2(u, k-1) W_2(i-u, k-1)\,, \tag{17}$$

where $r_1 = \max(0, i - 2^{k-1})$ and $r_2 = \min(i, 2^{k-1})$.

If k is odd there are three ways that the event $\{\text{win}(g), e(g) = i, h(g) = k\}$ can

occur. One is if both of the children of g are wins. The number of ways for this to occur is

$$\sum_{u=r_1}^{r_2} W_2(u, k-1)W_2(i-u, k-1), \tag{18}$$

where r_1 and r_2 are as in (17).

The second way is if the first child of g is a win and the second is a loss. The number of ways this can happen is

$$\begin{aligned}&\sum_{u=r_1}^{r_2} W_2(u, k-1)L_2(i-u, k-1) = \\ &\quad = \sum_{u=r_1}^{r_2} W_2(u, k-1)\left(\binom{2^{k-1}}{i-n} - W_2(i-u, k-1)\right).\end{aligned} \tag{19}$$

The third way is if the first child of g is a loss and the second is a win. The number of ways this can happen is also given by (19).

Thus for k odd, the total number of ways the event $\{\text{win}(g), e(g) = i, h(g) = k\}$ can occur is

$$\begin{aligned}W_2(i, k) &= \sum_{u=r_1}^{r_2} W_2(u, k-1)W_2(i-u, k-1) \\ &\quad + 2\sum_{u=r_1}^{r_2} W_2(u, k-1)\left(\binom{2^{k-1}}{i-u} - W_2(i-u, k-1)\right) \\ &= \sum_{u=r_1}^{r_2} W_2(u, k-1)\left[\binom{2^{k-1}}{i-u} - W_2(i-u, k-1)\right].\end{aligned} \tag{20}$$

3.3. The probability of correct decision

Suppose a player is choosing a move at some node g of height k in a b-ary Pearl game, using a minimax search to depth d. To avoid computational difficulties we restrict $b = 2$. If one of g's children (say, g_1) is a forced win and the other (g_2) is a forced loss, then we define a correct move to be a move to g_1 if the player is Max, or to g_2 if the player is Min. Since we are only interested in the probability of making a correct decision in the case where it makes a difference what move is made, we do not define a correct move if both children are forced wins or both are forced losses.[3]

[3] Actually, the choice of move may still make a difference if both nodes are forced wins or both are forced losses. This could occur, for example, if one player's evaluation function were more accurate on the nodes of the subtree rooted at g_1 than on the nodes of the subtree rooted at g_2. However, we have no way to predict such an occurrence.

Since Max moves to the node of highest minimax value and Min moves to the node of lowest minimax value, a correct decision will be made if $e_{d-1}(g_1) > e_{d-1}(g_2)$, and an incorrect decision will be made if $e_{d-1}(g_1) < e_{d-1}(g_2)$. If $e_{d-1}(g_1) = e_{d-1}(g_2)$, then the player must choose among g_1 and g_2 at random, whence the probability of correct decision will be 1/2. Hence the probability of correct decision at g is

$$\begin{aligned} D(d, k) &= \Pr[e_{d-1}(g_1) > e_{d-1}(g_2)] + \tfrac{1}{2}\Pr[e_{d-1}(g_1) = e_{d-1}(g_2)] \\ &= \sum_{j=0}^{2^{k-1}} \{\Pr[e_{d-1}(g_1) \geq j+1, e_{d-1}(g_2) = j] \\ &\qquad + \tfrac{1}{2}\Pr[e_{d-1}(g_1) = e_{d-1}(g_2) = j]\}, \end{aligned} \tag{21}$$

since $\mathrm{sq}(g_1) = \mathrm{sq}(g_2) = 2^{k-1}$. If we define

$$\mathrm{mw}(i, d, k) = \Pr[e_d(g) \geq i \mid h(g) = k, \mathrm{win}(g)] \tag{22}$$

and

$$\mathrm{ml}(i, d, k) = \Pr[e_d(g) \geq i \mid h(g) = k, \mathrm{loss}(g)], \tag{23}$$

then

$$\begin{aligned} D(d, k) &= \sum_{j=0}^{2^{k-1}} \{\mathrm{mw}(j+1, d-1, k-1)[\mathrm{ml}(j, d-1, k-1) \\ &\qquad - \mathrm{ml}(j+1, d-1, k-1)] \\ &\qquad + \tfrac{1}{2}[\mathrm{mw}(j, d-1, k-1) - \mathrm{mw}(j+1, d-1, k-1)] \\ &\qquad \times [\mathrm{ml}(j, d-1, k-1) - \mathrm{ml}(j+1, d-1, k-1)]\} \\ &= \sum_{j=0}^{2^{k-1}} \tfrac{1}{2}[\mathrm{mw}(j, d-1, k-1) + \mathrm{mw}(j+1, d-1, k-1)] \\ &\qquad \times [\mathrm{ml}(j, d-1, k-1) - \mathrm{ml}(j+1, d-1, k-1)]. \end{aligned} \tag{24}$$

