The Last Player Theorem*

Dana S. Nau

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

Recommended by Patrick J. Hayes and J. Gashnig

ABSTRACT

Game trees are an important model of decision-making situations, both in artificial intelligence and decision analysis, but many of the properties of game trees are not well understood. One of these properties is known as biasing: when a minimax search is done to an odd search depth, all moves tend to look good, and when it is done to an even search depth, all modes tend to look bad.

One explanation sometimes proposed for biasing is that whenever a player makes a move his position is 'strengthened', and that the evaluation function used in the minimax search reflects this. However, the mathematical results in this paper suggest that biasing may instead be due to the errors made by the evaluation function.

1. Introduction

Game trees are important both in artificial intelligence and decision analysis as models of situations in which series of decisions must be made. Many aspects of game trees are ill understood, and this paper is concerned with one of them. In particular, a widely observed phenomenon known as 'biasing' or 'manic-depressive behavior' is discussed, and mathematical results are presented which provide a partial explanation for why it occurs.

The game tree model casts a decision-making situation as a game between the decision maker and an opponent, who may either be a real opponent (as in the case of a parlor game) or a representation of the responses of the decision-maker's environment to his actions. The arcs of the game tree represent actions or *moves* by the two players, and the nodes of the tree represent the states resulting from the moves. The usual game tree model

*The material in this paper is adapted from a portion of the author's Ph.D. dissertation which was completed at Duke University in August of 1979. This work was supported in part by a National Science Foundation graduate fellowship, in part by a James B. Duke graduate fellowship, and in part by N.S.F. grant number ENG-7822159 to the Laboratory for Pattern Analysis at the University of Maryland.

0004-3702/82/0000–0000/$02.75

requires strict alternation of play between the two players. Situations not fitting this restriction can easily be handled by inserting null moves wherever necessary.

To make a decision using a game tree, payoffs or *utility values* are assigned to the terminal nodes, which are used to compute utility values for successively shallower nodes in the tree until values have been computed for the children of the root node. An optimal decision is to move to whichever child of the root has the largest utility value.

In decision analysis a number of different procedures or *criteria* are used to compute utility values [6, 18]. In computer science the minimax procedure (which is the same as the maximin criterion of decision analysis) is almost universally used; and hence the name *minimax values* for utility values. Computer science research on game trees has concentrated on ways to compute or estimate minimax values without searching the entire game tree [1–5, 7, 11, 13, 14, 15].

The details of the minimax procedure are amply discussed elsewhere [5, 8, 11]. Minimaxing is done only on zero-sum games, and the minimax value for a given player at a given node is the payoff he will receive if both players play perfectly from that position on. Thus at each node, the minimax value for one player is the negative of the minimax value for his opponent.

Most game trees of any significant depth are too large to search completely, but good results have been obtained by searching game trees to some limited depth, using a heuristic evaluation function to estimate the minimax values of the nodes at this depth; and applying minimaxing to these values as if they were the real minimax values [5, 8, 11]. This procedure we call a *heuristic game tree search*, and the values computed by it we call *estimated minimax values*.

When a heuristic game tree search is done, a phenomenon known as *biasing* or 'manic-depressive behavior' usually occurs: if the tree is searched to an odd depth, the estimated minimax values for the children of the root all tend to be high, and if the tree is searched to an even depth, they all tend to be low [8, 16, 17]. Thus the player doing the search tends to perceive himself as winning or losing, depending on whether the search depth is odd or even, respectively.

An explanation sometimes proposed for biasing is that a player's move somehow 'strengthens' his position and that the evaluation function reflects this. The difficulty with this explanation is that with a *perfect* evaluation function (i.e., one which always returns the true minimax value of a node) the estimated minimax value would remain constant, independent of the search depth. The main result presented in this paper indicates that biasing may be due instead to errors made by the evaluation function. This result we call the 'Last Player Theorem'.

