Pathology on Game Trees Revisited, and an Alternative to Minimaxing*

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

ABSTRACT

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 the game tree with such procedures would in general yield a better decision. However, recent investigations have revealed the existence of many game trees and evaluation functions which are 'pathological' in the sense that searching deeper consistently degrades *the decision.*

This paper extends these investigations in two ways. First, it is shown that whenever the evaluation function satisfies certain properties, pathology will occur on any *game tree of high enough constant branching factor. This result, together with Monte Carlo studies on actual games, gives insight into the causes of pathology. Second, an investigation is made of a possible cure for pathology: a probabilistic decision procedure which does not use minimaxing. Under some conditions, this procedure gives results superior to minimaxing.*

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, 20, 22]. However, a recent investigation by Nau [8, 9, 12] demonstrated the existence of a large class of game trees which are *pathological* in the following sense: for a wide class of evaluation functions, searching deeper consistently

*This work was supported by NSF Grants ENG-7822159 and MCS-8117391 to the Laboratory for Pattern Analysis at the University of Maryland.

Artificial Intelligence 21 (1983) 221–244
0004–3702/83/0000–0000/$03.00

degrades the quality of the decision. Pathology has been more recently investigated by Beal [2], Bratko and Gams [5], and Pearl [19].

Given that pathology can occur, why does it not occur in games such as chess and checkers? This question has been investigated by both Nau [10] and Pearl [19].

Nau [10] has examined two closely related classes of games, one pathological and one nonpathological. Examination of the differences between these two classes of games suggests that if the amount of 'strength' or 'weakness' in a board position tends to be roughly the same for sibling nodes, then pathology is less likely. This may be one reason why games such as chess and checkers are not pathological.

Pearl [19] has demonstrated that for a certain class of games and a wide class of evaluation functions, the evaluation function error must be reduced by over 50% at each successive search depth in order to avoid pathology. However, if these game trees are modified to include leaf nodes at random at all levels of the game tree, pathology disappears. The occurrence of such 'traps' or quick endings to the game may be another reason for the absence of pathology in chess, checkers, and similar games.

One purpose of the current paper is to further investigate the conditions under which pathology occurs. To this end, a theorem is proved which generalizes the pathology theorems proved in [8, 12]. This theorem states that if certain restrictions on the game tree and evaluation function are satisfied, pathology will occur whenever the branching factor of the game tree is sufficiently large. This theorem, together with the results of Monte Carlo studies on pathological and nonpathological games, provides additional support for the conjecture that pathology is less likely when sibling nodes have similar strengths.

This paper also investigates a possible cure for pathology. Pearl [19] has suggested that pathology might be avoided by mapping the usual static evaluation function into one which returns the probability that a node is a forced win, and replacing the minimax decision procedure by a decision procedure which treats the evaluation function values as independent probabilities. We examine a modification of this approach which takes an ordinary evaluation function, normalizes its values to fall in the interval [0, 1], and uses these values as approximations of the probability that a node is a forced win.

Using Monte Carlo simulation techniques, the probability of making a correct choice of move using this probabilistic decision procedure is investigated for two classes of games—a class of pathological games and a class of nonpathological games. For the nonpathological games, the procedure yields a probability of correct decision almost exactly the same as that obtained by a minimax search to the same depth. For the pathological games, the probabilistic decision procedure avoids pathology, and gives significantly higher probabilities of correct decision than minimaxing.

Intuitively, if the probabilistic decision procedure has a better probability of yielding the correct decision than minimaxing on the pathological games, then a player using this procedure should win more of these games than an opponent using minimaxing. To find out how *many* more games, a tournament of 3200 games is played between the two players. Somewhat disappointingly, the probabilistic decision procedure wins only marginally more games than are won by using minimaxing. Possible reasons for this are discussed in the last two sections of this paper.

Section 2 contains preliminary material. Section 3 presents the pathology theorem (which is proved in Appendix A), and Section 4 discusses the related Monte Carlo studies. Section 5 discusses the probabilistic decision procedure and the Monte Carlo studies comparing it to minimaxing. Section 6 contains concluding remarks.

2. Preliminaries

This section contains definitions of many of the terms used in this paper, as well as a few preliminary mathematical results.

By a *game* we mean a zero-sum, perfect information game between two players, in which there is strict alternation of play between the two players. At his move, each player may be allowed only finitely many possible choices of moves (although we do not require that the game end after a finite number of moves).

Let G be the game tree for such a game. Then each node of G corresponds to a position in the game, with the root, root(G), corresponding to the game's beginning. Each arc of G corresponds to a move in the game, 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 path from root(G) to position g; i.e., the number of moves it takes to get to node g. By depth(G) we mean

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

We define the *utility value* $u(g)$ to be the payoff which the player who moves to g would receive if both players played perfectly from that point on. Since G is a zero-sum game, the payoff for that player's opponent would be $-u(g)$. The utility value of g may be computed using the following 'negamax' formula [6]:

$$u(g) = \begin{cases} \text{the payoff for the player who moves to } g, & \text{if } g \text{ is a leaf,} \\ -\max\{u(g') \mid g' \text{ is a child of } g\}, & \text{otherwise.} \end{cases} \tag{2.1}$$

Let $b > 0$ and $d > 0$ be integers. G is a *b–ary* game tree (or game tree of *branching factor b*) if every nonleaf node of G has exactly b children. G is *complete to depth d* if no node of depth less than d is a leaf.

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 [23].

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

$$mm(d, g) = \begin{cases} e(g), & \text{if } d = 0, \\ -\max\{mm(d-1, g') \mid g' \text{ is a child of } g\}, & \text{otherwise.} \end{cases} \tag{2.2}$$

Pruning procedures such as alpha-beta [6, 14], SCOUT [16], SSS* [21] and B* [3], have been developed to speed the computation of $\mathrm{mm}(d, g)$.

One way to choose a move at node g is to choose whichever child g' of g has the highest depth $d-1$ minimax value $\mathrm{mm}(d-1, g')$. If more than one node has this value, the choice may be made at random from the set of all children of g having this value. We call this decision procedure a *depth d minimax search*, since it involves evaluating the nodes at depth d in the subtree of G rooted at g.

Consider the equation

$$x = (1-(1-x)^b)^b .$$

It is known [11] that for each value of b, this equation has exactly one solution in the interval (0, 1). This value we call $w(b)$. As an example, $w(2) = \frac{1}{2}(3-\sqrt{5})$, or approximately 0.382. Other values of $w(b)$ are given in [11]. $w(b)$ is a monotonically decreasing function, with

$$\lim_{b \to \infty} w(b) = 0 . \tag{2.3}$$

Other properties of $w(b)$ have been investigated by Nau [8, 11], Baudet [1] and Pearl [16].