We now discuss how to compute mw and ml. Since $e(g)$ is never greater than 2^k,

$$\mathrm{mw}(2^k, 0, k) = \Pr[e(g) = 2^k | h(g) = k, \mathrm{win}(g)] \tag{25}$$

and

$$\mathrm{ml}(2^k, 0, k) = \Pr[e(g) = 2^k \mid h(g) = k, \mathrm{loss}(g)], \tag{26}$$

for $k = 0, 1, 2, \ldots$. These values can be computed from (9) and (10). From (9) and

(10) we can then recursively compute

$$\mathrm{mw}(i, 0, k) = \mathrm{mw}(i+1, 0, k) + \Pr[e(g) = i \mid h(g) = k, \mathrm{win}(g)] \tag{27}$$

and

$$\mathrm{ml}(i, 0, k) = \mathrm{ml}(i+1, 0, k) + \Pr[e(g) = i \mid h(g) = k, \mathrm{loss}(g)]\,, \tag{28}$$

for $i = 2^{k-1}$ to 0.

Suppose k is even, g is a node of height k with children g_1 and g_2, and $d \leq k$. Then g is a min node, so $e_d(g) \geq i$ only if $e_{d-1}(g_1) \geq i$ and $e_{d-1}(g_2) \geq i$. Furthermore, g is a win only if both g_1 and g_2 are wins, so

$$\begin{aligned}
\mathrm{mw}(i, d, k) &= \Pr[e_d(g) \geq i \mid \mathrm{win}(g_1), \mathrm{win}(g_2)] \\
&= \Pr[e_{d-1}(g_1) \geq i \mid \mathrm{win}(g_1), \mathrm{win}(g_2)] \\
&\quad \times \Pr[e_{d-1}(g_2) \geq i \mid \mathrm{win}(g_1), \mathrm{win}(g_2)] \\
&= \Pr[e_{d-1}(g_1) \geq i \mid \mathrm{win}(g_1)] \Pr[e_{d-1}(g_2) \geq i \mid \mathrm{win}(g_2)] \\
&= (\mathrm{mw}(i, d-1, k-1))^2\,.
\end{aligned} \tag{29}$$

But g is a loss if either g_1 or g_2 is a loss, so

$$\begin{aligned}
\mathrm{ml}(i, d, k) &= \Pr[e_d(g) \geq i \mid \mathrm{loss}(g)] \\
&= \Pr[e_d(g) \geq i \mid \mathrm{loss}(g_1), \mathrm{loss}(g_2)] \Pr[\mathrm{loss}(g_1), \mathrm{loss}(g_2) \mid \mathrm{loss}(g)] \\
&\quad + \Pr[e_d(g) \geq i \mid \mathrm{loss}(g_1), \mathrm{win}(g_2)] \Pr[\mathrm{loss}(g_1), \mathrm{win}(g_2) \mid \mathrm{loss}(g)] \\
&\quad + \Pr[e_d(g) \geq i \mid \mathrm{win}(g_1), \mathrm{loss}(g_2)] \Pr[\mathrm{win}(g_1), \mathrm{loss}(g_2) \mid \mathrm{loss}(g)] \\
&= \Pr[e_{d-1}(g_1) \geq i \mid \mathrm{loss}(g_1)] \Pr[e_{d-1}(g_2) \geq i \mid \mathrm{loss}(g_2)] \\
&\quad \times \Pr[\mathrm{loss}(g_1), \mathrm{loss}(g_2)] / \Pr[\mathrm{loss}(g)] \\
&\quad + \Pr[e_{d-1}(g_1) \geq i \mid \mathrm{loss}(g_1)] \Pr[e_{d-1}(g_2) \geq i \mid \mathrm{win}(g_2)] \\
&\quad \times \Pr[\mathrm{loss}(g_1), \mathrm{win}(g_2)] / \Pr[\mathrm{loss}(g)] \\
&\quad + \Pr[e_{d-1}(g_1) \geq i \mid \mathrm{win}(g_1)] \Pr[e_{d-1}(g_2) \geq i \mid \mathrm{loss}(g_2)] \\
&\quad \times \Pr[\mathrm{win}(g_1), \mathrm{loss}(g_2)] / \Pr[\mathrm{loss}(g)] \\
&= \frac{\mathrm{ml}(i, d-1, k-1)^2 (w_2)^2}{1 - w_2} \\
&\quad + 2 \frac{\mathrm{ml}(i, d-1, k-1) \mathrm{mw}(i, d-1, k-1) w_2 (1 - w_2)}{1 - w_2}\,.
\end{aligned} \tag{30}$$

