Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 620–642
Artificial Intelligence
www.elsevier.com/locate/artint

Is real-valued minimax pathological?

Mitja Luštrek [a,*], Matjaž Gams [a], Ivan Bratko [b]

[a] *Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
[b] *Faculty of Computer and Information Science, University of Ljubljana, Tržaška cesta 25, 1000 Ljubljana, Slovenia*

Received 4 January 2005; received in revised form 9 January 2006; accepted 15 January 2006

Available online 17 February 2006

Abstract

Deeper searches in game-playing programs relying on the minimax principle generally produce better results. Theoretical analyses, however, suggest that in many cases minimaxing amplifies the noise introduced by the heuristic function used to evaluate the leaves of the game tree, leading to what is known as pathological behavior, where deeper searches produce worse results. In most minimax models analyzed in previous research, positions' true values and sometimes also heuristic values were only losses and wins. In contrast to this, a model is proposed in this paper that uses real numbers for both true and heuristic values. This model did not behave pathologically in the experiments performed. The mechanism that causes deeper searches to produce better evaluations is explained. A comparison with chess is made, indicating that the model realistically reflects position evaluations in chess-playing programs. Conditions under which the pathology might appear in a real-value model are also examined. The essential difference between our real-value model and the common two-value model, which causes the pathology in the two-value model, is identified. Most previous research reports that the pathology tends to disappear when there are dependences between the values of sibling nodes in a game tree. In this paper, another explanation is presented which indicates that in the two-value models the error of the heuristic evaluation was not modeled realistically.

Keywords: Game playing; Minimax; Pathology; Game tree; Real value; Chess

1. Introduction

The minimax principle lies at the heart of almost every game-playing program. Programs typically choose the best move by searching the game tree, heuristically evaluating the leaves and then propagating their values to the root using the minimax principle. In practice, deeper searches generally produce better results. However, mathematical analysis indicates that under certain seemingly sensible conditions, the opposite is true: minimaxing amplifies the error of the heuristic evaluation [2,10]. This phenomenon is called the minimax pathology [10]. Nau [14] described this as "doing worse by working harder". Evidently, game trees of real games must be different from game trees used in the theoretical analyses in some way that eliminates the pathology. Several explanations of what property of game trees of real games might be responsible have been put forth: similarity of positions close to each other [3,11], presence of stabilizing game-tree nodes with reliable estimates [4,16] etc. And while these properties can be shown to eliminate

* Corresponding author.
E-mail addresses: mitja.lustrek@ijs.si (M. Luštrek), matjaz.gams@ijs.si (M. Gams), bratko@fri.uni-lj.si (I. Bratko).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.01.006

the pathology, the question to what extent they are actually present in game trees of real games remains. Also, the mechanism by which they achieve its goal is sometimes inadequately explained.

This paper attempts to address both of these problems: we model game trees in a way that we believe reflects well the properties of real games and we explain the mechanism that eliminates the pathology. In our game trees, positions close to each other are similar. But unlike most models used in previous research, in which positions' true values, and sometimes also heuristic values, could only be losses or wins, we use real numbers for both true and heuristic values. This makes it possible to model the similarity between a position's descendants in a more natural way, and enables a more direct comparison to game-playing programs, which typically use a sizable range of integer values. The model can be analyzed mathematically to explain the mechanism that makes minimax effective. Multiple types of error and different experimental settings, among them the approximation of real values by a limited number of discrete values, are used to determine when exactly is the pathology present in a real-value model. Some light is also shed on the reasons for pathological behavior of the two-value models used in previous research.

The paper is organized as follows. Section 2 is a short introduction to the minimax pathology with some historical overview. Section 3 presents our model of minimax. Section 4 gives a mathematical analysis of minimax with real-number values. Section 5 compares our model to a chess program. Section 6 explains when and how the pathology appears in a real-value model and examines the differences between real-value and two-value models. Section 7 concludes the paper and points out some areas where further research is needed.

2. Related work

The minimax pathology was discovered independently by Nau [10] and Beal [2]. It was later more thoroughly analyzed by many authors, among them Bratko and Gams [4], whose notation we adopt. In this section, we present a basic minimax model like the one by Beal and others.

A game tree consists of nodes representing game positions and edges representing moves. In the basic model, positions can be lost or won. Negamax notation is used in this section, i.e. nodes are marked as lost or won from the perspective of the side to move. Two situations are possible: a node has at least one lost descendant, in which case the node itself is won because one can always choose the move leading to the descendant lost for the opponent; or a node has only won descendants, in which case the node itself is lost because all moves lead to positions won for the opponent. This is shown in Fig. 1; losses are marked with "−" and wins with "+".

Virtually all research on the minimax pathology assumed game trees to have a uniform branching factor b. In the basic model, the value of each leaf is independent of other leaves' values. Let d be the depth of search and k_i the probability of a loss at ith ply. Plies are numbered upwards: 0 for the lowest ply of search and d for the root.

Since a node can only be lost if all of its descendants are won, the relation between the values of k at consecutive plies is governed by Eq. (1).

$$k_{i+1} = (1 - k_i)^b. \tag{1}$$

The goal is to calculate the probability of incorrectly evaluating the root given the probability of an incorrect evaluation at the lowest ply of search. Two types of evaluation error are possible: a loss can be mistaken for a win (false win) or a win for a loss (false loss). Let p_i and q_i be the probabilities of the respective types of error at ith ply. False wins occur in nodes where all descendants should be won, but at least one of them is a false loss. Therefore p_{i+1} can be calculated according to Eq. (2).

$$p_{i+1} = \frac{1}{k_{i+1}}(1 - k_i)^b\big(1 - (1 - q_i)^b\big) = 1 - (1 - q_i)^b. \tag{2}$$

False losses occur in nodes where some descendants should be lost, but all of those are false wins instead, while all won descendants retain their true values. Therefore q_{i+1} can be calculated according to Eq. (3).

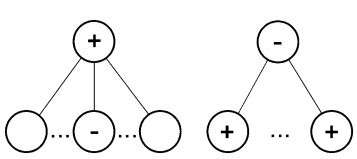

Fig. 1. Types of nodes.

$$q_{i+1} = \frac{1}{1-k_{i+1}} \sum_{j=1}^{b} \binom{b}{j} k_i^j (1-k_i)^{b-j} p_i^j (1-q_i)^{b-j}$$
$$= \frac{(k_i p_i + (1-k_i)(1-q_i))^b - ((1-k_i)(1-q_i))^b}{1-k_{i+1}}. \quad (3)$$

If only games where both sides have an equal chance of winning at the root of the game tree are considered, k_d must be 0.5 and Eq. (1) can be used to calculate k for other plies. Values for p_0 and q_0 have to be chosen and Eqs. (2) and (3) can be used to calculate p_d and q_d. If error at the root is defined as either $p_d k_d + q_d(1-k_d)$ [4] or $1/2(p_d + q_d)$ [16], it turns out that with increasing d, the error converges towards 0.5, rendering minimax useless.

First attempts to explain this phenomenon were made by Bratko and Gams [4] and Beal [3]. Both came to the conclusion that the reason minimax is effective in real games is that sibling nodes have similar values. Bratko and Gams argued that this property causes the formation of subtrees with reliably evaluated roots, which have a stabilizing effect. They did not verify whether this is indeed the case in real games. Beal showed that if a sufficient number of nodes that have all the descendants either lost or won are present in a game tree, the pathology disappears. He successfully verified his claim on king and pawn versus king chess endgame, but his results are not conclusive because such an endgame is not a typical chess situation. Nau [11,13] used a simple game designed for minimax analysis to show that a strong dependence between a node and its descendants eliminates the pathology. Pearl [16] suspected that real games do not have such a strong dependence. He argued that early terminations (also called blunders), which carry reliable evaluations, are the main reason for the elimination of the pathology. Both early terminations and subtrees with similar node values proposed by Bratko and Gams [4] result in relatively error-free nodes, so these two properties use basically the same mechanism to eliminate the pathology. Pearl's calculations show that the number of necessary early terminations in a game tree with $b = 2$ is 5%, but attempts by Luštrek and Gams [9] to verify this experimentally indicate that 20% is needed with $b = 2$ and even more with larger branching factors. Schrüfer [19] showed that if the error, particularly the probability of a false loss, is sufficiently low and certain additional conditions are satisfied, a game tree is not pathological. He did not investigate whether these conditions are present in real games. Althöfer [1] showed that in the trees that are pathological according to Schrüfer, the pathology persists even if some other back-up procedure is used instead of minimax.

Some researchers used models with multiple heuristic values, while they kept the true values restricted to losses and wins. Nau [11,13] used multiple heuristic values simply because they were a natural choice for the game he studied. In [12], he was concerned with proving that in certain circumstances, when the evaluation function has a limited number of possible values and the branching factor is large enough, the pathology is inevitable. Later [13] he extended the proof to multiple true values. Pearl [16] also considered multiple heuristic values and concluded that with regard to the pathology, they behave the same way as two values. Scheucher and Kaindl [18] were the first to consider multiple heuristic values crucial for the elimination of the pathology. They used them to design a heuristic evaluation function representative of what game-playing programs use. According to this function, not only were the values of the leaves of the game tree not independent of each other, it also implied that at lower plies of the game tree, positions where error is less likely are encountered more often than at higher plies. These two characteristics eliminated the pathology, but it remained unclear why the error should depend on the depth of search exactly the way they proposed.

A model with an arbitrary number of heuristic values and the same number of true values was already studied by Bratko and Gams [4]. They used completely independent leaf values and this model behaved similarly to the two-value version. Sadikov et al. [17] used multiple values in their analysis of king and rook versus king chess endgame. They managed to explain the pathology in backed-up position evaluations, but it is not known whether their conclusion applies to cases other than the endgame they studied. Some preliminary results of the research fully described in this paper were published by Luštrek [8].