Section 2 contains a simple version of the Last Player Theorem which applies to games without draws. Section 3 generalizes the theorem to games with arbitrary payoffs, and discusses games whose outcomes are ‘loss’, ‘draw’, and ‘win’ as an example. Section 4 explains how the theorem applies to evaluation functions, and Section 5 contains concluding remarks. Lemmas used in proving the theorem appear in the appendix.

2. Games without Draws

Let us define a *complete n-ary game tree of depth d* as a game tree for which every node of depth less than d has exactly n children, and every node of depth d has no children. Since we require that the play strictly alternates between the two players, the same player will have the last move regardless of what moves the players make during the game. If d is odd this will be the player who has the first move in the game; if d is even it will be the other player. Let us call this ‘last player’ Max and his opponent Min.

Let $G_{n,d}$ be a complete n-ary game tree of depth d whose terminal nodes are independently randomly labeled ‘win for Max’ and ‘loss for Max’, with probabilities p_{win} and p_{loss}, respectively (thus $p_{\text{loss}} = 1 - p_{\text{win}}$). Let g be the root of $G_{n,d}$ and let $p_{n,d}$ be the probability that Max can force a win at g. If $d = 0$, then g is a terminal node, so

$$p_{n,0} = p_{\text{win}}.$$

If $d > 0$, then g has n subtrees, each of which is a complete n-ary game tree of depth $d - 1$. On each of these subtrees the probability that Max can force a win is $p_{n,d-1}$. If $d > 0$ is odd, then Max has the first move, so he can force a win on $G_{n,d}$ if he can force a win at any of g's children. Thus if $d > 0$ is odd, then

$$p_{n,d} = 1 - (1 - p_{n,d-1})^n.$$

If $d > 0$ is even, then Min has the first move, so Max can force a win on $G_{n,d}$ only if he can force a win at each of g's children. Thus if $d > 0$ is even, then

$$p_{n,d} = (p_{n,d-1})^n = (1 - (1 - p_{n,d-2})^n)^n.$$

Thus for every $d \geq 0$,

$$p_{n,2(d+1)} = (1 - (1 - p_{n,2d})^n)^n.$$

Therefore, according to Lemma A.6 (see the appendix), there is a threshold w_n which depends only on n such that

$$\lim_{d\to\infty} p_{n,2d} = \begin{cases} 0 & \text{if } p_{\text{win}} < w_n, \\ 1 & \text{if } p_{\text{win}} > w_n, \end{cases}$$

and

$$p_{n,2d} = w_n \quad \forall d \quad \text{if } p_{\text{win}} = w_n.$$

Thus

$$\lim_{d\to\infty} p_{n,2d+1} = 1 - (1 - \lim_{d\to\infty} p_{n,2d})^n = \begin{cases} 0 & \text{if } p_{\text{win}} < w_n, \\ 1 & \text{if } p_{\text{win}} > w_n, \end{cases}$$

and

$$p_{n,2d+1} = 1 - (1 - w_n)^n = 1 - w_n \quad \forall d \quad \text{if } p_{\text{win}} = w_n.$$

This proves the following theorem.

Theorem 2.1. (1) *If* $p_{\text{win}} > w_n$, *then* $\lim_{d\to\infty} p_{n,d} = 1$;
(2) *if* $p_{\text{win}} < w_n$, *then* $\lim_{d\to\infty} p_{n,d} = 0$;
(3) *if* $p_{\text{win}} = w_n$, *then*

$$p_{n,d} = \begin{cases} 1 - w_n & \textit{if } d \textit{ is odd}, \\ w_n & \textit{if } d \textit{ is even}. \end{cases}$$

According to Theorem 2.1, whenever $p_{\text{win}} > w_n$ the probability that Max can force a win on $G_{n,d}$ converges to 1 as the depth d increases. For $n = 2$, w_n is less than 0.382, and w_n converges monotonically to 0 as n increases (see Corollary A.3 in the appendix). This convergence is illustrated in Table 1. Thus the probability that Max has a forced win converges to 1 for most values of n and p_{win}.

