On Minimax Game Tree Search Pathology and Node-Value Dependence

*Liwu Li** and *T. A. Marsland*
Computing Science Department, University of Alberta
Edmonton, Canada T6G 2H1

July 27, 1990

Abstract

Here we are concerned with the mysterious phenomenon of minimax game-tree search pathology, where it appears and how it happens. It is commonly believed that the strategy of searching-deeper for computer game-playing programs can enhance the accuracy of position evaluation and increase the possibility of detecting the correct move. The strategy has been successfully implemented in practice; but it has not been justified theoretically, and various previous investigations of the phenomenon were based on uniform trees whose terminal node values are independent from each other. Those trees fail to account for the apparent relationship between the node values for common games and, therefore, cannot be used to explain this pathology. Here, we present a new method to introduce node-value dependence into board-splitting games and relate the pathological phenomenon to the dependence. In particular, we study the effect of minimax searching-deeper for the games by using an evaluation of invariant accuracy with respect to the search depths; we also examine the relationship between the quality of minimax search and the node value dependence of the game trees by assuming a real evaluation function described by Nau [8]. The results of both approaches reveal that the pathological phenomenon is related to weak node value dependence and confirm that searching deeper is effective for game trees with strong node-value dependence.

1 Introduction

Computer game playing is a process of making decisions based on a game tree. It needs to search the tree to compute the merit values for the available moves. Since the number of nodes in the tree usually grows exponentially with its depth, it is not feasible to do a complete search, and almost all the computer game-playing programs use some depth-limited heuristics [6]. Heuristic look-ahead search is a successful technique, and there is a strong evidence that increasing the depth of the search would improve the quality of the decision for choosing a correct move [11, 14]. However, the investigations presented in the literature [8, 12, 1] show that, given a certain theoretical model of the errors made by an evaluation function, there exists an infinite class of game trees which are *pathological*. By pathological we mean that as

*Current address of the author is Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

long as the search does not reach the end of the games (in which case, a correct decision is guaranteed), searching deeper does not increase the probability of making a correct decision, but will instead cause the decision to become increasingly random [8].

The phenomenon of minimax game-tree search pathology, that a deeper search degrades the decision quality, has attracted much attention in the literature. For example, by assuming a uniform game-tree model, Beal [1] showed that the error probability will not be reduced by minimax back-up, and fixed-depth backed-up values are less trustworthy than the static values. A class of board-splitting games, which are called Pearl's games, was used by Nau [8] to demonstrate the existence of minimax search pathology for an obvious evaluation function. For these uniform game trees, Pearl [12] assumed an accuracy-invariant position evaluation and showed that the minimax process introduces a spurious noise which may cause the pathological phenomenon. In summary, the conclusion in the literature is that minimax back-up cannot reduce the errors presented in static evaluation, and several explanations or speculations for the success of minimax search for chess or other games have been proposed.

The results about game-tree search pathology mentioned above were based on the assumption that the merit values of sibling nodes in a game tree are independent from each other [8, 12] or are loosely dependent that the *win* node proportion at each level is the same [1]. This is apparently not a satisfactory model for common games. Here, we present a different method to assign merit values to terminal nodes in uniform trees. This method introduces some degree of dependence between the values of related nodes. The resulting trees correspond to a class of board-splitting games and provide a basis for observing the minimax process. With these games, we can investigate the relationship between the quality of decision made by a minimax search and the dependence measure of the merit values of related nodes in the trees. The investigation begins with some simple cases of extremely strong or weak dependence. Then, the effect of minimax search with respect to eliminating evaluation errors is studied by assuming a bi-valued evaluation function whose accuracy does not vary with the search depth. (This assumption was used by Pearl [12].) We also test the performance of the minimax process with respect to choosing the correct move by searching to various depths with a piece-counting evaluation function of Nau [8]. By a piece we mean a square labeled with *win* status in the corresponding portion of the game board. Mathematical formulas are developed for computing the probabilities of making a correct decision. Numerical results obtained from these formulas are used to show the effect of node-value dependence on minimax pathology.

2 Related Works

Uniform trees have been extensively used in the literature to study the efficiencies of different game-tree search algorithms [4, 3, 13, 7]. These trees differ from each other in assigning artificial merit values to their nodes. To examine the phenomenon of minimax game-tree search pathology, several methods were used to assign merit values for terminal nodes in a uniform tree. For example, a model was presented by Beal [2] to show that a minimax game-tree search method, named *locked-value search*, was beneficial. In Beal's model, true values, which correspond to our notion of merit values, have the minimax relationship; they

are distributed so that at each level, the proportions of *wins* for the player to move is the same, and across each level they tend to form clusters of identical values. Our assignment of merit values also follows the minimax relationship, but the distribution of *wins* following a node will be conditioned on the merit value of the node so that the dependence of successor nodes on their predecessors can be introduced. Therefore, an essential difference between Beal's model and ours is that the latter tries to simulate the observation that for common games like chess, strong positions are likely to be followed by strong positions, but the former simulates it by enforcing a clustering tendency.

The uniform trees used by Nau [8] and Pearl [12] have the same deficiency as the above mentioned Beal's model. Each of the terminal nodes in those trees is assigned a status, which is *win* or *loss*, with a fixed probability p or $1-p$. In Nau's work, for a b-ary tree, p was chosen as the unique solution w_b of the equation $(1-x)^b = x$ in the interval $(0,1)$. Therefore, at an even (odd) level, the proportion of *wins* is w_b ($1-w_b$, respectively). In other words, when a node has a *win* merit value, this value should not affect the proportion of *wins* for its successors. In Pearl's analysis, when we choose $p < w_b$ or $p > w_b$, the probability for the root position to have a *win* merit value is either very close to 0 or very close to 1, obviously unrealistic. An important observation about those trees is that no evidence of minimax benefits can be observed on them, which is contrary to the success of minimax search in practice.

To introduce incrementally changed node values, Nau [8] used the *incremental games* to show the absence of minimax search pathology for them. These board-splitting games are set up by randomly choosing the integer 1 or -1 with probability q or $1-q$ for each arc of the uniform trees. If the sum of the arc values along the path from a terminal node back to the root is positive, a merit value *win* is assigned to the node, *loss* otherwise. The incremental games cannot be used to illustrate the distinction between game trees with respect to minimax search pathology. Therefore, they cannot reveal the characteristics of game trees which are possibly related to minimax search pathology. A further question about incremental games is the discrepancy between the strength of an interior node (which is defined as the sum of the arc values on the path leading to it from the root) and the merit value of the node (which is the value backed up from terminal nodes by a minimax process). We have not found an intuitive interpretation for the discrepancy, which might contribute to the difficulty of a mathematical analysis for these games.

We should mention the two important approaches of studying minimax game-tree search pathology, established by Nau [8] and Pearl [12], respectively. Nau used a real evaluation function for the class of Pearl's games and developed basic mathematical formulas to demonstrate the pathological phenomenon; Pearl assumed error probabilities for position evaluation and discussed how these probabilities would be changed along with the minimax back-up process. Here, we use both approaches to analyze the relationship between the node-value dependence and the pathological phenomenon. The analysis produces mathematical formulas and results different from both those of Nau [8] and those of Pearl [12].

3 Assigning Dependent Node Values for Uniform Trees

3.1 The Terminology for Game Trees

A two-player zero-sum perfect-information game can be described by a *game tree*, where each node represents a game position, and each possible move from a position corresponds to an arc which leads to another node, called a *child* or *successor*. The two players, called Max and Min, take strict alternate turns to move. Thus, the tree consists of Min nodes and Max nodes, which represent positions that are the turns of the players Min and Max, respectively. The *degree* of a node is defined as the number of its children. A *terminal node* has a degree of 0. The *depth* of a node in a game tree is the number of arcs or moves required to reach it from the *root node*, which represents the starting position. In a *uniform tree*, all the terminal nodes are at the same depth d, and all the interior nodes have the same degree. We also use the notion of height to describe a uniform tree or a node g in it. The *height* of node g is the number of arcs required to reach a terminal node from g in the tree; the *height* of a uniform tree is defined as the height of its root node.

In our discussion, it is essential to distinguish a *search tree*, the part of a game tree explored by a computer program, from the game tree itself, and distinguish the value returned by an evaluation function for a position from the merit value of the position. In this paper, we assume both game trees and search trees are uniform trees. The values determined by an evaluation function for the terminal nodes of a search tree are called *utility values*. The *merit values* for the terminal nodes in the game tree, which represent the terminal positions of the game, are determined by the game rules. We represent the merit values of terminal nodes in a game tree with a partial function

$$\phi : V \rightarrow D,$$

where V is the set of nodes in the game tree, and D is the set of all possible merit values. Under the assumption that both players play their best, the function ϕ can be extended to a total function

$$\Phi : V \rightarrow D,$$

which determines the merit value $\Phi(g)$ for each node g of the game tree. The extension is provided by the *minimax back-up* process, which is described as follows. For a terminal node g, we have

$$\Phi(g) = \phi(g);$$

for an interior node g, the merit value $\Phi(g)$ is related to that of its children according to the rule: if g is a Max node,

$$\Phi(g) = max\{\Phi(g_i) | g_i \text{ is a child of } g\};$$

if g is a Min node,

$$\Phi(g) = min\{\Phi(g_i) | g_i \text{ is a child of } g\}.$$

In this way, each node g in a game tree is assigned a unique merit value $\Phi(g)$.

Based on the intuition that an evaluation function should provide an accurate estimate of $\Phi(g)$ for the terminal nodes g in a search tree, the utility values $\psi(g)$ are also backed up

to the shallower nodes g' by the minimax process. The backed-up utility values are denoted as $\Psi(g')$.

