Alcom-FT Technical Report Series
ALCOMFT-TR-02-108

The Secret of Selective Game Tree Search, when Using Random-Error Evaluations

⋆ ⋆⋆

U. Lorenz, B. Monien

Department of Mathematics and Computer Science
Paderborn University
Germany

Abstract. Game tree search deals with the problems that arise, when computers play two-person-zero-sum-games such as chess, checkers, othello etc. The greatest success of game tree search so far, was the victory of the chess machine 'Deep Blue' vs. G. Kasparov[14], the best human chess player in the world at that time. In spite of the enormous popularity of computer chess and in spite of the successes of game tree search in game playing programs, we do not know much about a useful theoretical background that could explain the usefulness of (selective) search in adversary games.
We introduce a combinatorial model, which allows us to model errors of a heuristic evaluation function, with the help of coin tosses. As a result, we can show that searching in a game tree will be 'useful' if, and only if, there are at least two leaf-disjoint strategies which prove the root value. In addition, we show that the number of leaf-disjoint strategies, contained in a game tree, determines the order of the quality of a heuristic minimax value. The model is integrated into the context of average-case analyses.

1 Introduction

When a game tree is so large that it is not possible to find a correct move, there are two standard approaches for computers to play games. In the first approach, the algorithms work in two phases. Initially, a subtree of the game tree is chosen for examination. This subtree may be a full width, fixed depth tree, or any other subtree rooted at the starting position. Thereafter, a search algorithm heuristically assigns evaluations to the leaves and propagates these numbers up the tree according to the minimax principle. Usually the chosen subtree is examined by the help of the $\alpha\beta$-algorithm [5] or one of its variants. As far as the error frequency is concerned, it does not make any difference whether the envelope is examined by the $\alpha\beta$-algorithm or by a pure minimax algorithm.

⋆ Supported by the German Science Foundation (DFG) project *Efficient Algorithms For Discrete Problems And Their Applications*
⋆⋆ Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

In both cases, the result is the same. Only the effort to get the result differs drastically.

The heuristic minimax value of such a static procedure already leads to high quality approximations of the root value. However, there are several improvements that form the selected subtree more individually. These lead us to a second class of algorithms which work in only one phase and which form the tree shape dynamically at execution time. Some of the techniques are domain independent such as Nullmoves [2], Fail High Reductions [3], or 'Conspiracy Number Search'. Conspiracy Number Search was introduced by D. McAllister [9]. J. Schaeffer [13] interpreted the idea and developed a search algorithm that behaves well on tactical chess positions. We presented an efficient flexible search algorithm which can deal with Conspiracy Numbers[6][8]. We implemented the algorithm in the chess program 'P.ConNerS', which was the first one that could win an official FIDE Grandmaster Tournament [7]. The success was widely recognized in the chess community.

In spite of the overwhelming successes of game tree search in practice, we do not know much about a theoretical background, which could explain how errors of the heuristic evaluation process are filtered out by the minimax-procedure, resp. why the flexible, adaptive algorithms work better in practice than static approaches do.

Pearl [12] examined game trees, assuming that WIN and LOSS are randomly, and indepenently from each other, assigned to the terminal positions. As a result the outcome at the root does only depend on the fraction of WIN-leaves. G. Schrüfer proposed a model in which he could explain the helpfulness of depth d search trees. Furthermore, he was able to distinguish between games where depth d-uniform searching is helpful and ones where it is detrimental [1]. The game trees in his model are random events themselves. A similar model, refined by the concept of quiescence, was used by Kaindl and Scheucher [4].

The results of this paper are applicable for all concrete (i.e. not randomly chosen, not vague) game trees, and they are the most general ones that we know of. We use a simple basic model which has been inspected before with non-encouraging results only [10, 11]:

Model: An arbitrary but fixed finite game tree G is given. Each of its nodes has a value 1 or 0. These values follow the minimax principle, and we call them 'real' (=true) values. We perform coin tosses at the leaves of G. Thus, each leaf gets a second value, which we call a 'heuristic' one. With a probability of p a leaf gets the same heuristic value as its real one. With a probability of $1-p$ it gets the complementary value. We assume p to be equal for all leaves. The heuristic value of G's root is the minimax value of the heuristic leaf values. The question of interest is the following: With which probability will the heuristic minimax value and the real value of G's root be equal to each other?

Insights: We show that the 'number of leave-disjoint strategies' is a key-term of error analyses in game trees.