Let $0 \leqslant x \leqslant 1$, and let $b \geqslant 0$ and $i \geqslant 0$ be integers. We define

$$s_i(b, x) = \begin{cases} x, & \text{if } i = 0, \\ 1-(1-s_{i-1}(b, x))^b, & \text{if } i > 0 \text{ is odd}, \\ (s_{i-1}(b, x))^b, & \text{if } i > 0 \text{ is even.} \end{cases} \tag{2.4}$$

It is proved [11] that

$$\lim_{i \to \infty} s_i(b, x) = \begin{cases} 1 & \text{if } x > w(b), \\ 0 & \text{if } x < w(b), \end{cases} \tag{2.5}$$

and that if $x = w(b)$, then

$$s_i(b, x) = \begin{cases} w(b), & \text{if } i \text{ is even,} \\ 1 - w(b), & \text{if } i \text{ is odd.} \end{cases} \tag{2.6}$$

$s_i(b, x)$ is important in analyzing the probability that a decision is correct when a minimax search is used.

3. A New Theorem about Pathology

By choosing either accurate or inaccurate evaluations for nodes far down in a game tree G, evaluation functions can be constructed whose performance either improves or degrades at increasing search depths. Thus no universal statements can be made about the quality of evaluation functions as a function of search depth. To avoid this difficulty, we consider the error in $e(g)$ to be stochastic in nature. Thus the members of $\{e(g) \mid g \text{ is a node of } G\}$ are taken to be discrete random variables (which are not necessarily independent or identically distributed).

Let h be the highest value that e is capable of returning on any node of G. Then $p > 1$ is a *dependence bound* for e if for every node g in G and for every set of constraints C on any subsets of $\{e(g') \mid g' \neq g \text{ and } \text{depth}(g') = \text{depth}(g)\}$,

$$\Pr[e(g) = h \mid C] \geqslant \Pr[e(g) = h]/p .$$

This means that the probability that e returns h depends only to a limited extent on the values of other nodes at the same level.

The evaluation functions investigated in [8, 12] are special cases of evaluation functions with dependence bounds. Theorem 3.1 below deals with the set of all evaluation functions having dependence bounds, and deals with a much more general set of game trees than the game trees investigated in [8, 12]. Thus the pathology theorem in [8, 12] is a special case of Theorem 3.1.

Theorem 3.1. *Let G be a b-ary game tree, and let g be a node of G. Let e be an evaluation function for G with dependence bound p,*[1] *and let*

$$c = \inf_{g \in G} \Pr[e(g) = h] ,$$

where h is the highest value that e can return on any node of G. Let $\text{tie}(d, g)$ *denote the event that all children of g receive the same minimax value when a depth d minimax search is done. If the subtree rooted at g is complete to at least depth d, then*

$$\Pr[\text{tie}(d, g)] > (s_{d-1}(b, c/p))^b . \tag{3.1}$$

[1]Theorem 3.1 could easily be generalized by assuming that the dependence bound p and the resulting equation (3.1) only apply to the first k levels of the game tree for some k. To avoid complicating the presentation, we do not do this here.

Proof. See the Appendix A.

Corollary 3.2. *Suppose that the hypotheses of Theorem* 3.1 *hold and that G is complete to every depth* $d < \infty$. *If* $c/p > w(b)$, *then*

$$\lim_{d \to \infty} \Pr[\text{tie}(d, g)] = 1 .$$

Proof. Immediate from Theorem 3.1 and (2.5).

The significance of Theorem 3.1 is as follows. Suppose the hypotheses of Theorem 3.1 hold on some game of depth k whose branching factor b is such that $c/p > w(b)$ (from (2.3), this is achieved if b is sufficiently large). Then from (2.5),

$$\lim_{d \to \infty} s_d(b, c/p) = 1 .$$

As shown in [16], this convergence is superexponentially fast. But from Theorem 3.1, as long as $d < k$,

$$\Pr[\text{tie}(d, g)] > (s_{d-1}(b, c/p))^b .$$

This means that as long as the search depth is less than k, increasing the search depth makes it increasingly likely that all nodes get exactly the same minimax value h. Thus with increasing search depth, it becomes increasingly likely that the choice of move must be made at random, with all possible choices of move being equally likely.

4. Two Games

Theorem 3.1 suggests that games with large branching factors are more likely to be pathological. In this section, we examine the behavior of two classes of games as a function of the branching factor.

4.1. Descriptions of the games

The following is a trivial generalization of a class of games invented by Judea Pearl [17] for use in analyzing the computational complexity of game tree search procedures. A P-game (known in [10] as a Pearl game) is played on a chessboard measuring $b^{k/2}$ by $b^{k/2}$ rather then 8 by 8, where $b > 1$ and $k > 0$ are integers. The initial playing board for a P-game is constructed by randomly assigning each square of the board the value 1 with probability p or the value -1 with probability $1 - p$, where p is a constant and $0 \leq p \leq 1$ (for example, see Fig. 1). For reasons described in [10], we choose $p = w(b)$.

The two players move in strict alteration. A move for the first player consists of dividing the board vertically into b sections of equal width, and discarding all but one of the sections. A move for his opponent consists of dividing what

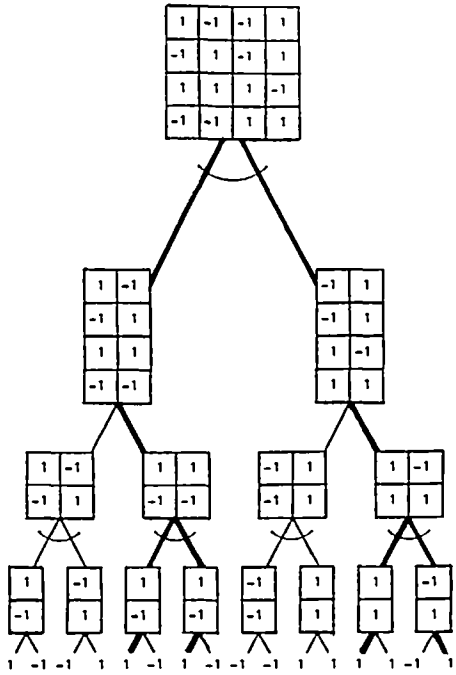

FIG. 1. A game tree for a binary P-game of depth 4. The initial playing board appears at the root of the tree. Since the depth is even, the second player is the 'last player'. The last player has a forced win in this particular game tree, as indicated by the solution tree drawn in boldface.

remains of the board horizontally into b sections of equal height, and discarding all but one of them. 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 -1, his opponent wins. Note that the same player will always have the last move. We call this player the *last player.*

The game tree for a P-game is a complete b-ary game tree of depth k, with random, identically distributed leaf node values (for example, see Fig. 1). For this reason, the value of a node in a P-game is independent of the values of other nodes at the same depth. Such independence does not occur in games such as chess or checkers. In these games, the board positions usually change incrementally, so that each node is likely to have children of similar strength.