3. A model with real-number values

There are many games where the final outcome is multivalued, typically expressed as an integer (Othello, bridge…). In such games, multivalued position values, both true and heuristic, are obviously needed. In games where the outcome can only be a loss, a win and perhaps a draw (chess, checkers…), multiple values might only seem necessary to express the uncertainty of the heuristic evaluation function, not as true values. However, even unlimited

resources to determine the value of a position cannot remove the uncertainty: in a losing position, the best one can do against a fallible and not fully known opponent is to evaluate moves in terms of the probability of a loss against this particular opponent. More importantly, in a winning position, multiple values make it possible to maintain a *direction* of play, gradually moving toward a win, not just maintaining a won position without achieving the final goal. Multiple values are necessary to differentiate between multiple winning (or losing, if only such are available) moves. Scheucher and Kaindl [18] already discussed the need for multiple values, although they were only concerned with heuristic values. They demonstrated on chess that a two-valued evaluation function performs poorly compared to a multivalued one. Given that multivalued position values are necessary for playing well, it is natural to also use multivalued true values. These values are true in the sense that they guide a program to play optimally. In our minimax model we use real values instead of multiple discrete values. This facilitates the mathematical explanation of why minimax works. The issue of granularity, i.e. how many distinct values should be used, is studied separately in Section 6.2. In particular, in Section 6.2 we address the question: Are the desirable properties of our real-valued model retained when real values are replaced by a limited number of discrete values?

3.1. Description of the model

In our model, the values of sibling nodes are distributed around the value of their parent. The rationale for such a distribution is that the descendants of a position are only one move away from it and are therefore unlikely to have a significantly different value. Dependence between the values of nodes and their descendants results in dependence between the values of leaves, which is a departure from the Section 2's assumption of independent leaf values. Negamax notation is no longer used, but the branching factor is still constant.

According to our model, a game tree is constructed from the root down. A so-called auxiliary value of 0 is assigned to the root and the auxiliary values of its descendants are distributed around it. The process is repeated recursively on each descendant until the maximum depth $d_{\max}$ required for a given experiment is reached. It should be noted that the auxiliary values generally do not respect the min/max rule. They serve only to establish the desired relation between the values of the leaves. Such a model is similar to Nau's incremental [11] or N-games [12] and to the model used by Scheucher and Kaindl [18], as well as to some earlier analyses of the alpha-beta algorithm [5,7,15].

The auxiliary values of the leaves of a tree with depth $d = d_{\max}$ are considered true values. True values at depths $d < d_{\max}$ are obtained by performing minimax starting with the true values of the leaves. In experiments with minimax search to depth d, static heuristic values are obtained by corrupting the true values at depth d with error representing the fallibility of the heuristic evaluation function. The error at the lowest ply of search, i.e. at depth d, is called static error. Backed-up heuristic values are obtained by performing minimax on the static heuristic values. The procedure is illustrated in Fig. 2.

The error is normally distributed noise. Normal distribution is chosen because this is the usual practice when no useful information on the error is available. Two types of error can be observed in a node: *position error*, which is the difference between the true and the heuristic value of the node, and *move error*, which is the probability of choosing a wrong move because of position error at the node's descendants.

The model has a number of parameters:

- b—branching factor,
- type of distribution of nodes' auxiliary values around the auxiliary value of their parent,
- σ_v—standard deviation of the above distribution,

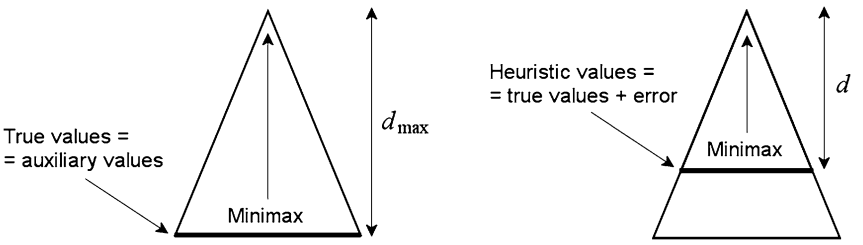

Fig. 2. Construction of game trees.

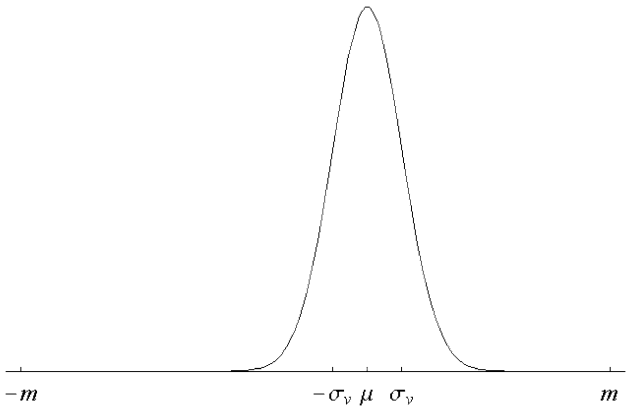

Fig. 3. Parameters of the model.

- $[-m, m]$—the interval within which nodes' values must lie,
- σ_e—standard deviation of static position error.

Some of the parameters are illustrated in Fig. 3; the curve represents probability density function of the distribution of sibling nodes' auxiliary values (in this case normal) and μ is the auxiliary value of their parent.

3.2. Experimental results

We investigate the behavior of minimax by randomly generating game trees and observing position and move error. The size of a game tree exceeds b^d, so computations with large branching factors or depths are very time-consuming. We conducted experiments with $b = 2$ and $b = 5$. The results were similar in all cases, so with the exception of the first experiment (Table 1 and Fig. 4), only the results with $b = 2$ are presented. Only depths of up to 10 were used, because observations at greater depths yield no additional insight.

Only relative values of σ_v, m and σ_e are important, so σ_v can be fixed to 1. In Table 1 and Fig. 4, results with normal distribution of nodes' values, $m = \infty$ and $\sigma_e = 0.2$ are presented. The parameters are chosen rather arbitrarily, but they will be examined later on: the value of each parameter will be varied while the others will be kept unchanged. The results are averaged over 10,000 game trees for $b = 2$ and 2,500 game trees for $b = 5$. For each tree, there are 10 repetitions with randomly generated noisy values. This will be the case in all experiments in the paper with the exception of Section 6.2. Average position error is defined as the sum of absolute differences between the true and the heuristic value of the root divided by the number of trees. Average move error is defined as the number of trees where a non-optimal move is chosen at the root divided by the number of all trees. Both position and move error at the root with respect to the depth of search are given in Table 1 and Fig. 4. Move error is more important because the choice

Table 1
Position and move error with the initial parameter settings

d	$b = 2$		$b = 5$	
	Position error	Move error	Position error	Move error
0	0.1599	–	0.1586	–
1	0.1588	0.0361	0.1545	0.1008
2	0.1549	0.0356	0.1457	0.0959
3	0.1521	0.0350	0.1403	0.0938
4	0.1501	0.0339	0.1344	0.0908
5	0.1478	0.0340	0.1283	0.0852
6	0.1460	0.0335	0.1242	0.0809
7	0.1438	0.0330	0.1177	0.0789
8	0.1415	0.0325	0.1138	0.0764
9	0.1387	0.0315	0.1108	0.0749
10	0.1361	0.0314	0.1051	0.0746

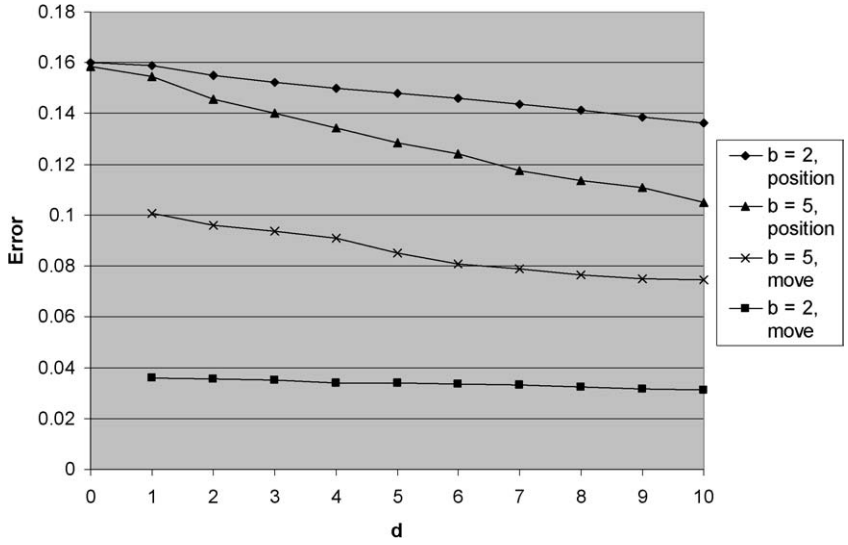

Fig. 4. Position and move error with the initial parameter settings.

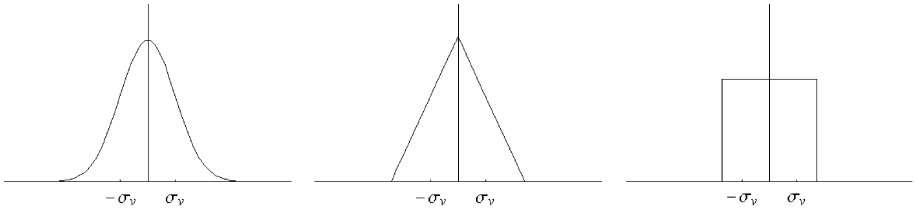

Fig. 5. Types of distribution of nodes' auxiliary values around the auxiliary value of their parent.

Table 2
Move error with different distributions of nodes' auxiliary values

d	Normal	Triangular	Uniform
1	0.0361	0.0366	0.0359
2	0.0356	0.0355	0.0352
3	0.0350	0.0351	0.0350
4	0.0339	0.0347	0.0345
5	0.0340	0.0338	0.0338
6	0.0335	0.0340	0.0339
7	0.0330	0.0332	0.0336
8	0.0325	0.0332	0.0331
9	0.0315	0.0317	0.0326
10	0.0314	0.0315	0.0314

of move is the purpose of minimax. Also, the behavior of both types of error in the experiments was similar, so only move error will be presented in the rest of the section.