The number of leaf-disjoint strategies (LDSs), which all prove the root value, determines to what degree the root value is approximated by a heuristic minimax value. If there are not at least two such LDSs, a minimax value will not lead to better approximations than a direct heuristic evaluation (with respect to a reasonable definition of 'better').

Note that our results are applicable for all game trees. They are neither limited to fixed-depth trees nor to trees with clustering leaf-values.

Example: In order to prove that the value of G's root (see Fig. 1) to be 0, we do only need to inspect the subtree $S_1 := \{v1, v2, v3, v5, v8, v9, v10, v12, v14\}$. We call this subtree (which proves the upper bound to be 0) a *strategy* which is contained in the game tree G. The nodes of $S_2 := \{v1, v2, v3, v5, v8, v9, v10, v11, v14\}$ build a strategy for the root value 0, as well. In contrast to $S_3 := \{v1, v2, v3, v5, v7, v9, v10, v11, v13\}$, however, S_2 is not *leaf-disjoint* to S_1.

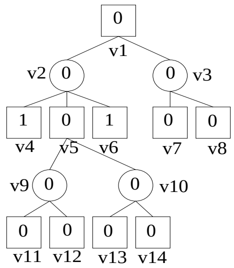

Fig. 1. An arbitrary Game Tree G

The paper is organized as follows: In Chapter 2 we introduce some general definitions, concerning game tree search. In Chapter 3 we analyze the model and discuss the results. We also link our results to Conspiracy Number Search. Chapter 4 contains the proofs, which profit from a new elegant technique of error analysis.

2 Definitions and Notations

2.1 General Game Tree Definitions

Definition 1. *In this paper $G = (T, h)$ will be a* game tree, *if $T = (V, E)$ is a tree and $h : L(G) \to \{0, 1\}$ is a function, $L(G)$ being the set of leaves of T.*
Remark: We identify the nodes of a game tree G with positions of the underlying game and the edges of T with moves from one position to the next.

Definition 2. *There are two players MIN and MAX. MAX has to move on even levels of the game tree, MIN on the other levels. This defines a player function $p : V \to \{MAX, MIN\}$.*

Definition 3. *Let $G = (T, h)$ be a game tree and $v \in V$ a node of T. The function minimax : $V \to \{0, 1\}$ is inductively defined by*

$$minimax(v) := \begin{cases} h(v) & \text{if } v \in L(G) \\ \max\{minimax(v') \mid (v, v') \in E\} & \text{if } p(v) = MAX \\ \min\{minimax(v') \mid (v, v') \in E\} & \text{if } p(v) = MIN \end{cases}$$

We call minimax(v) the *minimax value* of v. The minimax value of the root of G is denoted by minimax(G).

Definition 4. *Let G be a game tree, $s \in \{MIN, MAX\}$, with root $v \in V$. Let $\Gamma(u)$ denote the set of u's successors. A* strategy for player *s, $S_s = (V_s, E_s)$, is a subtree of G, inductively defined by*

- *$v \in V_s$*
- *If $u \in V_s$ is an internal node of T with $p(u) = s$ then there is exactly one $u' \in \Gamma(u)$ with $u' \in V_s$ and $(u, u') \in E_s$.*
- *If $u \in V_s$ is an internal node of T with $p(u) = \overline{s}$, then $\Gamma(u) \subset V_s$, and for all $u' \in \Gamma(u)$ is $(u, u') \in E_s$.*

As a consequence of that, the minimax-evaluation of a strategy S either provides us with a lower (in case S is a MAX-strategy) or with an upper (in case S is a MIN-strategy) bound of the root value of G. When a strategy S gives us 1 as a lower bound, resp. 0 as an upper bound, we say that S 'proves' the root value.

Definition 5. *Two strategies will be called* leaf-disjoint *if they have no leaf in common.*

Definition 6. *The* depth *of a game tree is the maximum distance between the root of G and its leaves.*

3 Error Analysis

When we agree on splitting an error analysis into a certain number of errors and the errors' positions, one can analyze how errors propagate in game trees from different points of view: Firstly, the errors may be positioned by a friend of ours (some kind of best-case scenario). Let S be the set of leaves of a smallest strategy, that proves the real value of the root. Let n be the number of leaves of G. Then we can make at least $n - |S|$ many errors at the leaves without the error reaching the root of G.