3.2 A Class of Board-Splitting Games

We shall present a class of board-splitting games, called node-dependent games, or simply D-games.[1] The D-games have the same rule as Pearl's games and Nau's incremental games. The playing board for a D-game is measured $b^{\lceil h/2 \rceil} \times b^{\lfloor h/2 \rfloor}$ of unit squares, for some integers b and h. The initial configuration is described by the assignment of *win* or *loss* for each square. A *move* for the first player consists of horizontally dividing the remaining board into b parallel parts and choosing one of them; a *move* for the second player consists of vertically dividing the remaining board into b parallel columns and choosing one of them. If the last player can get a *win*-square, this player wins the game; otherwise, the opponent wins. To make the recursive calculation of some probabilities easy and independent of the height h of game trees, we assume that the player who makes the last move is the player Max.

The D-games, which will be defined shortly, differ from Pearl's and Nau's games by the way of assigning *win* and *loss* values for the board squares. Informally speaking, our definition of a D-game simulates the process of a game design. We can imagine that to design a game, first, an initial configuration of the game must be set up, and then, a set of rules are specified to transform one position to others. When plotting the initial configuration, we should expect the game to be a "fair" one, which means that the probability for a player to win the game is about 0.5, or a little higher if the advantage of making the first move is taken into account. Therefore, we need a condition to "describe" the expected probability for the initial configuration. When comparing alternative move rules, we would expect each player to have a chance to continue the game and a chance to win the game. These chances will correspond to the conditional probabilities used in the definition of the D-games, which relate the probabilities of the possible merit values for a position to that of its children, and "describe" the game rules.

The game trees of D-games can also be seen as a variation of the branch-weighed trees, which were defined by Newborn [10] and can be described iteratively as follows. First, the root node receives a static value 0. After an interior node g receives a static value $m(g)$, the b arcs c_i, $1 \leq i \leq b$, following node g are assigned random values $m(c_i)$; the children g_i of g receive the values $m(g) + m(c_i)$ as their static values. The above step is repeated until all the terminal nodes in a uniform tree receive their static values. In many ways, this technique for building trees is similar to that used later for probabilistically ordered trees [7]. For the D-games, instead of choosing the arc values, we directly assign random values to the children g_i of a node g. To reduce the incompatibility present in the branch-weighed trees, the random values received by the children g_i are treated as their merit values $\Phi(g_i)$, which would be backed up from the terminal nodes in Pearl's or Nau's games. In this way, the merit values of the children of a node g can be related to that of g to simulate the node-value dependence.

In the D-game trees, the merit values are determined in a *top-down* manner. First, the merit value of the root node is defined as a random variable. After the merit value of

[1] The letter D stands for "dependent".

an interior node g is determined, we randomly choose the merit values for its children g_i according to a set of probabilities which are conditioned on the merit value of g. In the following definition, we use the conventional notation $\Pr[E]$ for the probability of event E, and $\Pr[E|C]$ for the probability of event E given condition C.

Definition 3.1 Given integers b, d and a finite set D of values, a *node-dependent game tree*, or simply, a *D-game tree*, is a uniform tree of branching degree b and depth d that is set up with the following three conditions:

1. The root node is randomly assigned with a value v from set D with a probability distribution $P_0(v)$ for $v \in D$.

2. For an interior Max node g, the merit values $\Phi(g_i)$ $(1 \leq i \leq b)$ of the children are randomly determined by a set of conditional probabilities

$$\Pr[\Phi(g_1) = v_1, \Phi(g_2) = v_2, \ldots, \Phi(g_b) = v_b | \Phi(g) = v_0], \tag{1}$$

   where $v_i \in D$ for $0 \leq i \leq b$.

3. For an interior Min node g, the merit values $\Phi(g_i)$ $(1 \leq i \leq b)$ of the children are randomly determined by a set of conditional probabilities

$$\Pr[\Phi(g_1) = v'_1, \Phi(g_2) = v'_2, \ldots, \Phi(g_b) = v'_b | \Phi(g) = v'_0], \tag{2}$$

   where $v'_i \in D$ for $0 \leq i \leq b$. □

Although a D-game tree can be regarded as a variation of the branch-weighed trees, it is "better" in two aspects. First, the domain D for the merit values can be any fixed set, which is independent of both the degree b and the depth d of the D-game tree. For bi-valued D-games, we can directly determine the merit values from set $D = \{win, loss\}$.[2] For a branch-weighted tree, the sums of arc weights along paths from the root node to terminal nodes can be arbitrarily large and must be "normalized" before they are assigned to the terminal nodes. Second, the conditional probabilities specified in conditions 2 and 3 of Definition 3.1 provide an amenable control over the properties of the D-game tree. As a consequence, we gain a flexibility for modeling games with different dependence measures for related nodes. As demonstrated later, because of this flexibility, we can investigate the relationship between minimax game-tree search pathology and node value dependence, which appears in common games. To define a minimax game tree, Definition 3.1 has to satisfy some conditions.

In a D-game tree, the merit values are determined by the probability distribution $P_0(v)$ specified for the root node and the conditional probabilities specified in conditions 2 and 3. Since these probabilities refer to *merit* values, in addition to some completeness requirements, they must satisfy the *consistency* requirements, which reflect the minimax relationship between the merit value of a parent node and that of its children. For example, a Max node g has merit value $\Phi(g) = v_0$ if and only if all its children have merit values less than or equal

[2] Bi-valued binary trees played an important role in analysis of tree search procedures and investigation of minimax pathology in the literature.

to v_0, and at least one child's merit value is v_0. From these requirements, the conditional probabilities specified in (1) must satisfy the following logical implications:

$$\text{if } \forall_i.(v_i \neq v_0) \text{ then } \Pr[\Phi(g_1) = v_1, \Phi(g_2) = v_2, \ldots, \Phi(g_b) = v_b | \Phi(g) = v_0] = 0,$$
$$\text{if } \exists_i.(v_i > v_0) \text{ then } \Pr[\Phi(g_1) = v_1, \Phi(g_2) = v_2, \ldots, \Phi(g_b) = v_b | \Phi(g) = v_0] = 0.$$

The conditional probabilities in equation (2) have to satisfy some similar propositions.

Note that in both Pearl's game trees and branch-weighed trees, all the terminal nodes take a number as static values with equal probabilities. But Definition 3.1 does not enforce this for D-game trees. Therefore, we assume the conditional probabilities in (1) (and correspondingly (2)) satisfy

$$\Pr[\Phi(g_1) = v_1, \Phi(g_2) = v_2, \ldots, \Phi(g_b) = v_b | \Phi(g) = v_0] =$$
$$\Pr[\Phi(g_1) = w_1, \Phi(g_2) = w_2, \ldots, \Phi(g_b) = w_b | \Phi(g) = v_0]$$

for each pair of vectors $\mathbf{v} = \langle v_1, v_2, \ldots, v_b \rangle$ and $\mathbf{w} = \langle w_1, w_2, \ldots, w_b \rangle$, provided $\mathbf{v}$ is a permutation of $\mathbf{w}$. By simple induction on depth, we can show that for any two nodes g and g' at the same depth in a D-game tree, we have

$$\Pr[\Phi(g) = v] = \Pr[\Phi(g') = v]$$

for any $v \in D$, i.e., $\Phi(g)$ and $\Phi(g')$ are identically distributed random variables.

In this paper, we shall consider bi-valued binary D-game trees. In other words, the degree b in these trees is fixed to 2, the merit value domain $D = \{win, loss\}$, and the final value *loss* for the last player means losing and *win* means winning. We shall use $d = 2k$ or $d = 2k + 1$, for some integer $k \geq 0$, to represent the depth of a node in a game tree or search tree. For these trees, the probability $P_0(win)$ specified in condition 1 of Definition 3.1 will be denoted as p_0, and some conditional probabilities are trivial: for a Max node g

$$\Pr[\Phi(g_1) = loss, \Phi(g_2) = loss | \Phi(g) = loss] = 1;$$

for a Min node g

$$\Pr[\Phi(g_1) = win, \Phi(g_2) = win | \Phi(g) = win] = 1.$$

In addition to these conditional probabilities, conditions 2 and 3 in Definition 3.1 will be respectively replaced by conditions:

2'. For an interior *win* Max node g, the children's merit values, denoted as $\Phi(g_1)$ and $\Phi(g_2)$, are randomly determined by the conditional probabilities

$$\Pr[\Phi(g_1) = win, \Phi(g_2) = win | \Phi(g) = win] = f_1,$$
$$\Pr[\Phi(g_1) = win, \Phi(g_2) = loss | \Phi(g) = win] = \tfrac{1}{2}(1 - f_1),$$
$$\Pr[\Phi(g_1) = loss, \Phi(g_2) = win | \Phi(g) = win] = \tfrac{1}{2}(1 - f_1)$$

with $0 \leq f_1 \leq 1$.

3'. For an interior *loss* Min node g, the children's merit values are randomly determined by the conditional probabilities

$$\Pr[\Phi(g_1) = loss, \Phi(g_2) = loss | \Phi(g) = loss] = f_2,$$
$$\Pr[\Phi(g_1) = win, \Phi(g_2) = loss | \Phi(g) = loss] = \tfrac{1}{2}(1 - f_2),$$
$$\Pr[\Phi(g_1) = loss, \Phi(g_2) = win | \Phi(g) = loss] = \tfrac{1}{2}(1 - f_2)$$

with $0 \leq f_2 \leq 1$.

The parameters f_1 and f_2 describe the dependence among the merit values of sibling nodes as well as among that of children and their parent. They will be called *dependence factors*. As noted before, both children g_1 and g_2 of a node g have merit value *win* (or *loss*) with the same probability; as a result, all the terminal nodes in a D-game tree have the same probability to receive value *win* (or *loss*). This fact implies that the merit values assigned to the terminal positions in a D-game are identically distributed. A position (node) will be called a *win* position (node) g if it has $\Phi(g) = win$, which is also denoted as $\Phi(g) = 1$ or $win(g)$. Similarly, we denote a *loss* position (node) g with $\Phi(g) = loss$, $\Phi(g) = 0$ or, simply, $loss(g)$.