As can be seen in Table 1 and Fig. 4, increased depth of search is generally beneficial.

Table 2 and Fig. 6 show how the choice of *the distribution of nodes' auxiliary values* affects move error at the root with respect to the depth of search. Normal, triangular and uniform distributions were tested to investigate the effect of varying bias towards the mean—they are shown in Fig. 5. The other parameters remain unchanged: $b = 2$, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 2 and Fig. 6, the choice of the distribution has very little effect on the behavior of move error. Apparently it matters only that nodes' values are similar to the value of their parent, not the exact shape of the similarity.

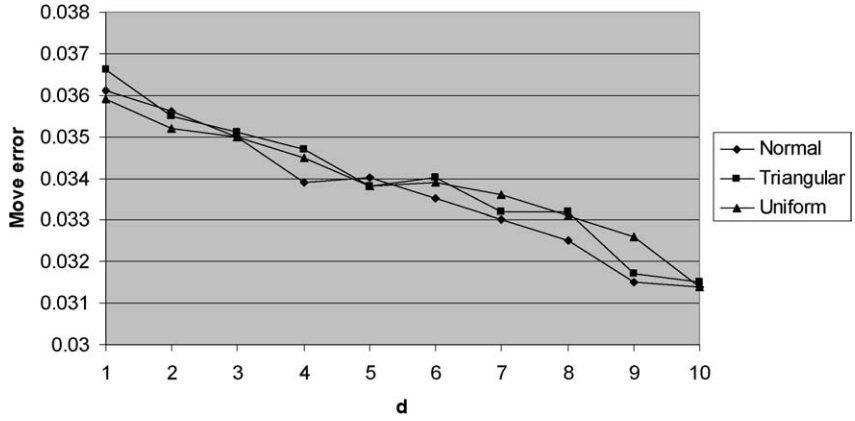

Fig. 6. Move error with different distributions of nodes' auxiliary values.

Table 3
Move error with different values of m

d	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=\infty$
1	0.2040	0.0820	0.0500	0.0383	0.0371	0.0361
2	0.1995	0.0791	0.0482	0.0386	0.0371	0.0356
3	0.1952	0.0779	0.0478	0.0388	0.0365	0.0350
4	0.1928	0.0761	0.0472	0.0367	0.0365	0.0339
5	0.1886	0.0748	0.0458	0.0369	0.0357	0.0340
6	0.1861	0.0742	0.0462	0.0364	0.0352	0.0335
7	0.1837	0.0725	0.0436	0.0351	0.0343	0.0330
8	0.1797	0.0691	0.0437	0.0348	0.0336	0.0325
9	0.1776	0.0704	0.0440	0.0352	0.0334	0.0315
10	0.1766	0.0677	0.0430	0.0342	0.0327	0.0314

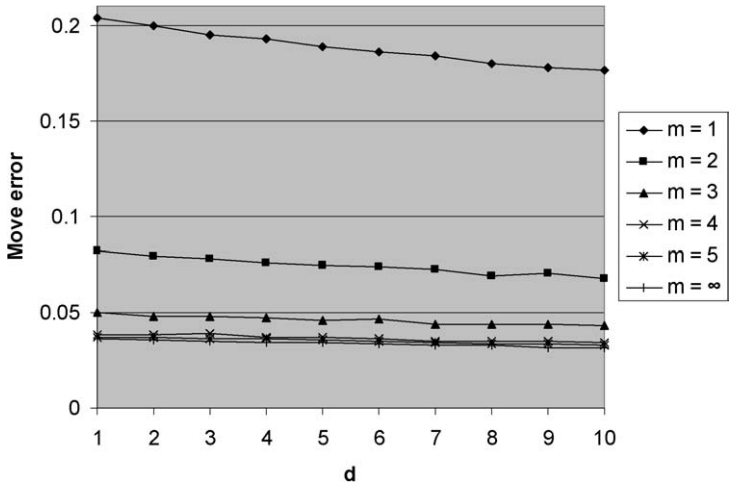

Fig. 7. Move error with different values of m.

The initial setting of $m=\infty$ means that nodes' values are unconstrained, i.e. that however good or bad a position is, it can always get better or worse respectively. In real games, position values are not unbounded: there is checkmate in chess, maximum or minimum number of pieces in Othello etc. In our model, the bound is implemented by simply setting the nodes' values that fall outside the $[-m, m]$ interval to $-m$ or m. Table 3 and Fig. 7 show how *the interval within which the nodes' values must lie* affects move error at the root with respect to the depth of search. The other

Table 4
Move error with different values of σ_e

d	$\sigma_e = 1$	$\sigma_e = 0.5$	$\sigma_e = 0.2$	$\sigma_e = 0.1$
1	0.1682	0.0914	0.0361	0.0191
2	0.1619	0.0873	0.0356	0.0191
3	0.1532	0.0847	0.0350	0.0191
4	0.1455	0.0819	0.0339	0.0185
5	0.1373	0.0788	0.0340	0.0186
6	0.1308	0.0756	0.0335	0.0185
7	0.1224	0.0735	0.0330	0.0187
8	0.1172	0.0723	0.0325	0.0182
9	0.1095	0.0685	0.0315	0.0184
10	0.1019	0.0661	0.0314	0.0180

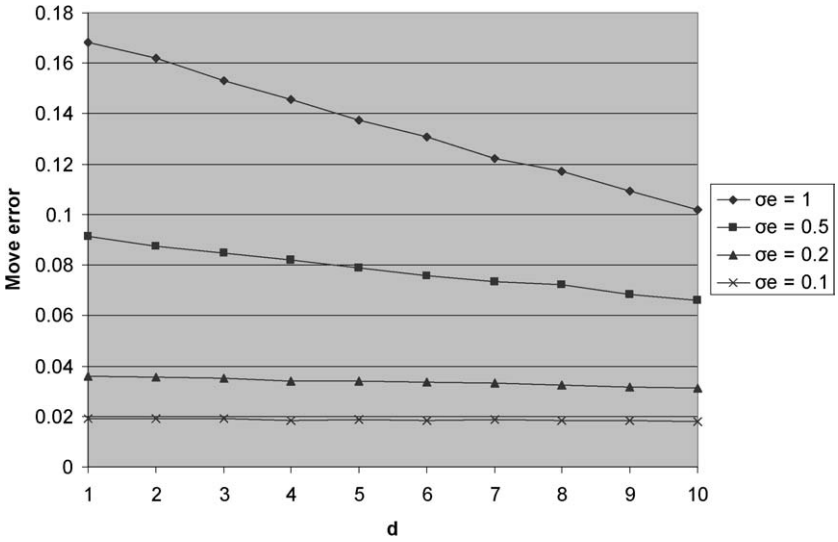

Fig. 8. Move error with different values of σ_e.

parameters remain unchanged: $b = 2$, distribution of nodes' values (when the values are unconstrained) is normal and $\sigma_e = 0.2$.

As can be seen in Table 3 and Fig. 7, the pathology is not present regardless of the value of m. However, m affects the magnitude of move error: when m is small, move error is large because the nodes' values are relatively closer to each other, which makes position error more likely to change their order. When m increases to the relatively small value of 4, results are very similar to those with $m = \infty$, and when it increases to 5, they are virtually identical.

Table 4 and Fig. 8 show how *standard deviation of static position error* affects move error at the root with respect to the depth of search. The other parameters remain unchanged: $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 4 and Fig. 8, increased depth of search is beneficial for all σ_e. The benefit is less pronounced when σ_e is small, but this is to be expected since game tree search would not be beneficial at all if a perfect evaluation function were available.

We can conclude that in our model, minimax is not pathological for a wide range of parameter settings.

4. Mathematical explanation

In this section we attempt a mathematical explanation of the experimental observations in Section 3. For simplicity of the analysis we will assume that the difference between the true values of sibling nodes is always 1, which leads

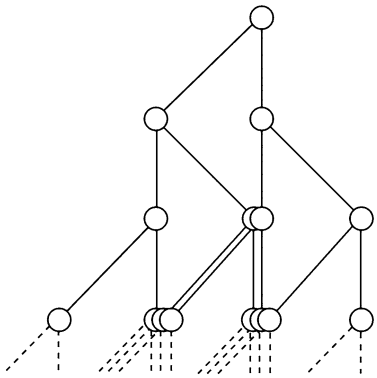

Fig. 9. Model used in the mathematical explanation.

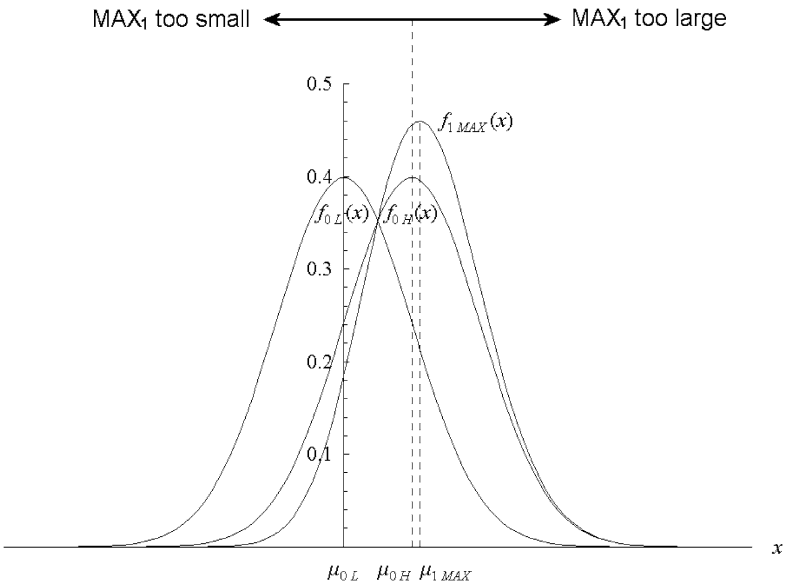

Fig. 10. Probability density functions of heuristic values of a pair of sibling nodes at ply 0 and their parent.

to the model sketched in Fig. 9. Hopefully this simplification does not affect qualitative conclusions of the analysis. Reasons why this should be the case are discussed later in this section. Also, only $b = 2$ is considered.