Secondly, the positioner of the errors may be our enemy (some kind of worst-case scenario). If x is the number of leaf-disjoint strategies in G, that all prove the root value, we will be able to make at least $x-1$ errors without the error reaching the root. When there are exactly x faulty evaluations, they can be positioned in such a way that the root value gets faulty, too.

Thirdly, and much more interesting is the question of what happens, when we presume a certain error rate (in relation to n) and when these errors are arbitrarily or average-like positioned. For simplification, let us assume that errors occur randomly.

Let us perform coin tosses on all leaves of G. With a probability of p each leaf gets the same heuristic value as its real value. With a probability of $1-p$ it gets the complementary value. We assume p to be equal for all leaves. For the inner nodes of G we build the minimax value that is based on the possibly incorrect heuristic values. The question of interest is, with which probability the heuristic value and the real value at the root of G will be the same.

Because of the tree structure the non-error probabilities of leaf-disjoint subtrees are independent of each other. The probability of making a correct evaluation at the root of a game tree G is a polynomial Q_G in the non-error probability p of a direct evaluation of a node. We will call $Q_G(p)$ the *polynomial of quality for* G.

Let v be a MAX-node. Then, in principle and without any loss of generality, we come to one of the following two situations:

1. v has got the real value 1:

   Let $v_1, \ldots, v_b$ be the successors of the node v, the root of the example tree (figure 2). Let $g_1(p), \ldots, g_b(p)$ be the probabilities for the event that the heuristic values $h_i, i \in \{1 \ldots b\}$ are equal to the real values $w_1 \ldots w_b$ of the nodes $v_1 \ldots v_b$. Now, we can compute the probability $Q_{G1}(p)$ that the heuristic minimax value of v is equal to the real value of v:

   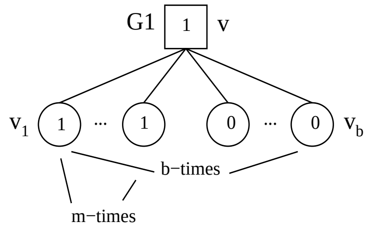

   Fig. 2. MAX-node v with real value 1

   $Q_{G1}(p)$ is equal to the probability that not all successors of v get the heuristic value 0:

   $$Q_{G1}(p) = 1 - (\prod_{i=1}^{m}(1 - g_i(p)) \cdot \prod_{i=m+1}^{b} g_i(p))$$

2. v has got the real value 0 (figure 3):

   Let $v_1, \ldots, v_b$ be the successors of the node v again, v being the root of the example game tree. If we know the probabilities $g_1(p), \ldots, g_b(p)$ that a heuristic value $h_i, i \in \{1 \ldots b\}$ is equal to the real value $w_i = 0$ of the node v_i, we will compute the probability $Q_{G2}(p)$, that the heuristic value h corresponds to the real value of the node v with a probability of $Q_{G2}(p) = \prod_{i=1}^{b} g_i(p)$.

   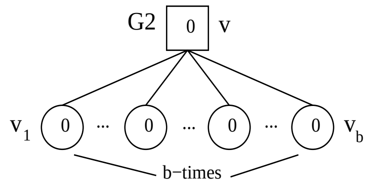

   Fig. 3. MAX-node with real value 0

Notion 1. *Let $Q'_G(p)$ resp. $Q_G^{(1)}(p)$ denote the first derivative of $Q_G(p)$. We will call a game tree G* useful, *if $Q'_G(1) = 0$. (Because we will see that $Q'_G(1) = 0$ or $Q'_G(1) \geq 1$ holds for all polynomials of quality, this criterion will give us a clear distinction between 'good' and 'bad' game trees, if the evaluator is 'good enough'.)*

Let $Q_G(p)$ be a polynomial of quality.

Lemma 1. *$Q'_G(1) = 0$ or $Q'_G(1) \geq 1$ for all game trees.*

Lemma 2. *If $Q'_G(1) > 0$, $Q'_G(1)$ will express the number of leaves, which are able to change the root value of G by a single flip of a leaf value.*

Theorem 1. *$Q'_G(1) = 0$ if, and only if, the game tree G contains at least 2 leaf-disjoint strategies, both proving the real value of the root of G.*

Lemma 1 reveals that there are two contrary types of game trees. Game trees which are clearly useful and such ones that are not useful at all. Theorem 1 specifies the useful game trees to be those that contain several leaf-disjoint strategies. Game trees with no two leaf-disjoint strategies have a damaging error behaviour.