The following games were originally defined in [10], where they were called incremental games. The definition was inspired by a similar game tree model defined by Newborn [13] and later used by Lindstrom [7]. This class of games provides one way to model incremental variation in node strength. An N-game has the same size playing board, the same moves, and the same criterion for winning as a P-game, but the initial playing board is set up differently. To set

up the board, each arc of the game tree is independently, randomly given the value 1 with probability q or -1 with probability $1-q$, where q is a constant and $0 \leq q \leq 1$. (We arbitrarily choose $q = \frac{1}{2}$.) 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 is given the value 1 if the corresponding leaf node of the game tree has positive strength, and the value -1 otherwise.

4.2. An evaluation function

Let G be a game tree for a b-ary P-game or N-game, and g be a node in G. We denote the depth of the subtree rooted at g by height(g), and the number of squares in g by size(g). Thus size(g) = $b^{\text{height}(g)}$. If the player to move to g is the 'last player' defined at the beginning of Section 4.1, then the more '1' squares there are in g the more likely it is that g is a win. If it is the other player that has the move to g, then the more '-1' squares there are in g the more likely it is that g is a win. Thus an obvious evaluation function for G is

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

where

$$v = \begin{cases} 1, & \text{if the last player has the move to } g, \\ -1, & \text{otherwise.} \end{cases}$$

Suppose height(g) = k, and suppose a depth d minimax search is used to choose between two children g' and g'' of g, where g' is a forced win and g'' is a forced loss. Then the *correct* decision is to move to g'. Now, g will always be choosen if $\text{mm}(d-1, g') > \text{mm}(d-1, g'')$, and g' will be chosen half of the time if $\text{mm}(d-1, g') = \text{mm}(d-1, g'')$. Thus the correct decision will be made with probability

$$D = \Pr[\text{mm}(d-1, g') > \text{mm}(d-1, g'')] + \tfrac{1}{2}\Pr[\text{mm}(d-1, g') = \text{mm}(d-1, g'')]. \tag{4.1}$$

4.3. Results about the games

Monte Carlo simulation techniques can be used to find out whether P-games and N-games are pathological when the branching factor is large. Given integers b, k, and n, we can create as many b-ary P-games (or N-games) of depth k as we wish, using a random number generator to determine which squares have value -1 and which have value $+1$. We discard games for which the root does not have at least one forced win child and one forced loss child. For each of the games that is left, we select one forced win child g' of the root and one forced loss child g'' of the root, and compute $\text{mm}(d-1, g')$ and $\text{mm}(d-1, g'')$. If out of a series of n games we have $\text{mm}(d-1, g') > \text{mm}(d-1, g'')$ on m of them and $\text{mm}(d-1, g') = \text{mm}(d-1, g'')$ on p of them,

then the fraction $D' = (m + \frac{1}{2}p)/n$ approximates the probability of correct decision D.

To test whether increasing the branching factor of a game causes it to become pathological, it is important that the game be nonpathological for small branching factors. Thus we use $k \leq 7$ since pathology does not occur in binary P-games and N-games when $k \leq 7$ [10]. To avoid examining games that are trivially short, we use $k \geq 5$. Using $k = 5, 6, 7$ and $n = 3200$, Monte Carlo simulations have been done on b-ary P-games and N-games for several values of b. The results are displayed in Table 1. As can be seen, none of the N-games is pathological. However, P-games of sufficiently large branching factor are pathological in the following sense: as long as the search terminates at nodes more than one move away from the end of the game (i.e., $d < k - 1$), the probability of correct decision decreases as the search depth increases. These results are interesting for two reasons, as discussed below.

TABLE 1. The fraction D' of correct decisions at nodes of height k in b-ary P-games and N-games, as a function of search depth d using minimaxing with the evaluation function e_1. The results come from Monte Carlo simulations involving 3200 games for each combination of k and b. Note that (1) on the P-games tested, pathology occurs if b is made sufficiently large, and (2) on the N-games tested, pathology does not occur

		P-games					N-games				
k	d	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$
5	1	0.842	0.760	0.711	0.692	0.675	0.941	0.936	0.959	0.968	0.985
5	2	0.867	0.779	0.701	0.671	0.649	0.969	0.967	0.976	0.987	0.997
5	3	0.891	0.777	0.708	0.664	0.638	0.982	0.976	0.987	0.994	0.998
5	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
5	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
6	1	0.809	0.702	0.656	0.637	0.619	0.936	0.941	0.966	0.972	0.985
6	2	0.806	0.699	0.641	0.612	0.602	0.953	0.961	0.978	0.990	0.996
6	3	0.825	0.713	0.632	0.602	0.592	0.976	0.978	0.992	0.996	0.999
6	4	0.833	0.692	0.619	0.577	0.565	0.987	0.983	0.992	0.999	0.999
6	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
6	6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
7	1	0.762	0.666	0.613	0.594	*[a]	0.924	0.936	0.948	0.969	*
7	2	0.769	0.655	0.602	0.566	*	0.955	0.960	0.974	0.992	*
7	3	0.777	0.653	0.606	0.559	*	0.964	0.964	0.977	0.992	*
7	4	0.797	0.640	0.586	0.555	*	0.980	0.982	0.988	0.996	*
7	5	0.811	0.624	0.569	0.538	*	0.985	0.985	0.994	0.998	*
7	6	1.000	1.000	1.000	1.000	*	1.000	1.000	1.000	1.000	*
7	7	1.000	1.000	1.000	1.000	*	1.000	1.000	1.000	1.000	*

[a] * indicates that it was not feasible to compute results for $b = 6$ when $k = 7$, because of the excessive computing time required when b and k are large.

First, recall that Theorem 3.1 predicts the occurrence of pathology for large branching factors on games satisfying certain conditions. Neither the class of P-games nor the class of N-games satisfies all of these conditions, yet pathology occurs for large branching factors anyway on the P-games. Clearly, the consequences of the theorem hold for a larger class of games than those satisfying its preconditions.

Second, it is interesting to note that when the evaluation function e_1 is used, the P-games satisfy more of the preconditions of Theorem 3.1 than the N-games do. In particular, the preconditions of Theorem 3.1 are as follows:

(1) The probability of the evaluation function returning the highest value in its range must be positive for all nodes in the game tree. (This is because $w(b) > 0$ for all b, whence c must be positive in order to obtain $c > w(b)$ for sufficiently large b.) Neither the P-games nor the N-games satisfy this condition when e_1 is used, for $e_1(g)$ cannot possibly return 1 if g is a forced loss node.

(2) The probability that the evaluation function returns the highest value in its range must depend only to a limited extent on the values of other nodes at the same depth. (This is the definition of an evaluation function with dependence bound.) The class of P-games satisfies this condition when e_1 is used, since sibling nodes are independent. However, the class of N-games does not satisfy this condition. A proof of this may be constructed by examining any leaf node g for which $e_1(g') = 1$ for every sibling g' of g. The more such siblings g has (each of which has $e_1(g') = 1$), the more likely it is that the parent of g is a node of sufficiently high strength that $e_1(g)$ must also be 1.

This suggests that games satisfying the second condition are more likely to be pathological for large branching factors than games that do not.