Suppose k is odd instead of even. Then g is a max node, so $e_d(g) < i$ only if $e_{d-1}(g_1) < i$ and $e_{d-1}(g_2) < i$. Now, g is a win if either of g_1 or g_2 is a win, so as with (30),

$$\begin{aligned}
\mathrm{mw}(i, d, k) &= 1 - \Pr[e_d(g) < i \mid \mathrm{win}(g)] \\
&= 1 - \Big\{ \Pr[e_{d-1}(g_1) < i \mid \mathrm{win}(g_1)] \Pr[e_{d-1}(g_2) < i \mid \mathrm{win}(g_2)] \\
&\qquad \times \Pr[\mathrm{win}(g_1), \mathrm{win}(g_2)] / \Pr[\mathrm{win}(g)] \\
&\qquad + \Pr[e_{d-1}(g_1) < i \mid \mathrm{win}(g_1)] \Pr[e_{d-1}(g_2) < i \mid \mathrm{loss}(g_2)] \\
&\qquad \times \Pr[\mathrm{win}(g_1), \mathrm{loss}(g_2)] / \Pr[\mathrm{win}(g)] \\
&\qquad + \Pr[e_{d-1}(g_1) < i \mid \mathrm{loss}(g_1)] \Pr[e_{d-1}(g_2) < i \mid \mathrm{win}(g_2)] \\
&\qquad \times \Pr[\mathrm{loss}(g_1), \mathrm{win}(g_2)] / \Pr[\mathrm{win}(g)] \Big\} \\
&= 1 - \left[\frac{(1 - \mathrm{mw}(i, d-1, k-1))^2 (w_2)^2}{1 - w_2} \right. \\
&\quad \left. + 2 \frac{(1 - \mathrm{mw}(i, d-1, k-1))(1 - \mathrm{ml}(i, d-1, k-1)) w_2 (1 - w_2)}{1 + w_2} \right].
\end{aligned} \tag{31}$$

But g is a loss only if both g_1 and g_2 are losses, so as with (29),

$$\begin{aligned}
\mathrm{ml}(i, d, l) &= 1 - \Pr[e_d(g) < i \mid \mathrm{loss}(g)] \\
&= 1 - (\Pr[e_{d-1}(g_1) < i \mid \mathrm{loss}(g_1)] \Pr[e_{d-1}(g_2) < i \mid \mathrm{loss}(g_2)]) \\
&= 1 - (1 - \mathrm{ml}(i, d-1, k-1))^2 .
\end{aligned}$$

Thus for $b = 2$, $k = 0, 1, \ldots$, $d = 0$ to k, and $i = 0$ to 2^k, $\mathrm{mw}(i, d, k)$ can be computed from (25) through (30).

4. Numerical Results for Pearl's Game

4.1. Analytical results

Let g be a node of height k in a binary Pearl game, and suppose one child of g is a forced loss node and the other is a forced win. Then the probability of correct decision $D(d, k)$ at g for all search depths $d = 1, 2, \ldots, k$ may be computed from the equations in Section 3.

A computer program was written to do these computations. Its output is given in Table 2 and Fig. 2. As can be seen from Fig. 2, $D(d, k)$ tends to increase with d for $k \leq 7$. But as k increases, it is increasingly common for $D(d, k)$ to *decrease* as d increases (as long as $d \leq k - 2$).

It was unfeasible to run the program for $k > 15$, because the computation

TABLE 2. Probability of correct decision $D(d, k)$ as a function of search depth d and node height k in a binary Pearl game. In each case the probability of a terminal node being a forced win is w_2