Before we study minimax game tree search pathology, we establish a property in the following theorem, which distinguishes the D-games from Pearl's games.

Theorem 3.1 *Given a positive real number p_0 as the probability for the root node that is a Max node of a D-game tree to take value win, if the dependence factors f_1 and f_2 satisfy the equation*

$$p_0 \left(1 - \frac{1}{4}(1 + f_1)(1 + f_2)\right) = \frac{1}{2}(1 - f_2),$$

every node at an even depth in the D-game tree takes value win with the same probability, which is p_0.

Proof:

For the three-level subtree shown in Fig. 1, let the Max node g_0 take value *win* with probability p_0. Then, the Min node g_1 takes value *win* with a probability of

$$p_0 \times f_1 + p_0 \times \frac{1}{2}(1 - f_1) = p_0 \times \frac{1}{2}(1 + f_1),$$

and the Max node g_3 takes value *win* with a probability of

$$p_0 \times \frac{1}{2}(1 + f_1) + (1 - p_0 \times \frac{1}{2}(1 + f_1)) \times \frac{1}{2}(1 - f_2).$$

Therefore, if f_1 and f_2 satisfy

$$p_0 \times \frac{1}{2}(1 + f_1) + (1 - p_0 \times \frac{1}{2}(1 + f_1)) \times \frac{1}{2}(1 - f_2) = p_0,$$

which is

$$p_0 \left(1 - \frac{1}{4}(1 + f_1)(1 + f_2)\right) = \frac{1}{2}(1 - f_2),$$

Max node g_3 will take value *win* with the probability p_0. This proves the theorem. □

A similar equation can be derived for D-games which have a Min root node.

Here, we can compare D-games with Pearl's games with the following fact. As shown by Nau [9, 8], if both the leaf nodes and the root node of a binary bi-valued Pearl's game tree of even height take the value *win* with the same probability w_2, w_2 is equal to the unique solution to equation

$$x = (1 - x)^2$$

in the range (0, 1).

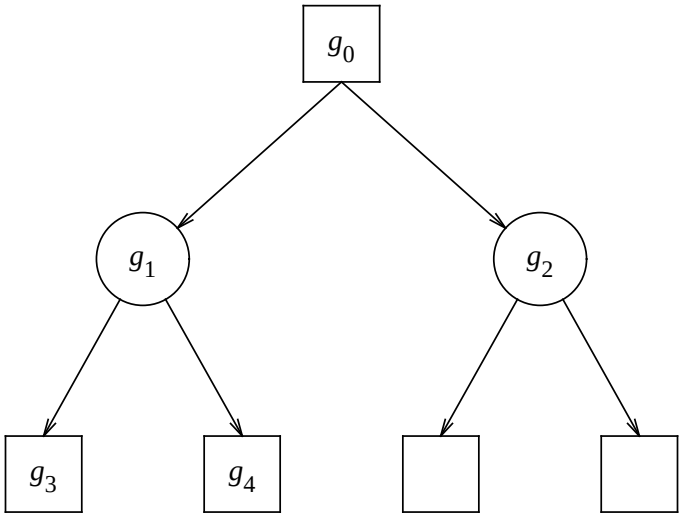

Figure 1: A three-level subtree.

4 Special Cases for Multi-Valued Evaluation

In this section, we use some D-games, which are established with extreme dependence factors, to illustrate Definition 3.1 and the relationship between the probabilities defined there and the decision made by a minimax search for choosing the correct moves. But first, we describe a position evaluation function, named as ψ, which was first used by Nau [8] to analyze Pearl's games. Let g be a node in the game tree of a board-splitting game like a Pearl's game, Nau's game or D-game, let g_1 and g_2 be the children of g. If g_1 has more 1-squares in its board configuration than g_2, it would seem more likely that g_1 is a *win* position than g_2. The number of *win*-terminal positions, or 1-squares, contained in a game position is a reasonable estimate for the strength of the corresponding node in board-splitting game trees. The number of 1-squares in the position of node g will be denoted by $\psi(g)$, which is the *utility value* of the position when g is a terminal node of a search tree.

When a D-game is established with an extreme dependence factor, such as f_1 or f_2 equal to 0 or 1, there is a relationship between the merit values of successors of some nodes. For example, when $f_1 = 1$, the successors' values for a Max node g are totally correlated so that

$$\Phi(g_1) = 1 \text{ iff } \Phi(g_2) = 1.$$

Similarly, if $f_2 = 1$, the successors' merit values for a Min node g have the following relation:

$$\Phi(g_1) = 0 \text{ iff } \Phi(g_2) = 0.$$

The extreme value $f_1 = 0$ implies that for any Max node g,

$$\text{if } \Phi(g_1) = 1 \text{ then } \Phi(g_2) = 0,$$
$$\text{if } \Phi(g_2) = 1 \text{ then } \Phi(g_1) = 0,$$

and $f_2 = 0$ implies that for any Min node g,

$$\text{if } \Phi(g_1) = 0 \text{ then } \Phi(g_2) = 1,$$
$$\text{if } \Phi(g_2) = 0 \text{ then } \Phi(g_1) = 1.$$

We shall show that some of these extreme values of dependence factors do not lead to minimax search pathology.

When $f_1 = 1$, each interior *win* Max node has two *win* successors. Based on this fact, we can prove that a *win* node will have more 1-squares in its configuration than a *loss* node of the same height h in D-games, where h can be any natural number. This property implies that searching deeper in a D-game tree with $f_1 = 1$ will not decrease the probability of making a correct decision.

Proposition 4.1 *For a D-game that is generated with $f_1 = 1$, the evaluation function ψ can be used to choose the correct move with a search to depth d for any $d \geq 1$.*
Proof:

Let the terminal nodes in a search tree have height h. By a simple induction on h, we can prove that a *win* node g of height h has 2^h 1-squares in its configuration (i.e., all the terminal nodes underneath g have a merit value 1), and a loss node of the same height h has less than 2^h 1-squares in the corresponding position. Therefore, if node g is at the minimax search front, we can see that a move leads to *win* if and only if the minimax backed-up value for the corresponding node is equal to 2^h, and a move leads to *loss* if and only if the backed-up value is less than 2^h. Therefore, the player at the root node can correctly choose a move by evaluating the terminal nodes of the search tree with function ψ. □

We can also prove a similar conclusion for a D-game tree that is generated with $f_2 = 1$. In fact, if $f_2 = 1$, all the terminal nodes in a D-game tree underneath a *loss* node must have merit value *loss*, and the evaluation function ψ can always be used to find the best move with minimax search.

When $f_1 = 0$, each interior *win* Max node has exactly one *win* successor; when $f_2 = 0$, each interior *loss* Min node has exactly one *loss* successor. Based on this fact, we can prove that in a D-game tree that is generated with $f_1 = f_2 = 0$, a *win* node has more 1-squares in its configuration than a *loss* node of the same height, and all *win* nodes of the same height have the same number of 1-squares, and so do all *loss* nodes of the same height. These properties imply that searching deeper in a D-game tree which is generated with $f_1 = f_2 = 0$ will not decrease the probability of making a correct decision at the root node. The proof of the following proposition is simple and can be found in the first author's thesis [5].

Proposition 4.2 *For a D-game that is generated with $f_1 = f_2 = 0$, the evaluation function ψ can be used to choose the correct move with a search to depth d for any integer $d \geq 1$.* □

When only one of the two dependence factors f_1 and f_2 is 0, the above conclusion is no longer true. In fact, when $f_1 = 0$, $f_2 \neq 0$ or $f_1 \neq 0$, $f_2 = 0$, we have a positive probability to generate D-game trees which have an interior *win* node g and *loss* node g' of the same height with

$$\psi(g) < \psi(g').$$

Therefore, the properties mentioned in Propositions 4.1 and 4.2 do not hold for these combinations of dependence factors. In fact, we shall see that minimax search with the evaluation function ψ may make a wrong decision with a positive probability.

5 Minimax Error Propagation in D-Game Trees

We shall use binary bi-valued D-game trees with different dependence factors f_1 and f_2 to study the merits of minimax search. Particularly, we assume a bi-valued position evaluation function ψ which has a predetermined error probability for all search depths, and study the relationship between the error probabilities implied by the minimax back-up and the dependence factors of the D-game trees.

5.1 Minimax Error-Propagation

Let G denote a search tree for a binary bi-valued D-game tree, which is generated by dependence factors f_1 and f_2. We assume that each terminal node g of G is assigned a utility value by a bi-valued evaluation ψ, with $\psi(g) = 1$ ($\psi(g) = 0$) to specify a strong (weak, respectively) position. We assume the evaluation does not examine the features of the sibling position. The consequence of this assumption is that, given the merit value $\Phi(g)$, the utility value $\psi(g)$ of a node g is independent of the merit value of g's sibling node. In other words, $\psi(g)$ depends only on the portion of the D-game tree that is rooted at g. The utility value $\Psi(g)$ of a shallower node g in the search tree G is calculated by the minimax back-up process. This value would constitute somewhat unreliable prediction on the merit value $\Phi(g)$ of the position. An evaluation error is made when $\Psi(g) = 1$ is assigned to a node g in G which in reality represents a *loss* position ($\Phi(g) = 0$), and *vice versa*, a value of $\Psi(g) = 0$ is assigned to a *win* node g.

