Received: 18 May 2022 | Revised: 16 January 2023 | Accepted: 13 February 2023

DOI: 10.1111/coin.12570

ORIGINAL ARTICLE

Computational Intelligence WILEY

The phenomenon of decision oscillation: A new consequence of pathology in game trees

Mark Levene | **Trevor Fenner**

Department of Computer Science and Information Systems, Birkbeck College, University of London, London, UK

Correspondence
Mark Levene, Department of Computer Science and Information Systems, Birkbeck College, University of London, London WC1E 7HX, UK.
Email: m.levene@bbk.ac.uk

Abstract

Random minimaxing studies the consequences of using a random number for scoring the leaf nodes of a full width game tree and then computing the best move using the standard minimax procedure. Experiments in Chess showed that the strength of play increases as the depth of the lookahead is increased. Previous research by the authors provided a partial explanation of why random minimaxing can strengthen play by showing that, when one move dominates another move, then the dominating move is more likely to be chosen by minimax. This paper examines a special case of determining the move probability when domination does not occur. Specifically, we show that, under a uniform branching game tree model, whether the probability that one move is chosen rather than another depends not only on the branching factors of the moves involved, but also on whether the number of ply searched is odd or even. This is a new type of game tree pathology, where the minimax procedure will change its mind as to which move is best, independently of the true value of the game, and oscillate between moves as the depth of lookahead alternates between odd and even.

KEYWORDS

game tree pathology, minimax algorithm, random minimaxing

1 | INTRODUCTION

The *minimax* procedure is the fundamental search algorithm for deciding the next move to play in two-player perfect information games between *white* and *black*;[1-3] Chess, Checkers, Othello and Go are examples of such games. The minimax procedure is utilised by constructing a full-width δ+1-ply game tree (with $\delta \geq 0$) from the current position (with white, maximizing, to move), where nodes represent game positions and arcs represent legal moves from one position to another. The value of the root is determined by computing a *score* for each leaf value using a static evaluation function and backing up these scores using the minimax rule. In practice, many refinements have been developed in order to improve the performance of the minimax search algorithm.[3,4]

Here, we concentrate on the classical minimax search algorithm, noting that the results of the paper would apply to any variant of the algorithm that guarantees the same outcome as classical minimax, such as alpha-beta pruning.[4] Many of the variants of the minimax search algorithm[4] would not necessarily return the same minimax value as classical minimax and therefore would demand separate study.

In order to measure the utility of the minimax procedure, we follow Reference 5 in using a random static evaluation function that returns a natural number uniformly distributed between 1 and α, with $\alpha > 1$; this variation of the minimax procedure is called *random minimaxing*. In this way, we can decouple the effectiveness of the minimax procedure from the accuracy of the static evaluation function, allowing us to study the minimax procedure in its raw form. The experiments carried out by Beal and Smith,[5] using random minimaxing in Chess, produced the interesting result that the strength of play increases as the depth of the lookahead is increased. In an attempt to understand this result, we showed in Reference 6 that, when one move *dominates* another move, then the dominating move is more probable, that is, more likely to be chosen by the minimax procedure. A move by white dominates another when choosing the former move over the latter would give less choice for black, at all levels, when it is black's turn to move, and subsequently more choice for white when it is white's turn to move. Under random minimaxing, domination can be viewed as a form of *mobility*, which lends support to the hypothesis that the minimax procedure is successful in games (such as Chess and Othello) where having greater mobility is considered to be advantageous.

A problem that was left open in Reference 6 was to determine the probability of one move being chosen rather than another when domination does not occur. In this paper we tackle this problem for the case where the subgame trees rooted at the nodes representing the moves have *uniform branching factors*. The branching factor of such a subgame tree T_i can be viewed as the average number of choices a player would have in subsequent moves, given that the move that leads to the root of T_i was actually chosen. We call this model of a game tree the *uniform branching model*.

Our main result is that modelling a game tree in this way leads to *decision oscillation*, which is a form of pathological behaviour in the following sense. We show that, under reasonable assumptions, whether the probability of one move being chosen is greater than that of another move depends not only on the branching factors of the moves involved but also on whether the number of ply searched is odd or even. Decision oscillation implies that the minimax procedure may change its mind as to which move is best, independently of the true value of the game, resulting in the move choice alternating between two moves as the search deepens. A similar phenomenon, called the *odd/even effect* (in which the backed-up value oscillates each time the depth of search increases by one),[7] has been observed in practical game-playing programs. However, for computer

chess programs, it has been suggested that this phenomenon is related to *quiescence*,[1] which is not related to decision oscillation as described here.

Game tree pathology has been studied for over three decades.[8] Under the uniform game tree model, when the scores of the leaf nodes are independent identically distributed random variables restricted to the values 1 (*win*) or 0 (*loss*), the term *pathological* means that, as the depth of lookahead increases, the probability that a move chosen by the minimax procedure is correct tends to the probability that a randomly chosen move is correct.[9] A related form of minimax pathology, called the "last player theorem", was exhibited by Nau,[10] who showed that, under the uniform game tree model with win/loss values at the leaf nodes, the probability that the last player appears to have a winning strategy tends to one as the depth increases. (Nau's game tree model is slightly different from ours in that he assumes that the last player is maximizing, while we assume that the first player is maximizing.)