5. A Decision Criterion Better than Minimaxing?

Our entire discussion of pathology has been predicated on the use of minimaxing. One might want to consider what happens when other kinds of decision procedures are used. Pearl [18, 19] has proposed the following approach. Suppose we have a special evaluation function $e^*(g)$ which returns the probability that g is a forced win (for the player to move to g) 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 $e^*(g')$ for every node g' 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

$$\Pr[h \text{ is a win node}] = \{1 - \Pr[h' \text{ is a win node}] \mid h' \text{ is a child of } h\} \qquad (5.1)$$

to successively shallower nodes in the subtree. Pearl suggests that the same product rule should be used to propagate $e(g)$ even when $e(g)$ only approximates the actual probability that g is a forced win given its examined features. The procedure 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 independent probabilities of the occurrence of forced wins. This amounts to replacing the 'negamax formula'

$$\mathrm{mm}(d, g) = \begin{cases} e(g), & \text{if } d = 0, \\ -\max\{\mathrm{mm}(d-1, g') \mid g' \text{ is a child of } g\}, & \text{otherwise.} \end{cases} \tag{5.2}$$

by the formula

$$\mathrm{pe}(d, g) = \begin{cases} e(g), & \text{if } d = 0, \\ \prod\{1 - \mathrm{pe}(d-1, g') \mid g' \text{ is a child of } g\}, & \text{otherwise.} \end{cases} \tag{5.3}$$

We call $\mathrm{pe}(d, g)$ the *depth d probability estimate for g*.

One difficulty with the above approach is that it is not clear how to estimate the probability $e^*(g)$ that g is a forced win given the features of g. Another difficulty is that the multiplication rule used in (5.1) and (5.3) assumes independent probabilities. Since the characteristics of sibling modes may be interdependent (as they are, for example, in N-games), this rule is incorrect for many games.

Regardless of these difficulties, this approach still should be investigated. It may not yield completely accurate values, but neither does minimaxing—and perhaps it will avoid the pathology that can arise when minimaxing is done. In this section, we examine the following questions.

(1) Can pathology be avoided by using probability estimation?

(2) In games which are not pathological, does probability estimation yield as high a probability of correct decision as minimaxing?

(3) Even if probability estimation yields a higher probability of correct decision than minimaxing, does it allow significantly more games to be won?

5.1. Probability estimation on P-games

Can pathology be avoided by using probability estimation? As an answer to this question, Table 2 contains the results of a Monte Carlo simulation on P-games. The simulation was similar to that of Section 4, but was done using probability estimation instead of minimaxing. Because of computational difficulties, only binary games were investigated. Since the evalutaiton function e_1 returns values between 0 and 1, it was used without modification.

For purposes of comparison, Table 3 contains analytic results taken from [10] concerning minimaxing in binary P-games, and Table 4 gives the percentage differences between the values contained in Table 2 and the corresponding values in Table 3. Note the following patterns.

(1) When minimaxing is used (Table 3) and k is larger than about 7 or 8, pathology occurs, in the following sense: as long as the search does not go

TABLE 2. 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 P-games. The results come from a Monte Carlo simulation involving 3200 games for each value of k

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.947	0.906	0.842	0.809	0.762	0.729	0.694	0.670	0.643	0.621	0.620
2	1.000	0.934	0.876	0.816	0.772	0.732	0.695	0.669	0.641	0.620	0.619
3	1.000	1.000	0.904	0.840	0.779	0.746	0.698	0.673	0.642	0.623	0.619
4		1.000	1.000	0.865	0.815	0.754	0.708	0.676	0.647	0.625	0.622
5			1.000	1.000	0.837	0.782	0.718	0.683	0.643	0.629	0.622
6				1.000	1.000	0.802	0.747	0.684	0.654	0.630	0.623
7					1.000	1.000	0.777	0.710	0.664	0.637	0.628
8						1.000	1.000	0.747	0.695	0.646	0.638
9							1.000	1.000	0.711	0.658	0.637
10								1.000	1.000	0.684	0.660
11									1.000	1.000	0.677
12										1.000	1.000
13											1.000

TABLE 3. 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 P-games. The results were computed analytically in [10]

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.947	0.902	0.849	0.805	0.765	0.731	0.701	0.675	0.652	0.633	0.616
2	1.000	0.914	0.872	0.807	0.769	0.725	0.695	0.666	0.644	0.623	0.607
3	1.000	1.000	0.893	0.830	0.773	0.730	0.692	0.663	0.638	0.617	0.600
4		1.000	1.000	0.825	0.790	0.728	0.694	0.658	0.633	0.611	0.594
5			1.000	1.000	0.806	0.741	0.691	0.658	0.629	0.607	0.589
6				1.000	1.000	0.698	0.705	0.650	0.627	0.602	0.585
7					1.000	1.000	0.699	0.656	0.619	0.599	0.581
8						1.000	1.000	0.581	0.640	0.585	0.574
9							1.000	1.000	0.592	0.595	0.570
10								1.000	1.000	0.517	0.592
11									1.000	1.000	0.521
12										1.000	1.000
13											1.000
14											
15											

TABLE 4. Percentage difference $100\% \times (D' - D)/D$, where D' is taken from Table 2 and D is taken from Table 3

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.0%	0.4%	−0.8%	0.5%	−0.4%	−0.3%	−1.0%	−0.7%	−1.4%	−1.9%	0.6%
2	0.0%	2.2%	0.5%	1.1%	0.4%	1.0%	0.0%	0.5%	−0.5%	−0.5%	2.0%
3	0.0%	0.0%	1.2%	1.2%	0.8%	2.2%	0.9%	1.5%	0.6%	1.0%	3.2%
4		0.0%	0.0%	4.8%	3.2%	3.6%	2.0%	2.7%	2.2%	2.3%	4.7%
5			0.0%	0.0%	3.8%	5.5%	3.9%	3.8%	2.2%	3.6%	5.6%
6				0.0%	0.0%	14.9%	6.0%	5.2%	4.3%	4.7%	6.5%
7					0.0%	0.0%	11.2%	8.2%	7.3%	6.3%	8.1%
8						0.0%	0.0%	28.6%	8.6%	10.4%	11.1%
9							0.0%	0.0%	20.1%	10.6%	11.8%
10								0.0%	0.0%	32.3%	11.5%
11									0.0%	0.0%	9.9%
12										0.0%	0.0%
13											0.0%

closer than one move away from the end of the game (i.e., $d < k - 1$), the probability of correct decision decreases as the search depth increases.

(2) When probability estimation is used (Table 2), pathology does not occur for any tested value of k. Instead, for all values of k examination, the probability of correct decision increases as the search depth increases.

(3) In most cases, the probability of correct decision using probability estimation is greater than or equal to the probability of correct decision using a minimax search to the same depth (see Table 4). However, the difference is not very large except for large search depths.

Thus, overall, probability estimation appears to be somewhat better than minimaxing on P-games.