A node at ply $i + 1$ has two descendants whose heuristic values are random variables L_i (lower value) and H_i (higher value) with means $\mu_{i\,L}$ and $\mu_{i\,H}$. Due to normally distributed error, static heuristic values of sibling nodes are distributed normally with means μ_{0L} and μ_{0H} and standard deviation σ_e. Their means are also the nodes' true values. Their probability density functions are given in Eqs. (4).

$$f_{0L}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu_{0L})^2}{2\sigma_e^2}}, \qquad f_{0H}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu_{0H})^2}{2\sigma_e^2}}. \tag{4}$$

At a max ply $i + 1$, the value of a node is a random variable $MAX_{i+1} = \max(L_i, H_i)$ and its probability density function is calculated according to Eq. (5).

$$f_{i+1MAX}(x) = f_{iH}(x)P(L < x) + f_{iL}(x)P(H < x) = f_{iH}(x) \int_{-\infty}^{x} f_{iL}(l)\,\mathrm{d}l + f_{iL}(x) \int_{-\infty}^{x} f_{iH}(h)\,\mathrm{d}h. \tag{5}$$

Probability density functions of L_0, H_0 and MAX_1 are shown in Fig. 10. Unlike L_0 and H_0, MAX_1 is not distributed normally. Its probability density function was obtained by numerical integration.

As can be seen in Fig. 10, the curve of the parent is narrower than the curves of its descendents, meaning that the variance of the parent is smaller. Because the difference between the true values of sibling nodes is constant, the difference between the means of the heuristic values of sibling nodes is constant as well. Therefore a smaller variance means a smaller probability that the lower-valued sibling becomes, due to position error, the higher-valued sibling, i.e. a smaller move error. If the mean of the parent were equal to the mean of the higher descendant, a smaller variance would also mean a smaller position error. Since the mean of the parent is somewhat larger, this is not necessarily the case and will be examined later on.

Let us first explain what causes the reduction of variance. If there were no interaction between the descendants, f_{i+1MAX} would be equal to f_{iH}. Since this is not true, two cases must be considered. When MAX_{i+1} is too large ($MAX_{i+1} > \mu_{iH}$), the most likely reason is that H_i is too large ($H_i > \mu_{iH}$). The less likely reason is that L_i is much too large ($L_i > \mu_{iH}$). This is marked in Fig. 10 on the top. When MAX_{i+1} is too small ($MAX_{i+1} < \mu_{iH}$), this is caused by H_i being too small ($H_i < \mu_{iH}$), but it can be compensated by L_i being larger than H_i ($H_i < L_i < \mu_{iH}$). The corrective effect of L_i in the latter case is greater than the disturbing effect in the former case, so the impact of L_i it is more likely to be beneficial than harmful.

At a min ply, the probability density function is calculated according to Eq. (6).

$$f_{i+1MIN}(x) = f_{iL}(x)P(H > x) + f_{iH}(x)P(L > x) = f_{iL}(x)\int_x^\infty f_{iH}(h)\,\mathrm{d}h + f_{iH}(x)\int_x^\infty f_{iL}(l)\,\mathrm{d}l. \tag{6}$$

Probability density functions at ply 0 are given by Eqs. (4), so probability density function at any ply can be calculated by repeatedly using Eqs. (5) and (6). This is shown in Fig. 11; $\mu_{iH} - \mu_{iL} = 1$ for all i.

As can be seen in Fig. 11, the higher a ply, the narrower the curve of the probability density function of a node's heuristic backed-up value. This means that the variance of the heuristic values and consequently move error decrease with the depth of search. True values in all the cases in Fig. 11 are 0, so it can also be seen that heuristic values exhibit a slight positive bias, most of which is accrued at the lowest ply of search. This is consistent with observations by Sadikov et al. [17].

Let ME_{i+1} be move error at ply $i+1$. An erroneous move at ply $i+1$ is chosen when the values of a pair of sibling nodes at ply i are switched, i.e. $L_i > H_i$, so ME_{i+1} is calculated according to Eq. (7).

$$ME_{i+1} = P(L_i > H_i) = \int_{-\infty}^{\infty} f_{iH}(h)\left(\int_h^\infty f_{iL}(l)\,\mathrm{d}l\right)\mathrm{d}h. \tag{7}$$

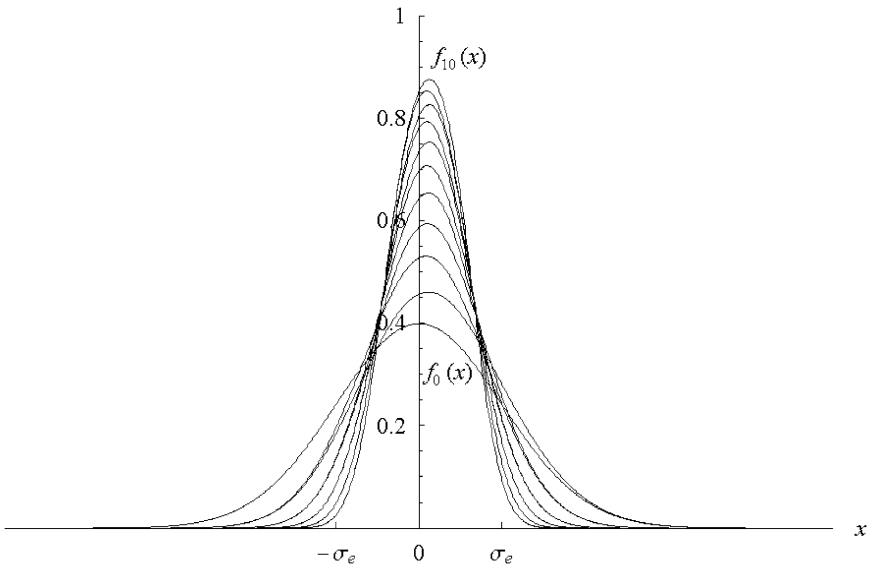

Fig. 11. Probability density functions of the values of nodes at plies 0 through 10.

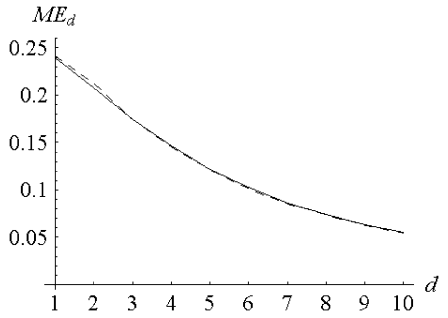

Fig. 12. Move error at the root as a function of the depth of search (dashed line shows Monte Carlo experimental results).

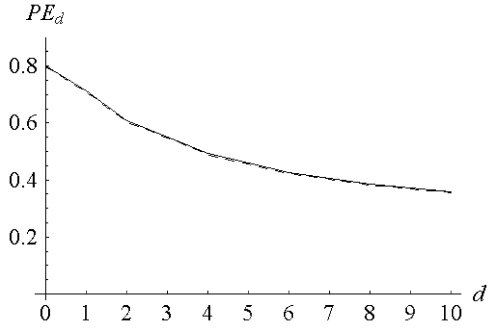

Fig. 13. Position error at the root as a function of the depth of search (dashed line shows Monte Carlo experimental results).

How move error changes with increasing depth of search is shown in Fig. 12; ME_d is move error at the root when the depth of search is d, $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$. The solid line results from calculations according to Eq. (7). The dashed line shows the results of Monte Carlo experiments using the model of this section (the difference between the true values of sibling nodes always 1). They serve to verify the correspondence between the mathematical analysis and random experiments.

As can be seen in Fig. 12, increased depth of search reduces move error at the root.

Let t_i be the true value of a node at ply i and $f_{i\,E}$ probability density function of its heuristic (erroneous) value E_i. Position error PE_i is the average absolute difference between t_i and E_i and is calculated according to Eq. (8).

$$PE_i = \int_{-\infty}^{\infty} |t_i - x| f_{i\,E}(x)\,\mathrm{d}x. \tag{8}$$

How position error changes with increasing depth of search is shown in Fig. 13; PE_d is position error at the root when the depth of search is d, $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$. The solid line results from calculations according to Eq. (8). The dashed line again results from Monte Carlo experiments.

As can be seen in Fig. 13, increased depth of search reduces position error at the root, which is in accordance with the experimental results in Section 3. This means that the bias observed in Fig. 10 and Fig. 11 is not large enough to cause position error to behave pathologically, probably because alternating max and min plies cause alternating positive and negative bias. To verify this hypothesis, behavior of move and position error in a tree with only max plies was investigated. The results are shown in Fig. 14; $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$.

As can be seen in Fig. 14, move error is still not pathological, while position error behaves pathologically for $d > 3$. This result is similar to what Sadikov et al. [17] observed in their experiments on king and rook versus king chess endgame, although their trees consisted of both max and min plies. The reason for this discrepancy is not clear.

In the analysis presented in this section, a constant difference between the true values of sibling nodes is assumed. However, in game trees of real games as well as game trees used in the experiments in Section 3, the difference between the true values of sibling nodes is not constant. The simplification in the analysis is not as problematic as it may appear, though. This is because we investigate the search of a *single* game tree to different depths, therefore no

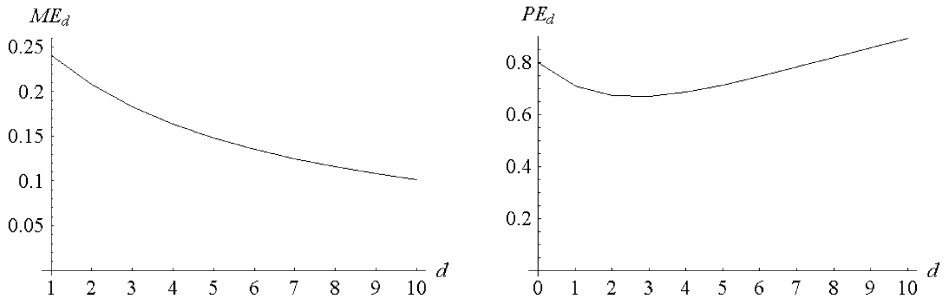

Fig. 14. Move error (left) and position error (right) at the root as a function of the depth of search in a tree with only max plies.

matter how the difference between the true values of sibling nodes varies within the tree, the variation is the same in searches to all depths. Hence the difference affects the error in a similar fashion in all searches and consequently does not affect the pathology.