One explanation for the apparent absence of pathology in real games could be that the scores of leaf positions are seldomly independent. Beal[11] showed that, when sibling node dependence is taken into account, the error probability decreases with the depth of lookahead, and Pearl[12] showed that if the density of "traps" exceeds a certain threshold then pathology can be avoided. This explanation for the lack of pathology, when real games contain numerous "traps" that lead to early termination of the game, is based on the fact that for such terminal nodes the score assigned to the position by the evaluation function reflects the true value of the game. A further refinement was suggested by Scheucher and Kaindl,[13] who showed that the use of multivalued evaluation functions with a non-uniform error distribution allows for improved evaluation quality, so that evaluation errors decrease when the depth of search is increased. Luštrek et al.[14] extended the model further to include real values for both heuristic and true values, showing that pathology can be eliminated in this model due to reduced variance of the values returned by the minimax procedure.

Despite these results, it was shown by Sadikov et al.[15] that in a king and rook versus king chess endgame, with errors introduced into the evaluation function, pathology still exists in the sense that deeper search does not reduce the evaluation error. They conjecture that the cause is the *bias* introduced by the minimax procedure, where the bias is defined as the difference between the true and minimax values of a position, averaged over all the positions examined during a minimax search. Interestingly, despite the fact that evaluation accuracy may decrease with deeper searches, it was shown that bias affected all evaluations more or less equally, and therefore it did not affect the relative ordering of moves with respect to their quality, Hence, the decision accuracy of minimax was not impaired in this context and was more accurate with deeper searches.

Schrüfer[16] uses a model similar to that of Nau,[9] that is, game trees have a uniform branching factor and a finite depth, and only two heuristic values, win and loss, are considered. However, the error attached to the heuristic value returned by the evaluation function is modelled using two different sets of independent identically distributed random variables. This is to distinguish the probability of assigning a win value to a leaf when it is a loss and the probability of assigning a loss value to a leaf when it is a win (in Nau[9] the probability of these two types of error are equal). Schrüfer[16] showed that, for the errors to decrease with increasing depth of search, the probability of having a single "good" candidate move should be small.

Notwithstanding this long history of studying minimax pathology, there is continuing interest in the problem. Lorenz and Monien[17] investigated arbitrary (not necessarily uniform) win/loss game trees, using the same probability model as in Reference 9. They show that, in order for minimax to be non-pathological in this more general model, there must be at least two leaf-disjoint

strategies that prove the true value at the root of the game tree. A more recent paper by Nau et al.[18] used simulations and experimental tests to show that pathology is more likely to occur in practice when the heuristic evaluation function has a small number of possible values, the branching factor of the game trees is high, and the local similarity values of nearby nodes in game trees is low. Another recent paper by Zuckerman et al.[19] suggests that local pathologies in the game tree occur due to the error of the static evaluation function used when applying the minimax procedure. The authors propose to minimise the error by tracking these local pathologies in tandem with computing the minimax value. Also, even more recently, a paper by Liu et al.[20] suggests a variant to the minimax algorithm that overcomes pathology by optimising the minimax backup rule.

It is also worth mentioning that the newer Monte Carlo tree search (MCTS) algorithm[21,22] has proved to be competitive with the minimax algorithm when combined with reinforcement learning methods.[23] However, we note that in Reference 24 it was shown that the Stockfish Chess engine,[25] which deploys minimax rather than MCTS, outperforms a state of the art MCTS-based Chess engine on solving a well-known Chess endgame puzzle. In addition,[26] demonstrated that a best-first variant of minimax is competitive with state of the art reinforcement learning methods that use MCTS. To combine the strengths of both MTCS and minimax, hybrid search strategies have been introduced.[27]

The layout of the rest of the paper is as follows. In Section 2, we provide notational background for the minimax procedure and then, in Section 3, we introduce the definitions and assumptions used in the paper. In Section 4, we adapt the results from Reference 10 to random minimaxing of game trees. In Section 5, we show that the uniform branching model leads to decision oscillation for random minimaxing. Finally, in Section 6, we give our concluding remarks.

2 | THE MINIMAX PROCEDURE

We assume that the reader is familiar with the basic *minimax* procedure.[2] However, we briefly recall some of the definitions from Reference 6 relevant to this paper.

A *game tree* T is a special kind of tree, whose nodes represent game positions and arcs represent *legal* moves from one position to another; the *root* node represents the current position. In general, we will not distinguish between the nodes and the positions they represent, nor between the arcs and the moves they represent. Furthermore, when no confusion arises, we will refer to the position arrived at as a result of making a move as the move itself. We are assuming a two-player zero-sum perfect information game between the first player, called *white*, and the second player, called *black*, where the game has three possible outcomes: *win* for white (i.e., loss for black), *loss* for white (i.e., win for black), or *draw* (see Reference 28 for a precise definition of a game).

