

# Error Propagation in Game Trees <sup>★</sup>

Benjamin Doerr<sup>1</sup> and Ulf Lorenz<sup>2</sup>

<sup>1</sup> Mathematisches Seminar II, Christian-Albrechts-Universität zu Kiel,  
Ludewig-Meyn-Str. 4, D-24098 Kiel, Germany,  
[bed@numerik.uni-kiel.de](mailto:bed@numerik.uni-kiel.de),

WWW home page: <http://www.numerik.uni-kiel.de/~bed/>

<sup>2</sup> Department of Mathematics and Computer Science  
Paderborn University  
Germany

**Abstract.** Game tree search is the core of most attempts to teach computers play games. We present a fairly general theoretical analysis on how evaluation error influence the value estimation of a game position. We extend the work of Lorenz and Monien [7] in two directions. Firstly, we allow arbitrary game values. By a different approach, we show that also in this setting the number of leaf-disjoint strategies proving a particular property is a key notion. This number precisely describes the order of growth of the heuristic game value in terms of the quality of the leaf evaluation heuristics. Secondly, in allow random nodes (rolls of a die). Surprisingly, this changes the situation: Still the number of leaf-disjoint strategies ensures robustness against leaf evaluation errors, but the converse is not true. An average node may produce additional robustness like further leaf-disjoint strategies.

## 1 Introduction

When a game tree is too large to find an optimal move, the most successful approach for computers to play a chess-like two-player game is the following: The game tree search algorithm chooses a sub-game tree for examination. This can either be done static (e.g. a fixed-depth tree with depth  $t$  is chosen), or it can be done dynamically as e.g. in Conspiracy Number Search [8, 10, 5] or as in  $\alpha\beta$ -search with extensions like the Nullmove heuristic [3]. The algorithm heuristically assigns values to the artificial leaves and propagates these numbers up according to the minimax principle.

---

<sup>★</sup> The first author was partially supported by the graduate school 'Effiziente Algorithmen und Mehrskalmethoden', Deutsche Forschungsgemeinschaft. The second author was partially supported by the Future and Emerging Technologies programme of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-2001-33116 (FLAGS).

For playing 2-person zerosum games, building the heuristic minimax value is the by far most successful approach in the computer games history. When Shannon [11] proposed a design for a chess program in 1949 — as it is in its core still used by all modern game playing programs — it seemed quite reasonable that searching deeper leads to better results. Many years later, however, first theoretical analyses suggested that searching deeper will lead to *worse* results. In order to explain the error behavior, there have been published several papers by Pearl [9], G. Schrüfer, Althöfer [1, 2], Kaindl and Scheucher [4]. Lorenz and Monien [7] presented a one-to-one relation between the number of leaf-disjoint strategies and the robustness of game trees for minimax game trees with values 0 and 1. This was the first result for concrete (i.e. not randomly chosen, vague) game trees. In order to make game tree searching leave the field of parlour games towards the use in money-relevant practice, like in planning tasks, it is certainly unavoidable to understand the underlying principles. One question is which structural properties make a game tree an error-filter resp. an error amplifier. We build on the model of Lorenz and Monien [7] by both allowing arbitrary game values and random nodes.

**Model:** An arbitrary but fixed finite game tree  $G$  is given. Each of its leaves has a value in  $[0, 1]$ . These values are propagated to inner nodes following the so called mimav principle: There are MIN-, MAX-, and AVG-nodes in the tree. At a MAX-node  $v$ , the value of  $v$  is determined as the maximum over the values of  $v$ 's sons, at a MIN-node  $v$ , the value of  $v$  is determined as the minimum over the values of  $v$ 's sons. At an AVG-node the value of  $v$  is built as the (weighted) average over the values of  $v$ 's sons. A typical AVG-player is a die. The resulting values are again between 0 and 1, and we call them 'true' values.

Each leaf gets a second value, which we call a 'heuristic' one: There is a parameter  $p \in [0, 1]$  that describes the quality of the heuristics. With probability  $p$  are the heuristic and the true value of a leaf equal. We assume  $p$  to be equal for all leaves. In the special case that true and heuristic leaf values are from  $\{0, 1\}$ ,  $p$  describes the probability of a flip. The heuristic value of  $G$ 's root is the mimav value of the heuristic leaf values.

In this paper we investigate the influence of the game tree structure on the robustness of the heuristic value, i.e., the probability that the heuristic mimav value equals the true one of the root or deviates from it only slightly.