d \ k	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.000	0.9474	0.9018	0.8491	0.8051	0.7648	0.7306	0.7006	0.6748	0.6524	0.6329	0.6160	0.6013	0.5884
2	1.000	1.000	0.9137	0.8724	0.8075	0.7685	0.7255	0.6952	0.6662	0.6438	0.6235	0.6070	0.5924	0.5802
3		1.000	1.000	0.8929	0.8303	0.7727	0.7299	0.6922	0.6627	0.6377	0.6174	0.6003	0.5861	0.5740
4			1.000	1.000	0.8252	0.7904	0.7280	0.6936	0.6577	0.6331	0.6108	0.5941	0.5795	0.5679
5				1.000	1.000	0.8062	0.7415	0.6911	0.6584	0.6293	0.6075	0.5892	0.5750	0.5631
6					1.000	1.000	0.6983	0.7050	0.6502	0.6269	0.6024	0.5854	0.5702	0.5589
7						1.000	1.000	0.6986	0.6564	0.6188	0.5986	0.5807	0.5670	0.5552
8							1.000	1.000	0.5810	0.6404	0.5850	0.5743	0.5616	0.5518
9								1.000	1.000	0.5925	0.5947	0.5698	0.5558	0.5464
10									1.000	1.000	0.5172	0.5924	0.5416	0.5381
11										1.000	1.000	0.5210	0.5524	0.5401
12											1.000	1.000	0.5009	0.5545
13												1.000	1.000	0.5011
14													1.000	1.000
15														1.000

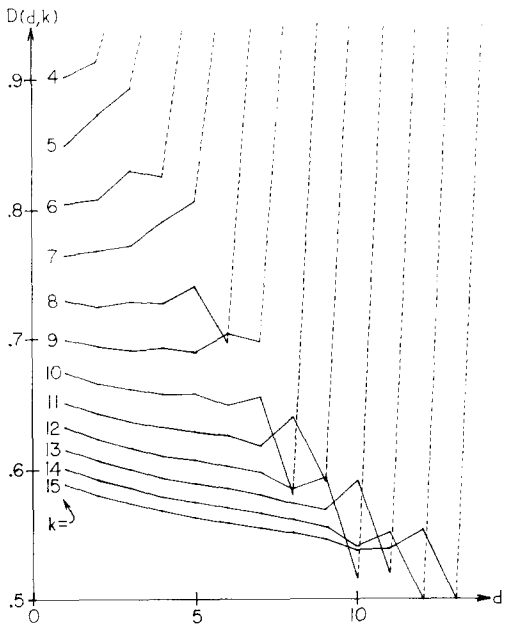

FIG. 2. Probability of correct decision $D(d, k)$ as a function of d, for $d = 1, 2, \ldots, k-2$. If $d = k-1$ or $d = k$, then $D(d, k) = 1$, as indicated by the dotted lines.

takes exponentially increasing time and space requirements.[4] However, the author expects that the tendency for $D(d, k)$ to decrease as d increases holds for all $k > 15$. A rigorous proof of this would be quite difficult because of the complexity of the expressions used to compute $D(d, k)$.

Some readers have speculated that the reason pathology does not occur on games such as chess or checkers is that the evaluation functions in such games become more accurate toward the end of the game, thus making deeper searches more accurate. Experts on game-playing computer programs consulted by this author disagree with such statements [17, 20], but it is interesting to consider the same issues with respect to Pearl's game.

One easy way to measure the accuracy of $e(g)$ as a function of the distance k from g to the end of the game is to look at the probability of correct decision at g when evaluating g's children directly. This probability is $D(1, k)$, which is graphed as a function of k in Fig. 3. Note that e increases exponentially in accuracy as the end of the game approaches. Thus deeper searches make use of board information which is dramatically more accurate, and the pathological behavior illustrated in Fig. 2 occurs despite this fact.

[4] The program was started with $k = 16$ on a Vax 11/780, and had taken about 48 hours of CPU time without finishing, when the system crashed.

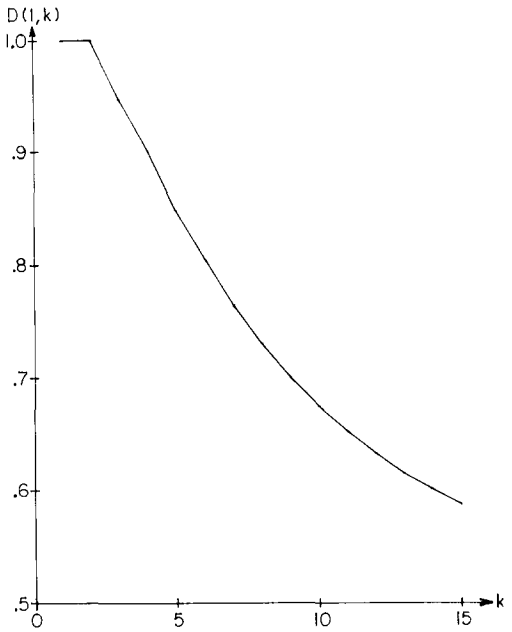

FIG. 3. $D(1, k)$ as a function of k. This provides a measure of the accuracy of $e(g)$ when g is at distance k from the end of the game.