The *level* of a node n in T is defined recursively as follows: if n is the root of T then its level is zero, otherwise the level of n is one greater than the level of its parent node. Nodes of T at even levels are called *max-nodes* and those at odd levels are called *min-nodes*. At a max-node it is white's turn to move and at a min-node it is black's turn to move. We assume that T is a δ+1-ply game tree, with $\delta \geq 0$, where the number of ply is one less than the number of levels of T. (Thus the root of T is at level zero and the leaves are at level δ+1.) Non-leaf nodes of a game tree are called *internal nodes*. A *full-width* game tree satisfies the condition that there is an arc for each legal move from every position represented by an internal node. We will assume that all game trees are full-width.

We define the following two operators on T:

1. *root_moves*(T) is the set of children of the root of T, that is, the set of nodes representing the possible positions arrived at after white makes a single move;
2. T[n] is the subgame tree of T rooted at a node n; if n is the root of T then T[n] = T.

We let minimax(T, δ+1, *score*, α) denote a procedure that returns the leaf node of the *principal variation* chosen by minimaxing,[2] where T is the δ+1-ply game tree whose root represents the current position, *score* is a *static evaluation function*, and α is a natural number representing the maximum possible score.

We assume that the scoring of leaf nodes is computed by the function *score*, which returns a natural number between 1 and α inclusive, with $\alpha > 1$. We denote the set $\{1, 2, \ldots, \alpha\}$ by $[\alpha]$. For the purpose of scoring, we assume that all leaf nodes are distinct although, in practice, two distinct leaf nodes may represent the same position (for example, through a transposition of moves[3]).

In general, it is possible that there is more than one principal variation, in which case we assume that the minimax procedure returns the set of leaf nodes of all the principal variations. For our purposes, it is sufficient to know whether or not a particular leaf node could be returned by the minimax procedure. We note that normally, in practical implementations, the leaf node of only one principal variation is returned.

The score assigned to an internal node n of T during the evaluation of minimax(T, δ+1, *score*, α) is called the *backed-up* score of n and is denoted by $sc(n)$; when n is a leaf node $sc(n) = score(n)$. The backed-up score of a subgame tree T[n] is $sc(n)$, the score of its root n; so the backed-up score of T is the score of the leaf node of any principal variation.

3 | RANDOM MINIMAXING

For given δ, *score* and α, the procedure minimax(T, δ+1, *score*, α) defines a *strategy* for playing a particular combinatorial game. We assume, from now on, that successive calls of *score* return a sequence of independent random integers uniformly distributed between 1 and α. In this case, we will refer to the induced minimax strategy as *random minimaxing*.[5,6,29]

We are interested in the probability that any given node lies on a principal variation. For a node $n \in root\_moves$(T), we define $prob(n, i)$ to be the probability that n is on a principal variation of T and $sc(n) = i$. This may be calculated as follows: count the number of assignments of scores to the leaf nodes of T such that n is on a principal variation of T and $sc(n) = i$, and then divide this count by α^N, the total number of assignments of scores to the leaf nodes of T, where N is the number of leaf nodes of T. We now define $prob(n)$, the probability that a node $n \in root\_moves$(T) is on a principal variation of T, by

$$prob(n) = \sum_{i=1}^{\alpha} prob(n, i). \tag{1}$$

From now on, we assume that $n \in root\_moves$(T) and that T[n] has a uniform *branching factor* $b \geq 1$, where b may depend on n. Suppose first that m is a max-node in T[n] and that m' is a child of m. We note that the distribution of $sc(m')$ is the same for all the children of m. We define

$$x(i) = prob\left(sc(m') \leq i\right), \tag{2}$$

where $1 \leq i \leq \alpha$. Then, since m is a max-node,

$$prob\,(sc(m) > i) = 1 - x(i)^b. \tag{3}$$

Now suppose conversely that m is a min-node in T[n]. In this case we define

$$x(i) = prob\left(sc(m') > i\right). \tag{4}$$

Then, since m is a min-node,

$$prob\,(sc(m) \leq i) = 1 - x(i)^b. \tag{5}$$

We therefore define

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

and consider the recurrence relation

$$x_{t+1} = F_b(x_t), \tag{6}$$

for $t \geq 0$ (see [Reference 6, Equation (9)] for a justification of this formula), where

$$x_0 = x_0^i = \begin{cases} \frac{i}{\alpha} & \text{if } \delta \text{ is even, i.e., the leaves of T are min-nodes,} \\ \frac{\alpha - i}{\alpha} & \text{if } \delta \text{ is odd, i.e., the leaves of T are max-nodes,} \end{cases}$$

for some $i \in [\alpha] \cup \{0\}$. It follows that $0 \leq x_t \leq 1$ for all $t \geq 0$.

Starting with x_0, we can compute the corresponding sequence x_t, for $1 \leq t \leq \delta$, by successive applications of F_b as in (6). We denote the final value x_δ of the above sequence by $F_b^\delta(x_0)$, where F_b^δ is the δ-fold composition of F_b with itself; we call F_b^δ the *propagation* function for T[n].