We show that leaf-disjoint strategies again are the key term. If there are  $k$  leaf-disjoint strategies proving that the game value is at least  $\gamma$ , the heuristic value is at least  $\gamma$  with probability  $1 - O(\varepsilon^k)$  for a leaf heuristics quality of  $1 - \varepsilon$ ,  $\varepsilon$  small. If we allow only MAX- and MIN-nodes, also the converse holds.

Surprisingly, this behaviour changes if AVG-nodes are included. They may generate additional robustness in the above sense, even though only few leaf disjoint strategies exist.

The paper is organized as follows: In Chapter 2 we introduce some general definitions concerning  $2\frac{1}{2}$  person game tree search. We discuss our definition of error critically. Chapter 3 consists of the analysis for multi-valued minimax values as well as for mimav values. Chapter 4 contains the proofs.

## 2 Definitions and Notations

**Definition 1.** We call  $G = (T, h)$  a game tree, if  $T = (V, E)$  is a tree and  $h : L(G) \rightarrow [0, 1]$  is a function,  $L(G)$  being the set of leaves of  $T$ .  $\Gamma(u)$  denotes the set of successors of a node  $v \in V$ . The depth of a game tree is the maximum distance between the root of  $G$  and its leaves.

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 three players *MIN*, *MAX*, and *AVG*. This defines a player function  $p : V \rightarrow \{MAX, MIN, AVG\}$ .

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

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

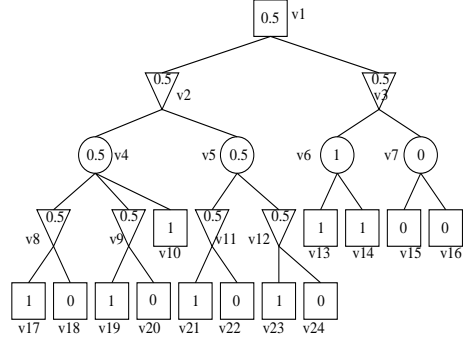
We call  $mimav(v)$  the *mimav value* of  $v$ .

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

- $r \in V_s$ .
- 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  $p(u) = \bar{s}$ , then  $\Gamma(u) \subset V_s$ , and for all  $u' \in \Gamma(u)$  is  $(u, u') \in E_s$ .
- If  $u \in V_s$  is an internal node of  $T$  with  $p(u) = AVG$ , then for some subset  $C(u)$  of  $\Gamma(u)$  is  $C(u) \subset V_s$ , and for all  $u' \in C(u)$  is  $(u, u') \in E_s$ .

As a consequence, the mimav-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 for the root value of  $G$ .

**Example:** In order to prove that the value of  $G$ 's root (see Fig. 1) is at most 0.5, we only need to inspect the subtree  $S_1 := \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{12}, v_{14}, v_{16}, v_{17}, v_{18}, v_{23}, v_{24}\}$ . Hence this subtree is a strategy proving an upper bound of 0.5. The nodes of  $S_2 := \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{11}, v_{13}, v_{16}, v_{17}, v_{18}, v_{21}, v_{22}\}$  form such a strategy as well. In contrast to  $S_3 := \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_{11}, v_{13}, v_{15}, v_{19}, v_{20}, v_{21}, v_{22}\}$ , however,  $S_2$  is not leaf-disjoint to  $S_1$ .



**Fig. 1.** A game tree  $G$

**Definition 5.** Strategies are called leaf-disjoint if they have no leaf in common.

**Definition 6.** Let  $G$  be a game tree and  $S$  be a strategy with  $x$  as lower/upper bound in  $G$ .  $S$  is called a minimal strategy if it proves  $x$  to be a lower/upper bound of  $G$ 's value and if it is possible to destroy the strategy by changing just one leaf value.

**Model for Heuristics:** We assume the heuristics evaluating the leaves to be of the following kind: There is a particular parameter  $p \in [0, 1]$  that describes the quality of the heuristics. If  $p = 1$ , then the heuristics detects the true value of a leaf surely. If  $p = 0$ , then the heuristics shall produce no reliable information: We assume that for each leaf  $\ell$  there is a fixed (but unknown) distribution  $p_\ell : [0, 1] \rightarrow [0, 1]$  such that  $\text{supp}(p_\ell) := \{x \in [0, 1] \mid p_\ell(x) \neq 0\}$  is finite,  $\{0, 1\} \subseteq \text{supp}(p_\ell)$  and such that our heuristics (in the case  $p = 0$ ) estimates the value of  $\ell$  to be  $x$  with probability  $p_\ell(x)$ . Thus in particular  $\sum_{x \in \text{supp}(p_\ell)} p_\ell(x) = 1$ . We also assume the  $p_\ell(\tau) = 0$ , if  $\tau$  is the true value of  $\ell$ . If  $p$  is in between 0 and 1, we choose between the two extremal cases proportional to  $p$ : With probability  $p$ , our heuristics detects the true value of  $\ell$ , with probability  $(1 - p) \cdot p_\ell(x)$  it outputs a faulty value  $x \in [0, 1]$ .