Although Theorem 2.1 is an asymptotic result, the convergence is generally quite rapid. This is illustrated in Table 2 and Fig. 1. For example, if $p_{\text{win}} \geq 1/2$, $n \geq 4$, and $d \geq 5$, then the probability that Max has a forced win is 1.0000, correct to four decimal places.

TABLE 1. Approximate values of w_n for $n = 1, 2, \ldots, 50$. For $p_{\text{win}} < w_n$ the probability that Max has a forced win converges to 0 as d increases. For $p_{\text{win}} > w_n$ the probability converges to 1

n	w_n	n	w_n	n	w_n	n	w_n	n	w_n
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

TABLE 2. The probability of Max (the 'last player') having a forced win on a complete n-ary game tree of depth d, for various values of n, d, and p_{win}

p_{win}	d	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
1/3	0	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333
1/3	1	0.5556	0.7037	0.8025	0.8683	0.9122	0.9414	0.9610
1/3	2	0.3086	0.3485	0.4147	0.4936	0.5762	0.6556	0.7273
1/3	3	0.5220	0.7234	0.8826	0.9667	0.9942	0.9994	1.0000
1/3	4	0.2725	0.3786	0.6069	0.8442	0.9657	0.9960	0.9998
1/3	5	0.4708	0.7601	0.9761	0.9999	1.0000	1.0000	1.0000
1/3	6	0.2216	0.4391	0.9078	0.9995	1.0000	1.0000	1.0000
1/3	7	0.3941	0.8235	0.9999	1.0000	1.0000	1.0000	1.0000
1/3	8	0.1553	0.5585	0.9997	1.0000	1.0000	1.0000	1.0000
1/3	9	0.2865	0.9140	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	10	0.0821	0.7635	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	11	0.1575	0.9868	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	12	0.0248	0.9608	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	13	0.0490	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	14	0.0024	0.9998	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	15	0.0048	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	16	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/3	17	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	0	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
1/2	1	0.7500	0.8750	0.9375	0.9688	0.9844	0.9922	0.9961
1/2	2	0.5625	0.6699	0.7725	0.8532	0.9098	0.9466	0.9692
1/2	3	0.8086	0.9640	0.9973	0.9999	1.0000	1.0000	1.0000
1/2	4	0.6538	0.8959	0.9893	0.9997	1.0000	1.0000	1.0000
1/2	5	0.8802	0.9989	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	6	0.7747	0.9966	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	7	0.9492	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	8	0.9010	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	9	0.9902	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	10	0.9805	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	11	0.9996	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	12	0.9992	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	13	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1/2	14	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	1	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667
2/3	2	0.8889	0.9630	0.9877	0.9959	0.9986	0.9995	0.9998
2/3	3	0.7901	0.8930	0.9515	0.9796	0.9918	0.9968	0.9988
2/3	4	0.9560	0.9988	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	5	0.9138	0.9963	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	6	0.9926	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	7	0.9852	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	8	0.9998	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	9	0.9996	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2/3	11	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

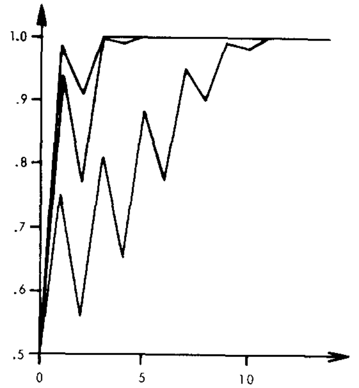

FIG. 1. The probability of Max having a forced win on a complete n-ary game tree of depth d, as a function of d. From bottom to top the curves are for $n = 2$, 4, and 6. In each case $p_{\mathrm{win}} = \frac{1}{2}$.

3. Games with Arbitrary Payoffs

Theorem 2.1 trivially generalizes to trees whose terminal nodes are independently randomly assigned the outcomes ‘loss’, ‘draw’, and ‘win’, or any other ordered set of payoffs.