For any given value of i, we may substitute x_t for $x(i)$ in equations (2) to (5), for appropriate values of t. It is then easy to verify that, when δ - t is *even*, x_t is the probability that the score of a min-node of T[n] at level $\delta + 1$- t *does not exceed* i, and that, when δ - t is *odd*, x_t is the probability that the score of a max-node of T[n] at level $\delta + 1$- t *exceeds* i.

For $t = \delta$, we have

$$x_\delta = F_b^\delta(x_0) = prob\,(sc(n) \leq i). \tag{7}$$

We observe that $F_b(0) = 1$ and $F_b(1) = 0$, otherwise $0 < F_b(x) < 1$. Thus, when x is 0 or 1, if δ is even then $F_b^\delta(x) = x$, and if δ is odd then $F_b^\delta(x) = 1 - x$.

From now on, unless stated otherwise, we will assume that there are exactly two nodes, n_1 and n_2 in *root_moves* (T), and that the subgame trees T [n_1] and T [n_2] have uniform branching factors b_1 and b_2, respectively.

The following lemma (cf. Lemma 5.1 in Reference 6) expresses the probability of a move in terms of propagation functions.

Lemma 1. *Suppose root_moves(T) = $\{n_1, n_2\}$. Then, for $i \in [\alpha]$,*

$$prob(n_1, i) = \left(F^{\delta}_{b_1}(x^i_0) - F^{\delta}_{b_1}(x^{i-1}_0)\right) F^{\delta}_{b_2}(x^i_0),$$
$$prob(n_2, i) = \left(F^{\delta}_{b_2}(x^i_0) - F^{\delta}_{b_2}(x^{i-1}_0)\right) F^{\delta}_{b_1}(x^i_0). \tag{8}$$

Proof. Since the root of T is a max-node, and $prob(n_1, i)$ and $prob(n_2, i)$ are independent, we have

$$\begin{aligned} prob(n_1, i) &= prob(sc(n_1) = i) \times prob(sc(n_2) \leq i) \\ &= (prob(sc(n_1) \leq i) - prob(sc(n_1) \leq i - 1)) \times prob(sc(n_2) \leq i). \end{aligned}$$

The result then follows from (7). ∎

The following corollary generalises Lemma 1 to the case when the root of T has more that two children.

Corollary 1. *Suppose that $n \in$ root_moves(T) has branching factor b, and that all other nodes in root_moves(T) also have uniform branching factors. Then, for each $i \in [\alpha]$,*

$$prob(n, i) = \left(F^{\delta}_{b}(x^i_0) - F^{\delta}_{b}(x^{i-1}_0)\right) \prod_{n' \in sib(n)} F^{\delta}_{b'}(x^i_0) = \left(1 - \frac{F^{\delta}_{b}(x^{i-1}_0)}{F^{\delta}_{b}(x^i_0)}\right) \Omega(T, i), \tag{9}$$

where sib(n) denotes the set of sibling nodes of n, b' is the branching factor of n', and $\Omega(T, i)$ is independent of n.

4 | THE LAST PLAYER THEOREM FOR RANDOM MINIMAXING

In this section we restate the relevant results from Reference 10 in the context of our formulation of random minimaxing, which differs slightly from that in Reference 10: it is assumed that the last player is maximizing, while we make the assumption that the first player (root node) is maximizing.

The following theorem and corollary follow directly from the results in the Appendix and Table 1 in Reference 10.

Theorem 1. *For all branching factors $b \geq 1$:*

(a) There exists exactly one value $x \in [0, 1]$ such that $1 - x^b = x$; we denote this value
(b) by v_b and call it the critical value for b.
  (i) If $0 \leq x < v_b$ then $1 - x^b > v_b$.
  (ii) If $v_b < x \leq 1$ then $1 - x^b < v_b$.
(c) If $b = 1$ then $1 - (1 - x^b)^b = x$ for all x.
(d) If $b > 1$ then:
  (i) $1 - (1 - x^b)^b < x$ when $0 < x < v_b$;

(ii) $1-(1-x^b)^b > x$ *when* $v_b < x < 1$;
(iii) $1-(1-x^b)^b = x$ *when* $x = v_b$;

(e) $\lim_{b\to\infty} v_b = 1$ *and the convergence is strictly monotonic increasing.*
(f) $\frac{1}{2} \leq v_b < 1$ *and the inequality is strict except when* $b = 1$.

(We note that $v_b = 1 - w_b$, where w_b is the threshold value in Reference 10.)

Corollary 2. *Let* $x_0 \in [0,1]$ *and let* $b > 1$. *We define the sequence* x_t *as in (6).*

(a) *If* $0 \leq x_0 < v_b$ *then* $\lim_{t\to\infty} x_t = 0$ *for even* t.
(b) *If* $0 \leq x_0 < v_b$ *then* $\lim_{t\to\infty} x_t = 1$ *for odd* t.
(c) *If* $v_b < x_0 \leq 1$ *then* $\lim_{t\to\infty} x_t = 1$ *for even* t.
(d) *If* $v_b < x_0 \leq 1$ *then* $\lim_{t\to\infty} x_t = 0$ for odd t.
(e) *If* $x_0 = v_b$ then $x_t = v_b$ *for all* $t \geq 0$.