**Remark:** It is important for practice to assume that we cannot examine the complete game tree and that we are forced to heuristically evaluate some internal nodes. More exactly, we assume that a subtree is cut from the top of the allover game tree and that the artificial leaves of this subtree are heuristically evaluated. The resulting values are to be updated in the tree as if they were true values.

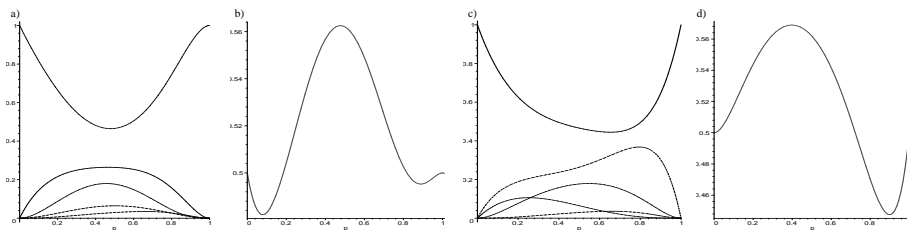
**Example:** Let the possible values of a leaf  $\ell$  be 0, 0.5 and 1. Let its true value be 0.5. For  $p = 0$  the probability to see 0.5 is 0 and the probabilities to see 0 or 1 are arbitrary, but both greater than 0 and in their sum just  $1 - p = 1$ .

So let us assume that for  $p = 0$  the probability to get a heuristic value of 0 is  $p_\ell(0) = 0.4$  and the one for 1 is  $p_\ell(1) = 0.6$ . Then for  $p = 0.5$  the corresponding probabilities are 0.2 and 0.3.

**Remark:** We think this error model quite reasonable. In particular it satisfies the following properties:

- (i) In the case of perfection it is clear that  $p = 1$  means that all heuristic values are exactly equal to the true ones.
- (ii) The parameter  $p$  allows to model a heuristic evaluator which makes an arbitrarily small number of mistakes. This fits to our effort to examine heuristic evaluators near perfection.
- (iii) Moreover, if one inspects the situation at inner nodes of  $G$  a very similar situation will be found:  
For  $p = 1$  the correct value is detected. If  $p < 1$  there will be one polynomial for each value  $y$ , describing the probability to detect just the value  $y$ .

For each node  $v$  of the tree and each  $\gamma \in [0, 1]$  we define a mapping  $q(v, \gamma) : [0, 1] \rightarrow [0, 1]$  by setting  $q(v, \gamma)(p)$  to be the probability the value of node  $v$  is evaluated to  $\gamma$  provided that our leaf heuristics detects the true value of each leaf with probability  $p$  independently for each leaf. In particular,  $q(v, \gamma)(1)$  is zero for all  $\gamma$  except the true value of the node  $v$  (where it is one). We say that  $q(v, \cdot)$  is *finite*, if  $q(v, \gamma)$  is the zero mapping for all but a finite number of  $\gamma$ .



**Fig. 2.** Specific polynomials and the corresponding polynomial for the expected value at the root of  $G$  (cf. Fig. 1).

Figure 2a) shows the probabilities to see the root value of our example tree as 0, 0.25, 0.5, 0.75 or 1, depending on the leaf-error probability  $1 - p$ . The uppermost curve describes the probability that the root evaluates to 0.5. Its first derivative at  $p = 1$  is zero. This is due to the fact that there are two leaf-disjoint strategies proving this value.

Let  $H$  arise from  $G$  by flipping the value of node v13 from 1 to 0. Figure 2c) shows the corresponding probabilities. As a consequence, the first derivative of the 0.5-polynomial (it is again the most upper one) becomes greater than zero. We have

no two leaf-disjoint strategies. Near perfection, the number of derivatives being zero at  $p = 1$  tells us how well the game tree filters errors.

One might argue that the behaviour of a polynomial  $q_e$  which describes the expected value of the game tree (examples are Figure 2b) and d) for the game trees  $G$  and  $H$ ) is more interesting than the whole bundle of value-specific polynomials. However: on the one hand the expectation polynomial was resistant against our attempts to analyse it and, moreover, is  $q'_e(v, x)(1) = 0$  if and only if  $q'(v, x)(1) = 0$ ,  $x$  being the true value of  $v$ .  $q_e$  brings no further insight.