Theorem 2. *$Q_G^{(n)}(1) = Q_G^{(n-1)}(1) = \ldots = Q_G^{(1)}(1) = 0 \Leftrightarrow$ there are $n+1$ leaf-disjoint strategies below v that prove the real value of v.*

Let G be an arbitrary game tree, Q_G its polynomial of quality. Taylor's theorem, $f(p) = f(1) + f'(1)(p-1) + \ldots + \frac{f^{(n)}(1)}{n!}(p-1)^n + R_{n+1}(p)$, leads us to $|Q_G(p) - Q_G(1)| = O((1-p)^{n+1})$ if, and only if there are $n+1$-many leaf-disjoint strategies in the game tree G. Near the point $x = 1$ the number of leaf-disjoint strategies, contained in G, determines the order of the quality of a search.

Interpretation: Figure 4 presents us with three different courses of $Q(p)$ and the identity function. In the case of $Q(p) \leq p$, we do not see any motivation for performing any search at all. It is obviously more effective to directly evaluate the root. Thus, we see that the search tree is only useful, (in an intuitive sense, here) when $Q(p) > p$, since only then is the error rate of the computed minimax value of the root smaller than the error rate of a direct evaluation of the root. Thus, if $Q_G(p)$ and $Q_H(p)$ are polynomials of quality and $Q'_G(1) = 0$ and $Q'_H(1) \geq 1$, there will be an $\epsilon > 0$ such that for all $p \in [1-\epsilon, 1]$ it is $Q_G(p) > Q_H(p)$, and thus G is more useful than H.

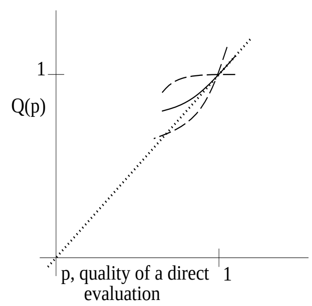

Fig. 4. Possible courses of $Q(p)$

Example: Let us re-inspect the game tree G in Figure 1. Let H arise from G by flipping the value of node v11 from 0 to 1. As a consequence, there are no two leaf-disjoint strategies in H for the root value.

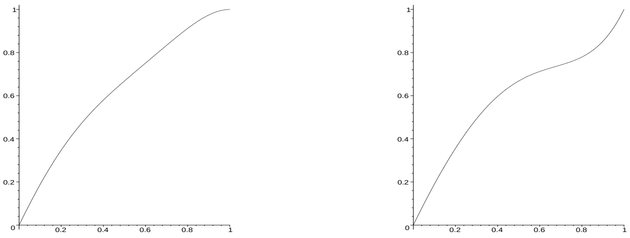

Fig. 5. $Q_{\epsilon(G)}(p)$ (on the left), $Q_{\epsilon(H)}(p)$ (on the right)

Figure 5 shows, how the non-error probabilities at the roots grow with increasing p at the leaves.

We derive the following conjectures:

- It is not the number of strategies, contained in G, that is the key to the quality of a search, but the number of leaf-disjoint ones. There may be thousands of different strategies, if there are not at least two leaf-disjoint ones, the game tree is not helpful.
- The branching factor (i.e. the degree of the inner nodes) of a game tree seems to have only little impact on the approximability of decisions taken in game trees. Indeed, it vanishes in our analysis. This gives us reason to hope that game tree search methods can also be used in games with quite a high branching factor, which has sometimes be doubted. Especially the branching factor argument has led to the general opinion that Go is not suited for game tree search.
- For chess, there are many heuristic-based ideas how to form a game tree. What is the effect of these heuristics such as Fail High Reductions or Singular Extensions? We guess that these heuristics do nothing but increase the number of leaf-disjoint strategies in game trees, concerning their real values. Conspiracy Number Search (CNS) can be interpreted as a meta-heuristic which tries to increase the number of LDSs, too. Indeed, with CNS you try to examine game trees which contain several LDSs, but seen from the side of the heuristic values.

Integration of the model: An objection to the model, which we use here, might be the fact that in practice (e.g. in chess programs) the evaluation-errors are not independent from each other, or that they do not occur randomly at all. Anyway, the used model is definitely a nice and elegant tool for analyses and it is not (as it is no model in the world) necessarily a one-to-one description of the reality. We can easily set the model in the context of average case analyses. Let us call an average-case error analysis, in which the number of occurring errors is weighted according to a binomial distribution, a *relaxed average-case analysis.* The term expresses that we are interested in the problem of how errors propagate in minimax trees when we presume an error rate of 'approximately x percent'.