To study the propagation of errors along with the minimax back-up of evaluation results, let us focus on a three-level subtree G' inside G, which is shown in Fig. 1. We assume the root node g_0 of G' is a Max node at depth $2(k-1)$, the two children of node g_0 are g_1 and g_2, and the children of g_1 are g_3 and g_4, which are located at depth $2k$. We use symbol Ψ_i to denote the utility value of node g_i for $0 \leq i \leq 4$, and Φ_i to denote the merit value of node g_i. Using these notations, the propagations of Φ and Ψ should follow the minimax back-up. More specifically, we have

$$\Phi_0 = \begin{cases} \mathit{win} & \text{if } \Phi_1 = \mathit{win} \text{ or } \Phi_2 = \mathit{win}, \\ \mathit{loss} & \text{if } \Phi_1 = \mathit{loss} \text{ and } \Phi_2 = \mathit{loss}; \end{cases}$$

$$\Phi_1 = \begin{cases} \mathit{win} & \text{if } \Phi_3 = \mathit{win} \text{ and } \Phi_4 = \mathit{win}, \\ \mathit{loss} & \text{if } \Phi_3 = \mathit{loss} \text{ or } \Phi_4 = \text{loss}; \end{cases}$$

$$\Psi_0 = \begin{cases} 1 & \text{if } \Psi_1 = 1 \text{ or } \Psi_2 = 1, \\ 0 & \text{if } \Psi_1 = 0 \text{ and } \Psi_2 = 0; \end{cases}$$

$$\Psi_1 = \begin{cases} 1 & \text{if } \Psi_3 = 1 \text{ and } \Psi_4 = 1, \\ 0 & \text{if } \Psi_3 = 0 \text{ or } \Psi_4 = 0. \end{cases}$$

The informedness of the utility value Ψ_i at a node g_i is quantified by two parameters [12]:[3]

$$\gamma_i = \Pr[\Psi_i = 1 | \Phi_i = loss],$$
$$\delta_i = \Pr[\Psi_i = 0 | \Phi_i = win],$$

for $i = 0, \ldots, 4$. Since Φ_1 and Φ_2 (Φ_3 and Φ_4) are identically distributed random variables, it is reasonable to assume $\gamma_1 = \gamma_2$, $\delta_1 = \delta_2$ ($\gamma_3 = \gamma_4$, $\delta_3 = \delta_4$). The dependence factors f_1 and f_2 are involved in calculating error propagation. By error propagation we mean determining γ_0 and δ_0 from γ_3 and δ_3. But first, we calculate γ_1 and δ_1. In the computation, we shall use the equation

$$\Pr[\Psi_i = v_1 | \Phi_i = v_2, \Phi_j = v_3] = \Pr[\Psi_i = v_1 | \Phi_i = v_2],$$

where nodes g_i and g_j are sibling nodes, like g_1 and g_2 or g_3 and g_4, and the values v_1, v_2, $v_3 \in \{0, 1\}$. This equation reflects our assumption that given its merit value, the utility value of a node does not relate to that of its sibling node. Since node g_1 is a Min node, by the minimax back-up rules for Φ and Ψ and the definition of dependence factor f_2, we have

$$\begin{aligned}
\gamma_1 &= \Pr[\Psi_1 = 1 | \Phi_1 = loss] \\
&= \Pr[\Psi_3 = \Psi_4 = 1 | \Phi_1 = loss] \\
&= \Pr[\Psi_3 = \Psi_4 = 1 | \Phi_3 = \Phi_4 = loss] \times \Pr[\Phi_3 = \Phi_4 = loss | \Phi_1 = loss] \\
&\quad + \Pr[\Psi_3 = \Psi_4 = 1 | \Phi_3 = win, \Phi_4 = loss] \times \Pr[\Phi_3 = win, \Phi_4 = loss | \Phi_1 = loss] \\
&\quad + \Pr[\Psi_3 = \Psi_4 = 1 | \Phi_3 = loss, \Phi_4 = win] \times \Pr[\Phi_3 = loss, \Phi_4 = win | \Phi_1 = loss] \\
&= \Pr[\Psi_3 = 1 | \Phi_3 = loss] \times \Pr[\Psi_4 = 1 | \Phi_4 = loss] f_2 \\
&\quad + \Pr[\Psi_3 = 1 | \Phi_3 = win] \times \Pr[\Psi_4 = 1 | \Phi_4 = loss] \frac{1}{2}(1 - f_2) \\
&\quad + \Pr[\Psi_3 = 1 | \Phi_3 = loss] \times \Pr[\Psi_4 = 1 | \Phi_4 = win] \frac{1}{2}(1 - f_2) \\
&= \gamma_3^2 f_2 + \gamma_3(1 - \delta_3)(1 - f_2),
\end{aligned}$$

$$\begin{aligned}
\delta_1 &= \Pr[\Psi_1 = 0 | \Phi_1 = win] \\
&= \Pr[\Psi_3 = \Psi_4 = 0 | \Phi_1 = win] \\
&\quad + \Pr[\Psi_3 = 1, \Psi_4 = 0 | \Phi_1 = win] \\
&\quad + \Pr[\Psi_3 = 0, \Psi_4 = 1 | \Phi_1 = win] \\
&= \Pr[\Psi_3 = \Psi_4 = 0 | \Phi_3 = \Phi_4 = win] \\
&\quad + \Pr[\Psi_3 = 1, \Psi_4 = 0 | \Phi_3 = \Phi_4 = win] \\
&\quad + \Pr[\Psi_3 = 0, \Psi_4 = 1 | \Phi_3 = \Phi_4 = win] \\
&= \Pr[\Psi_3 = 0 | \Phi_3 = win] \times \Pr[\Psi_4 = 0 | \Phi_4 = win] \\
&\quad + \Pr[\Psi_3 = 1 | \Phi_3 = win] \times \Pr[\Psi_4 = 0 | \Phi_4 = win] \\
&\quad + \Pr[\Psi_3 = 0 | \Phi_3 = win] \times \Pr[\Psi_4 = 1 | \Phi_4 = win]
\end{aligned}$$

[3] Here we avoid using the letters α and β, since they have been widely used in minimax game-tree search for search window bounds.

$$= \delta_3^2 + 2\delta_3(1-\delta_3)$$
$$= 2\delta_3 - \delta_3^2.$$

Since node g_0 is a Max node, we have

$$\begin{aligned}
\gamma_0 &= \Pr[\Psi_0 = 1 | \Phi_0 = loss] \\
&= \Pr[\Psi_1 = \Psi_2 = 1 | \Phi_0 = loss] \\
&\quad + \Pr[\Psi_1 = 1, \Psi_2 = 0 | \Phi_0 = loss] \\
&\quad + \Pr[\Psi_1 = 0, \Psi_2 = 1 | \Phi_0 = loss] \\
&= \Pr[\Psi_1 = \Psi_2 = 1 | \Phi_1 = \Phi_2 = loss] \\
&\quad + \Pr[\Psi_1 = 1, \Psi_2 = 0 | \Phi_1 = \Phi_2 = loss] \\
&\quad + \Pr[\Psi_1 = 0, \Psi_2 = 1 | \Phi_1 = \Phi_2 = loss] \\
&= \Pr[\Psi_1 = 1 | \Phi_1 = loss] \times \Pr[\Psi_2 = 1 | \Phi_2 = loss] \\
&\quad + \Pr[\Psi_1 = 1 | \Phi_1 = loss] \times \Pr[\Psi_2 = 0 | \Phi_2 = loss] \\
&\quad + \Pr[\Psi_1 = 0 | \Phi_1 = loss] \times \Pr[\Psi_2 = 1 | \Phi_2 = loss] \\
&= \gamma_1^2 + 2\gamma_1(1-\gamma_1) \\
&= 2\gamma_1 - \gamma_1^2 \\
&= 2\left(\gamma_3^2 f_2 + \gamma_3(1-\delta_3)(1-f_2)\right) - \left(\gamma_3^2 f_2 + \gamma_3(1-\delta_3)(1-f_2)\right)^2.
\end{aligned}$$

$$\begin{aligned}
\delta_0 &= \Pr[\Psi_0 = 0 | \Phi_0 = win] \\
&= \Pr[\Psi_1 = \Psi_2 = 0 | \Phi_0 = win] \\
&= \Pr[\Psi_1 = \Psi_2 = 0 | \Phi_1 = \Phi_2 = win] \times \Pr[\Phi_1 = \Phi_2 = win | \Phi_0 = win] \\
&\quad + \Pr[\Psi_1 = \Psi_2 = 0 | \Phi_1 = win, \Phi_2 = loss] \times \Pr[\Phi_1 = win, \Phi_2 = loss | \Phi_0 = win] \\
&\quad + \Pr[\Psi_1 = \Psi_2 = 0 | \Phi_1 = loss, \Phi_2 = win] \times \Pr[\Phi_1 = loss, \Phi_2 = win | \Phi_0 = win] \\
&= \Pr[\Psi_1 = 0 | \Phi_1 = win] \times \Pr[\Psi_2 = 0 | \Phi_2 = win] f_1 \\
&\quad + \Pr[\Psi_1 = 0 | \Phi_1 = win] \times \Pr[\Psi_2 = 0 | \Phi_2 = loss] \frac{1}{2}(1-f_1) \\
&\quad + \Pr[\Psi_1 = 0 | \Phi_1 = loss] \times \Pr[\Psi_2 = 0 | \Phi_2 = win] \frac{1}{2}(1-f_1) \\
&= \delta_1^2 f_1 + \delta_1(1-\gamma_1)(1-f_1) \\
&= \left(2\delta_3 - \delta_3^2\right)^2 f_1 + \left(2\delta_3 - \delta_3^2\right)\left(1 - \gamma_3^2 f_2 - \gamma_3(1-\delta_3)(1-f_2)\right)(1-f_1).
\end{aligned}$$

Let $\gamma' = \gamma_0$, $\delta' = \delta_0$, $\gamma = \gamma_3$, $\delta = \delta_3$, the derived equations are

$$\gamma' = 2\left(\gamma^2 f_2 + \gamma(1-\delta)(1-f_2)\right) - \left(\gamma^2 f_2 + \gamma(1-\delta)(1-f_2)\right)^2, \tag{3}$$

$$\delta' = \left(2\delta - \delta^2\right)^2 f_1 + \left(2\delta - \delta^2\right)\left(1 - \gamma^2 f_2 - \gamma(1-\delta)(1-f_2)\right)(1-f_1). \tag{4}$$

The values of γ and δ express the probabilities that an error is made at depth $2k$, while γ' and δ' reflect the errors after the evaluation results are rolled back from depth $2k$ to depth

$2(k-1)$. Therefore, formulas (3) and (4) indicate the evaluation error propagation pattern in a bi-valued binary D-game tree established with dependence factors f_1 and f_2.