### 3 Error Analysis

Recall that we denote the probability that our heuristics (with quality parameter  $p$ ) outputs the value  $\gamma$  for the node  $v$  by  $q(x, \gamma)(p)$ . Our assumption fixes  $q(v, \gamma)$  for all leaves. Since we assume that the leaf values are propagated through the mimav principle, we obtain the following for non-leaf nodes. Let us denote by

$$\text{supp}(q(v, \cdot)) := \{\gamma \in [0, 1] \mid q(v, \gamma)(p) \neq 0 \text{ for some } p \in [0, 1]\}.$$

We call  $q(v, \cdot)$  *finite*, if  $\text{supp}(q(v, \cdot))$  is finite.

**Lemma 1.** *Let  $v$  be a non-leaf node with children  $v_1, \dots, v_b$ .*

(i) *If  $v$  is a MAX-node and all  $q(v_i, \cdot), i \in [b]$  are finite, then*

$$q(v, \gamma) = \sum_{(\gamma_1, \dots, \gamma_b) \in [0, \gamma]^b \setminus [0, \gamma]^{[b]}} \prod_{i=1}^b q(v_i, \gamma_i)$$

*for all  $\gamma \in [0, 1]$ . In particular,  $q(v, \cdot)$  is finite.*

(ii) *If  $v$  is a MIN-node and all  $q(v_i, \cdot), i \in [b]$  are finite, then*

$$q(v, \gamma) = \sum_{(\gamma_1, \dots, \gamma_b) \in [\gamma, 1]^b \setminus [\gamma, 1]^{[b]}} \prod_{i=1}^b q(v_i, \gamma_i)$$

*for all  $\gamma \in [0, 1]$ . In particular,  $q(v, \cdot)$  is finite.*

(iii) *If  $v$  is an AVG-node and all  $q(v_i, \cdot), i \in [b]$  are finite, then*

$$q(v, \gamma) = \sum_{\frac{1}{b}(\gamma_1 + \dots + \gamma_b) = \gamma} \prod_{i=1}^b q(v_i, \gamma_i)$$

*for all  $\gamma \in [0, 1]$ . In particular,  $q(v, \cdot)$  is finite.*

(iv)  $q(v, \gamma)$  is a polynomial function for all  $\gamma \in [0, 1]$ .

In the following, we do not distinguish between polynomials  $q \in \mathbb{R}[p]$  and the corresponding polynomial functions  $p \mapsto q(p)$ . Since this correspondence is one-one, there is no danger of ambiguity. By item (iv) of the above lemma we may write  $q(v, \Gamma) := \sum_{\gamma \in \Gamma} q(v, \gamma)$  for arbitrary subsets  $\Gamma \subseteq [0, 1]$  without worrying about existence of such sums. We use the shorthands

$$q(v, \leq \gamma) := q(v, [0, \gamma]), q(v, < \gamma) := q(v, [0, \gamma]), q(v, \geq \gamma) := q(v, [\gamma, 1]), q(v, > \gamma) := q(v, ]\gamma, 1]).$$

Clearly, such functions are polynomials as well. For an arbitrary real mapping  $q$  we write  $q^{(i)}$  to denote its  $i$ th derivative. We put  $\kappa(q) = 0$ , if  $q(1) \neq 1$ , and else  $\kappa(q) := \sup\{n \in \mathbb{N} \mid \forall 1 \leq i \leq n-1 : q^{(i)}(1) = 0\}$ . Hence  $\kappa(q) = 1$ , if  $q(1) = 1$  and  $q'(1) \neq 0$ , for example. For polynomials, we have

**Lemma 2.** *Let  $q$  be a non-zero polynomial. Then  $\kappa(q)$  is the largest  $n \in \mathbb{N}_0$  such that  $(x-1)^n$  divides  $q-1$  in the ring of polynomials.*

**Lemma 3.** *Let  $G$  be an arbitrary game tree (allowing any kind of nodes). Denote the root by  $r$  and its true value by  $\tau$ . Let  $\gamma \leq \tau$  and  $k \geq 1$ .*

- (i) *Assume that the root value remains at least  $\gamma$ , even if the value of up to  $k$  leaves is changed<sup>1</sup>. Then  $\kappa(q(r, \geq \gamma)) = k+1$ .*
- (ii) *Conversely,  $\kappa(q(r, \geq \gamma)) = k+1$  implies that the value of up to  $k$  arbitrarily chosen leaves may be changed without the root value dropping below  $\gamma$ .*