In the next section we will also need the following lemma.

Lemma 2. *For all* $b > 1$, v_b *is irrational.*

Proof. Suppose to the contrary that, for some $b > 1$, $v_b = \frac{y}{z}$ where y and z are co-prime integers. Then $y > 1$, since, by Theorem 1 (f), $v_b > \frac{1}{2}$. Now, if u is a prime factor of y, it is also a prime factor of z^b since, by Theorem 1 (a),

$$z^b - y^b = yz^{b-1}.$$

The result now follows, since u is then also a prime factor of z, which contradicts the fact that y and z are co-prime. ∎

5 | A DECISION OSCILLATION THEOREM FOR RANDOM MINIMAXING

We now present our main results, showing that the choice of move under random minimaxing is pathological in the sense that the choice depends not only on the branching factor of the possible moves, but also on whether the depth of search is even or odd. The result depends on modelling game trees using the uniform branching model, and is in contrast to the situation with non-uniform game trees where domination may be present.[6]

The following definition imposes some reasonable constraints on the maximum score α, and the critical values, v_{b_1} and v_{b_2}, for branching factors b_1 and b_2. The intuition behind the constraints is that the maximum score should be large enough.

Definition 1 (Admissible maximum scores). We assume that $b_1 > b_2$ and that v_{b_1} and v_{b_2} are defined as in Theorem 1. (Note that $v_{b_1} > v_{b_2}$ by Theorem 1 (e).) We say that the natural number α, $\alpha > 2$, is *admissible* if, for some $i \in [\alpha]$, $v_{b_2} < x_0^i < v_{b_1}$.

The following lemma, which is an immediate consequence of Definition 1 and Theorem 1 (f), implies that, in practice, admissibility of α is not too restrictive provided α is large enough.

Lemma 3. *Suppose that* $b_1 > b_2$. *Let us call* α *large if* $\alpha > \frac{1}{v_{b_1} - v_{b_2}}$. *Then, if* α *is large,* α *is admissible.*

The following theorem establishes our main result, that random minimaxing leads to decision oscillation under the uniform branching model.

Theorem 2. *Suppose that* $b_1 > b_2 > 1$ *and that* α *is admissible. Then, for large enough* δ*,*

1. $prob(n_1) > prob(n_2)$ *if and only if* δ *is even,*
2. $prob(n_1) < prob(n_2)$ *if and only if* δ *is odd.*

Proof. Clearly, since δ is either even or odd, we only need to prove the "if" parts of (a) and (b).

(a) Assume that δ is even and let $i \in [\alpha]$. Then x_0^i is strictly increasing with i.
We first compute the value of $prob(n_1, i)$ for each $i \in [\alpha]$. It follows from Lemma 2 that, since $b_1 > b_2 > 1, x_0^i \neq v_{b_1}$ and $x_0^i \neq v_{b_2}$ for all $i \in [\alpha]$. So there are three cases to consider:
(A) $x_0^{i-1} < x_0^i < v_{b_1}$. By Corollary 2 (a), $\lim_{\delta\to\infty} F_{b_1}^{\delta}(x_0^j) = 0$, for $j = i$ and $i-1$; so $prob(n_1, i) \to 0$ by Lemma 1.
(B) $v_{b_1} < x_0^{i-1} < x_0^i$. By Corollary 2 (c), $\lim_{\delta\to\infty} F_{b_1}^{\delta}(x_0^j) = 1$, for $j = i$ and $i-1$; so $prob(n_1, i) \to 0$ by Lemma 1.
(C) $x_0^{i-1} < v_{b_1} < x_0^i$. This implies that $v_{b_2} < x_0^i$. So, by parts (a) and (c) of Corollary 2, $\lim_{\delta\to\infty} F_{b_1}^{\delta}(x_0^{i-1}) = 0$, $\lim_{\delta\to\infty} F_{b_1}^{\delta}(x_0^i) = 1$ and $\lim_{\delta\to\infty} F_{b_2}^{\delta}(x_0^i) = 1$; thus $prob(n_1, i) \to 1$ by Lemma 1.
Clearly there exists a unique $i \in [\alpha]$ satisfying case (C), which implies that $prob(n_1) \to 1$ (see also [Reference 30, Theorem 1]).
We now similarly compute the value of $prob(n_2, i)$ for $i \in [\alpha]$. There are again three cases to consider:
(D) $x_0^{i-1} < x_0^i < v_{b_2}$. As in (A) $prob(n_2, i) \to 0$.
(E) $v_{b_2} < x_0^{i-1} < x_0^i$. As in (B) $prob(n_2, i) \to 0$.
(F) $x_0^{i-1} < v_{b_2} < x_0^i$. This implies that $x_0^i < v_{b_1}$, by the definition of admissibility. By Corollary 2 (a) $\lim_{\delta\to\infty} F_{b_1}^{\delta}(x_0^i) = 0$, so $prob(n_2, i) \to 0$, by Lemma 1.
It follows that $prob(n_2) \to 0$ and thus, for large enough δ, $prob(n_1) > prob(n_2)$ as required.