5.2 Benefit Analysis of Minimax Search

In terms of the above defined probabilities $\gamma, \delta, \gamma', \delta'$, the problem of whether minimax search helps choose a correct move can be described as whether the following inequalities hold:

$$\gamma' < \gamma, \quad \delta' < \delta. \tag{5}$$

In fact, an answer of the problem depends on the specific values of γ, δ, f_1 and f_2: for some combinations of these values, the answer is positive, for some others, the answer is negative. To establish (5), we need to find the values for γ, δ, f_1 and f_2 so that

$$2(\gamma^2 f_2 + \gamma(1-\delta)(1-f_2)) - (\gamma^2 f_2 + \gamma(1-\delta)(1-f_2))^2 < \gamma, \tag{6}$$

$$(2\delta - \delta^2)^2 f_1 + (2\delta - \delta^2)(1 - \gamma^2 f_2 - \gamma(1-\delta)(1-f_2))(1-f_1) < \delta. \tag{7}$$

We shall prove a stronger conclusion that for some D-game trees the minimax back-up process reduces uniformly the errors made by evaluation function ψ for the terminal nodes in a search tree. In the next theorem, we assume the initial errors produced by the evaluation function are reasonably small, for example, $\gamma < 0.5$ and $\delta < 0.25$. The preliminary choice of 0.25 for δ is simply for ease of proof, and the threshold for δ should be higher than 0.25.

Theorem 5.1 *Given an integer k and a search tree G whose terminal nodes g are located at depth $2k$, let $\gamma = \Pr[\psi(g) = 1 | \Phi(g) = loss]$, $\delta = \Pr[\psi(g) = 0 | \Phi(g) = win]$ represent the evaluation error probabilities for terminal nodes g and $\gamma^{(i)} = \Pr[\Psi(g_i) = 1 | \Phi(g_i) = loss]$, $\delta^{(i)} = \Pr[\Psi(g_i) = 0 | \Phi(g_i) = win]$ represent the error probabilities for shallower nodes g_i at depth $2i$ with $0 \le i < k$, where $\Psi(g_i)$ is determined by the evaluation function $\psi(g)$. For any $\gamma < 0.5$ and $\delta < 0.25$, there are real numbers $F_1 = F_1(\gamma, \delta)$ and $F_2 = F_2(\gamma, \delta)$ in the interval (0, 1) such that, if the D-game tree has dependence factors $f_1 > F_1$ and $f_2 > F_2$, we have*

$$\gamma^{(i)} < H_1^{k-i}\gamma, \quad \delta^{(i)} < H_2^{k-i}\delta,$$

where the two real numbers $0 < H_1 < 1$, $0 < H_2 < 1$ are determined from γ, δ, f_1, f_2.

Proof:

Let γ' and δ' be defined as (3) and (4), respectively. First, we need to find real numbers H_1 and H_2 such that

$$\frac{\gamma'}{\gamma} < H_1, \quad \frac{\delta'}{\delta} < H_2. \tag{8}$$

By formulas (3) and (4), we need to prove

$$\gamma'/\gamma = (\gamma f_2 + (1-\delta)(1-f_2))(2 - \gamma^2 f_2 - \gamma(1-\delta)(1-f_2)) < 1,$$
$$\delta'/\delta = \delta(2-\delta)^2 f_1 + (2-\delta)(1 - \gamma^2 f_2 - \gamma(1-\delta)(1-f_2))(1-f_1) < 1.$$

Since $1 - \delta < 1$ and $2 - \delta < 2$, the above inequalities are implied by

$$2(\gamma f_2 + (1-f_2)) < 1,$$
$$4\delta f_1 + 2(1-f_1) < 1.$$

Therefore, to satisfy (8), we can choose

$$f_1 > F_1 = \frac{1}{2-4\delta},$$
$$f_2 > F_2 = \frac{1}{2(1-\gamma)},$$
$$H_1 = 2(\gamma f_2 + (1-f_2)),$$
$$H_2 = 4\delta f_1 + 2(1-f_1).$$

By simple induction, we can prove that when $f_1 > F_1$ and $f_2 > F_2$, we have $\gamma^{(i)}/\gamma^{(i+1)} < H_1, \delta^{(i)}/\delta^{(i+1)} < H_2$ for $0 \le i < k$, where $\gamma^{(k)} = \gamma$, $\delta^{(k)} = \delta$. This completes the proof. □

The above analysis implies that if the merit values of related nodes in the D-game trees are strongly correlated (f_1 and f_2 are relatively large), and the evaluation error measures (γ and δ) are reasonably small, the probability of a backed-up wrong utility value reduces at a geometrical rate. For these values of γ, δ, f_1 and f_2, pathology does not appear in the minimax search. This benefit gained from minimax search is illustrated in Fig. 2, where three series of points, corresponding to dependence factors $f_1 = f_2 = 0.7$, $f_1 = f_2 = 0.8$ and $f_1 = f_2 = 0.9$, respectively, are drawn on the γ-δ-plane. The common starting point (0.3, 0.2) of the series represents the initial evaluation error probabilities, $\gamma = 0.3$ and $\delta = 0.2$, at depth $2k$ in a D-game tree, and the other points, computed with formulas (3) and (4), represent the backed-up errors at depths $2(k-1), 2(k-2)$, etc. in the D-game tree. Note that the error almost disappears at depth $2(k-j)$ for large j. Therefore, if the search tree G is high enough, the static evaluation error incurred at the search front will eventually be *eliminated* by the minimax back-up process.

We can compare our result with that of Pearl [12]. For uniform trees with random, independent terminal node values, if one assumes that the accuracy of the static evaluation function ψ does not improve with increasing depth (i.e., no visibility improvement), then minimax search pathology is inevitable [12]. That conclusion is directly contrary to the result of Theorem 5.1, which says that under some reasonable assumptions about the accuracy of the static evaluation function, *the minimax process can eliminate the effect of the evaluation errors for game trees which are set up with large dependence factors.*

The next theorem describes how to determine the values for dependence factors which make one-step deeper minimax search less reliable.

Theorem 5.2 *Given an integer k and a search tree G, where terminal nodes g are located at depth $2k$, let $\gamma = \Pr[\psi(g) = 1 | \Phi(g) = loss]$, $\delta = \Pr[\psi(g) = 0 | \Phi(g) = win]$ represent the error probabilities of the evaluation function $\psi(g)$, and $\gamma' = \Pr[\Psi(g') = 1 | \Phi(g') = loss]$, $\delta' = \Pr[\Psi(g') = 0 | \Phi(g') = win]$, the error probabilities for the nodes g' at depth $2(k-1)$, where $\Psi(g')$ is determined by the minimax back-up from value $\psi(g)$. For any $\gamma < 0.3$ and $\delta < 0.25$, there are real numbers $F_1 = F_1(\gamma, \delta)$ and $F_2 = F_2(\gamma, \delta)$ such that if the D-game tree is established with dependence factors $f_1 < F_1$ and $f_2 < F_2$, we have*

$$\gamma' > \gamma, \quad \delta' > \delta.$$

Proof:

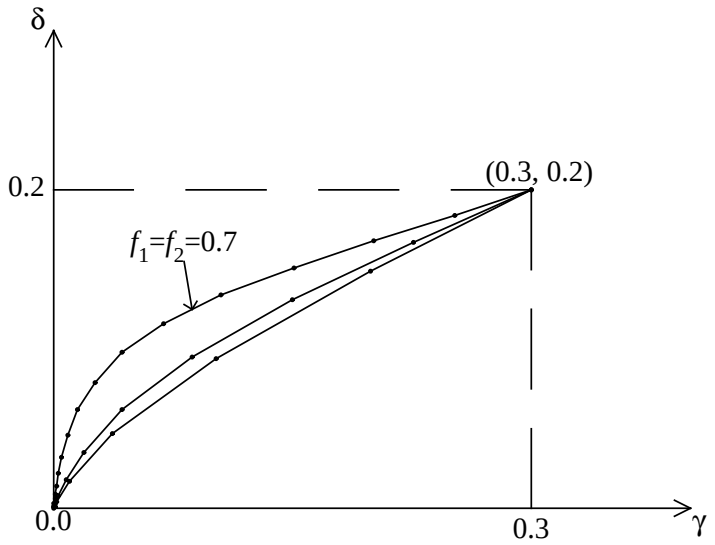

Figure 2: Declining error probabilities with minimax back-up.

Note that for given dependence factors f_1 and f_2, the relationship between γ', δ' and γ, δ are determined by formulas (3) and (4). First, we show that we can find an upper bound F_2 such that for any $f_2 < F_2$, we have $\gamma' > \gamma$, which is

$$2(\gamma^2 f_2 + \gamma(1-\delta)(1-f_2)) - (\gamma^2 f_2 + \gamma(1-\delta)(1-f_2))^2 > \gamma.$$

In fact, when $\gamma < 0.3$ and $\delta < 0.25$, we have

$$\begin{aligned} & 2\gamma^2 f_2 - (\gamma^2 f_2)^2 - 2\gamma^2 f_2 \gamma(1-\delta)(1-f_2) \\ & = \gamma^2 f_2 (2 - \gamma^2 f_2 - 2\gamma(1-\delta)(1-f_2)) \\ & > \gamma^2 f_2 (2 - 0.3^2 - 2 \times 0.3) > 0. \end{aligned}$$

Therefore, by formula (3), we have

$$\begin{aligned} \gamma' & > 2\gamma(1-\delta)(1-f_2) - \gamma^2(1-\delta)^2(1-f_2)^2 \\ & > \gamma\left(2(1-\delta)(1-f_2) - \gamma(1-\delta)^2(1-f_2)^2\right). \end{aligned}$$

Hence, to satisfy $\gamma' > \gamma$, we can choose f_2 so that

$$2(1-\delta)(1-f_2) - \gamma(1-\delta)^2(1-f_2)^2 > 1.$$

In fact, the inequality is satisfied when

$$f_2 < F_2 = 1 - \frac{1-\sqrt{1-\gamma}}{\gamma(1-\delta)}.$$

By deleting the small positive term $\epsilon = (2\delta - \delta^2)^2 f_1$ from the right side of formula (4), we can see that the inequality

$$\delta < (2\delta - \delta^2)(1 - \gamma^2 - \gamma(1-\delta))(1-f_1)$$

implies $\delta' > \delta$. In fact, we can take any

$$f_1 < F_1 = 1 - \frac{1}{(2-\delta)(1-\gamma^2-\gamma(1-\delta))} \tag{9}$$

to satisfy the above inequality.