5.2. Probability estimation on N-games

In games which are nonpathological, is probability estimation as accurate as minimaxing? It would be difficult to answer this question for all games. However, one can certainly examine specific classes of games. Table 5 contains Monte Carlo results about the probability of correct decision for probability estimation in binary N-games. For purposes of comparison, Table 6 contains Monte Carlo results taken from [10] concerning minimaxing in binary N-games. As shown in Table 7, probability estimation appears to perform slightly more poorly in N-games than minimaxing does. However, the percent difference between the probability of correct decision is in all cases less than 1.3%. This figure is small enough that the differences may have been caused by random variation in the Monte Carlo simulations. Thus on at least one class of

TABLE 5. 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 N-games. The results come from a Monte Carlo simulation involving 3200 games for each value of k

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.982	0.970	0.941	0.936	0.924	0.933	0.914	0.920	0.914	0.913	0.910
2	1.000	0.981	0.963	0.949	0.944	0.945	0.929	0.928	0.929	0.924	0.922
3	1.000	1.000	0.981	0.973	0.959	0.962	0.943	0.940	0.940	0.937	0.931
4		1.000	1.000	0.985	0.972	0.968	0.957	0.948	0.950	0.943	0.936
5			1.000	1.000	0.983	0.978	0.968	0.964	0.957	0.952	0.948
6				1.000	1.000	0.988	0.980	0.973	0.962	0.962	0.952
7					1.000	1.000	0.987	0.984	0.972	0.969	0.960
8						1.000	1.000	0.992	0.979	0.977	0.967
9							1.000	1.000	0.987	0.984	0.976
10								1.000	1.000	0.991	0.978
11									1.000	1.000	0.992
12										1.000	1.000
13											1.000

TABLE 6. 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 N-games. The results come from a Monte Carlo simulation involving 3200 games for each value of k

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.982	0.970	0.941	0.936	0.924	0.933	0.914	0.920	0.914	0.913	0.910
2	1.000	0.978	0.969	0.953	0.955	0.947	0.938	0.939	0.934	0.924	0.926
3	1.000	1.000	0.982	0.976	0.964	0.959	0.952	0.950	0.940	0.944	0.935
4		1.000	1.000	0.987	0.980	0.966	0.962	0.960	0.950	0.951	0.943
5			1.000	1.000	0.985	0.979	0.968	0.969	0.958	0.958	0.947
6				1.000	1.000	0.985	0.983	0.974	0.965	0.963	0.959
7					1.000	1.000	0.984	0.983	0.975	0.972	0.965
8						1.000	1.000	0.988	0.982	0.978	0.976
9							1.000	1.000	0.986	0.987	0.981
10								1.000	1.000	0.988	0.985
11									1.000	1.000	0.986
12										1.000	1.000
13											1.000

Note. The above figures are better approximations of D' than the corresponding figures given in [10].

TABLE 7. Percentage difference $100\% \times (D'_5 - D'_6)/D'_6$, where D'_5 and D'_6 are the values of D' given in Tables 5 and 6, respectively

d	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$
1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2	0.0%	0.3%	−0.6%	−0.4%	−1.2%	−0.2%	−1.0%	−1.2%	−0.5%	0.0%	−0.4%
3	0.0%	0.0%	−0.1%	−0.3%	−0.5%	0.3%	−0.9%	−1.1%	0.0%	−0.7%	−0.4%
4		0.0%	0.0%	−0.2%	−0.8%	0.2%	−0.5%	−1.3%	0.0%	−0.8%	−0.7%
5			0.0%	0.0%	−0.2%	−0.1%	0.0%	−0.5%	−0.1%	−0.6%	0.1%
6				0.0%	0.0%	0.3%	−0.3%	−0.1%	−0.3%	−0.1%	−0.7%
7					0.0%	0.0%	0.3%	0.1%	−0.3%	−0.3%	−0.5%
8						0.0%	0.0%	0.4%	−0.3%	−0.1%	−0.9%
9							0.0%	0.0%	0.1%	−0.3%	−0.5%
10								0.0%	0.0%	0.3%	−0.7%
11									0.0%	0.0%	0.6%*
12										0.0%	0.0%
13											0.0%

nonpathological games, probability estimation performs almost identically to minimaxing.

5.3. Probability estimation versus minimaxing: two contests

Since probability estimation has a higher probability of correct decision than minimaxing on P-games, a player using it will generally make better moves than an opponent using minimaxing. Thus the player using probability estimation should win more games than his opponent, but it is not clear how many more. To find out how many more, the two decision procedures were pitted against each other in another Monte Carlo simulation. The simulation worked as follows: 1600 binary P-games of depth $k = 10$ were generated. For each game, and for each possible search depth d, two games were played: one in which the player using probability estimation played first and one in which the player using minimaxing played first. Thus both players and the same number of games on which they had forced wins, as well as the same number of chances to be the first player.

The above contest was done with both players using the evaluation function e_1. Let us define

$$e_2(g) = f(h, e_1(g))$$

where h is the height of g in the game tree, b is the branching factor (in this case 2), and

$$f(h, x) = \begin{cases} x, & \text{if } h = 0, \\ f(h-1, x^b), & \text{if } h > 0 \text{ is odd}, \\ (1-(1-f(h-2, x))^b)^b, & \text{if } h > 0 \text{ is even}. \end{cases}$$

Pearl [15] has suggested that $e_2(g)$ is a better approximation of $e^*(g)$ than $e_1(g)$ is, and that therefore probability estimation might do better in relation to minimaxing if e_2 is used instead of e_1. To test this conjecture, a second contest between probability estimation and minimaxing was run using e_2. The results of both contests are given in Table 8.

The results of the two contests were somewhat disappointing: probability estimation performed only marginally better than minimaxing. In fact, much of the difference in performance could be due to random variation in the Monte Carlo simulation.

Since probability estimation was shown in Tables 2–4 to have a higher probability of correct decision than minimaxing on binary P-games, Table 8 at first appears inconsistent with these tables. However, further thought reveals that no inconsistency exists. Tables 2–4 deal only with the probability of correct decision at individual positions in a game. However, winning a game depends not just on one correct decision, but on the entire series of decisions made by both players.

If two players E and M are playing a game and M makes a wrong move at one point, E will not necessarily win the game unless he takes advantage of M's mistake and continues to make good moves from that point on. If E makes a mistake later, then M may possibly recover and win the game. Thus unless E is

TABLE 8. Number of P-games won by a player using probability estimation playing against a player using minimaxing, with both players searching to the same depth d. The results come from Monte Carlo simulations of 1600 games each. For each search depth d, each player had a chance to start first, giving a total of 3200 games for each value of d

	Both players using $e_1(g)$		Both players using $e_2(g)$	
Search depth d	Number of wins	Percentage of wins	Number of wins	Percentage of wins
1[a]	1600	50.0%	1600	50.0%
2	1590	49.7%	1630	50.9%
3	1529	47.8%	1604	50.1%
4	1671	52.2%	1732	54.1%
5	1603	50.1%	1677	52.4%
6	1729	54.0%	1757	54.9%
7	1646	51.4%	1624	50.8%
8	1722	53.8%	1718	53.7%
9[a,b]	1600	50.0%	1600	50.0%
10[a,b]	1600	50.0%	1600	50.0%

[a]For search depths 1, 9, and 10, both players play identically.
[b]For search depths 9 and 10, both players play perfectly.