Let G be an arbitrary game tree with n leaves, and let s be the string of 0s and 1s, consisting of the real values of the leaves of G in a left to right order. Furthermore, let $s' \in \{0,1\}^n$ be a further string of length n. If the value $s(i)$ and $s(i')$ have the same value at a position i, we say 'the i-th leaf of s' has been correctly classified'. When we put all strings of length n into clusters $C_1 \ldots C_n$, with each cluster containing those strings which have i correctly classified items, there exists a certain number c_i for each cluster of strings, which tells us, in how many cases the root is correctly classified by a minimax-evaluation of G.
Since $Q_G(p)$ is indeed equal to $\sum_{i=0}^{n}$ Prob(the root value is correctly classified by the minimax-evaluation of G | there are exactly i correctly classified leaves) $\cdot$ Prob(exactly i leaves are correctly classified) $= \sum_{i=0}^{n} c_i/|C_i| \cdot \binom{n}{i} \cdot x^i \cdot (1-x)^{n-i}$, with x being equal to the probability p of our combinatorial model, the proposed model is nothing but a nice mathematical vehicle in which we perform a relaxed average-case error analysis.

4 Proofs

Let $Q_G(p)$ be a polynomial of quality.

Lemma 1: $Q'_G(1) = 0$ or $Q'_G(1) \geq 1$ for all game trees.
Theorem 1: $Q'_G(1) = 0$ if, and only if, the game tree G contains at least 2 leaf-disjoint strategies, both proving the real value of the root of G.

Without loss of generality we can recursively compose any game tree G with the help of the following three 'different' classes of depth-1 trees. They are different as far as their polynomials of quality are concerned.

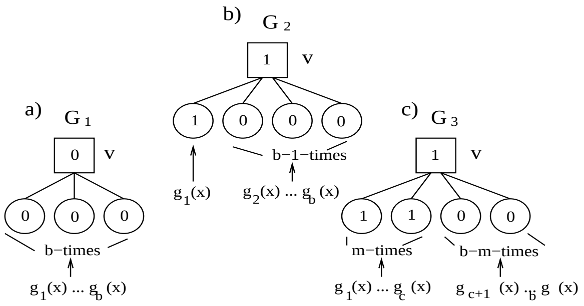

Fig. 6. Three quality-different (t=1)-trees

Figure 6a) shows a game tree rooted by a maxnode with value 0. Because of the minimax principle all successors are 0 as well. Figure 6b) is a game tree rooted by a maxnode with value 1. Exactly one successor of the root has a value of 1.

In figure 6c) there is more than one successor with a value of 1. By the help of section 4 we get polynomials of quality Q_1, Q_2 and Q_3 for the three types of game trees G_1, G_2 and G_3:

$Q_1(x) = g_1(x) \cdots g_b(x)$
$Q_2(x) = 1 - (1 - g_1(x)) \cdot g_2(x) \cdots g_b(x)$
$Q_3(x) = 1 - (1 - g_1(x)) \cdots (1 - g_m(x)) \cdot g_{m+1} \cdots g_b(x)$

The derivatives of these three polynomials are easily computable:
$Q_1'(x) = \sum_{i=1}^{b} g_1(x) \cdots g_i'(x) \cdots g_b(x)$
$Q_2'(x) = -(1 - g_1(x))' \cdot g_2(x) \cdots g_b(x) + \sum_{i=2}^{b}(1 - g_1(x)) \cdot g_2(x) \cdots g_i'(x) \cdots g_b(x)$
$Q_3'(x) = -\sum_{i=1}^{c}(1 - g_1(x)) \cdots (1 - g_i(x)) \cdots (1 - g_c(x)) \cdot g_{c+1}(x) \cdots g_b(x) - \sum_{i=c+1}^{b}(1 - g_1(x)) \cdots (1 - g_c(x)) \cdot g_{c+1}(x) \cdots g_i'(x) \cdots g_b(x)$

Because $g_i(1) = 1$ for all i, the following holds:
$Q_1'(1) = \underline{g_1'(1) + \ldots + g_b'(1)}$
$Q_2'(1) = \underline{g'(1)}$
$Q_3'(1) = \underline{0}$