5. Verification in a chess program

Whether the results of Sections 3 and 4 have bearing on real games can be verified by comparing the model proposed in Section 3 to real games. However, it is difficult to obtain true values of game-tree nodes of a sufficiently complex game. What can and often is done is to use values returned by a game-playing program. The game chosen is chess and the program is Crafty [6]. With ELO rating of 2,616, it is the highest-rated open-source chess program according to SSDF [20]. Being open-source is a requirement because the program needed to be modified for the experiment.

The comparison between the model and Crafty should be based on the values of the leaves. Distributions of leaf values are difficult to compare in practice, though, so the comparison is performed on the relations between the static values of nodes and their descendants in Crafty and auxiliary values of nodes and their descendants in our model. The values of the leaves are, after all, determined by this relation in both cases.

One tenth of 4.5 million positions visited in the course of a game played by Crafty were randomly chosen. All the positions generated during game-tree search were considered, not only positions actually occurring in the game. The differences between the static value of each node and the static values of all of its descendants were computed. Determining static values did not involve quiescence search. The results were discretized and the number of cases in each interval counted. In Fig. 15, the results are shown as the number of cases where the difference lies within an interval; the lower and upper 1% of the cases are omitted, otherwise the interesting interval could not be seen clearly since the extremes are -22.78 and 20.58.

As can be seen in Fig. 15, most nodes' static values lie close to the static value of their parent. Their distribution resembles normal. This relation corresponds reasonably well to the relation between the auxiliary values of nodes in our model, where the default distribution is also normal. Since Section 3 shows that the exact type of distribution is not crucial anyway, we feel that the results from Crafty can be considered a confirmation of the relevance of the model presented in Section 3 to real games.

6. When does pathology occur

If a two-value model can be pathological, as shown in Section 2, should not a real-value model exhibit pathology under certain conditions as well? They both aspire to model the error behavior of minimax in game-playing programs. The presence of the pathology in one and the complete absence in another would be an indication that one of them lacks some fundamental property. In this section we look into this apparent contradiction between the two models and search for conditions under which pathology would appear in a real-value model.

There are three main differences between the two-value model described in Section 2 and the real-value model described in Section 3:

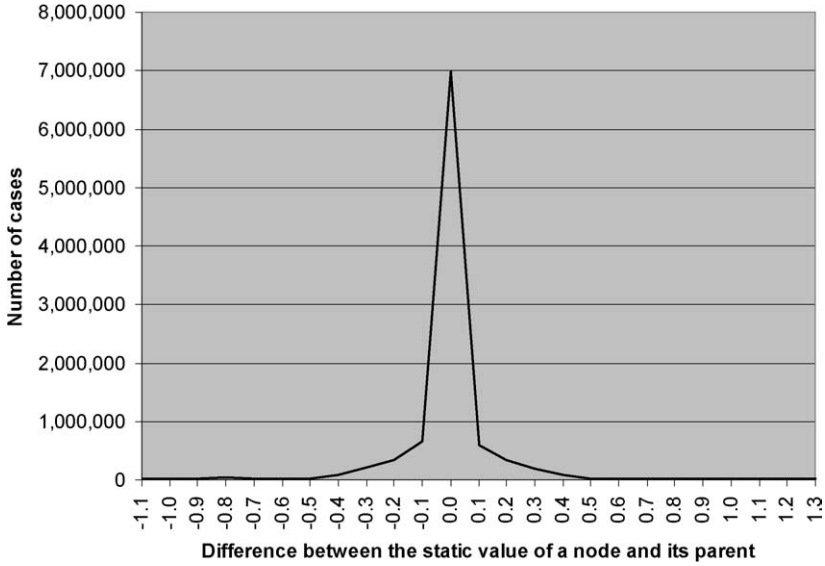

Fig. 15. Distribution of the differences between the static value of a node and its parent.

1. In the two-value model, the value of each leaf of the game tree is independent of all other leaves' values; in the real-value model, the leaves' values are not independent due to the nodes' auxiliary values being distributed around the auxiliary value of their parent.
2. In the two-value model, the error is measured as the probability of mistaking a loss for a win or vice versa and is set to the same value at the lowest ply of search regardless of the depth of search; in the real-value model, the error is measured as position error whose probability distribution is always the same at the lowest ply of search regardless of the depth of search, and move error for which this is not guaranteed as we show later in this section.
3. And of course one model uses two values and the other real values.

6.1. Independent-value model

If the first difference is removed, a game tree is constructed by simply setting the leaves' values independently of each other instead of using auxiliary intermediate values; the rest of the procedure follows the description in Section 3. We call such a model *independent-value* model as opposed to *dependent-value* model, which is the one described in Section 3. Table 5 and Fig. 16 show position and move error at the root with respect to the depth of search in the independent-value model; $b = 2$, distribution of leaves' values (this time around 0 instead of their parents' auxiliary values) is normal, $m = \infty$ and $\sigma_e = 0.2$.

Table 5
Position and move error at the root in the independent-value model

d	Position error	Move error
0	0.1598	–
1	0.1472	0.2952
2	0.1233	0.2916
3	0.1153	0.2863
4	0.1041	0.2824
5	0.0997	0.2744
6	0.0944	0.2720
7	0.0918	0.2686
8	0.0895	0.2635
9	0.0876	0.2578
10	0.0862	0.2577

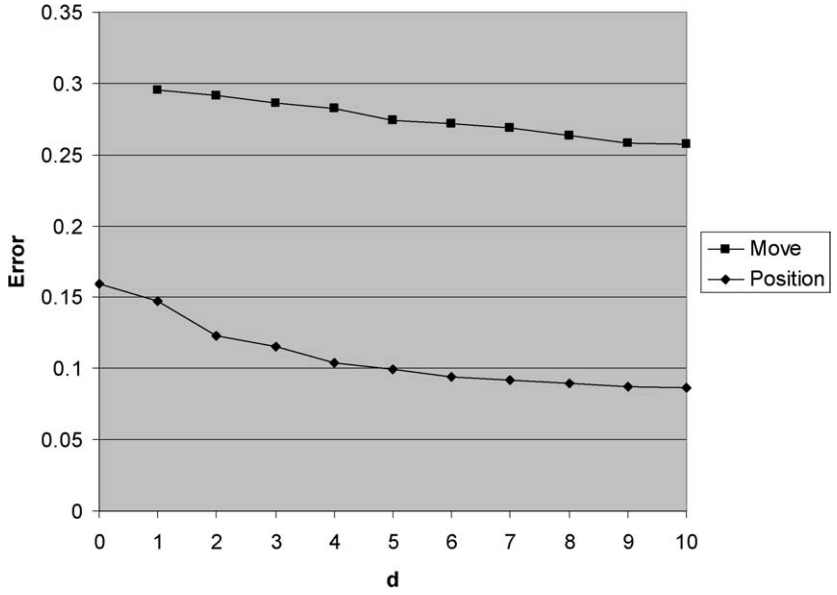

Fig. 16. Position and move error at the root in the independent-value model.

As can be seen in Table 5 and Fig. 16, the independent-value model with the chosen set of parameters is not pathological. Experiments with several different sets of parameters also yielded no pathology. Considering that models with independent values such as the one described in Section 2 are known to be pathological, this result is on one hand somewhat surprising. On the other hand, though, the explanation of Section 4 also applies to the independent-value model. Apparently two-value models are sufficiently different from real-value models that the pathology in one does not imply the pathology in the other, even if both are independent-valued.

6.2. Granularity

In our models, we used real numbers as both true and heuristic values, so in principle there could be an infinite number of them. Since digital computers do not use real values in the true mathematical sense, in our simulations these real numbers were approximated by high-density floating-point numbers. But we know that game-playing programs usually use as heuristic values a much smaller set of discrete values, most often represented by a range of integers. Therefore real values are not completely realistic for modeling typical game playing programs. We will now investigate whether this difference affects our main findings.