**Theorem 1.** *Let  $G$  be a game tree with root  $r$ . If Max-Player has  $k$  leaf-disjoint strategies proving a game value of at least  $\gamma$ , then  $\kappa(q(r, \geq \gamma)) \geq k$ . In particular, if  $k \geq 2$ , then  $q^{(1)}(r, \geq \gamma)(1) = \dots = q^{(k-1)}(r, \geq \gamma)(1) = 0$ . This is valid for all game trees with MIN, MAX and AVG players.*

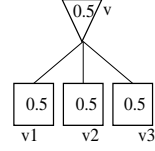
**Theorem 2.** *Let  $G$  be a minimaxgame tree with root  $r$  (only the players MIN and MAX). Then  $\kappa(q(r, \geq \gamma))$  is the number of leaf-disjoint strategies for Max-Player proving a game value of at least  $\gamma$ .*

Both theorems are analogously valid for upper bound strategies.

**Theorem 3.** *Let  $G$  be a minimaxgame tree with root  $r$  (only the players MIN and MAX). Let  $q(v, [\gamma - \delta, \gamma + \delta])$  denote the probability that the recognized heuristic value of the node  $v$  with true value  $\gamma$  is in between  $\gamma - \delta$  and  $\gamma + \delta$ . Then follows: If and only if the Max-Player has  $k \geq 2$  leaf-disjoint strategies proving a value of at least  $\gamma - \delta$  and at most  $\gamma + \delta$ , then  $q^{(1)}(r, [\gamma - \delta, \gamma + \delta])(1) = \dots = q^{(k-1)}(r, [\gamma - \delta, \gamma + \delta])(1) = 0$ . This is valid for  $\delta = 0$  in particular.*

Theorem 2 is not valid for mimav values as can easily be seen:

$G_c$  of figure 3 certainly contains no 2 LDSs showing a value at the root of  $\geq 0.2$ . The reason is that no single successor of  $v$  can prove 0.2 as lower bound. Nevertheless, you can change any of the three leaf values without that the value of  $v$  falls below 0.2. Thus  $q'(v, \geq 0.2)(1) = 0$ . Interestingly, if we want to say something about  $q(v, \geq 0.4)$ , none of the nodes  $v_1, v_2, v_3$  may be arbitrarily changed in order to get  $q'(v, \geq 0.4)(1) = 0$ . Nevertheless, there is a node dependent constant  $k = 0.3$  by which each successor of  $v$  may be lowered without that the bound 0.4 can be

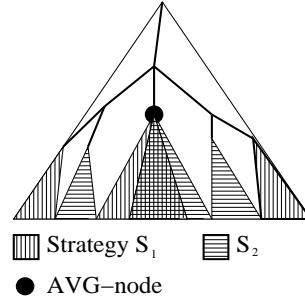


**Fig. 3.** Counter example: Game tree  $G_c$

hurt. Thus, if  $q'(v, \geq 0.4)(1) = 0$  then it is not possible to change the value of  $v$  below 0.4 by changing one leaf value. As a consequence, it is not possible to change the values of  $v_1, v_2$  and  $v_3$  below 0.2 by changing one single leaf value, and therefore it is necessary that  $q'(v_1, \geq 0.2)(1) = q'(v_2, \geq 0.2)(1) = q'(v_3, \geq 0.2)(1) = 0$ .

**Theorem 4.** Let  $G$  be a game tree with root  $r$ . The following statements are equivalent:

- Max-Player has at least 2 minimal strategies  $S_1$  and  $S_2$  proving a game value of at least  $\gamma$ , and it holds: a leaf  $l$  of  $S_1$  is either not in  $S_2$ , or on the path from  $l$  to  $r$  there is an AVG-node with one successor only in  $S_1$  and another successor only in  $S_2$ .
- $q^{(1)}(r, \geq \gamma)(1) = 0$
- you can change any single leaf value without that the value of  $r$  changes below  $\gamma$ .



**Fig. 4.** The structure of a nice mimav game tree  $G$

*Proof.* Induction over the depth of game trees. For a single leaf there is nothing to show. Let theorem 4 be valid for all game trees with depth less than  $t$ . If the root of a game tree with depth  $t$  is a MIN or a MAX node the theorem follows analogously to theorem 2. If the root is an AVG node it follows by the following observation.

Let  $G$  be a game tree with AVG-root  $v$  and successors  $v_1 \dots v_b$ . Let  $w, w_1, \dots, w_b$  their true values. Let  $\gamma$  be any number from  $]0, w]$ . Let  $q'(v \geq \gamma)(1) = 0$ , which

<sup>1</sup> by changing the value of some leaf  $\ell$  we always mean that its value is replaced by another possible value from  $\text{supp}(p_\ell)$ .