4.2. Monte Carlo simulation results

Rather than using the mathematical results from Section 3, it is also possible to estimate the probability of correct decision on binary Pearl games using Monte Carlo simulation techniques. This can be done as follows.

Given k and some positive integer n, create n binary Pearl games of depth $k-1$, using a random number generator to determine which squares have value 0 and which have value 1. Let m be the number of games that are forced wins. For $d = 1, 2, \ldots, k$ and $i = 0, 1, \ldots, 2^{k-1}$, let

$$\text{mw}'(i, d-1, k-1) = \text{number of win trees whose depth } d-1 \text{ minimax value is } i \text{ or greater} \Big/ m, \tag{33}$$

$$\text{ml}'(i, d-1, k-1) = \text{number of loss trees whose depth } d-1 \text{ minimax value is } i \text{ or greater} \Big/ n-m. \tag{34}$$

For all i and d, $\text{mw}'(i, d-1, k-1)$ approximates $\text{mw}(i, d-1, k-1)$ and $\text{ml}'(i, d-1, k-1)$ approximates $\text{ml}(i, d-1, k-1)$. Thus an approximation $D'(d, k)$ of $D(d, k)$ can be computed using (24).

Such a simulation has been done for $n = 3200$ and $k = 3, 4, \ldots, 14$. The results, which are shown in Table 3, match the results given in Section 4.1 fairly

TABLE 3. Probability of correct decision $D'(d, k)$ as a function of search depth d and node height k in a binary Pearl game, produced by a Monte Carlo simulation involving 3200 games for each value of k. In each case the probability of a terminal node being a forced win is w_2. R, the number of wins divided by 3200, approximates w_2 (0.38197) or $1 - w_2$ (0.61803) as k is odd or even, respectively

d \ k	3	4	5	6	7	8	9	10	11	12	13	14
1	0.9519	0.9067	0.8424	0.8035	0.7633	0.7282	0.6854	0.6596	0.6467	0.6142	0.6129	0.6220
2	1.000	0.9148	0.8732	0.8021	0.7692	0.7163	0.6730	0.6393	0.6366	0.6059	0.6028	0.6103
3	1.000	1.000	0.8938	0.8260	0.7716	0.7236	0.6710	0.6433	0.6208	0.5995	0.5926	0.5934
4		1.000	1.000	0.8226	0.7928	0.7204	0.6739	0.6400	0.6314	0.5944	0.5837	0.5878
5			1.000	1.000	0.8111	0.7353	0.6659	0.6457	0.6234	0.6044	0.5874	0.5924
6				1.000	1.000	0.6978	0.6897	0.6388	0.6352	0.5891	0.5837	0.5815
7					1.000	1.000	0.6956	0.6452	0.6267	0.5844	0.5832	0.5835
8						1.000	1.000	0.5854	0.6346	0.5798	0.5896	0.5751
9							1.000	1.000	0.5927	0.5858	0.5683	0.5630
10								1.000	1.000	0.5135	0.6047	0.5466
11									1.000	1.000	0.5201	0.5548
12										1.000	1.000	0.5000
13											1.000	1.000
14												1.000
R	0.3825	0.6113	0.3788	0.6322	0.3928	0.6163	0.3834	0.6259	0.3900	0.6297	0.3872	0.6250

TABLE 4. The error quantity $(D'(d,k)-D(d,k))/D(d,k)$ expressed as a percentage, where $D(d,k)$ and $D'(d,k)$ are as given in Tables 2 and 3. The row labeled 'R error' contains $(R-w_2)/w_2$ or $(R-(1-w_2))/(1-w_2)$ as k is odd or even, respectively, where R is as given in Table 3

d \ k	3	4	5	6	7	8	9	10	11	12	13	14
1	0.5%	0.5%	−0.8%	−0.2%	−0.2%	−0.3%	−2.2%	−2.3%	−0.9%	−3.0%	−0.5%	3.4%
2	0.0%	0.1%	0.1%	−0.7%	0.1%	−1.3%	−3.2%	−4.0%	−1.1%	−2.8%	−0.7%	3.0%
3	0.0%	0.0%	0.1%	−0.5%	−0.1%	−0.9%	−3.1%	−2.9%	−2.7%	−2.9%	−1.3%	1.2%
4		0.0%	0.0%	−0.3%	0.3%	−1.1%	−2.8%	−2.7%	−0.3%	−2.7%	−1.8%	1.4%
5			0.0%	0.0%	0.6%	−0.8%	−3.6%	−1.9%	−0.9%	−0.5%	−0.3%	3.0%
6				0.0%	0.0%	−0.1%	−2.2%	−1.8%	1.3%	−2.2%	−0.3%	2.0%
7					0.0%	0.0%	−0.4%	−1.7%	1.3%	−2.4%	0.4%	2.9%
8						0.0%	0.0%	0.8%	−0.9%	−0.9%	2.7%	2.4%
9							0.0%	0.0%	0.0%	−1.5%	−0.3%	1.3%
10								0.0%	0.0%	−0.7%	2.1%	0.9%
11									0.0%	0.0%	−0.2%	0.4%
12										0.0%	0.0%	−0.2%
13											0.0%	0.0%
14												0.0%
R error	0.1%	−1.1%	−0.8%	2.3%	2.8%	−0.3%	0.4%	1.3%	2.1%	1.9%	1.4%	1.1%