(b) This follows using a similar argument to that used in (a), utilising parts (b) and (d) of Corollary 2. ∎

We extend the notion of admissibility to game trees with more than two moves by letting b_1 and b_2 in Lemma 3 be the branching factors of any two moves in *root_moves*(T) satisfying $b_1 > b_2$. Thus, for α to be admissible, the conditions of Definition 1 must hold for the branching factors b_1 and b_2 of any two moves in *root_moves*(T) with $b_1 > b_2$.

As a corollary, we show that the main result can be extended to game trees with more than two moves in *root_moves*(T). For node $n_j \in root\_moves(\mathrm{T})$, from (9) we have

$$prob(n_j, i) = \left(F_{b_j}^{\delta}(x_0^i) - F_{b_j}^{\delta}(x_0^{i-1})\right) \prod_{n_k \in sib(n_j)} F_{b_k}^{\delta}(x_0^i), \tag{10}$$

where $sib(n_j)$ denotes the set of sibling nodes of n_j.

Corollary 3. *Suppose that root_moves(T) contains any number of moves, and let n_1 and n_2 be any two moves in root_moves(T) with branching factors b_1 and b_2 such $b_1 > b_2$. Suppose also that α is admissible. Then, for large enough δ,*

(a) $prob(n_1) > prob(n_2)$ if and only if δ is even.
(b) $prob(n_1) < prob(n_2)$ if and only if *δ is odd.*

Proof. We note from (10) that $prob(n_1, i)$ and $prob(n_2, i)$ are given by the expressions in (8) multiplied in each case by the same factor, namely the product in (10) taken over all moves in *root_moves*(T) except n_1 and n_2. The result then follows as in the proof of Theorem 2. ∎

It follows from the corollary that, for large enough δ, $prob(n_1) \neq prob(n_2)$ if $b_1 \neq b_2$, thus $prob(n_1) = prob(n_2)$ if and only if $b_1 = b_2$.

We now extend Theorem 2 to the case when $b_2 = 1$.

Theorem 3. *The result of Theorem 2 still holds when $b_2 = 1$.*

Proof. We first assume that δ is even, and show that $prob(n_1) > prob(n_2)$ in this case.

Using an identical argument to that made in the proof of Theorem 2, it follows that for cases (A) and (B), $prob(n_1, i) \to 0$ as $\delta \to \infty$. However, for case (C), we have $F^{\delta}_{b_2}(x^i_0) = x^i_0$ for all even δ by Theorem 1 (c). So, by Lemma 1 and (1), as $\delta \to \infty$,

$$prob(n_1) = prob(n_1, i) \to x^i_0 = \frac{i}{\alpha} = \frac{\lceil \alpha v_{b_1} \rceil}{\alpha}.$$

By Theorem 1 (c), $F^{\delta}_{b_2}(x^i_0) - F^{\delta}_{b_2}(x^{i-1}_0) = x^i_0 - x^{i-1}_0 = \frac{1}{\alpha}$. Also, $\lim_{\delta\to\infty} F^{\delta}_{b_1}(x^i_0)$ is 0 if $x^i_0 < v_{b_1}$ and 1 if $v_{b_1} < x^i_0$; so, by Lemma 1, as $\delta \to \infty$, $prob(n_2, i) \to 0$ or $\frac{1}{\alpha}$, respectively. Therefore, by (1), $prob(n_2) \to \frac{\alpha - \lfloor \alpha v_{b_1} \rfloor}{\alpha}$ as $\delta \to \infty$. Since $b_1 \geq 2$, we have $v_{b_1} \geq v_2 = \frac{\sqrt{5}-1}{2}$ by Theorem 1 (e). Straightforward calculation shows that $\lceil \alpha v_{b_1} \rceil + \lfloor \alpha v_{b_1} \rfloor > \alpha$, for $\alpha \geq 4$. So, for large enough δ, $prob(n_1) > prob(n_2)$ as required.

As in Theorem 2, the proof when δ is odd follows by a similar argument. ∎

We next extend the result of Theorem 3 to game trees with more than two moves. The proof of the following corollary is exactly analogous to that of Corollary 3.

Corollary 4. *The result of Corollary 3 still holds when $b_2 = 1$.*

A subgame tree is *level-regular* if all nodes at the same level have the same branching factor, that is, the same number of children.

The next extension of Corollary 3 is an interesting and stronger form of decision oscillation pathology. Intuitively, it states that the this result still holds when we attach, between each move $n_j \in$ *root_moves*(T) and the subgame tree T$[n_j]$ below it, an arbitrary level-regular τ-ply subgame tree T_j, where τ is an even integer.

Definition 2 (Level-regular extension of a game tree). Let T$_\tau$ be a τ-ply level-regular subgame tree with branching factors $\mathbf{r} = (r_1, r_2, \ldots, r_\tau)$ at levels $0, 1, \ldots, \tau - 1$, respectively, where τ is an *even* integer.