Note that when $\gamma < 0.3$ and $\delta < 0.25$, we have $0 < F_1 < 1$ and $0 < F_2 < 1$. □

This theorem implies that if the probabilities for an evaluation function ψ to make a wrong estimate of the merit values of a node g' at depth $2(k-1)$ and a node g at depth $2k$ are the same, one-step deeper search from g' to g may degrade the possibility of making a correct decision for the D-games which are established by dependence factors $f_1 < F_1$ and $f_2 < F_2$. In Fig. 3, there are three series of points, which are computed with formulas (3) and (4) by setting dependence factors $f_1 = f_2 = 0.1$, $f_1 = f_2 = 0.2$ and $f_1 = f_2 = 0.3$, respectively. The common starting point ($\gamma = 0.3$, $\delta = 0.2$) represents the error probabilities for terminal node evaluation, and the following points correspond to probabilities that the minimax backed-up values $\Psi(g^{(i)})$ are different from the merit values $\Phi(g^{(i)})$ for shallower nodes $g^{(i)}$ at depth $2i$. It is interesting to note that along with the minimax back-up process, the series migrate towards the point (0, 1), a similar phenomenon was observed by Pearl [12] for those game trees that are set up by independently assigning merit values to terminal nodes.

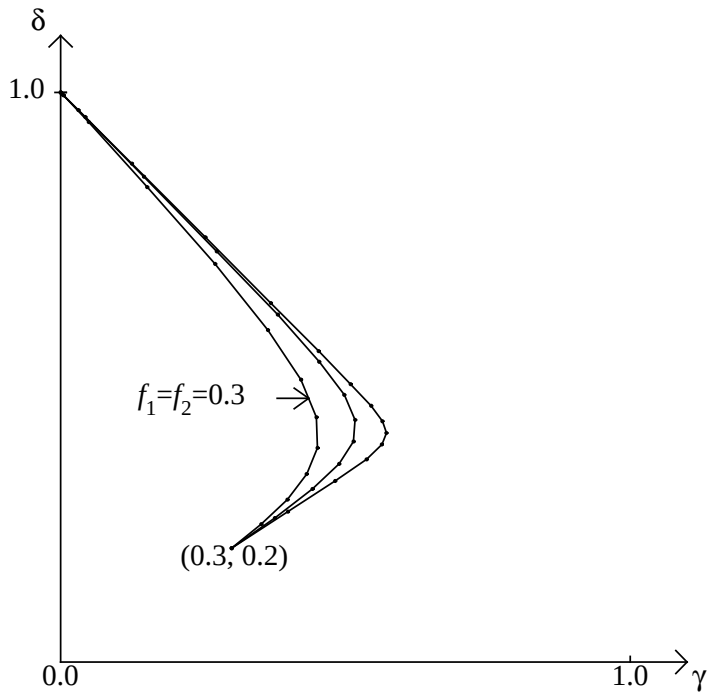

Figure 3: Degradation of minimax with deeper searches.

We have only discussed the back-up of evaluation results from Max nodes to Max nodes in D-games. Similar conclusions can be drawn for the minimax process from Min nodes to Min nodes. The interested reader can write formulas equivalent to (3) and (4) for the Min nodes to Min nodes minimax back-up and analyze them to obtain similar results.

In summary, by assuming predetermined error probabilities for position evaluation function ψ, we have shown that both pathology and non-pathology can appear in the minimax process. The answer to the question of whether there are benefits in minimax searching-deeper depends on the node value dependence of a D-game tree. For some reasonably accurate evaluation functions, if the D-game tree has a strong node value dependence, the evaluation errors made at the search front will be dramatically reduced along with the minimax back-up process. On the other hand, if the node value dependence is weak, searching-deeper with the minimax process cannot produce a more reliable decision than the static evaluation.

6 Mathematical Calculation for Decision Making

6.1 The Probability of Making a Correct Decision

An evaluation function ψ of Nau [8] will be applied to D-game trees to study minimax game-tree search pathology. The value $\psi(g)$ for a position g is defined as the number of 1-squares contained in the corresponding board portion. Suppose node g has a height of h in a D-game tree. The *depth d utility value* $\Psi^{(d)}(g)$ of g for integer $0 \le d \le h$ is defined by

$$\Psi^{(d)}(g) = \begin{cases} \psi(g) & \text{if } d = 0, \\ \min(\Psi^{(d-1)}(g_1), \Psi^{(d-1)}(g_2)) & \text{if } d > 0 \text{ and } g \text{ is a Min node}, \\ \max(\Psi^{(d-1)}(g_1), \Psi^{(d-1)}(g_2)) & \text{if } d > 0 \text{ and } g \text{ is a Max node}, \end{cases}$$

where g_1 and g_2 are the children of g in the game tree. (The superscripts will be omitted when they are clear from the context.) Choosing a move at g by a depth d search means choosing the child of g that has the best depth $d-1$ utility value. By the "best" value we mean the highest value if Max is to move at node g, or the least value if Min is to move. If both children receive the same value, the player must choose one of them randomly. It is obvious that the probability of making a correct decision depends on the accuracy of the depth $d-1$ utility values of g_1 and g_2. In the following, we discuss how to compute the probability of making a correct decision at such a node g in a D-game tree.

A node g has a utility value $\psi(g) = i$ if and only if $\psi(g_1) = j$ and $\psi(g_2) = i - j$ for some integer $0 \le j \le i$. If node g is a Max node, we have

$$\begin{aligned}
&\Pr[\psi(g) = i | h(g) = h, win(g)] \\
&\quad = \sum_{j=0}^{i} \Pr[\psi(g_1) = j, \psi(g_2) = i - j | h(g) = h, win(g)] \\
&\quad = \sum_{j=0}^{i} (\Pr[\psi(g_1) = j, \psi(g_2) = i - j | h(g_1) = h(g_2) = h - 1, win(g_1), win(g_2)] \\
&\qquad\qquad \times \Pr[win(g_1), win(g_2) | win(g)] \\
&\qquad + \Pr[\psi(g_1) = j, \psi(g_2) = i - j | h(g_1) = h(g_2) = h - 1, win(g_1), loss(g_2)] \\
&\qquad\qquad \times \Pr[win(g_1), loss(g_2) | win(g)] \\
&\qquad + \Pr[\psi(g_1) = j, \psi(g_2) = i - j | h(g_1) = h(g_2) = h - 1, loss(g_1), win(g_2)]
\end{aligned}$$

$$\times \Pr[loss(g_1), win(g_2)|win(g)])$$

$$= \sum_{j=0}^{i} (\Pr[\psi(g_1) = j|h(g_1) = h-1, win(g_1)]\Pr[\psi(g_2) = i-j|h(g_2) = h-1, win(g_2)]f_1$$
$$+\Pr[\psi(g_1) = j|h(g_1) = h-1, win(g_1)]\Pr[\psi(g_2) = i-j|h(g_2) = h-1, loss(g_2)](1-f_1)),$$

and

$$\begin{aligned}
&\Pr[\psi(g) = i|h(g) = h, loss(g)] \\
&\quad = \sum_{j=0}^{i} \Pr[\psi(g_1) = j, \psi(g_2) = i-j|h(g) = h, loss(g)] \\
&\quad = \sum_{j=0}^{i} \Pr[\psi(g_1) = j, \psi(g_2) = i-j|h(g_1) = h(g_2) = h-1, loss(g_1), loss(g_2)] \\
&\quad = \sum_{j=0}^{i} (\Pr[\psi(g_1) = j|h(g_1) = h-1, loss(g_1)]\Pr[\psi(g_2) = i-j|h(g_2) = h-1, loss(g_2)]).
\end{aligned}$$

When g is a Min node, we can obtain formulas for the utility value distribution by applying substitutions of $win/loss$ and f_1/f_2 in the above argument. When $h(g) = 0$, g is a terminal node in the game tree, and we can determine the probabilities

$$\begin{aligned}
&\Pr[\psi(g) = 0|h(g) = 0, win(g)] = 0, \quad \Pr[\psi(g) = 1|h(g) = 0, win(g)] = 1, \\
&\Pr[\psi(g) = 0|h(g) = 0, loss(g)] = 1, \quad \Pr[\psi(g) = 1|h(g) = 0, loss(g)] = 0.
\end{aligned}$$

For a D-game tree with given dependence factors f_1 and f_2, the distribution of utility value $\psi(g)$ for node g at different height can be recursively calculated from these initial probabilities.