Let $G_{n,d}$ be a complete n-ary game tree of depth d whose terminal nodes are labeled with independent, identically distributed random variables with p.d.f. $f(x)$. For each real number x, let $F(x) = \mathbf{P}[X \geqslant x]$, where X is any random variable whose p.d.f. if f.[1] Let g be the root of $G_{n,d}$, and let $f_{n,d}$ be the p.d.f. for the minimax value $M_{n,d}$ of g. For each real number x let $F_{n,d}(x) = \mathbf{P}[M_{n,d} \geqslant x]$.

If $d = 0$, then g is a terminal node, so $f_{n,0} = f$ and

$$F_{n,0} = F. \tag{3.1}$$

If $d > 0$, then g has n subtrees, each of which is a complete n-ary game tree of depth $d - 1$. For each of these subtrees, the probability that the minimax value of its root is at least x is $F_{n,d-1}(x)$. If $d > 0$ is odd, then Max has the first move, so the minimax value of g is the maximum of the minimax values of the children of g. Thus for $d > 0$ odd,

$$F_{n,d}(x) = 1 - (1 - F_{n,d-1}(x))^n. \tag{3.2}$$

If $d > 0$ is even, then Min has the first move, so the minimax value of g is the minimum of the minimax values of the children of g. Thus for $d > 0$ even,

$$F_{n,d}(x) = (F_{n,d-1}(x))^n = (1 - (1 - F_{n,d-2}(x))^n)^n. \tag{3.3}$$

[1]For this paper it is more convenient to define F in this way than to use the cumulative distribution function.

Thus for every $d \geq 0$,

$$F_{n,2(d+1)}(x) = (1 - (1 - F_{n,2d}(x))^n)^n.$$

Thus in exactly the same way that we proved Theorem 2.1, we get the following.

Theorem 3.1. *For every real number* x;

(1) *if* $F(x) > w_n$, *then* $\lim_{d\to\infty} F_{n,d}(x) = 1$;
(2) *if* $F(x) < w_n$, *then* $\lim_{d\to\infty} F_{n,d}(x) = 0$;
(3) *if* $F(x) = w_n$, *then*

$$F_{n,d}(x) = \begin{cases} 1 - w_n & \text{if } d \text{ is odd}, \\ w_n & \text{if } d \text{ is even}. \end{cases}$$

Theorem 3.1 has much the same import as Theorem 2.1. If the probability that Max has a payoff of at least x on a terminal node of $G_{n,d}$ exceeds w_n, then as the depth d increases the probability that Max can force the game to end on a node with a payoff of at least x converges to 1.

As an example, let $G_{n,d}$ be any complete n-ary game tree of depth d whose terminal nodes are independently randomly labeled 'loss', 'draw', and 'win', with the probabilities p_{loss}, p_{draw}, and p_{win}, respectively. If we represent 'loss', 'draw', and 'win' by -1, 0, and 1, respectively, and let f be the discrete p.d.f. defined by

$$f(-1) = p_{\text{loss}}, \qquad f(0) = p_{\text{draw}},$$
$$f(1) = p_{\text{win}}, \qquad f(x) = 0 \text{ otherwise},$$

then $F_{n,d}(1)$ is the probability that Max has a forced win on $G_{n,d}$. This probability is given in Table 2 and Fig. 1 as before.

4. Evaluation Functions

Let G be a game tree of constant branching factor n and depth greater than d. Let g be the root of G, and $g_1, g_2, \ldots, g_n$ be its children. A depth-d heuristic search at g normally means using a heuristic evaluation function to provide estimates of the minimax values of some or all of the nodes at depth d.[2] These estimates are used as if they were the real minimax values, to compute estimated minimax values for the shallower nodes of G. The move normally chosen is to whichever of $g_1, g_2, \ldots, g_n$ has the best estimated value.

Let Max be whichever player has the move at depth $d - 1$, and suppose the evaluation function returns independent, identically distributed random vari-

[2]If a pruning technique such as alpha–beta is used, not all of these nodes may be evaluated.