closely. In particular, the relative error (as shown in Table 4) is in all cases no greater than 4%. This provides additional confirmation of the existence of pathology in Pearl's game.

5. A Nonpathological Game

5.1. Motivation and definitions

The preceding sections have demonstrated the existence of pathology in Pearl's game. It is well known that pathology does not occur in games such as chess or checkers. For example, the most successful chess-playing computer programs have achieved their success by searching the game tree as deeply as possible, even at the expense of using a faster but less accurate evaluation function [3, 16, 18, 19]. It is natural to wonder what kinds of differences between these games and Pearl's game might be responsible for this difference in behavior.

As pointed out in the last section, the occurrence of pathology in Pearl's game does not seem to be due to the evaluation function: pathology in Pearl's game occurs despite the fact that the evaluation function increases exponentially in accuracy toward the end of the game, whereas evaluation functions for chess are notoriously inaccurate in the endgame [17, 18].

We now discuss another major difference between Pearl's game and games such as chess or checkers which might be relevant to the occurrance or absence of pathology. In games such as chess or checkers, the evaluation function value of a node is usually positively correlated with the evaluation function value of its parent. This also occurs in Pearl's game. However, there is another kind of dependency among node values which Pearl's game does not have.

In games such as chess or checkers, positions are often characterized as 'strong' and 'weak'. Strong nodes are likely to be win nodes, and are likely to have high minimax values. Weak nodes are likely to be loss nodes, and are likely to have low minimax values. Since board positions change incrementally, a strong node is likely to have strong children, and a weak node is likely to have weak children. Thus the minimax values of sibling nodes (or other closely related nodes) are likely to be similar. Therefore, the game tree is likely to be differentiated into sections containing many strong nodes and few weak nodes, and sections containing many weak nodes and few strong nodes.

The above property does not occur in Pearl's game. In particular, let P be a Pearl game, and let g and g' be any two nodes at the same depth in P. Then the minimax values of g and g', being functions of independent random variables, are independent of each other.

In order to investigate games in which node strength changes incrementally, we define a class of games which we call *incremental games*.[5] Let d be a

[5] The approach used here was inspired by a game tree model proposed by Newborn [12, p. 157] and later used by Lindstrom [7, p. 41].

positive integer. We define a b-ary incremental game G of depth d as a game having the same size playing board, the same moves, and the same criterion for winning as a b-ary Pearl game of depth d. However, the initial playing board is set up differently.

To set up the playing board for an incremental game G, each arc of the game tree for G is independently, randomly given the value 1 with some probability q or the value -1 with probability $1-q$, where q is a constant such that $0 \leq q \leq 1$. The *strength* of a node g in the game tree is defined as the sum of the arc values on the path from g back to the root. A square in the playing board for G is given the value 1 if the corresponding terminal node of the game tree has positive strength, and the value 0 otherwise. We arbitrarily choose $q = 1/2$.

We use the same evaluation function e for incremental games as we do for Pearl games, for the same reasons. The depth d minimax value $\bar{e}_d(g)$ for a node g in an incremental game is defined in the same way as $e_d(g)$ was defined for Pearl's game.

Suppose a player is choosing a move at some node g of height k in a binary incremental game, using a minimax search to depth d. If one of g's children (say, g_1) is a forced win and the other (g_2) is a forced loss, then (as with Pearl's game) we define a *correct* move to be a move to g_1 if the player is Max, or to g_2 if the player is Min.