To see whether the issue of granularity affects the pathology, the number of possible heuristic values, or grains, in our models was restricted. The granularity of true values was set to be equal to the granularity of heuristic values, otherwise errors would occur even with the best possible evaluation function, simply because it would be too coarse-grained to differentiate between similar true values. The effect of granularity turned out to be strongly tied to the branching factor. Table 6 and Fig. 17 show the number of grains needed to avoid the pathology with respect to the branching factor in the dependent- and the independent-value model; the distribution of nodes' values in the dependent-value model is normal and in the independent-value model uniform, $d_{\max} = 6$, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 6 and Fig. 17, the dependent-value model is quite robust in the sense that even with a low granularity, it is not pathological. Evaluation functions of modern chess programs have ranges of several thousand values, so these programs are not likely to be pathological due to low granularity. The independent-value model is much more sensitive to the granularity: the granularity required to avoid the pathology $g(b)$ seems to be exponential in the branching factor. Eq. (9) gives the least-square-error approximation of the experimental data.

$$g(b) = 7.5084 \times 1.3208^b - 0.5169. \tag{9}$$

Table 6
Required granularity to avoid the pathology in the dependent- and the independent-value model

b	Dependent-value model	Independent-value model
2	10	13
3	15	17
4	19	22
5	19	29
6	19	39
7	19	52
8	19	71
9	19	90
10	19	121

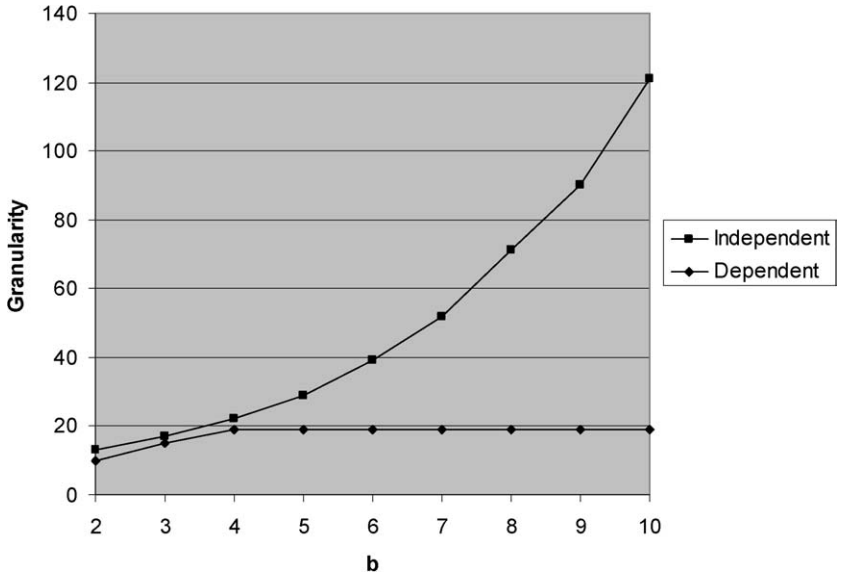

Fig. 17. Required granularity to avoid the pathology in the dependent- and the independent-value model.

The average branching factor of chess is estimated to be between 30 and 40. $g(30) = 31{,}670$ and $g(40) = 511{,}706$, so a game with a chess-like branching factor, typical chess program granularity and independent leaf values would probably be pathological.

This result is similar to what Nau [12] predicted: if the heuristic evaluation function returns its minimum and maximum values with a non-zero probability and the branching factor is large enough, the pathology will ensue. With the introduction of the granularity, every value in our model is indeed returned with a non-zero probability. Scheucher and Kaindl [18] also investigated the issue of granularity. Their conclusion was that it has no effect as long as it is not too small for the evaluation function to express all the knowledge of the game. In our experiments this was never the case, because the granularity of the heuristic values was equal to the granularity of the true values, but the pathology still appeared. Scheucher and Kaindl's did not encounter it because their work was based on the assumption that a simple chess evaluation function requires 84 distinct values, and since their model was dependent-valued, our experiments suggest that much fewer distinct values are required for the pathology to appear.

6.3. Constant static move error

In the experiments up to this point, whenever searches to different depths were performed, static position error was always constant. This is a reasonable assumption, since position error directly models the fallibility of the heuristic

Table 7
Static move error in the dependent- and the independent-value model

d	Dependent-value model	Independent-value model
1	0.0369	0.2952
2	0.0373	0.2602
3	0.0375	0.2249
4	0.0376	0.1946
5	0.0384	0.1636
6	0.0396	0.1375
7	0.0408	0.1130
8	0.0432	0.0936
9	0.0486	0.0760
10	0.0628	0.0628

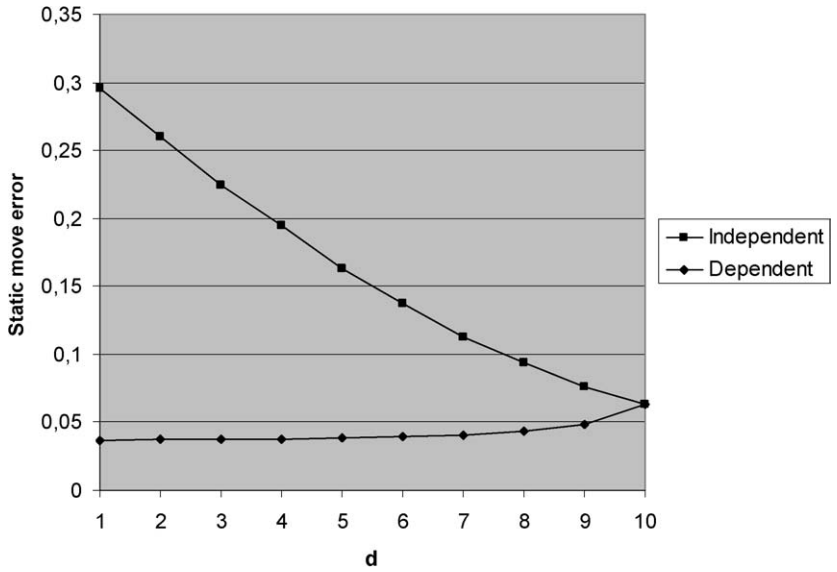

Fig. 18. Static move error in the dependent- and the independent-value model.

evaluation function and it is generally accepted that the heuristic evaluation function performs equally well at all plies within the horizon of a single search. However, it is also not an unreasonable assumption that move error at the lowest ply of search, or static move error, should be constant.

Table 7 and Fig. 18 show static move error with respect to the depth of search in the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$ and $\sigma_e = 0.2$.

Perhaps surprisingly, as can be seen in Table 7 and Fig. 18, static move error is not constant. In the dependent-value model, it increases with the depth of search, while in the independent-value model, it decreases with the depth of search. Differences in static move error when searching to different depths are particularly large in the independent-value model.

Why static move error changes with the depth of search can be explained by observing the average difference between the true values of sibling nodes at successive plies. Table 8 and Fig. 19 show the results for the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$, $\sigma_e = 0.2$ and $d_{\max} = 10$.

As can be seen in Table 8 and Fig. 19, the behavior of the average difference between the true values of sibling nodes is the opposite of the behavior of static move error (shown in Table 7 and Fig. 18). This is not surprising: it has already been observed that small differences between backed-up values of sibling nodes cause large move errors and vice versa.

Table 8
The average difference between the true values of sibling nodes at successive plies in a game tree with $d_{\max} = 10$ in the dependent- and the independent-value model

Ply	Dependent-value model	Independent-value model
9	1.9313	0.1711
8	1.9191	0.2132
7	1.9040	0.2622
6	1.8925	0.3232
5	1.8763	0.4016
4	1.8231	0.4913
3	1.7580	0.6108
2	1.6464	0.7467
1	1.4638	0.9297
0	1.1286	1.1286

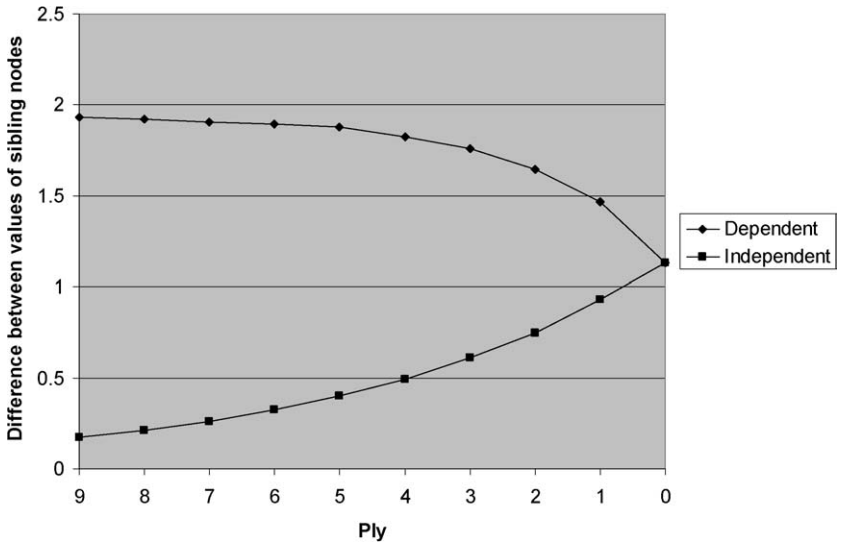

Fig. 19. The average difference between the true values of sibling nodes at successive plies in a game tree with $d_{\max} = 10$ in the dependent- and the independent-value model.

Why true backed-up values of sibling nodes in the dependent-value model are statistically further apart at higher plies can be explained by analyzing the relation between the auxiliary values of a pair of sibling nodes and the corresponding true backed-up values. Consider a pair of non-leaf sibling nodes in a dependent-value game tree. Let a_L and a_H be the auxiliary values of these sibling nodes, so that $a_L \leqslant a_H$. Let L^* and H^* be the corresponding true backed-up values; they are random variables that acquire values from the populations of the possible subtrees rooted in the two sibling nodes. The means of L^* and H^* are μ_{L^*} and μ_{H^*}. Both subtrees rooted in the two sibling nodes are constructed by the same random mechanism starting with the root values a_L and a_H, therefore $\mu_{H^*} - \mu_{L^*} = a_H - a_L$. Consider the difference $D = H^* - L^*$. The expected value of D, μ_D, is equal to $\mu_{H^*} - \mu_{L^*} = a_H - a_L$. This follows from the fact that the expected value of the difference between two random variables is equal to the difference between the expected values of the two variables. This property is demonstrated in Eq. (10), where X and Y are two independent random variables whose probability density functions are f_X and f_Y.

$$\mu_{X-Y} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - y) f_X(x) f_Y(y) \, \mathrm{d}x \, \mathrm{d}y$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x f_X(x) f_Y(y)\,\mathrm{d}x\,\mathrm{d}y - \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y f_X(x) f_Y(y)\,\mathrm{d}x\,\mathrm{d}y$$

$$= \int_{-\infty}^{\infty} x f_X(x)\left(\int_{-\infty}^{\infty} f_Y(y)\,\mathrm{d}y\right)\mathrm{d}x - \int_{-\infty}^{\infty} y f_Y(y)\left(\int_{-\infty}^{\infty} f_X(x)\,\mathrm{d}x\right)\mathrm{d}y$$

$$= \int_{-\infty}^{\infty} x f_X(x) 1\,\mathrm{d}x - \int_{-\infty}^{\infty} y f_Y(y) 1\,\mathrm{d}y = \mu_X - \mu_Y. \tag{10}$$

Regarding the probability of move error, the parameter that matters is the absolute difference between the true values of the two sibling nodes: $|D| = |H^* - L^*|$. The expected value of $|D|$ is greater than μ_D which is easily seen from Eq. (11).

$$\mu_D = \int_{-\infty}^{\infty} x f_D(x)\,\mathrm{d}x \leqslant \int_{-\infty}^{\infty} |x| f_D(x)\,\mathrm{d}x = \mu_{|D|}. \tag{11}$$

So for auxiliary values of any pair of sibling nodes, the corresponding true backed-up values are further apart on average. This is also illustrated in Fig. 20.