TABLE 3. The p.d.f. for the minimax value M of a child of the root of a complete n-ary game tree of depth d for various values of n and d. The evaluation function's p.d.f. is $f(x) = 0.2$ for $x = -2, -1, 0, 1$, and 2, and $f(x) = 0$ elsewhere.

n	d	P[M = −2]	P[M = −1]	P[M = 0]	P[M = 1]	P[M = 2]
2	1	0.2000	0.2000	0.2000	0.2000	0.2000
2	2	0.0400	0.1200	0.2000	0.2800	0.3600
2	3	0.0784	0.2160	0.2960	0.2800	0.1296
2	4	0.0061	0.0805	0.2619	0.4090	0.2424
2	5	0.0123	0.1536	0.4098	0.3656	0.0588
2	6	0.0002	0.0273	0.3039	0.5546	0.1141
2	7	0.0003	0.0539	0.4987	0.4341	0.0130
2	8	0.0000	0.0029	0.3028	0.6684	0.0259
2	9	0.0000	0.0059	0.5121	0.4813	0.0007
2	10	0.0000	0.0000	0.2683	0.7304	0.0013
2	11	0.0000	0.0001	0.4646	0.5354	0.0000
2	12	0.0000	0.0000	0.2159	0.7841	0.0000
2	13	0.0000	0.0000	0.3852	0.6148	0.0000
2	14	0.0000	0.0000	0.1483	0.8517	0.0000
2	15	0.0000	0.0000	0.2747	0.7253	0.0000
2	16	0.0000	0.0000	0.0754	0.9246	0.0000
2	17	0.0000	0.0000	0.1452	0.8548	0.0000
2	18	0.0000	0.0000	0.0211	0.9789	0.0000
2	19	0.0000	0.0000	0.0417	0.9583	0.0000
2	20	0.0000	0.0000	0.0017	0.9983	0.0000
2	21	0.0000	0.0000	0.0035	0.9965	0.0000
2	22	0.0000	0.0000	0.0000	1.0000	0.0000
2	23	0.0000	0.0000	0.0000	1.0000	0.0000
4	1	0.2000	0.2000	0.2000	0.2000	0.2000
4	2	0.0016	0.0240	0.1040	0.2800	0.5904
4	3	0.0064	0.0921	0.3275	0.4524	0.1215
4	4	0.0000	0.0001	0.0329	0.5627	0.4044
4	5	0.0000	0.0004	0.1250	0.8478	0.0267
4	6	0.0000	0.0000	0.0002	0.8970	0.1028
4	7	0.0000	0.0000	0.0010	0.9989	0.0001
4	8	0.0000	0.0000	0.0000	0.9996	0.0004
4	9	0.0000	0.0000	0.0000	1.0000	0.0000
4	10	0.0000	0.0000	0.0000	1.0000	0.0000
6	1	0.2000	0.2000	0.2000	0.2000	0.2000
6	2	0.0001	0.0040	0.0426	0.2155	0.7379
6	3	0.0004	0.0239	0.2249	0.5894	0.1614
6	4	0.0000	0.0000	0.0002	0.3476	0.6521
6	5	0.0000	0.0000	0.0014	0.9216	0.0769
6	6	0.0000	0.0000	0.0000	0.6187	0.3813
6	7	0.0000	0.0000	0.0000	0.9969	0.0031
6	8	0.0000	0.0000	0.0000	0.9817	0.0183
6	9	0.0000	0.0000	0.0000	1.0000	0.0000
6	10	0.0000	0.0000	0.0000	1.0000	0.0000

ables from some p.d.f. $f(x)$. Then the p.d.f. for the estimated value of each node at depth $k \leq d$ is obviously $f_{n,d-k}$. In particular, the p.d.f. for the estimated values of $g_1, g_2, \ldots, g_n$ is $f_{n,d-1}$. Because of this, Theorem 3.1 predicts that in most cases the estimated minimax values for $g_1, g_2, \ldots, g_n$ are likely to be high. Thus it will appear that Max is winning. For d odd, Max is the player doing the search; for d even, it is his opponent. This is precisely the biasing effect discussed in Section 1.