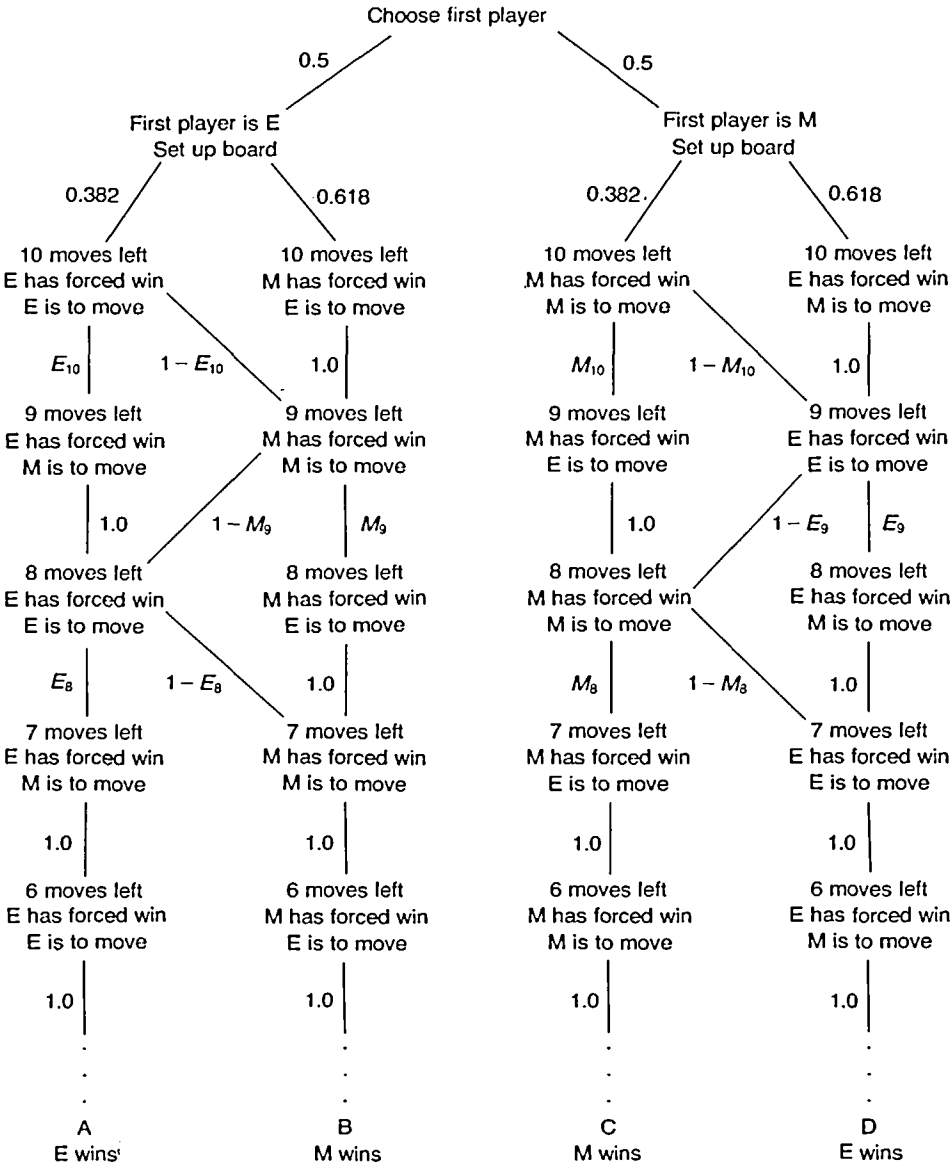

FIG. 2. A Markov chain representing all possible outcomes of binary P-games of depth 10 between a player E who uses probability estimation and a player M who uses minimaxing, with both players using the evaluation function e_1. States corresponding to game tree nodes of height less than 6 are omitted since both players are playing perfectly from height 7 on.

playing significantly better than M throughout the game, E will not win significantly more games than M. Suppose E is using probability estimation and M is using minimaxing, with both players using the evaluation function e_1. As can be seen from Table 4, player E will play significantly better at certain points in a P-game than player M, but for most of the game E's probability of correct decision is not significantly better than M's.

As an example, consider P-games of depth 10 played by the players E and M, where both players are searching to depth 6. The possible courses of events in these games are modeled by the Markov chain of Fig. 2. The first two levels of the chain represent choosing who goes first and setting up the playing board (it can be shown [11] that the first player has a forced win at the beginning of the game with probability 0.382). Each subsequent state represents which player is to move and which player has a forced win, and the arcs leaving the state represent choices of correct or incorrect moves. The probability that player E wins is the sum of the probabilities of reaching either state A or state D.

The transition probabilities E_j and M_j in Fig. 2 are the probabilities of correct decision for E and M, respectively, when j moves are left. If we approximate E_j and M_j by the figures given for $d = 6$ and $k = j$ in Tables 2 and 3, respectively,[2] then we get $E_{10} = 0.684$, $E_9 = 0.747$, and $E_8 = 0.802$, whereas $M_{10} = 0.650$, $M_9 = 0.705$, and $M_8 = 0.698$. From simple algebra, it follows that the probability that the game will end at a node where E wins is 0.539, which is quite close to the figure 0.540 given in Table 8.

6. Summary and Discussion

This paper has dealt with two main topics: an investigation of some of the conditions under which pathology occurs, and an examination of a decision procedure which is in some respects better than minimaxing.

6.1. Conditions which cause pathology

Since the discovery of game tree pathology in 1979 [8], a major open question has been why pathology does not occur in games such as chess and checkers.

In games such as chess and checkers, some game positions are 'strong' and other are 'weak', and since the board positions usually change incrementally, the children of any given node are all likely to have similar strengths. In [10], it was conjectured that this is one reason why such games are not pathological.

[2] This approximation is by no means exact, since the values in Tables 2 and 3 deal only with the case where one of the two children of the current node is a forced win and the other is a forced loss, and where the a priori probability of a leaf node being a forced win for the last player is 0.382. In actuality, both, one, or neither of the children may be forced wins; and the probability of a leaf node being a forced win will be biased from the initial value of 0.382 as the moves taken by the players lead toward subtrees containing greater or smaller numbers of forced win nodes.

The first part of the current paper contains an investigation into the causes of pathology which supports this conjecture.

The first part of this investigation has consisted of formulating and proving a theorem (Theorem 3.1) stating that for game trees satisfying certain preconditions, pathology will occur if the branching factor is sufficiently large. One of the conditions necessary for this theorem to be applicable is the following independence condition: the value returned by the evaluation function on a node of the game tree must be dependent only to a limited extent on the values returned on the node's siblings. This condition clearly will not be satisfied if similar nodes have similar strengths and if the evaluation function value of a node correlates with the node's strength.