After evaluating the values $\psi(g)$ for nodes g at the search front, these values should be backed up to the root, where a player can decide a move. First, we can determine

$$\begin{aligned}
&\Pr[\Psi^{(0)}(g) = i|h(g) = h-d, win(g)] = \Pr[\psi(g) = i|h(g) = h-d, win(g)], \\
&\Pr[\Psi^{(0)}(g) = i|h(g) = h-d, loss(g)] = \Pr[\psi(g) = i|h(g) = h-d, loss(g)]
\end{aligned}$$

at the search front. For a shallower Max node g at depth r with $0 < r < d$, since the utility values are backed with minimax process, $\Psi(g) = i$ if and only if both children have a utility value less than or equal to i and at least one child has utility i. Therefore, we have

$$\begin{aligned}
&\Pr[\Psi(g) = i|h(g) = h-r, win(g)] \\
&\quad = \Pr[\Psi(g_1) < i, \Psi(g_2) = i|h(g) = h-r, win(g)] + \Pr[\Psi(g_1) = i, \Psi(g_2) < i|h(g) = h-r, win(g)] \\
&\qquad\qquad +\Pr[\Psi(g_1) = i, \Psi(g_2) = i|h(g) = h-r, win(g)] \\
&\quad = (2\Pr[\Psi(g_1) < i, \Psi(g_2) = i|h(g_1) = h(g_2) = h-r-1, win(g_1), win(g_2)] \\
&\qquad +\Pr[\Psi(g_1) = i, \Psi(g_2) = i|h(g_1) = h(g_2) = h-r-1, win(g_1), win(g_2)]) \\
&\qquad\qquad \times \Pr[win(g_1), win(g_2)|win(g)] \\
&\quad +2(\Pr[\Psi(g_1) < i, \Psi(g_2) = i|h(g_1) = h(g_2) = h-r-1, win(g_1), loss(g_2)] \\
&\qquad +\Pr[\Psi(g_1) = i, \Psi(g_2) < i|h(g_1) = h(g_2) = h-r-1, win(g_1), loss(g_2)]
\end{aligned}$$

$$
\begin{aligned}
&+\Pr[\Psi(g_1)=i,\Psi(g_2)=i|h(g_1)=h(g_2)=h-r-1,win(g_1),loss(g_2)])\\
&\qquad\times\Pr[win(g_1),loss(g_2)|win(g)]\\
=\;&(2\sum_{j=0}^{i-1}\Pr[\Psi(g_1)=j|h(g_1)=h-r-1,win(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,win(g_2)]\\
&+\Pr[\Psi(g_1)=i|h(g_1)=h-r-1,win(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,win(g_2)])f_1\\
&+(\sum_{j=0}^{i-1}\Pr[\Psi(g_1)=j|h(g_1)=h-r-1,win(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)]\\
&+\Pr[\Psi(g_1)=i|h(g_1)=h-r-1,win(g_1)]\\
&\qquad\times\sum_{j=0}^{i-1}\Pr[\Psi(g_2)=j|h(g_2)=h-r-1,loss(g_2)]\\
&+\Pr[\Psi(g_1)=i|h(g_1)=h-r-1,win(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)])(1-f_1),
\end{aligned}
$$

$$
\begin{aligned}
&\Pr[\Psi(g)=i|h(g)=h-r,loss(g)]\\
&\quad=\Pr[\Psi(g_1)<i,\Psi(g_2)=i|h(g)=h-r,loss(g)]+\Pr[\Psi(g_1)=i,\Psi(g_2)<i|h(g)=h-r,loss(g)]\\
&\qquad+\Pr[\Psi(g_1)=i,\Psi(g_2)=i|h(g)=h-r,loss(g)]\\
&\quad=2\sum_{j=0}^{i-1}\Pr[\Psi(g_1)=j|h(g_1)=h-r-1,loss(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)]\\
&\quad+\Pr[\Psi(g_1)=i|h(g_1)=h-r-1,loss(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)].
\end{aligned}
$$

When g is a Min node, the minimax process determines a utility value $\Psi(g)=i$ if and only if both children of g have utility values greater than or equal to i and at least one of them has the utility value i. Note that the largest possible utility value is 2^{h-d}. Therefore, we can calculate the distribution of utility value $\Psi(g)$ for Min nodes g with the following formulas.

$$
\begin{aligned}
&\Pr[\Psi(g)=i|h(g)=h-r,loss(g)]\\
&\quad=(2\sum_{j=i+1}^{2^{h-d}}\Pr[\Psi(g_1)=j|h(g_1)=h-r-1,loss(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)]\\
&\qquad+\Pr[\Psi(g_1)=i|h(g_1)=h-r-1,loss(g_1)]\\
&\qquad\times\Pr[\Psi(g_2)=i|h(g_2)=h-r-1,loss(g_2)])f_2
\end{aligned}
$$

$$
\begin{aligned}
&+\Big(\sum_{j=i+1}^{2^{h-d}} \Pr[\Psi(g_1)=j \mid h(g_1)=h-r-1, loss(g_1)] \\
&\qquad\qquad \times \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, win(g_2)] \\
&\quad +\Pr[\Psi(g_1)=i \mid h(g_1)=h-r-1, loss(g_1)] \\
&\qquad\qquad \times \sum_{j=i+1}^{2^{h-d}} \Pr[\Psi(g_2)=j \mid h(g_2)=h-r-1, win(g_2)] \\
&\quad +\Pr[\Psi(g_1)=i \mid h(g_1)=h-r-1, win(g_1)] \\
&\qquad\qquad \times \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, loss(g_2)]\Big)(1-f_2),
\end{aligned}
$$

and

$$
\begin{aligned}
&\Pr[\Psi(g)=i \mid h(g)=h-r, win(g)] \\
&\quad = 2\sum_{j=i+1}^{2^{h-d}} \Pr[\Psi(g_1)=j \mid h(g_1)=h-r-1, win(g_1)] \\
&\qquad\qquad \times \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, win(g_2)] \\
&\qquad +\Pr[\Psi(g_1)=i \mid h(g_1)=h-r-1, win(g_1)] \\
&\qquad\qquad \times \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, win(g_2)].
\end{aligned}
$$

In the above formulas, we have

$$
\begin{aligned}
&\Pr[\Psi(g_1)=i \mid h(g_1)=h-r-1, win(g_1)] = \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, win(g_2)], \\
&\Pr[\Psi(g_1)=i \mid h(g_1)=h-r-1, loss(g_1)] = \Pr[\Psi(g_2)=i \mid h(g_2)=h-r-1, loss(g_2)].
\end{aligned}
$$

The only difference between the above derived formulas and those used in Section 5 is that the latter back up *binary* evaluation results. Therefore, we can say that the above formulas encompass those used in Section 5 as special cases.

Suppose a player is to choose a move at node g of height h in a D-game tree by searching to depth d with $d \leq h$. If one of g's children (say, g_1) is a *win* node and the other (g_2) is a *loss* node, we can define a correct move to be either moving to g_1 if the player is Max or moving to g_2 if the player is Min. We are only interested in the probability of a correct decision in the case where it makes a difference what move is made, a correct move is not defined if both children are *win* or *loss* nodes [8]. Since player Max will move to the node of the highest utility and player Min to the node of the lowest utility, a correct move will be made if $\Psi^{(d-1)}(g_1) > \Psi^{(d-1)}(g_2)$, and an incorrect decision will be made if $\Psi^{(d-1)}(g_1) < \Psi^{(d-1)}(g_2)$. If $\Psi^{(d-1)}(g_1) = \Psi^{(d-1)}(g_2)$, the player must choose among g_1 and g_2 at random. Therefore, the probability of a correct decision with depth d search at a node g of height h is

$$
\begin{aligned}
&D(d,h) \\
&\quad = \Pr[\Psi^{(d-1)}(g_1) > \Psi^{(d-1)}(g_2)] + \frac{1}{2}\Pr[\Psi^{(d-1)}(g_1) = \Psi^{(d-1)}(g_2)] \\
&\quad = \sum_{i=0}^{2^{h-d}} \left(\Pr[\Psi^{(d-1)}(g_1) \geq i+1, \Psi^{(d-1)}(g_2) = i] + \frac{1}{2}\Pr[\Psi^{(d-1)}(g_1) = \Psi^{(d-1)}(g_2) = i]\right)
\end{aligned}
$$

$$= \sum_{i=0}^{2^{h-d}} \Big(\sum_{t=i+1}^{2^{h-d}} \Pr[\Psi^{(d-1)}(g_1) = t | h(g_1) = h-1, win(g_1)] \times \Pr[\Psi^{(d-1)}(g_2) = i | h(g_2) = h-1, loss(g_2)]$$
$$+ \frac{1}{2} \Pr[\Psi^{(d-1)}(g_1) = i | h(g_1) = h-1, win(g_1)] \times \Pr[\Psi^{(d-1)}(g_2) = i | h(g_2) = h-1, loss(g_2)] \Big).$$

Since the probabilities $\Pr[\Psi^{(d-1)}(g_k) = i | h(g_k) = h-1, win(g_k)]$ and $\Pr[\Psi^{(d-1)}(g_k) = i | h(g_k) = h-1, loss(g_k)]$ for $k = 1, 2$ and $0 \leq i \leq 2^{h-d}$ can be calculated, we can develop a program to calculate $D(d, h)$.