Auxiliary values at all plies are distributed equally, so the average difference between the auxiliary values of a pair of sibling nodes is the same for all plies. Auxiliary values at ply 0 are also true values, therefore backed-up true values at ply 1 and higher are further apart on average than true values at ply 0. What makes $\mu_{|D|}$ larger at each successive ply is that the enlarging effect is cumulative.

Why true backed-up values of sibling nodes in the independent-value model are closer to each other at higher plies can be explained as follows. The values of the leaves are distributed within an interval. Applying maximum to groups of them results in values concentrated in a smaller interval. Applying minimum to groups of values from the new interval results in values concentrated in an even smaller interval, etc. This is illustrated in Fig. 21.

The effect of the average difference between the values of sibling nodes on static move error can be compensated by adjusting static position error to achieve constant static move error: in the dependent-value model, it has to be increased in shallower searches, while in the independent-value model, it has to be decreased in shallower searches. Table 9 and Fig. 22 show move error at the root with respect to the depth of search in the dependent- and the independent-value model with constant static move error; $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 9 and Fig. 22, constant static move error strengthens the benefits of minimax in the dependent-value model, but produces the pathology in the independent-value model. This is consistent with previous

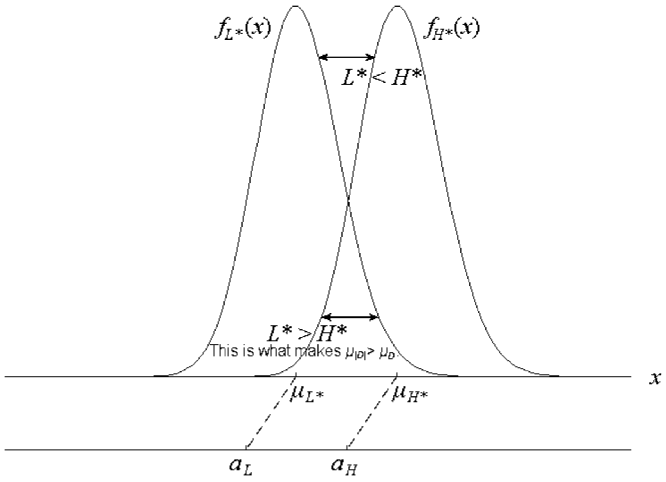

Fig. 20. Nodes' true backed-up values higher up in a tree are further apart in the dependent-value model.

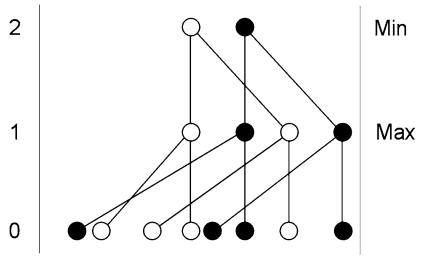

Fig. 21. Explanation for nodes' backed-up values higher up in a tree being closer to each other in the independent-value model.

Table 9
Move error at the root in the dependent- and the independent-value model with constant static move error

d	Dependent-value model	Independent-value model
1	0.0628	0.0628
2	0.0614	0.0734
3	0.0605	0.0914
4	0.0599	0.1069
5	0.0557	0.1303
6	0.0523	0.1501
7	0.0494	0.1756
8	0.0476	0.2064
9	0.0418	0.2317
10	0.0314	0.2577

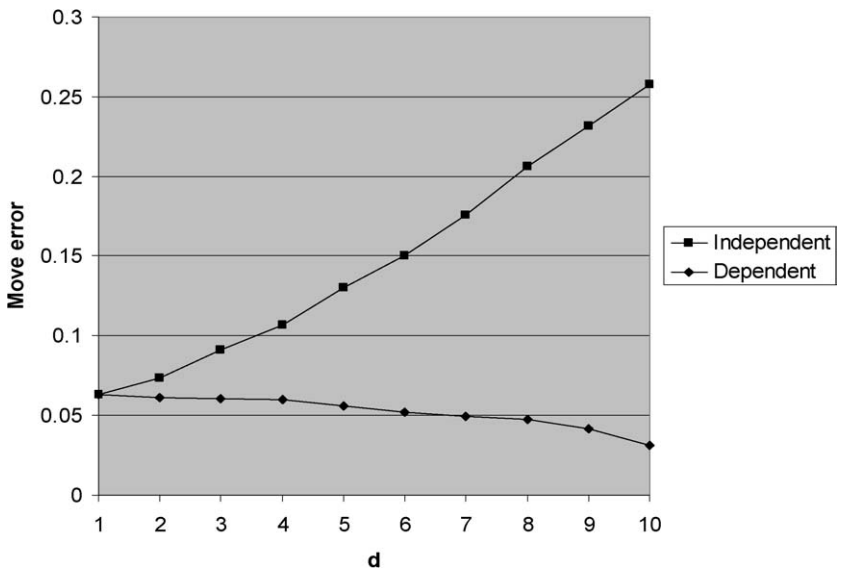

Fig. 22. Move error at the root in the dependent- and the independent-value model with constant static move error.

research by Beal [3], Bratko and Gams [4], and Nau [11,12], concerned with two-value models. But it should be noted that in two-value models, error is defined somewhat differently from move error in our real-value model.

Table 10
Two-value error at the root and at the lowest ply of search in the dependent- and the independent-value model

d	Dependent-value model		Independent-value model	
	Root	Static	Root	Static
0	0.0486	0.0486	0.2072	0.2072
1	0.0469	0.0321	0.1987	0.1893
2	0.0470	0.0328	0.1770	0.1873
3	0.0466	0.0264	0.1663	0.1655
4	0.0465	0.0268	0.1551	0.1563
5	0.0453	0.0227	0.1465	0.1316
6	0.0445	0.0232	0.1433	0.1203
7	0.0435	0.0210	0.1411	0.0955
8	0.0433	0.0211	0.1364	0.0865
9	0.0422	0.0199	0.1293	0.0700
10	0.0414	0.0202	0.1268	0.0592

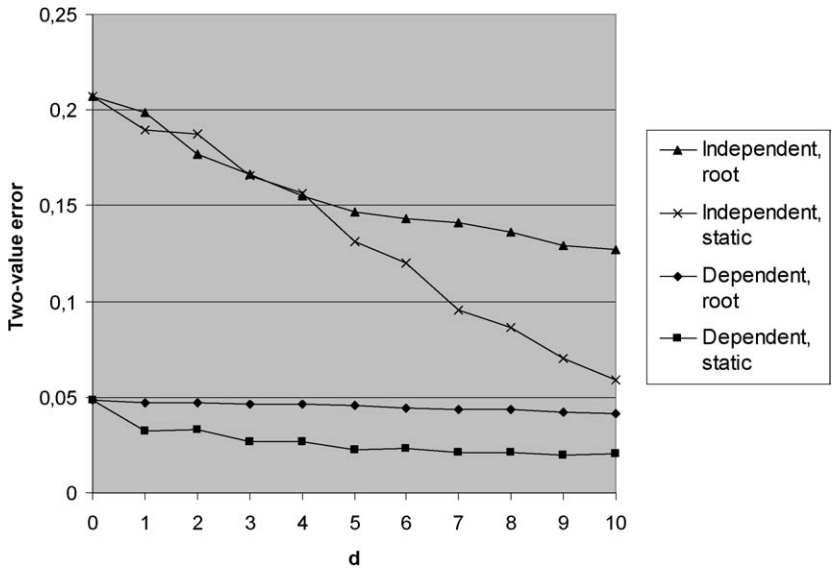

Fig. 23. Two-value error at the root and static two-value error in the dependent- and the independent-value model.

6.4. Conversion to two values

A real-value model can be reduced to a two-value model by converting the real-number values to losses and wins. To do this, a threshold must be established: the values below it are considered losses and the values above it wins. If the probability of a loss at the root is to be 0.5, the threshold should be the median backed-up real value at the root.