As an example, suppose an evaluation function returns the values -2, -1, 0, 1, and 2, each with probability 0.2. In Table 3 equations (3.1), (3.2), and (3.3) are used to compute the p.d.f. for the minimax values for $g_1, g_2, \ldots, g_n$ for various values of n and d. Note that as d increases, the probability that the minimax value is 1 rapidly converges to 1.

5. Discussion

Although no attempt has been made to do this, it should be fairly easy to generalize Theorems 2.1 and 3.1 to a much larger class of game trees. Such results would predict the occurrence of biasing on most game trees, but only under the assumption that the evaluation function returns identically distributed, independent random values. There are two difficulties with these restrictions.

First, evaluation functions generally do not return identically distributed values but instead return values which are intended to reflect the strengths and weaknesses of game positions. A mathematical model for this is developed in the author's Ph.D. dissertation [8], and again biasing is found to occur. Related work can be found in [9] and [10].

Second, evaluation functions generally operate deterministically rather than probabilistically. Without restricting ourselves to one specific game and one specific evaluation function, we cannot develop a mathematical model which takes this into account. However, evaluation functions are not perfect estimators of minimax values, for otherwise a game tree search would be totally unnecessary. Since the errors made by evaluation functions are largely unpredictable, they can be thought of as being random.

If the evaluation function's errors are considered to introduce some randomness into the values returned by the function, then results in this paper would lead one to predict biasing on a wide variety of game trees. That this prediction is indeed correct suggests that biasing is at least partially due to errors made by the evaluation function.

Appendix

This appendix contains statements and proofs of the lemmas needed to prove Theorems 2.1 and 3.1. Lemmas A.1 and A.4 and their corollaries state a number of properties of the sequence $\{w_n\}_{i=1}^{\infty}$. Lemma A.6 uses these properties to determine the limiting values of sequences of the form

$$x_{i+1} = (1-(1-x_i)^n)^n,$$

and this result is used to prove Theorems 2.1 and 3.1.

Lemma A.1. *Let $n \geq 1$. Then there is exactly one $x \in [0, 1]$ such that $(1-x)^n = x$. This value we call w_n.*

Proof. Let $f_n(x) = (1-x)^n - x$. Then $(1-x)^n = x$ if and only if $f_n(x) = 0$. $f_n(0) = 1$ and $f_n(1) = -1$, so by the intermediate value theorem, there is a point $z_0 \in (0, 1)$ for which $f_n(z_0) = 0$.

Suppose there is another point $z_1 \in (0, 1)$ such that $f_n(z_1) = 0$. Without loss of generality we may assume that $z_0 < z_1$. Then by the mean value theorem, there is a $y \in (z_0, z_1)$ such that $f_n'(y) = 0$. But

$$f_n'(x) = n(1-x)^{n-1}(-1) - 1 = -n(1-x)^{n-1} - 1,$$

so if $f_n'(y) = 0$, then $(1-y)^{n-1} = -1/n < 0$. But since $0 < y < 1$, this cannot be. Thus z_1 cannot exist.

Corollary A.2. *Let $n \geq 1$ and $x \in [0, 1]$. If $x < w_n$, then $(1-x)^n > w_n$. If $x > w_n$, then $(1-x)^n < w_n$.*

Proof. Obviously $(1-x)^n$ is decreasing for $x \in [0, 1]$. Thus if $x < w_n$, then $(1-x)^n > (1-w_n)^n = w_n$, and if $x > w_n$, then $(1-x)^n < (1-w_n)^n = w_n$.

Corollary A.3. $\lim_{n\to\infty} w_n = 0$, *and the convergence is monotonic.*

Proof. Let $0 < \varepsilon < 1$. Then $0 < 1 - \varepsilon < 1$, so there is an N such that for every $n > N$, $0 < (1-\varepsilon)^n < \varepsilon$. From Lemma A.1 and Corollary A.2, if $\varepsilon \leq w_n$, then $(1-\varepsilon)^n \geq w_n$, which is a contradiction. Thus $\varepsilon > w_n$. Thus, since $w_n > 0$ for every n, $\lim_{n\to\infty} w_n = 0$. To prove that the convergence is monotonic, let $1 \leq m < n$, and suppose $w_m \leq w_n$. Then

$$w_n = (1-w_n)^n < (1-w_n)^m \leq (1-w_m)^m = w_m,$$

which is a contradiction. Thus if $1 \leq m < n$, then $w_m > w_n$.