For a given move $n \in root\_moves(\mathrm{T})$, let $\mathrm{T}'[n]$ be the $(\tau+\delta)$-ply subgame tree obtained by substituting the subgame tree $\mathrm{T}[n]$ for each leaf node of T_τ. We construct $\mathrm{T}'[n]$ in this way for each node $n \in root\_moves(\mathrm{T})$, where the branching factors $r_1, r_2, \ldots, r_\tau$ for different nodes n may be different. Let T' be the game tree where $root\_moves(\mathrm{T}')$ consists of the root nodes of $\mathrm{T}'[n]$ for $n \in root\_moves(\mathrm{T})$. We call T' a *level-regular extension* of T.

For a level-regular extension T' of T the propagation function for $\mathrm{T}'[n]$ is

$$F^\tau_{\mathbf{r}}(F^\delta_b(\cdot)),$$

where

$$F^\tau_{\mathbf{r}} = F_{r_1}(F_{r_2}(\cdots(F_{r_\tau}(\cdot))\cdots)).$$

Corollary 5. *The result of Corollary 3 holds for any level-regular extension of T when* $b_2 > 1$.

Proof. Let $n' \in root\_moves(\mathrm{T}')$ with propagation function $F^\tau_{\mathbf{r}}(F^\delta_b(\cdot))$. By the continuity of this function and Corollary 2, for any $x_0 \neq v_b$,

$$\lim_{\delta\to\infty} F^\tau_{\mathbf{r}}(F^\delta_b(x_0)) = F^\tau_{\mathbf{r}}\left(\lim_{\delta\to\infty} F^\delta_b(x_0)\right) = \lim_{\delta\to\infty} F^\delta_b(x_0),$$

since $F^\tau_{\mathbf{r}}(0) = 0$ and $F^\tau_{\mathbf{r}}(1) = 1$ for even τ.

Thus, when δ is large enough, the attached subgame tree T_τ serves to propagate, essentially unchanged, the values passed to it by the subgame trees $\mathrm{T}[n]$ substituted for its leaves. It follows that $prob(n)$ and $prob(n')$ tend to the same value as δ tends to infinity, which yields the result. ∎

Corollary 5 is interesting because it indicates that, provided the branching factor is uniform at the lower levels of each subgame tree $\mathrm{T}[n]$, the branching factors at the top levels have little effect on the probability of a move. We conjecture that Corollary 5 still holds for an arbitrary extension T_τ of T, even when T_τ is not level-regular, provided the probability distribution of the leaves is suitably constrained.

6 | CONCLUDING REMARKS

We have shown that, under the uniform game tree model, random minimaxing exhibits decision oscillation. That is, for moves whose subgame trees have different branching factors the minimax procedure will most probably change its selection of best move depending on whether the depth of lookahead is odd or even, provided the depth is large enough.

The results in this paper suggest that, in order to understand which of two moves is more likely to be chosen, we need to take into account the structure of the subgame trees rooted at these moves. Corollary 5 indicates that the lower levels of the tree have more influence on the minimax value. However, it may be the case that using game specific information, such as sibling dependence, decision oscillation will be eliminated.

A non-uniform random static evaluation function is obviously a more realistic model for a real game than a uniform random function. In a typical game-specific application, the evaluation tries to produce the same move ordering as would a function that estimated the probability of winning. The evaluation function is often a linear combination of terms, to which a squashing function (such as a sigmoid function; see Chap. 4 of Reference 31) could be applied to estimate the probability of winning. In this typical arrangement, the leaf probability values in the range $[0, 1]$ are not uniformly distributed.

The results we have presented generalise to evaluations with and without a squashing function – the only requirement on the squashing function is that it be monotonic. (Monotonicity ensures that the same leaf is selected by minimax.) This is due to the fact that when a monotonic function is applied to the leaf evaluations, the minimax procedure will still choose the same move at every node in the tree. The typical squashing functions used in game-specific applications, for example, the logistic function, are monotonic. Thus the result applies to any model in which a uniform random distribution is used to model the linear evaluation, with a squashing function applied to map the linear evaluation to a probability of winning.

Finally, we note that it follows from Reference 32 that Theorem 1 and Corollary 2 can be generalised to random evaluation functions that are not necessarily either uniform or identically distributed, as long as they are independent. Moreover, Nau also showed that the condition of independence can be somewhat relaxed. It follows that decision oscillation will also hold in this more general setting provided the probability distribution of the leaves is suitably constrained.

We have shown that decision oscillation is present in a situation where domination does not occur. It is an open problem to investigate, in detail, the relationship between these two phenomena. In this context it would also be interesting to find weaker conditions than domination[6] that eliminate decision oscillation. Finally, it remains to be seen how these results relate to practical game playing systems, and whether there is any connection between decision oscillation and the odd/even effect.[7] It would also be interesting to investigate game pathology in the context of MCTS, as it was shown in Reference 33 that pathology may also arise when MCTS is deployed.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Mark Levene https://orcid.org/0000-0001-8632-4732
Trevor Fenner https://orcid.org/0000-0002-4239-7574