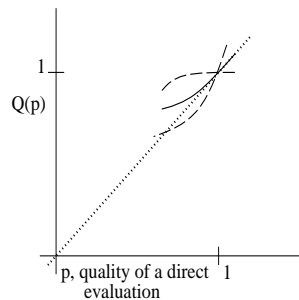
just means that you are allowed to misread at least one arbitrary leaf value without that you recognize the value of  $v$  smaller than  $\gamma$ . Then, there is a  $k \in [0, 1]$  by that any successor of  $v$  may be lowered without that the value of  $v$  falls below  $\gamma$ . Let  $w_1 \dots w_b$  the true values of  $v_1 \dots v_b$ . Then  $q'(v \geq \gamma)(1) = 0 \Leftrightarrow q'(v_i \geq w_i - k)(1) = 0 \forall v_i \in \{v_1, \dots, v_b\}$  with  $w_i - k > 0$ .  $\square$

There seems no simple combinatorial property of mimav game trees which expresses whether or not  $q^{(i)}(r, \geq \gamma)(1) = 0$  for  $i > 1$ . Again, let us inspect the example of figure 3. If it shall be possible to change two arbitrarily selected leaf values without that the value of  $v$  falls below 0.4 (resp. the sum over the values of  $v_1, v_2$  and  $v_3$  must stay above 12), then there are a variety of possibilities how e.g. the values of  $v_1$  and  $v_2$  can be changed so that their sum stays 8. Indeed it is quite difficult to encounter all these possibilities in the general case.

**Impact on practice:** Although theorem 4 is not as satisfying as its analogon for minimaxgame trees and although it is only valid for first derivatives, our new knowlwdge about the interconnection between inner structure of game trees and their robustness opens an opportunity for a heuristic search algorithm which works similar to the Controlled Conspiracy-2 Seach [6].

**Interpretation:** Figure 5 presents us with three different courses of  $q(\cdot)(p)$  and the identity function. In the case of  $q(\cdot)(p) \leq p$ , 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  $q(\cdot)(p) > p$ , 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  $q(r_1, \cdot)(p)$  and  $q(r_2, \cdot)(p)$  are polynomials of quality for game trees  $G$  and  $H$  with roots  $r_1$  and  $r_2$ , and if  $q'(r_1, \cdot)(1) = 0$  and  $q'(r_2, \cdot)(1) > 0$ ,

there will be an  $\epsilon > 0$  such that for all  $p \in [1 - \epsilon, 1]$  it is  $q(r_1, \cdot)(p) > q(r_2, \cdot)(p)$ , and thus  $G$  is more useful than  $H$ .



**Fig. 5.** Possible courses of  $Q(p)$

**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’ [7].

## 4 Proofs

Put  $\lambda(q) = \kappa(1 - q)$  for all polynomials  $q$ . Then  $\lambda(q) = 0$ , if  $q(1) \neq 0$ , and else  $\lambda(q) = \sup\{n \in \mathbb{N} \mid \forall i \in [n] : q^{(i-1)}(1) = 0\}$ . Thus  $\lambda(q)$  is the largest  $n \in \mathbb{N}_0$  such that  $(x - 1)^n$  divides  $q$  in the ring of polynomials (and  $\infty$ , if  $q = 0$ ). Note that by the uniqueness of prime factorization in the ring of polynomials, we have  $\lambda(\prod_{i=1}^b q_i) = \sum_{i=1}^b \lambda(q_i)$  for all polynomials  $q_1, \dots, q_b$ . Without proof, we give some elementary properties:

**Lemma 4.** *Let  $q_1, \dots, q_b : [0, 1] \rightarrow [0, 1]$  be polynomial functions such that  $q := \sum_{i=1}^b q_i \leq 1$ . Let  $q_1(1) = 1$ . Then  $\kappa(q) \geq \kappa(q_1)$ . Equality holds, if  $\lambda(q_i) > \kappa(q_1)$  for all  $i \geq 2$ .*

**Lemma 5.** *Let  $q_1, \dots, q_b$  be polynomial functions. Then  $\kappa\left(\prod_{i=1}^b q_i\right) = \min_{i \in [k]} \kappa(q_i)$ .*

**Lemma 6.** *Let  $G$  be a game tree. Denote the root by  $r$  and its true value by  $\tau$ . Let  $\gamma \leq \tau$  and  $k \geq 1$ .*