Any given game tree can be regarded as a recursive construction of $t = 1$-trees, as introduced above. By the fact that the polynomial of quality of any leaf is the identity function and by a simple implicit induction over the depth of game trees we see that Lemma 1, as well as Theorem 1 are correct. Example: Let Theorem 1 be proven for all game trees with a maximum depth of d. Let us now inspect a game G tree with a depth of $d + 1$, with real root value of e.g. 0. Therefore, all root successors $s_1 \ldots s_b$ of G have the real value 0. If there are two or more leaf-disjoint strategies contained in each subtree $\overline{G_1} \ldots \overline{G_b}$ below $s_1 \ldots s_b$, and if they all prove a value of 0, then $Q_{G_i}'(1) = 0$ for all $1 \leq i \leq b$. Therefore $Q_G'(1) = 0$, as well. If one or more of the subtrees $\overline{G_1} \ldots \overline{G_b}$ contains no two leaf-disjoint strategies, at least one of the addends is greater than 0 and thus $Q_G'(1) > 0$, as well.
Let G be an arbitrary game tree, and let v be its root. From now on, let $Q_G^{(n)}$ denote the n-th derivative of Q_G. By induction we will show

Theorem 2: $Q_G^{(n)}(1) = Q_G^{(n-1)}(1) = \ldots = Q_G^{(1)}(1) = 0 \Leftrightarrow$ there are $n + 1$ leaf-disjoint strategies below v that prove the real value of v.

Claim. The n-th derivative of a product of polynomials $h(x) := g_1(x) \cdots g_b(x)$ can be described as

$$\sum_{y_1 + \ldots y_b = n} a(y_1, \ldots, y_b) \cdot g_1^{(y_1)}(x) \cdots g_b^{(y_b)}(x)$$

with appropriate $a(y_1, \ldots, y_b) \in I\!N$ (natural numbers)

The derivatives of Q_1, Q_2 and Q_3

Analogously to the previous subsection we will examine the n-th derivative of Q_1, Q_2 and Q_3. We will examine them on the following assumptions: **(i)** For any game tree G and for all $i \leq n$ there is valid $Q_G^{(i-1)}(1) = \ldots = Q_G^{(1)}(1) = 0 \Leftrightarrow$ there are i leaf-disjoint strategies below v (the root of G) that prove the real value of v. **(ii)** For all $G \in \{G_1, G_2, G_3\}$ it is $Q_G^{(n-1)}(1) = \ldots = Q_G^{(1)}(1) = 0$ <u>and</u> there are n leaf-disjoint strategies below v that prove the real value of v. Moreover **(iii)**, for all $i \in \{1 \ldots n-1\}$ the following staement is valid: the sign of $Q_G^{(i)}(1)$ is equal to $(-1)^{i-1}$.

Remark: When we will start the induction itself, (i) and (iii) will form the induction hypothesis. (ii) will be derived from one of the assumption that will be made in the induction step: For a given $G \in \{G_1, G_2, G_3\}$ it is either $Q_G^{(n)}(1) = \ldots = Q_G^{(1)}(1) = 0$, or, there are $n+1$ leaf-disjoint strategies below v (the root of G) that prove the real value of v.

We start describing the derivatives now:
$Q_1^{(n)}(x) = \sum_{y_1+\ldots+y_b=n} a(y_1, \ldots, y_b) \cdot g_1^{(y_1)}(x) \cdots g_b^{(y_b)}(x)$ with appropriate $a(\ldots) \in I\!N$. With the help of assumption (b) we see that all addends that contain a derivative smaller than n are zero at the position $x = 1$. As $g_i(1) = 1$ for all i, we see that

$$\underline{Q_1^{(n)}(1) = \sum_1^b g_i^{(n)}(1)}$$

$Q_2^{(n)}(x) = (-1) \cdot \sum_{y_1+\ldots+y_b=n} a(y_1, \ldots, y_n) \cdot (1-g_1(x))^{(y_1)} \cdot g_2^{(y_2)}(x) \cdots g_b^{(y_b)}(x)$ with appropriate $a(\ldots) \in I\!N$. With the help of assumption (ii) we see that at $x = 1$ only one addend of the sum is non-equal to zero. We conclude that

$$\underline{Q_2^{(n)}(1) = g_1^{(n)}(1).}$$

The n-th derivative of $Q_3^{(n)}(x)$ is more complicated.
$Q_3^{(n)}(x) = (-1) \cdot \sum_{y_1+\ldots+y_b=n} a(y_1, \ldots, y_b) \cdot (1-g_1(x))^{(y_1)} \cdots (1-g_c(x))^{(y_c)} \cdot g_{c+1}^{(y_{c+1})}(x) \cdot \cdot \cdot g_b^{(y_b)}(x)$ with appropriate $a(\ldots) \in I\!N$.