This result is further supported by Monte Carlo simulations on two classes of games, which are called P-games and N-games. For the evaluation function used for these games, neither class of games satisfies all of the preconditions of Theorem 3.1, but the class of P-games does satisfy the independence condition described above. It is therefore quite interesting to note that pathology occurs on the P-games when the branching factor is made sufficiently large, but does not occur on the N-games (which do not satisfy this precondition) for any of the branching factors which have been tried.

It should be noted that none of the studies in this paper deal with what happens when a game tree search reaches a leaf node. Thus the results of these studies do not conflict with Pearl's conjecture [19] 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 contributing to the lack of pathology in these games.

6.2. An alternative to minimaxing

The other topic discussed in this paper is a game tree search procedure which does not use minimaxing, but instead uses a 'probability estimation' procedure.

Using Monte Carlo simulations, this procedure is compared against minimaxing, with both procedures searching to the same depth, on the P-games and N-games mentioned above. On the N-games, the probabilities of correct decision for probability estimation and minimaxing are almost identical; and on the P-games, probability estimation yields a higher probability of correct decision than minimaxing. However, further Monte Carlo studies indicate that probability estimation performs only marginally better than minimaxing in terms of the number of games that it wins against a minimax search to the same depth.

The major reason for the disappointing performance of probability estimation appears to be that in order for one player to win a game over an opponent, the player usually must play significantly better than his opponent throughout the game. As illustrated in Table 4, the probability of correct decision obtained by probability estimation is considerably better than that obtained by mini-

maxing during specific points in a P-game, but it is not significantly better during most of the game. Pearl [19, 18] also suggests that perhaps minimaxing can take advantage of a player's mistakes better than probability estimation, but there seems to be no good way to measure whether this is in fact true.

Possibly another reason for the disappointing performance of probability estimation is that the evaluation functions $e_1(g)$ and $e_2(g)$ are only approximations of the probability that g is a forced win. Better results could possibly be obtained by using an evaluation function $e^*(g)$ which returned the actual probability that g were a forced win given the features of g examined by e^*. However, such a function would be very difficult to compute for most games. Thus probability estimation is probably of limited use, especially since pruning schemes such as alpha-beta [6, 14] and SSS* [21] cannot be used with it.

Appendix A

Lemma A.1. *Let G be a b-ary game tree. Let e be an evaluation function for G with dependence bound p, and let*

$$c = \inf_{g \in G} \Pr[e(g) = h]$$

where h is the highest value e can return on any node of G. For every node g of G, if the subtree rooted at g is complete to at least depth d, then for every set of constraints C on any subset of

$$\{e(g') \mid \text{depth}(g') = \text{depth}(g) + d \text{ and } g' \text{ is not a descendent of } g\},$$

we have

$$\Pr[\text{mm}(d, g) = (-1)^d h \mid C] \geq s_d(b, c/p).$$

Proof (by induction on d). Let g be a node of G. From (2.4) and the definition of an evaluation function with a dependence bound it follows that for every set of constraints C on any subset of $\{e(g') \mid \text{depth}(g') = \text{depth}(g)+0 \text{ and } g' \neq g\}$,

$$\Pr[\text{mm}(0, g) = (-1)^0 h \mid C] = \Pr[e(g) = h \mid C] \geq c/p = s_0(b, c/p).$$

Thus the lemma holds for $d = 0$. Let $j > 0$, and suppose the lemma holds for $d = j - 1$. Let g be a node of G, and let the children of g be $g_1, g_2, \ldots, g_b$. From (2.1) it follows that $\text{mm}(j, g) = (-1)^j e(g')$ for some descendant g' of g. Thus if j is even, then $\text{mm}(j, g) \leq h$, and if j is odd, then $\text{mm}(j, g) \geq -h$. There are two cases to consider.

Case 1. j is even. Then $\text{mm}(j, g) \leq h$, and $\text{mm}(j-1, g) \geq -h$. Thus from (2.1), $\text{mm}(j, g) = h$ if and only if $\text{mm}(j-1, g_k) = -h$ for $k = 1, \ldots, b$. Therefore, for every set of constraints C on any subset of

$$\{e(g') \mid \text{depth}(g') = \text{depth}(g) + j \text{ and } g' \text{ is not a descendent of } g\},$$

$$\begin{aligned}
&\Pr[\mathrm{mm}(j, g) = h \mid C] = \\
&\quad = \Pr[\mathrm{mm}(j-1, g_m) = -h \text{ for all } m \mid C] \\
&\quad = \Pr[\mathrm{mm}(j-1, g_1) = -h \mid C] \\
&\qquad * \Pr[\mathrm{mm}(j-1, g_2) = -h \mid \mathrm{mm}(j-1, g_1) = -h \text{ and } C] \\
&\qquad \vdots \\
&\qquad * \Pr[\mathrm{mm}(j-1, g_m) = -h \mid \mathrm{mm}(j-1, g_k) = -h \\
&\qquad\qquad \text{for } k = 1, \ldots, m-1 \text{ and } C] \\
&\qquad \vdots \\
&\qquad * \Pr[\mathrm{mm}(j-1, g_b) = -h \mid \mathrm{mm}(j-1, g_k) = -h \\
&\qquad\qquad \text{for } k = 1, \ldots, b-1 \text{ and } C]. \qquad \text{(A.1)}
\end{aligned}$$

But for $m = 1, \ldots, b$,

$$\begin{aligned}
&\{e(g') \mid \mathrm{depth}(g') = \mathrm{depth}(g) + j \\
&\qquad \text{and } g' \text{ is not a descendent of } g\} \subseteq S_m,
\end{aligned}$$

where

$$\begin{aligned}
S_m = \{&e(g') \mid \mathrm{depth}(g') = \mathrm{depth}(g_m) + j - 1 \\
&\text{and } g' \text{ is not a descent of } g_m\}.
\end{aligned}$$

Thus C is a set of constraints on a subset of S_m. Furthermore, for $k = 1, \ldots, m-1$, $\mathrm{mm}(j-1, g_k)$ is a function only of $\{e(g') \mid g'$ has depth $j-1$ in the subtree rooted at $g_k\}$, which is a subset of S_m. Thus the statement '$\mathrm{mm}(j-1, g_k) = h$' is also a constraint on a subset of S_m. Therefore, from the induction assumption,

$$\begin{aligned}
&\Pr[\mathrm{mm}(j-1, g_m) = -h \mid \mathrm{mm}(j-1, g_k) = -h \\
&\qquad\qquad \text{for } k = 1, \ldots, m-1 \text{ and } C] \\
&\quad \geq s_{j-1}(b, c/p).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Pr[\mathrm{mm}(j, g) = (-1)^j h \mid C] = \\
&\quad = \Pr[\mathrm{mm}(j, g) = h \mid C] \\
&\quad \geq (s_{j-1}(b, c/p))^b \quad \text{(from (A.1) and (A.2))} \\
&\quad = s_j(b, c/p) \qquad \text{(from (2.4))}.
\end{aligned}$$

Thus the theorem holds for $d = j$ when j is even.