Corollary A.3 is illustrated in Table 1, which gives approximate values of w_1 through w_{50}. Graphically, w_n can be thought of as the intersection of the functions $y = (1-x)^n$ and $y = x$. For $n > 1$, w_n is also the unique intersection in the interval (0, 1) of the functions $y = (1-(1-x)^n)^n$ and $y = x$. This is proved in Lemma A.4, and is illustrated for w_2 in Fig. 2.

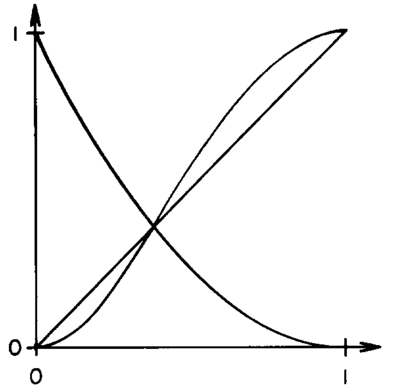

FIG. 2. Sketches of $y = (1-x)^2$, $y = (1-(1-x^2)^2$, and $y = x$.

Lemma A.4. *Let* $n > 1$, *and let*

$$g_n(x) = (1-(1-x)^n)^n - x$$

for all real x. Then g_n *has exactly one root in the interval* $(0, 1)$. *This is at* $x = w_n$.

Proof.

$$g_n(w_n) = (1-(1-w_n)^n)^n - w_n = (1-w_n)^n - w_n = w_n - w_n = 0,$$

so w_n is a root of g_n.

Suppose g_n has roots x_1 and x_2 in $(0, 1)$, with $x_1 < x_2$. In addition 0 and 1 are roots of g_n. So, from the mean value theorem, there are points y_1, y_2, and y_3 such that

$$0 < y_1 < x_1 < y_2 < x_2 < y_3 < 1$$

and

$$g_n'(y_1) = g_n'(y_2) = g_n'(y_3) = 0.$$

Therefore, applying the mean value theorem again, there are points z_1 and z_2 such that

$$0 < y_1 < z_1 < y_2 < z_2 < y_3 < 1$$

and

$$g_n''(z_1) = g_n''(z_2) = 0.$$

Now,

$$g_n'(x) = n^2(1-(1-x)^n)^{n-1}(1-x)^{n-1} - 1, \text{ so}$$
$$g_n''(x) = n^2(n-1)(1-(1-x)^n)^{n-2}(1-x)^{n-2}((n+1)(1-x)^n - 1).$$

For $x \in (0, 1)$

$$n^2(n-1)(1-(1-x)^n)^{n-2}(1-x)^{n-2} > 0,$$

so $g_n''(x) = 0$ if and only if $(n+1)(1-x)^n - 1 = 0$. There is exactly one $x \in (0, 1)$ for which this is true. Thus not both z_1 and z_2 can exist, so w_n is the only root of g_n in $(0, 1)$.

Corollary A.5. *Let $n > 1$. Then*
(1) *if $0 < x < w_n$, then $(1-(1-x)^n)^n < x$;*
(2) *if $w_n < x < 1$, then $(1-(1-x)^n)^n > x$;*
(3) *if $x = w_n$, then $(1-(1-x)^n)^n = x$.*

Proof. Let g_n be as in Lemma A.4. Then $g_n(0) = 0$, $g_n'(0) = -1$, and as shown in Lemma A.4, $g_n(x) \neq 0$ for $0 < x < w_n$. Therefore, by the intermediate value theorem, $g_n(x) < 0$ for $0 < x < w_n$. Similarly, $g_n(x) > 0$ for $w_n < x < 1$. Statement (3) follows immediately from Lemma A.4.