- (i) *Assume that the root value remains at least  $\gamma$ , even if the value of up to  $k$  leaves is changed<sup>2</sup>. Then  $\kappa(q(r, \geq \gamma)) = k + 1$ .*
- (ii) *Conversely,  $\kappa(q(r, \geq \gamma)) = k + 1$  implies that the value of up to  $k$  arbitrarily chosen leaves may be changed without the root value dropping below  $\gamma$ .*

*Proof.* Let  $L$  be the set of leaves,  $\mathcal{L}$  be the set of all subsets of  $L$  such that suitably changing the values of all their leaves changes the root value to something below  $\gamma$ . Then  $|L_0| \geq k + 1$  for all  $L_0 \in \mathcal{L}$ . Denote by  $P(L_0, p)$  the probability that the heuristics (with confidence parameter  $p$ ) errs on exactly the leaves in  $L_0$ . Then the probability that the root value is seen to be below  $\gamma$  is at most  $\sum_{L_0 \in \mathcal{L}} P(L_0, p)$  (we say ‘at most’, as not all errors on some  $\mathcal{L}$ -set need to change the root value to that extend). By Lemma 4 we have

<sup>2</sup> by changing the value of some leaf  $\ell$  we always mean that its value is replaced by another value from  $\text{supp}(p_\ell)$ .

$\kappa(q(r, \geq \gamma)) \geq \kappa(1 - \sum_{L_0 \in \mathcal{L}} P(L_0, p))$ . Since  $P(L_0, p) = p^{|L \setminus L_0|} (1-p)^{|L_0|}$  and  $|L_0| \geq k+1$ , we conclude  $\kappa(q(r, \geq \gamma)) = k+1$ .

To prove the converse (via contraposition), assume that there are  $k$  leaves changing the root value to below  $\gamma$ . Without loss of generality let  $k$  be minimal. Let  $\mathcal{L}$  be the set of all pairs  $(L_0, \delta)$  such that  $L_0$  is a set of leaves and  $\delta : L_0 \rightarrow [0, 1]$  an assignment of values to the leaves such that changing the values of all  $\ell \in L_0$  to  $\delta(\ell)$  induces the described root value change. By assumption,  $|L_0| \geq k$  for all  $(L_0, \delta)$  in  $\mathcal{L}$ . The probability that our heuristics evaluated the leaves in  $L_0$  as described by  $\delta$  and all remaining leaves correctly is

$$P((L_0, \delta), p) = \prod_{\ell \in L_0} q(\ell, \delta(\ell)) \prod_{\ell \in L \setminus L_0} q(\ell, \tau_\ell) = (1-p)^{|L_0|} p^{|L \setminus L_0|} \prod_{\ell \in L_0} p_\ell(\delta(\ell)).$$

Hence

$$\begin{aligned} q(r, \geq \gamma)(p) &= 1 - \sum_{(L_0, \delta) \in \mathcal{L}} P((L_0, \delta), p) \\ &= 1 - \sum_{(L_0, \delta) \in \mathcal{L}} (1-p)^{|L_0|} p^{|L \setminus L_0|} \prod_{\ell \in L_0} p_\ell(\delta(\ell)). \end{aligned}$$

By Lemma 4,  $\kappa(q(r, \geq \gamma)) = \kappa(q)$ , where

$$q(p) := 1 - (1-p)^k p^{|L|-k} \sum_{\substack{(L_0, \delta) \in \mathcal{L} \\ |L_0|=k}} \prod_{\ell \in L_0} p_\ell(\delta(\ell)).$$

Clearly,  $\kappa(q) = k$ , hence  $\kappa(q(r, \geq \gamma)) < k+1$ .

As is obvious from the proof, the particular structure of the nodes of the game tree is not important. Hence Lemma 6 holds also for game trees containing other nodes than just MIN-, MAX- and AVG-nodes (provided that the model is adopted suitably).

Now Theorem 1 follows easily from Lemma 6: If there are  $k$  leaf-disjoint strategies proving that the game value is at least  $\gamma$ , then changing any  $k-1$  leaves' values can not reduce the root value to below  $\gamma$  (simply because there is at least one strategy whose leaves' values are unchanged). Hence Lemma 6 shows that  $\kappa(q(r, \geq \gamma))$  is at least  $k$ .