We distinguish between 3 cases:

1. **$n < c$:** $Q_3^{(n)}(1) = 0$, because one factor of each addend is zero, and there are n leaf-disjoint strategies below v because of the definition of a strategy.
2. **$n = c$:** Let $S_{y_1,\ldots,y_b}(x)$ be any addend of $Q_3^{(n)}(x)$, at $x = 1$.
   (a) If there is an l with $l \leq c$ and ($y_l > 1$ or $y_l = 0$), it will follow $S_{y_1,\ldots,y_b}(1) = 0$, because $1 - g_l(1) = 0$.
   (b) If there is an l with $l > c$ and $y_l \neq 0$ it will follow $S_{y_1,\ldots,y_b}(1) = 0$. It is zero because the assumption implies that there is an index i with $i \leq c$ and $y_i = 0$.
   (c) Otherwise: $S_{y_1,\ldots,y_b}(1) = (-1)^c \cdot \prod_{i=1}^c g_i^{(1)}(1)$

   With $n = c$ we get $Q_3^{(n)}(1) = (-1) \cdot (-1)^n \cdot k \cdot \prod_{i=1}^n g_i^{(1)}(1)$, for a appropriate k $\in I\!N$. (In this case the sign of $Q_3^{(n)}(1)$ is (-1), if n is even, and $(+1)$ otherwise. That is because first derivatives of polynomials of quality are positive.)

3. $n > c$: Again, let $S_{y_1,\ldots,y_b}(x)$ be one of the addends of $Q_3^{(n)}(x)$.
   (a) If there is an l with $l \leq c$ and $y_l = 0$, we know that $S_{y_1,\ldots,y_b}(1) = 0$.
   (b) If there is an l with $l > c$ and $y_l > 0$, we get $\sum_{i=1}^{c} y_i \leq n-1$. $S_{y_1,\ldots,y_b}(x)$ has the form $(1-g_1(x))^{(y_1)} \cdots (1-g_c(x))^{(y_c)} \cdot X, X \in \mathbb{R}($ real numbers$)$. From assumption (ii) we know that there are n leaf-disjoint strategies below v. By the help of the definition of strategies we know that the sum of leaf-disjoint strategies below $v_1 \ldots v_c$ is n, as well. As $\sum_{i=1}^{c} y_i \leq n-1$, we can conclude that there is one successor $v_i, i \in \{1\ldots c\}$ that is supplied with more than y_i-many leaf-disjoint strategies. Thus, by the help of the assumption (i), we know that at least one of the $(1-g_i^{(y_i)}(x))$ becomes zero at the position $x=1$, for some $i \in \{1\ldots c\}$.
   (c) Last but not least there is the case of $\sum_{i=1}^{c} y_i \leq n$ and $\prod_{i=1}^{c} y_i > 0$. Here we get $Q_3^{(n)}(1) = -\sum_{y_1+\ldots+y_c=n} a(y_1,\ldots,y_c,0,\ldots,0) \cdot (1-g_1(x))^{(y_1)}(1) \cdots (1-g_c(x))^{(y_c)}(1)$ with appropriate $a(y_1,\ldots y_c,0,\ldots,0) \in \mathbb{N}$.

   With some proper $a(y_1,\ldots,y_c,0,\ldots,0) \in \mathbb{N}$ we can conclude

$$\underline{Q_3^{(n)}(1) = (-1)^{c+1} \sum_{y_1+\ldots+y_c=n} a(y_1,\ldots,y_c,0,\ldots,0) g_1(x)^{(y_1)}(1) \cdots g_c(x)^{(y_c)}(1)}$$

   With the help of assumption (iii) the sign of $Q_3^{(n)}(1)$ is equal to $(-1)\cdot(-1)^c \cdot \prod_1^c \mathrm{sign}(g_i^{(y_i)}(1))$, with $\sum_1^c y_i = n$: Let $k_i = y_i - 1, \forall i \in \{1\ldots c\}$. Thus $(-1)^{k_i}$ is the sign of $g_i^{(y_i)}(1)$. Obviously, $\sum_1^c k_i = n-c$, and therefore, the sign of $Q_c^{(n)}(1) = \prod_{i=1}^{c} k_i \cdot (-1) \cdot (-1)^c = (-1)^{(n-1)}$.