Lemma A.6. *Let $x_0 \in [0, 1]$, and let $n > 1$. For every integer $i \geq 0$, let*

$$x_{i+1} = (1-(1-x_i)^n)^n.$$

Then (1) *if $0 \leq x_0 < w_n$, then $\lim_{i\to\infty} x_i = 0$;*
(2) *if $w_n < x_0 \leq 1$, then $\lim_{i\to\infty} x_i = 1$;*
(3) *if $x_0 = w_n$, then $x_i = w_i$ for all i.*

Proof. To prove the first statement of the lemma suppose that $0 \leq x_0 < w_n$. From Corollary A.5, x_i is a bounded monotone sequence which, therefore, must have some limit $z \in [0, w_n)$. But

$$\begin{aligned}(1-(1-z)^n)^n &= (1-(1-\lim_{i\to\infty} x_i)^n)^n \\ &= \lim_{i\to\infty}(1-(1-x_i)^n)^n = \lim_{i\to\infty} x_{i+1} = z,\end{aligned}$$

and the only $z \in [0, w_n)$ for which this can be true is $z = 0$.

The above proves the first statement of the lemma. The proof of the second statement is almost identical, and the third follows immediately from Corollary A.5.

ACKNOWLEDGMENT

The author is grateful to Alan Biermann for his advice and encouragement. Results equivalent to Theorems 2.1 and 3.1 were independently discovered by Judea Pearl [12].

REFERENCES

1. Baudet, G.M., On the branching factor of the alpha–beta pruning algorithm, *Artificial Intelligence* **10** (1978) 173–199.
2. Biermann, A.W., Theoretical issues related to computer game playing programs, *Personal Computing* (Sept. 1978) 86–88.
3. Fuller, S.H., Gashnig, J.G. and Gillogly, J.J., Analysis of the alpha–beta pruning algorithm, Department of Computer Science, Carnegie–Mellon Univ. (July 1973).
4. Greenblatt, R.D., Eastlake, D.E. and Crocker, S.D., The Greenblatt chess program, *Proc. AFIPS Fall Joint Computer Conference* **31** (1967) 801–810.
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* (Rinehart and Winston, New York, 1978).
7. Lindstrom, G., Alpha–beta pruning on evolving game trees, Tech. Rept. UUCS 79–101, Department of Computer Science, Univ. of Utah (1979).
8. Nau, D.S., Quality of decision versus depth of search on game trees, Ph.D. Dissertation, Duke University (August 1979).
9. Nau, D.S., Decision quality as a function of search depth on game trees, Tech. Rept. TR-866, Computer Science Department, Univ. of Maryland (1980).
10. Nau, D.S., Quality of decision versus depth of search on game trees: a summary of results, *First Annual National Conference Artificial Intelligence* (August 1980).
11. Nilsson, N.J., *Problem-Solving Methods in Artificial Intelligence* (McGraw-Hill, New York, 1971).
12. Pearl, J., Studies in heuristics, Part 4: Asymptotic properties of minimax trees and game-searching procedures, Tech. Rept. UCLA-ENG-CSL-7981, School of Engineering and Applied Science, UCLA (1979).
13. Robinson, A.L., Tournament competition fuels computer chess, *Science* **204** (1979) 1396–1398.
14. Samuel, A.L., Some studies in machine learning using the game of checkers. II—Recent progress, *IBM J. Res. Develop.* **11** (1967) 601–617.
15. Stockman, G.C., A minimax algorithm better than alpha–beta?, *Artificial Intelligence* **12** (1979) 179–196.
16. Truscott, T.R., Personal communication (1979).
17. Truscott, T.R., Minimum variance tree searching, *Proc. First Internat. Symp. Policy Analysis and Information Systems* (1979) 203–209.
18. Tummala, V.M.R., *Decision Analysis with Business Applications* (Intext, New York, 1973).

Received January 1980; *revised version received June* 1980