Table 10 and Fig. 23 show two-value error (the probability of an incorrect evaluation of a position) at the root and two-value error at the lowest ply of search, or static two-value error, with respect to the depth of search in the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 10 and Fig. 23, neither the dependent- nor the independent-value model is pathological in terms of two-value error. Note that even though the error at the root is mostly larger than static error in both models, this does not mean that the models are pathological: it merely means that positions at higher plies are more difficult to evaluate as losses or wins than those at lower plies. If static error were constant, the increase of error through minimaxing would of course lead to the pathology. But as can be observed in Table 10 and Fig. 23, static two-value error is not constant, which also means that this experiment cannot be compared to the results of previous research outlined in Section 2. In order to make such a comparison possible, static position error must be adjusted so that static two-value error is constant. Table 11 and Fig. 24 show two-value error at the root with respect to the depth of search in

Table 11
Two-value error at the root in the dependent- and the independent-value model with constant static two-value error

d	Dependent-value model	Independent-value model
0	0.0202	0.0592
1	0.0306	0.0624
2	0.0277	0.0609
3	0.0354	0.0688
4	0.0349	0.0687
5	0.0391	0.0818
6	0.0390	0.0851
7	0.0436	0.0991
8	0.0402	0.1069
9	0.0438	0.1212
10	0.0414	0.1268

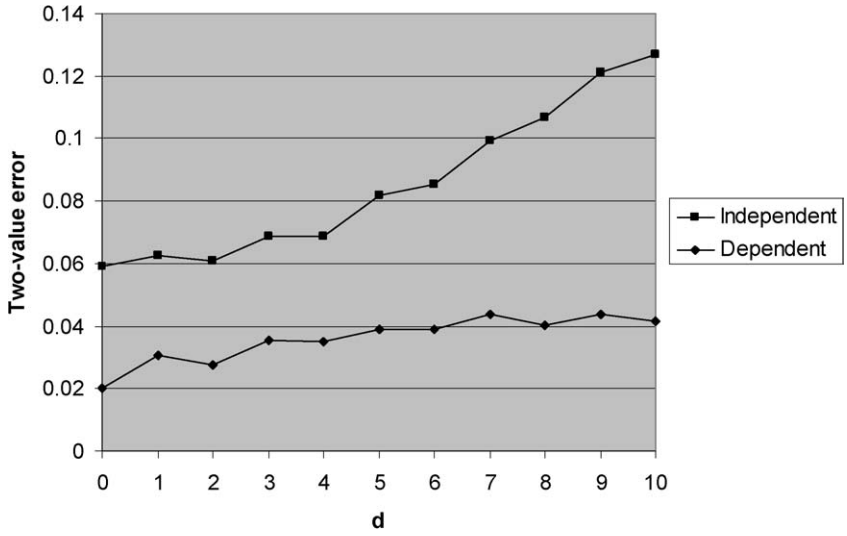

Fig. 24. Two-value error at the root of the dependent- and the independent-value model with constant static two-value error.

the dependent- and the independent-value model after this adjustment; $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 11 and Fig. 24, constant static two-value error produces pathology both in the dependent- and the independent-value model; pathology is stronger in the independent-value model. This result could be interpreted as an indication that the pathology may exist even in the dependent-value model. However, the constant static two-value error assumption in this experiment is not appropriate. A constant static two-value error is indeed assumed in the two-value model in Section 2, but in a model where true values are real numbers, static two-value error should decrease with depth. The reason is that at the lowest ply of shallower searches, nodes' values are on average closer to the threshold separating losses from wins, hence two-value error is more likely to occur in such nodes. This is illustrated in Fig. 25; the darker area represents a higher probability of two-value error.

Scheucher and Kaindl [18] investigated the relation between the nodes' heuristic values and two-value error in greater detail. One of the key points of their paper was precisely that static two-value error (called simply error, since they did not consider multiple types of error) should not be constant. If static two-value error in shallower searches should indeed be larger than in deeper searches, this explains why a basic two-value model such as the one described in Section 2 is pathological: it does not model the heuristic error of a real-value game appropriately. This way of modeling error might be suitable for a two-value game, but two (or three) values are not enough for playing real games like chess successfully, as was also demonstrated by Scheucher and Kaindl.

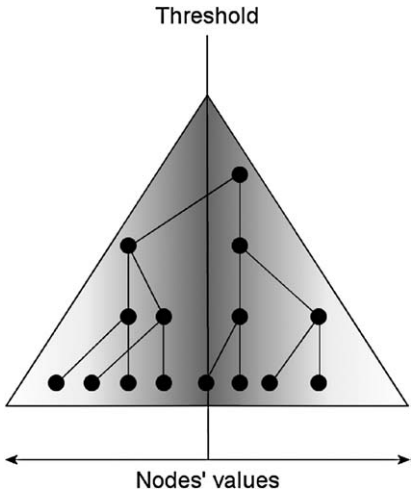

Fig. 25. Why static two-value error should not be constant.

7. Conclusion

In this paper we investigated the question whether minimax behaves pathologically in game trees where the true values of positions are real numbers. Our general conclusion is "no", which is in agreement with the practice of game-playing programs, but different from earlier theoretical analyses by several other authors who studied game trees where the true values of positions could only be losses and wins.

We constructed a minimax model where game tree nodes have real-number values and nodes' values are distributed around the value of their parent. The pathology was not present with any configuration of parameters tested.

A mathematical explanation of why minimax performs better with increased depth of search on game trees such as the ones built according to our model was provided. The reason is that minimum/maximum of pairs of random variables has a smaller variance than the variables themselves. If these random variables represent error, it is thus reduced through minimaxing. The means of the heuristic values have a slight bias with respect to the true values, but with alternating min and max plies, the bias alternates between negative and positive, preventing the heuristic values from moving too far away from the true values. Extending the theoretical analysis to trees with branching larger than two and general distribution of nodes' values around the value of their parent is left for further work.

The assumption that sibling nodes' values are normally distributed around the value of their parent was verified in the Crafty chess program. Static values of sibling nodes in the program turned out to be close to each other and their distribution around their parent's value similar to normal. This can be considered a confirmation that our model corresponds well to the situation in chess playing programs.

Considering that pathology is common in two-value models, we tried to find under what conditions, if any, it appears in real-value models. No pathology in terms of position error was found. By default, static position error was constant in our experiments. If, instead, static move error was set to be constant, our model was still not pathological, but a model with independent leaves' values did behave pathologically. This shows that similarity of nodes close to each other in a game tree can eliminate pathology.

In order to better model game-playing programs, a limited number of discrete values were assigned to positions instead of real values. Here the existence of the pathology depends on the granularity of heuristic values and on the branching factor. The chances of pathology increase with branching factor and decrease with granularity. In the dependent-value model, 19 values were sufficient to avoid the pathology in all the experiments. In these experiments, we could only reach branching factors of up to 10, but the required granularity did not rise after $b = 4$. In the dependent-value model, the required granularity increases fast with the branching factor, so a program for playing a game described by this model would require a very fine-grained evaluation function.

If real-value game trees are converted to two-value game trees, two-value error can be measured. Depending on the assumption regarding static heuristic error, dependent and independent-value models are either both pathological or both non-pathological. One assumption is that static position error (noise applied to the position values) is independent

of the depth of search. The other assumption is that static two-value error (probability of mistaking a lost position for won or vice versa) is independent of the depth of search. We will here refer to these two assumptions as assumption 1 and assumption 2 respectively. Both assumptions seem reasonable, but they generally cannot be both true in a real-value game tree. Assumption 1 does not produce pathology and is appropriate for real-value game trees. The basic two-value model of Beal [2] and others makes assumption 2, which causes pathology. This assumption is, however, not appropriate for real-value game trees. It might be suitable for two-value game trees, but playing most real-life games by two-value evaluations does not seem to be a good idea (as discussed also by Scheucher and Kaindl [18]).

Acknowledgement

This work was supported by the Slovenian Ministry of Science and Education. We would also like to thank Aleksander Sadikov, Mihael Perman and Dana S. Nau for helpful suggestions and discussion.

References

[1] I. Althöfer, Generalized minimax algorithms are no better error correctors than minimax itself, in: D.F. Beal (Ed.), Advances in Computer Chess, vol. 5, North-Holland, Amsterdam, 1989, pp. 265–282.
[2] D.F. Beal, An analysis of minimax, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 2, Edinburgh University Press, Edinburgh, 1980, pp. 103–109.
[3] D.F. Beal, Benefits of minimax search, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 3, Pergamon Press, 1982, pp. 17–24.
[4] I. Bratko, M. Gams, Error analysis of the minimax principle, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 3, Pergamon Press, 1982, pp. 1–15.
[5] S.H. Fuller, J.G. Gaschnig, J.J. Gillogly, An analysis of the alpha-beta pruning algorithm, Technical Report, Carnegie Mellon University, 1973.
[6] R. Hyatt, Home page, http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html, 2004-04-21.
[7] D.E. Knuth, R.W. Moore, An analysis of alpha-beta pruning, Artificial Intelligence 6 (4) (1975) 293–326.
[8] M. Luštrek, Minimax pathology and real-number minimax model, in: B. Zajc, A. Trost (Eds.), Proceedings of the Thirteenth International Electrotechnical and Computer Science Conference, vol. B, Portorož, Slovenia, Slovenian Section IEEE, Ljubljana, Slovenia, 2004, pp. 137–140.
[9] M. Luštrek, M. Gams, Minimaks in napaka pri ocenjavanju položajev, in: M. Bohanec, B. Filipič, M. Gams, D. Trček, B. Likar (Eds.), Proceedings A of the 6th International Multi-Conference Information Society IS 2003, 2003, pp. 89–92.
[10] D.S. Nau, Quality of decision versus depth of search on game trees, Ph.D. thesis, Duke University, 1979.
[11] D.S. Nau, An investigation of the causes of pathology in games, Artificial Intelligence 19 (3) (1982) 257–278.
[12] D.S. Nau, Decision quality as a function of search depth on game trees, J. Assoc. Comput. Mach. 30 (4) (1983) 607–708.
[13] D.S. Nau, Pathology on game trees revisited, and an alternative to minimaxing, Artificial Intelligence 21 (1, 2) (1983) 221–224.
[14] D.S. Nau, How to do worse by working harder: The nature of pathology on game trees, in: Proceedings of 1983 IEEE International Conference on Systems, Man, and Cybernetics, 1983.
[15] M.M. Newborn, The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores, Artificial Intelligence 8 (2) (1977) 137–153.
[16] J. Pearl, On the nature of pathology in game searching, Artificial Intelligence 20 (4) (1983) 427–453.
[17] A. Sadikov, I. Bratko, I. Kononenko, Search vs knowledge: Empirical study of minimax on KRK endgame, in: H.J. van den Herik, H. Iida, E. Heinz (Eds.), Advances in Computer Games: Many Games, Many Challenges, Kluwer Academic, Dordrecht, 2003, pp. 33–44.
[18] A. Scheucher, H. Kaindl, Benefits of using multivalued functions for minimaxing, Artificial Intelligence 99 (2) (1998) 187–208.
[19] G. Schrüfer, Presence and absence of pathology on game trees, in: D.F. Beal (Ed.), Advances in Computer Chess, vol. 4, Pergamon Press, 1986, pp. 101–112.
[20] B. Sjörgen, Swedish Chess Computer Association website, http://w1.859.telia.com/~u85924109/ssdf/, 2004-04-21.