**Lemma 7.** *Let  $\ell$  be a leaf with true value  $\tau$ . Then  $\kappa(q(\ell, \geq \gamma)) = 1$ , if  $0 < \gamma \leq \tau$ , and  $\kappa(q(\ell, \geq \gamma)) = 0$ , if  $\gamma > \tau$ .*

*Proof.* If  $\tau \geq \gamma > 0$  then  $q(\ell, \geq \gamma)(p) = 1 - (1-p)(\sum_{\delta < \gamma} p_\ell(\delta))$ . Since by assumption  $\sum_{\delta < \gamma} p_\ell(\delta) > 0$ , we have  $q(\ell, \geq \gamma)(1) \neq 0$ . Together with  $q(\ell, \geq \gamma)(1) = 1$ , we have  $\kappa(q(\ell, \geq \gamma)) = 1$ . On the other hand if  $\gamma > \tau$ , then clearly  $q(\ell, \geq \gamma)(1) \neq 1$  and thus  $\kappa(q(\ell, \geq \gamma)) = 0$ .

*Proof (Theorem 2).* We proceed by induction on the depth of  $G$ . For depth 0 we simply invoke Lemma 7. Let us therefore assume that  $G$  has depths greater than zero, and that Theorem 2 is true for all smaller depths. We need to treat the cases that the root  $r$  of  $G$  is a MAX– and MIN–node separately.

**Case 1:** Let  $r$  be a MAX–node with children  $v_1, \dots, v_b$ . Then

$$q(r, \geq \gamma) = 1 - \prod_{i=1}^b q(v_i, < \gamma)$$

by Lemma 1. We have

$$\begin{aligned} \kappa(q(r, \geq \gamma)) &= \kappa\left(1 - \prod_{i=1}^b q(v_i, < \gamma)\right) = \lambda\left(\prod_{i=1}^b (1 - q(v_i, \geq \gamma))\right) \\ &= \sum_{i=1}^b \lambda(1 - q(v_i, \geq \gamma)) = \sum_{i=1}^b \kappa(q(v_i, \geq \gamma)). \end{aligned}$$

By induction, each node  $v_i$  has  $\kappa(q(v_i, \geq \gamma))$  leaf-disjoint strategies proving a node value of at least  $\gamma$ . Each of these strategies (together with the root) is a strategy proving that the root value is at least  $\gamma$ .

**Case 2:** Let  $r$  be a MIN–node with children  $v_1, \dots, v_b$ . Then

$$q(r, \geq \gamma) = \prod_{i=1}^b q(v_i, \geq \gamma)$$

by Lemma 1. By Lemma 5,  $\kappa(q(r, \geq \gamma)) = \min_{i \in [b]} \kappa(q(v_i, \geq \gamma))$ . By induction, each node  $v_i$  has thus at least  $\kappa(q(r, \geq \gamma))$  leaf-disjoint strategies proving that the node value is at least  $\gamma$ . Putting together one such strategy for each  $v_i$  by definition yields a strategy for the root  $r$ . Hence there are at least  $\kappa(q(r, \geq \gamma))$  leaf-disjoint strategies proving a root value of at least  $\gamma$ .  $\square$

## 5 Conclusion

We presented a refined combinatorial model, which allows us to model errors of a heuristic evaluator in games with two players and a die, with the help of coin tosses.

The non-error probability of a so called heuristic mimav value at the root of any game tree  $G$  with root  $v$  is collection of polynomials in the non-error probability of the heuristic evaluation function at the leaves of  $G$ . Let  $q(v, x)$  be that polynomial. We were able to prove a one to one relationship between the number of leaf-disjoint strategies that all prove the mimav value of  $G$ , and the derivatives of  $q(v, x)(1)$ .

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 mimav value. We also got an easily understandable criterion for the usefulness of game tree searches with multiple possible values and with heuristic evaluations at all.

## References

1. I. Althöfer. Root evaluation errors: How they arise and propagate. *ICCA Journal*, 11(3):55–63, 1988.
2. I. Althöfer. Generalized minmax algorithms are no better error correctors than minmax itself. In D.F. Beal, editor, *Advances in Computer Chess 5*, pages 265–282. North-Holland, 1990.
3. C. Donninger. Null move and deep search. *ICCA Journal*, 16(3):137–143, 1993.
4. H. Kaindl and A. Scheucher. The reason for the benefits of minmax search. In *Proc. of the 11<sup>th</sup> IJCAI*, pages 322–327, Detroit, MI, 1989.
5. 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.
6. 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.
7. U. Lorenz and B. Monien. The secret of selective game tree search, when using random-error evaluations. *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)* (H. Alt, A. Ferreira eds), Springer LNCS, pages 203–214, 2002.
8. D.A. McAllester. Conspiracy Numbers for Min-Max searching. *Artificial Intelligence*, 35(1):287–310, 1988.
9. J. Pearl. *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Co., Reading, MA, 1984.
10. J. Schaeffer. Conspiracy numbers. *Artificial Intelligence*, 43(1):67–84, 1990.
11. C.E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.