Case 2. j is odd. Then $\mathrm{mm}(j, g) \geq -h$ and $\mathrm{mm}(j-1, g) \leq h$. Thus from (2.1), $\mathrm{mm}(j, g) = -h$ if there is at least one child g' of g such that $\mathrm{mm}(j-1, g') = h$. Therefore, for every set of constraints C on any subset of

$$\{e(g') \mid \mathrm{depth}(g') = \mathrm{depth}(g)+j \text{ and } g' \text{ is not a descendent of } g\},$$

$$\begin{aligned}
&\Pr[\mathrm{mm}(j, g) = -h \mid C] = \\
&\quad = \Pr[\mathrm{mm}(j-1, g_m) = h \text{ for some } m \mid C] \\
&\quad = 1 - \Pr[\mathrm{mm}(j-1, g_m) \neq h \text{ for all } m \mid C], \\
&\quad = 1 - (1 - \Pr[\mathrm{mm}(j-1, g_1) = h \mid C]) \\
&\qquad * (1 - \Pr[\mathrm{mm}(j-1, g_2) = h \mid \mathrm{mm}(j-1, g_1 \neq h \text{ and } C]) \\
&\qquad \cdot \\
&\qquad \cdot \\
&\qquad \cdot \\
&\qquad * (1 - \Pr[\mathrm{mm}(j-1, g_m) = h \mid \mathrm{mm}(j-1, g_k) \neq h \\
&\qquad\qquad \text{for } k = 1, \ldots, m-1 \text{ and } C]) \\
&\qquad \cdot \\
&\qquad \cdot \\
&\qquad \cdot \\
&\qquad * (1 - \Pr[\mathrm{mm}(j-1, g_b) = h \mid \mathrm{mm}(j-1, g_k) \neq h \\
&\qquad\qquad \text{for } k = 1, \ldots, b-1 \text{ and } C]).
\end{aligned}$$

In a manner similar to Case 1, it follows that

$$\begin{aligned}
\Pr[\mathrm{mm}(j, g) = (-1)^j h \mid C] &\geq 1 - (1 - s_{j-1}(b, c/p))^b \\
&= s_j(b, c/p).
\end{aligned}$$

Thus the theorem holds for $d = j$ when j is odd.

Proof of Theorem 3.1. Let $g_1, g_2, \ldots, g_b$ be the children of g, and let the subtree rooted at g be complete to at least depth d. Then

$$\begin{aligned}
&\Pr[\mathrm{tie}(d, g)] = \\
&\quad = \Pr[\mathrm{mm}(d-1, g_1) = \mathrm{mm}(d-1, g_2) = \cdots = \mathrm{mm}(d-1, g_b)] \\
&\quad \geq \Pr[\mathrm{mm}(d-1, g_i) = (-1)^{d-1} h \text{ for all } i] \\
&\quad \geq \Pr[\mathrm{mm}(d-1, g_1) = (-1)^{d-1} h] \\
&\qquad * \Pr[\mathrm{mm}(d-1, g_2) = (-1)^{d-1} h \mid \mathrm{mm}(d-1), g_1 = (-1)^{d-1} h] \\
&\qquad \cdot \\
&\qquad \cdot \\
&\qquad \cdot
\end{aligned}$$

$$*\Pr[\mathrm{mm}(d-1, g_b) = (-1)^{d-1}h \mid \mathrm{mm}(d-1), g_i = (-1)^{d-1}h \text{ for } i = 1, \ldots, b-1]$$

(from the definition of conditional probability)

$$\geq (s_{d-1}(b, c/p))^b \quad \text{(from Lemma A.1).}$$

ACKNOWLEDGMENT

The author appreciates the comments and suggestions made by Vipin Kumar, Mike Mutchler, Judea Pearl, and Donald Loveland.

REFERENCES

1. Baudet, G.M., On the branching factor of the alpha-beta pruning algorithm, *Artificial Intelligence* **10** (1978) 173–199.
2. Beal, D., An analysis of minimax, in: M.R.B. Clarke (Ed.), *Advances in Computer Chess* 2 (University Press, Edinburgh, 1980).
3. Berliner, H., The B* tree search algorithm: a best-first proof procedure, *Artificial Intelligence* **12** (1979) 23–40.
4. Biermann, A.W., Theoretical issues related to computer game playing programs, *Personal Computing* (Sept. 1978) 86–88.
5. Bratko, I. and Gams, M., Error analysis of the minimax principle, in: M.R.B. Clarke (Ed.), *Advances in Computer Chess* 3 (Pergamon Press, London, 1982).
6. Knuth, D.E. and Moore, R.W., An analysis of alpha-beta pruning, *Artificial Intelligence* **6** (1975) 293–326.
7. Lindstrom, G., Alpha-beta pruning on evolving game trees, Tech. Rept. UUCS 79–101, Department of Computer Science, University of Utah, Salt Lake City, UT, 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., Pathology on game trees: a summary of results, *Proc. First Annual National Conf. Artificial Intelligence* (1980) 102–104.
10. Nau, D.S., An investigation of the causes of pathology in games, *Artificial Intelligence* **19** (1982) 257–278.
11. Nau, D.S., The last player theorem, *Artificial Intelligence* **18** (1982) 53–65.
12. Nau, D.S., Decision quality as a function of search depth on game trees, *J. ACM* (1983) to appear.
13. Newborn, M.M., The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores, *Artificial Intelligence* **8** (1977) 137–153.
14. Nilsson, N.J., *Principles of Artificial Intelligence* (Tioga, Palo Alto, CA, 1980).
15. Pearl, J., personal communication, 1982.
16. Pearl, J., Asymptotic properties of minimax trees and game-searching procedures, *Artificial Intelligence* **14** (1980) 113–138.
17. Pearl, J., Colloquium talk, University Maryland, College Park, MD, 1980.
18. Pearl, J., Heuristic search theory: survey of recent results, *Proc. Seventh Internat. Joint Conf. Artificial Intelligence*, 1981.
19. Pearl, J., On the nature of pathology in game searching, Tech. Rept. UCLA-ENG-CSL-82-17, School of Engineering and Applied Science, University of California, Los Angeles, CA, 1982.

20. Robinson, A.L., Tournament competition fuels computer chess, *Science* **204** (1979) 1396–1398.
21. Stockman G.C., A minimax algorithm better than alpha-beta?, *Artificial Intelligence* **12** (1979) 179–196.
22. Truscott, T.R., Minimum variance tree searching, *Proc. First Internat. Symp. Policy Analysis and Information Systems* (1979) 203–209.
23. Truscott, T.R., personal communication, 1980.

Received July 1982; revised version received October 1982