Now, we can easily prove theorem 4-2 by induction. The induction hypothesis is:

For all $i \leq n$ the following is valid: $Q_G^{(i-1)}(1) = \ldots = Q_G^{(1)}(1) = 0 \Leftrightarrow$ there are i leaf-disjoint strategies below v (the root of G) that prove the real value of v. Moreover, the sign of $Q_G^{(i)}(1)$ is equal to $(-1)^{i-1}$.

The induction start has already been done in the previous subsection.

Induction step ($n \rightarrow n+1$): We start with '$\Leftarrow$'. There are $n+1$ leaf-disjoint strategies below v, the root of a game tree G. Thus, we know that there are n leaf-disjoint strategies, too. With the help of the induction hypothesis we know that the assumptions (i), (ii), and (iii) are fulfilled. By a simple, implicit induction over the depth of G, we see that the induction step is already done. We use the previously computed derivatives of Q_1, Q_2 and Q_3.

Now, we come to '$\Rightarrow$': Let Q_G be a polynomial of quality of a game tree G. Let $Q_G^{(n)} = \ldots = Q_G^{(1)} = 0$. Obviously, $Q_G^{(n-1)} = \ldots = Q_G^{(1)} = 0$, too. From the induction hypothesis we know that there are n leaf-disjoint strategies below the root of G, reasoning the real value of the root of G. Once again, we know that the assumptions (i), (ii), and (iii) are fulfilled. By a simple, implicit induction over the depth of G, we see that the induction step is already done. We must only consider the previously computed derivatives of Q_1, Q_2 and Q_3, and the fact that for all $i \in \{1\ldots n\}$ the sign of $Q_G^{(i)}(1)$ (G a game tree) is $(-1)^{i-1}$.

5 Conclusion

We presented a combinatorial model, that allows us to model errors of a heuristic evaluation function with the help of coin tosses. The non-error probability of a heuristic minimax value at the root of any game tree G is a polynomial in the non-error probability of the heuristic evaluation function at the leaves of G. Let $Q_G(x)$ be that polynomial. We were able to prove a one to one relationship between the number of leaf-disjoint strategies that all prove the minimax value of G, and the derivatives of $Q_G(1)$.
We showed that the number of leaf-disjoint strategies that are contained in a game tree, determines the order of the quality of a heuristic minimax value. We also got an easily understandable criterion for the usefulness of game tree searches with heuristic evaluations at all.

6 Acknowledgement

We would like to thank the unknown reviewers for their constructive and helpful comments.

References

1. I. Althöfer. Root evaluation errors: How they arise and propagate. *ICCA Journal*, 11(3):55–63, 1988.
2. C. Donninger. Null move and deep search. *ICCA Journal*, 16(3):137–143, 1993.
3. R. Feldmann. Fail high reductions. *Advances in Computer Chess 8 (ed. J. van den Herik)*, 1996.
4. H. Kaindl and A. Scheucher. The reason for the benefits of minmax search. In *Proc. of the 11 th IJCAI*, pages 322–327, Detroit, MI, 1989.
5. D.E. Knuth and R.W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.
6. U. Lorenz. Controlled Conspiracy-2 Search. *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS), (H. Reichel, S.Tison eds), Springer LNCS*, pages 466–478, 2000.
7. U. Lorenz. P.ConNers wins the 10^{th} Grandmaster Tournament in Lippstadt. *ICCA Journal*, 23(3):188–192, 2000.
8. U. Lorenz. Parallel controlled conspiracy number search. *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), ACM Press*, pages 320–321, 2001.
9. D.A. McAllester. Conspiracy Numbers for Min-Max searching. *Artificial Intelligence*, 35(1):287–310, 1988.
10. D.S. Nau. *Quality of Decision versus depth of search on game trees.* PhD thesis, Duke University, Durham, NC, 1979.
11. J. Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 20(4):427–453, 1983.
12. J. Pearl. *Heuristics – Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Co., Reading, MA, 1984.
13. J. Schaeffer. Conspiracy numbers. *Artificial Intelligence*, 43(1):67–84, 1990.
14. J. Schaeffer and A. Plaat. Kasparov versus Deep Blue: The Rematch. *ICCA Journal*, 20(2):95–125, 1997.