As with Pearl's game, the probability of correct decision at g is thus

$$\begin{aligned}\bar{D}(d,k) &= \Pr[\bar{e}_{d-1}(g_1) > \bar{e}_{d-1}(g_2)] + \tfrac{1}{2}\Pr[\bar{e}_{d-1}(g_1) = \bar{e}_{d-1}(g_2)] \\ &= \sum_{j=0}^{2^{k-1}} \{\Pr[\bar{e}_{d-1}(g_1) \geq j+1, \bar{e}_{d-1}(g_2) = j] \\ &\qquad + \tfrac{1}{2}\Pr[\bar{e}_{d-1}(g_1) = \bar{e}_{d-1}(g_2) = j)\} \end{aligned} \tag{35}$$

since $\mathrm{sq}(g_1) = \mathrm{sq}(g_2) = 2^{k-1}$. If we define

$$\overline{\mathrm{mw}}(i,d,k) = \Pr[\bar{e}_d(g) \geq i \mid h(g) = k, \mathrm{win}(g)] \tag{36}$$

and

$$\overline{\mathrm{ml}}(i,d,k) = \Pr[\bar{e}_d(g) \geq i \mid h(g) = k, \mathrm{loss}(g)], \tag{37}$$

then (analogously to (24)),

$$\begin{aligned}\bar{D}(d,k) = \sum_{j=0}^{2^{k-1}} \{&\overline{\mathrm{mw}}(j+1, d-1, k-1) \\ &\times [\overline{\mathrm{ml}}(j, d-1, k-1) - \overline{\mathrm{ml}}(j+1, d-1, k-1)] \\ &+ \tfrac{1}{2}[\overline{\mathrm{mw}}(j, d-1, k-1) - \overline{\mathrm{mw}}(j+1, d-1, k-1)] \\ &\times [\overline{\mathrm{ml}}(j, d-1, k-1) - \overline{\mathrm{ml}}(j+1, d-1, k-1)]\}\end{aligned}$$

$$= \sum_{j=0}^{2^{k-1}} \tfrac{1}{2}[\overline{\mathrm{mw}}(j, d-1, k-1) + \overline{\mathrm{mw}}(j+1, d-1, k-1)]$$
$$\times [\overline{\mathrm{ml}}(j, d-1, k-1) - \overline{\mathrm{ml}}(j+1, d-1, k-1)]\,. \tag{38}$$

5.2. Monte Carlo simulation results

Analyzing incremental games mathematically is considerably more complicated than analyzing Pearl games, and is not attempted here. However, since the Monte Carlo simulation studies described in Section 4.2 gave good results, it seems reasonable to simulate incremental games in a similar fashion.

As with Pearl's game, we may create n incremental games of depth $k-1$, using a random number generator to determine the arc values in the game trees. Let m be the number of games that are forced wins. For $d = 1, 2, \ldots, k$ and $i = 0, 1, \ldots, 2^{k-1}$, let

$$\overline{\mathrm{mw}}'(i, d-1, k-1) = \text{number of win trees whose depth } d-1 \text{ minimax value is } i \text{ or greater} \Big/ m\,, \tag{39}$$

$$\overline{\mathrm{ml}}'(i, d-1, k-1) = \text{number of loss trees whose depth } d-1 \text{ minimax value is } i \text{ or greater} \Big/ n-m\,. \tag{40}$$

As with Pearl's game, $\overline{\mathrm{mw}}'(i, d-1, k-1)$ approximates $\overline{\mathrm{mw}}(i, d-1, k-1)$ and $\overline{\mathrm{ml}}'(i, d-1, k-1)$ approximates $\overline{\mathrm{ml}}(i, d-1, k-1)$. Thus an approximation $\bar{D}'(d, k)$ of $\bar{D}(d, k)$ can be computed using (38).

Such a simulation has been done for $n = 3200$ and $k = 3, 4, \ldots, 14$. The results are shown in Table 5. As can be seen from this table, pathology does not occur for any size of incremental game examined.

6. Conclusions and Speculations

Sections 2 through 4 of this paper provide a practical example of a nonintuitive phenomenon which had previously been predicted theoretically: that in some games or game trees, searching deeper can consistently degrade the quality of a decision rather than improve it.

An obvious question is why pathology occurs in Pearl's game but not in games such as chess or checkers. Because of the markedly increasing accuracy of the evaluation function as the end of the game approaches, the pathology in Pearl's game does not seem to be due to any deficiency in the evaluation function. Indeed, the author suspects that pathological behavior would occur in Pearl's game for many (if not most) other reasonable evaluation functions.