REFERENCES

1. Shannon CE. Programming a computer for playing chess. *Philos Mag*. 1950;41:256-275.
2. Kaindl H. Tree searching algorithms. In: Marsland TA, Schaeffer J, eds. *Computers, Chess and Cognition*. Springer Verlag; 1990:133-158.
3. Marsland TA, Björnsson Y. From minimax to Manhattan. In: van den Herik HJ, Iida H, eds. *Games in AI Research*. Institute for Knowledge and Agent Technology IKAT, University of Maastricht; 2000: 5-17.
4. Bouzy B, Cazenave T, Corruble V, Teytaud O. Artificial intelligence for games. In: Marquis P, Papini O, Prade H, eds. *A Guided Tour of Artificial Intelligence Research*. Springer Nature; 2020:313-337.
5. Beal D, Smith MC. Random evaluation in chess. *Int Comput Chess Assoc J*. 1994;17:3-9.
6. Levene M, Fenner TI. The effect of mobility on minimaxing of game trees with random leaf values. *Artif Intell*. 2001;130:1-26.

7. Schaeffer J. The history heuristic and alpha-beta search enhancements in practice. *IEEE Trans Pattern Anal Mach Intell*. 1989;11:1203-1212.
8. Abramson B. Control strategies for two-player games. *ACM Comput Surv*. 1989;21:137-161.
9. Nau DS. Decision quality as a function of search depth on game trees. *J ACM*. 1983;30:687-708.
10. Nau DS. The last player theorem. *Artif Intell*. 1982;18:53-65.
11. Beal DF. Benefits of minimax search. In: Clarke MRB, ed. *Advances in Computer Chess 3*. Pergamon Press; 1982:17-24.
12. Pearl J. On the nature of pathology in game searching. *Artif Intell*. 1983;20:427-453.
13. Scheucher A, Kaindl H. Benefits of using multivalues functions for minimaxing. *Artif Intell*. 1998;99:187-208.
14. Luštrek M, Gams M, Bratko I. Is real-valued minimax pathological? *Artif Intell*. 2006;170:620-642.
15. Sadikov A, Bratko I, Kononenko I. Bias and pathology in minimax search. *Theor Comput Sci*. 2005;349:268-281.
16. Schrüfer G. Presence and absence of pathology in game trees. In: Beal DF, ed. *Advances in Computer Chess 4*. Pergamon Press, Oxford; 1986:101-112.
17. Lorenz U, Moien B. Error analysis in minimax trees. *Theor Comput Sci*. 2004;313:485-498.
18. Nau DS, Luštrek M, Parker A, Gams M, Bratko I. When is it better not to look ahead? *Artif Intell*. 2010;174:1323-1338.
19. Zuckerman I, Wilson B, Nau DS. Avoiding game-tree pathology in 2-player adversarial search. *Comput Intell*. 2018;34:542-561.
20. Liu C, Yan J, Ma Y, Zhao T, Zhang Q, Wei X. An adversarial search method based on an iterative optimal strategy. *Math*. 2020;8(9):1623.
21. Fu MC. A tutorial introduction to Monte Carlo tree search. Proceedings of the 2020 Winter Simulation Conference (WSC), Held virtually; 2020:1178-1193.
22. Świechowski M, Godlewski K, Sawick B, Mańdziuk J. Monte Carlo tree search: A review of recent modifications and applications. *Artif Intell Rev*. 2023;56:2497-2562.
23. Silver D, Hubert T, Schrittwieser J, Antonoglou I, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*. 2018;362:1140-1144.
24. Maharaj S, Polson N, Turk A. Chess AI: competing paradigms for machine intelligence. *Entropy*. 2022;24:13.
25. Romstad T, Costalba M, Kiiski J. Stockfish – Open Source Chess Engine; 2023. https://stockfish.org.
26. Cohen-Solal Q, Cazenave T. Minimax strikes back. Proceedings of the AAAI-21 Workshop on Reinforcement Learning in Games, Held virtually; 2021:8.
27. Baier H, Winands MHM. MCTS-minimax hybrids with state evaluations. *J Artif Intell Res*. 2018;62:193-231.
28. Guy RK. What is a game? In: Guy RK, ed. *Combinatorial Games, volume 43 of Proceedings of Symposia in Applied Mathematics*. American Mathematical Society; 1991:1-21.
29. Levene M, Fenner TI. A partial analysis of minimaxing game trees with random leaf nodes. *Int Comput Chess Assoc J*. 1995;18:20-33.
30. Pearl J. Asymptotic properties of minimax trees and game-searching procedures. *Artif Intell*. 1980;14:113-138.
31. Mitchell TM. *Machine Learning*. McGraw-Hill; 1997.
32. Nau DS. Pathology on game trees revisited, and an alternative to minimaxing. *Artif Intell*. 1983;21:221-244.
33. Nguyen KPN, Ramanujan R. Lookahead pathology in Monte-Carlo tree search. *Artif Intell*, arXiv:2212.05208 [cs.AI]. 2022. https://arxiv.org/abs/2212.05208.

How to cite this article: Levene M, Fenner T. The phenomenon of decision oscillation: A new consequence of pathology in game trees. *Computational Intelligence*. 2023;1-13. doi: 10.1111/coin.12570