6.2 Numerical Results

The formulas described above can be used to produce data, which are employed here to analyze the relationship between minimax game-tree search pathology and the node value dependence present in D-games. In particular, for some different values of dependence factors f_1 and f_2, we calculate the probabilities $D(d, h)$ of making a correct decision by searching to various depths d in a D-game tree of height h. The probabilities for D-game trees of height ten with dependence factors $f_1 = f_2 = i/10$ for integers $1 \leq i \leq 9$ are listed in Table 1. As Propositions 4.1 and 4.2 prove, when $f_1 = f_2 = 0$ or $f_1 = f_2 = 1$, the evaluation function ψ can determine the correct move by searching to any depth d, the corresponding probabilities $D(d, 10) = 1.000$ are not included here. As Table 1 shows, when $f_1 = f_2 = 0.1$, pathology is present, since searching to depth 2 is worse than to depth 1, i.e., one step search is worse than a static evaluation. As the dependence factors increase to $f_1 = f_2 = 0.6$, the pathology phenomenon disappears. The last row of Table 1 lists for each column in the table a numeral data, called *pathological sum* or PS, which is calculated in the following way: if $D(d-1, 10) > D(d, 10)$ for $1 < d \leq 10$, this "pathological" depth d search contributes the positive difference $D(d-1, 10) - D(d, 10)$ to the sum. Pathology sum is used here as an indicator (or approximate measure) of the degree of pathology. In the case $h = 10$, after rising to the peak at about $f_1 = f_2 = 0.32$, the pathological sums decrease strictly with increasing f_1 and f_2 until $f_1 = f_2 > 0.57$ when pathology totally disappears. Note that the PS value for $f_1 = f_2 = 0.2$ is less than the PS value for $f_1 = f_2 = 0.3$, at the same time, the probabilities $D(d, 10)$ in the second column are less than or equal to the corresponding probabilities in the third column. Therefore, we anticipate the existence of a better measuring method for pathology than the PS values. Even so, it is clear that pathology presents itself in the minimax search of D-game trees for small dependence factors. This conclusion also confirms the results presented in Theorems 5.1 and 5.2. In practice, since a player does not include the obvious uninteresting moves into a game tree, the game trees actually established in the computer have a strong positive correlation. An *uninteresting* move is one which probably leads to a loss for the player who is in a position to make the move. In terms of dependence factors, the corresponding game tree should have large values for f_1 and f_2. This observation helps explain why computer game-playing programs seem to avoid minimax pathology and do better by searching deeper.

Table 1 The Probabilities $D(d, 10)$ of Making Correct Decision for $f_1 = f_2$

d	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$	$f_1 = f_2$
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	0.608	0.652	0.721	0.801	0.876	0.934	0.972	0.992	0.999
2	0.600	0.644	0.712	0.794	0.874	0.937	0.978	0.996	1.000
3	0.598	0.640	0.710	0.796	0.881	0.947	0.984	0.998	1.000
4	0.593	0.635	0.706	0.795	0.884	0.953	0.988	0.999	1.000
5	0.592	0.635	0.708	0.803	0.896	0.963	0.993	1.000	1.000
6	0.571	0.627	0.702	0.802	0.901	0.969	0.995	1.000	1.000
7	0.657	0.643	0.701	0.807	0.914	0.977	0.997	1.000	1.000
8	0.656	0.587	0.595	0.715	0.891	0.979	0.998	1.000	1.000
9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PS	0.038	0.081	0.129	0.101	0.025	0.000	0.000	0.000	0.000

In Table 2, we present some data for the case where the dependence factors are related by $f_1 = i/10$, $f_2 = (10 - i)/10$ for integers $1 \leq i \leq 9$. As described by Proposition 4.1, when $f_1 = 1$ (or $f_2 = 1$), the pathological sum must be zero. In real games, since both players apply the same set of rules, the node value dependence with $f_1 >> f_2$ or vice versa, $f_1 << f_2$, should not occur. In other words, the difference $|f_1 - f_2|$ between the dependence factors for a real game must be relatively small, and the dependence factors should be related by $f_1 \approx f_2$, both of which are relatively large. Therefore, Table 2 should be of only theoretical interests.

Table 2 The Probabilities $D(d, 10)$ of Making Correct Decision for $f_1 + f_2 = 1$

d	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2	f_1 f_2
	0.1 0.9	0.2 0.8	0.3 0.7	0.4 0.6	0.5 0.5	0.6 0.4	0.7 0.3	0.8 0.2	0.9 0.1
1	0.947	0.898	0.878	0.872	0.876	0.887	0.906	0.935	0.976
2	0.968	0.917	0.888	0.876	0.874	0.882	0.900	0.932	0.980
3	0.980	0.924	0.891	0.880	0.881	0.894	0.917	0.953	0.992
4	0.991	0.946	0.907	0.888	0.884	0.893	0.917	0.957	0.996
5	0.997	0.963	0.918	0.898	0.896	0.910	0.938	0.978	0.999
6	0.999	0.979	0.934	0.907	0.901	0.914	0.946	0.989	1.000
7	1.000	0.994	0.957	0.915	0.914	0.926	0.969	0.997	1.000
8	1.000	0.996	0.963	0.879	0.891	0.968	0.995	1.000	1.000
9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PS	0.000	0.000	0.000	0.035	0.025	0.005	0.006	0.003	0.000

In the works of Nau [8] and Pearl [12], a special value w_2, which is the unique solution in the interval (0, 1) of equation $(1 - x)^2 = x$, was used in studying minimax search pathology. That value has a property: if w_2 is the probability that a terminal node in a binary uniform tree of even height has the value *win*, the root node has value *win* with the same probability.

Here, we investigate a similar case. Given a real number p_0 as the probability for the root node to take merit value *win*, by Theorem 3.1, we can determine values for dependence factors f_1 and f_2 so that the nodes at even depths have the same probability p_0 to take merit value *win*. For $p_0 = 0.5$ and for values of f_1 in the range (0, 1), we can use the equation of Theorem 3.1 to determine a value for f_2 (note that $f_2 > f_1$), and use these dependence factors to calculate $D(d, h)$ for various depths d and $h = 10, 12, 13$. The relationship between the pathological sums PS(f_1, f_2) and the dependence factors f_1 (and f_2) are illustrated in Fig. 4. Generally the pathological sums decrease as the dependence factors f_1 and f_2 increase. Some 100 sets of dependence factors were used in the calculation, which for D-game tree of height 10 involved the computation of 1000 probabilities $D(d, 10)$. Of these, only 167 pairs of adjacent probabilities $D(d-1, 10)$, $D(d, 10)$ contributed to the pathological sums. Therefore, we can say that minimax search pathology is a rare event.

Figure 4: Pathology sum (h=10, 12, 13) for $p_0 = 0.5$ at even depth nodes.

On the other hand, data from deeper searches suggests that although the pathological sum itself does not increase significantly, its frequency of occurrence rises for small dependence factors. From Tables 1 and 2, we can observe that the evaluation function $\psi(g)$ for nodes g of height 2 ($d = 8$) are extremely unreliable in measuring the strengths of g. On the basis of our model, we hypothesize that even in games like chess with high dependence factors, for sufficiently deep searches, minimax pathology will exist, but in such a weak form that it will not be distinguishable from normal errors in the evaluation of $\psi(g)$. We also conjecture

that with an evaluation function which improves $\psi(g)$ by recognizing some features of the board configurations, the relationship between the minimax game tree search pathology and dependence factors will be more prominent than we have observed here.

7 Concluding Remarks

To reveal the source of minimax game-tree search pathology in board-splitting games, we present a new method for assigning a status, which is either *win* or *loss*, to board squares. This method establishes D-game trees in a top-down manner to introduce a sort of node-value dependence among nodes and their children. By assuming a predetermined error probability for a bi-valued evaluation function and examining the error probabilities which are associated with different levels by the minimax backing-up process, our study shows a strong relationship between the node value dependence and the performance of minimax process with respect to eliminating the effects of evaluation errors. When the game tree is established with large values of dependence factors, the errors produced at the search front will eventually be eliminated by the minimax back-up; but when the dependence factors are small, a one-step deeper search will not generate a more reliable result. This relationship is also confirmed by another method, which calculates the probabilities of choosing a correct move with a real evaluation function of Nau [8] by searching to different depths in the game trees. By varying the dependence factors of these game trees, we can observe the performance improvement of minimax search when the dependence factors increase.

Our results can be used to answer questions about where minimax game tree search pathology appears and how it happens. When the node values are weakly related, the minimax back-up process will degrade the reliability of static evaluation made at the search front; when the node values are strongly related (with large dependence factors f_1 and f_2), the evaluation errors will be eliminated along with the minimax back-up process. This observation can be used to explain why previous studies could not show the positive effect of minimax search. This is simply because the models used there fail to account for the apparent dependence between the related positions of common games like chess.

Even though the results help computer-game players understand the mysterious phenomenon of game-tree search pathology, we still need to study the relationship between the dependence factors defined here and the node value dependence present in common games. An interesting research topic would be whether we can determine the values of dependence factors for common games with some statistical analysis. In this way, computer-game players can develop guidelines for making the decisions of whether or not to conduct a deeper search.

Acknowledgement

This work was partly supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant OPG 7902.

References

[1] D. F. Beal. An analysis of minimax. In M. R. B. Clarke, editor, *Advances in Computer*

Chess 2, pages 103–109. Pergamon Press, 1980.

[2] D. F. Beal. Benefits of minimax search. In M. R. B. Clarke, editor, *Advances in Computer Chess 3*, pages 17–24. Pergamon Press, 1982.

[3] H. Berliner. The B* tree search algorithm: Best-first proof procedure. *Artificial Intelligence*, 12:23–40, 1979.

[4] D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326, 1975.

[5] L. Li. *Probabilistic Analysis of Search.* PhD thesis, Department of Computing Science, University of Alberta, 1989. Also available as Technical Report TR90-5, Department of Computing Science, University of Alberta.

[6] T. A. Marsland. Computer chess methods. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Vol 1*, pages 159–171. Wiley & Sons, Inc., 1987.

[7] T. A. Marsland, A. Reinefeld, and J. Schaeffer. Low overhead alternatives to sss*. *Artificial Intelligence*, 31:185–199, 1987.

[8] D. S. Nau. An investigation of the causes of pathology in games. *Artificial Intelligence*, 19:257–278, 1982.

[9] D. S. Nau. The last player theorem. *Artificial Intelligence*, 18:53–65, 1982.

[10] M. M. Newborn. The efficiency of the alpha-beta search trees with branch-dependent terminal node scores. *Artificial Intelligence*, 8:137–153, 1977.

[11] M. M. Newborn. Computer chess: Recent progress and future expectation. In J. Moneta, editor, *Information Technology*, pages 189–192. North-Holland, 1978.

[12] J. Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 20:427–453, 1983.

[13] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Company, 1984.

[14] K. Thompson. Computer chess strength. In M. R. B. Clarke, editor, *Advances in Computer Chess 3*, pages 55–56. Pergamon Press, 1982.