A more likely cause of pathology was investigated in Section 5. This section described a class of games called incremental games which is identical to Pearl's game except for the following property: the strength of a board position

TABLE 5. Probability of correct decision $D'(d, k)$ as a function of search depth d and node height k in a binary incremental game, produced by a Monte Carlo simulation involving 3200 games for each value of k. In each case the probability of an arc of the game tree having value 1 is $q = 1/2$. R is the number of wins divided by 3200

d \ k	3	4	5	6	7	8	9	10	11	12	13	14
1	0.9602	0.9457	0.9366	0.9097	0.9095	0.8819	0.8904	0.8817	0.8908	0.8722	0.8737	0.8772
2	1.000	0.9743	0.9664	0.9397	0.9308	0.9059	0.9071	0.9006	0.9064	0.8857	0.8948	0.8982
3	1.000	1.000	0.9759	0.9650	0.9461	0.9317	0.9297	0.9181	0.9289	0.9076	0.9218	0.9176
4		1.000	1.000	0.9761	0.9722	0.9461	0.9485	0.9327	0.9435	0.9204	0.9319	0.9222
5			1.000	1.000	0.9755	0.9701	0.9578	0.9537	0.9530	0.9295	0.9394	0.9362
6				1.000	1.000	0.9778	0.9739	0.9621	0.9655	0.9410	0.9484	0.9457
7					1.000	1.000	0.9777	0.9765	0.9702	0.9579	0.9589	0.9539
8						1.000	1.000	0.9837	0.9795	0.9652	0.9687	0.9590
9							1.000	1.000	0.9800	0.9762	0.9736	0.9655
10								1.000	1.000	0.9804	0.9813	0.9732
11									1.000	1.000	0.9807	0.9794
12										1.000	1.000	0.9813
13											1.000	1.000
14												1.000
R	0.1294	0.6962	0.1734	0.6769	0.1906	0.6844	0.1850	0.6747	0.2025	0.6947	0.1831	0.6962

changes in an incremental manner so that closely related nodes (e.g., sibling nodes) have closely related minimax values. In contrast, the relative strengths (and the minimax values) of sibling nodes in Pearl's game are completely independent.

It is likely that in games such as chess and checkers the strength of a board position changes in a fashion closer to incremental games than to Pearl's game. Since the class of incremental games was shown to be nonpathological for the same evaluation function used for Pearl's game, this suggests that the incremental change in node strength in games such as chess and checkers is one of the reasons why such games are not pathological.

REFERENCES

1. Baudet, G.M., On the branching factor of the alpha-beta pruning algorithm, *Artificial Intelligence* **10** (1978) 173–199.
2. Berliner, H.J., A chronology of computer chess and its literature, *Artificial Intelligence* **10** (1978) 201–214.
3. Biermann, A.W., Theoretical issues related to computer game playing programs, *Personal Computing* **2** (1978) 86–88.
4. Fuller, S.H., Gaschnig, J.G., and Gillogly, J.J., Analysis of the alpha-beta pruning algorithm, Department of Computer Science, Carnegie-Mellon University, 1973.
5. Knuth, D.E., and Moore, R.W., An analysis of alpha-beta pruning, *Artificial Intelligence* **6** (1975) 293–326.
6. LaValle, I.H., *Fundamentals of Decision Analysis* (Holt, Rinehart and Winston, New York, 1978).
7. Lindstrom, G., Alpha-beta on evolving game trees, Tech. Rept. UUCS 79-101, Computer Science Department, University of Utah, 1979.
8. Nau, D.S., Quality of decision versus depth of search on game trees, Ph.D. Dissertation, Duke University, Durham, NC, 1979.
9. Nau, D.S., Decision quality as a function of search depth on game trees, Tech. Rep. TR-866, Computer Science Department, University of Maryland, 1980.
10. Nau, D.S., Pathology on game trees: a summary of results, *Proc. First National Conference on Artificial Intelligence* Stanford University, Stanford, CA (1980) 102–104.
11. Nau, D.S., The last player theorem, *Artificial Intelligence* **18**(1) (1982) 53–65.
12. Newborn, M.M., The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores, *Artificial Intelligence* **8** (1977) 137–153.
13. Nilsson, N.J., *Problem-Solving Methods in Artificial Intelligence* (McGraw-Hill, New York, 1971).
14. Pearl, J., Asymptotic properties of minimax trees and game-searching procedures, *Artificial Intelligence* **14** (1980) 113–138.
15. Pearl, J., Colloquium talk, University of Maryland, 1980.
16. Robinson, A.L., Tournament competition fuels computer chess, *Science* **204** (1979) 1396–1398.
17. Thompson, K., Private communication, Bell Telephone Laboratories, Murray Hill, NJ, 1981.
18. Truscott, T.R., Personal communication, Computer Science Department, Duke University, 1979.
19. Truscott, T.R., Minimum variance tree searching, *Proc. First International Symposium on Policy Analysis and Information Systems* Duke University, Durham, NC (1979) 203–209.
20. Truscott, T.R., Private communication, Duke University, Durham, NC, 1981.
21. Tummala, V.M.R., *Decision Analysis with Business Applications* (Intext, New York, 1973).

Received February 1981; revised version received January 1982