SEARCH VERSUS KNOWLEDGE: AN EMPIRICAL STUDY OF MINIMAX ON KRK

A. Sadikov, I. Bratko, I. Kononenko

University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, 1000 Ljubljana, Slovenia

aleksander.sadikov@fri.uni-lj.si

Abstract This article presents the results of an empirical experiment designed to gain insight into what is the effect of the minimax algorithm on the evaluation function. The experiment's simulations were performed upon the KRK chess endgame. Our results show that dependencies between evaluations of sibling nodes in a game tree and an abundance of possibilities to commit blunders present in the KRK endgame are not sufficient to explain the success of the minimax principle in practical game-playing as was previously believed. The article shows that minimax in combination with a noisy evaluation function introduces a bias into the backed-up evaluations and argues that this bias is what masked the effectiveness of the minimax in previous studies.

Keywords: minimax principle, KRK chess endgame, evaluation-function quality, bias

1. Introduction

Over twenty years ago Beal (1980) set out to analyze whether and why values backed up from minimax search are more trustworthy than the heuristic values themselves. He constructed a simple mathematical model to analyze the minimax algorithm. To his surprise the analysis of the model showed that the backed-up values were actually somewhat less trustworthy than the heuristic values themselves. He then wrote: “This result is disappointing. It was hoped that the analysis would show that the probability of error reduced with backing-up.” A couple of years later two articles (Beal, 1982; Bratko and Gams, 1982) simultaneously conducted further analysis into why minimax does yield good results in practical game-playing while apparently backed-up values seem less reliable; both articles reached the same conclusion. They argued that the true values of sibling nodes in a game tree are not independent of one another. This clustering of similar values is a major feature in practical games and it was this phenomenon that Beal's mathematical model did not account for. The problem with the minimax paradigm under the assumption of independence of sibling values was also confirmed by Nau (1982, 1983), who called this a search-depth pathology in game trees. In a simulation Nau (1982) introduced strong dependencies between sibling nodes and discovered that this can cause search-depth pathology to disappear.

However, Pearl (1984) partly disagreed with the conclusion reached by Beal, Bratko, Gams and Nau, and claimed that while strong dependencies between sibling nodes in a game tree can eliminate the pathology, practical games like chess do not possess dependencies of sufficient strength. He pointed out that few chess positions are so strong that they cannot be spoiled abruptly if one really tries hard to do so. He concluded that the success of minimax in game-playing programs is "based on the fact that common games do not possess a uniform structure but are riddled with early terminal positions, colloquially named blunders, pitfalls or traps. Close ancestors of such traps carry more reliable evaluations than the rest of the nodes, and when more of these ancestors are exposed by the search, the decisions become more valid." Moreover, Schrüfer (1986) and its follow-up (Althöfer, 1989) did some further analysis of pathology in game trees. Especially interesting is their observation that to avoid pathology, an evaluation function must, among other things, have negligible probability of underestimating a position from the perspective of the player to move.

All of the above studies have two things in common: (a) they accept the empirical evidence that the minimax principle works in practical game-playing programs and (b) they try to model mathematically the minimax algorithm and theoretically deduce what happens when heuristic values assigned to leaves are backed-up towards the root of the game tree. To make such mathematical analysis feasible the researchers are forced to make certain assumptions about the game they model and to make simplifications in their model. Thus, the results of these models are always to be viewed with the acknowledgement of this assumptions and simplifications in the back of one's mind. In contrast to that, our approach in this article is to take (part of) a real game with a real evaluation function and observe empirically what is going on when we change the search depth and the quality of the evaluation function. We have at our disposal an absolutely correct evaluation function which we can corrupt in a controlled way. We also have a minimax search engine that is capable of searching to very high depths because of its efficient implementation.

The next section describes our choice of the game, the evaluation function and its artificial corruption, as well as the search engine. Section 3 presents the results for various settings of our simulation parameters and gives our explanations for the observed phenomena. In Section 4 we give our conclusions and some ideas for further work.

2. Experimental Design

We have decided to centre our simulations on a simple subset of chess: the KRK endgame. In this endgame White has a King and a Rook, while Black has only a King. The goal for White is to mate the opponent, striving to do so in as little moves as possible. There are two possible outcomes of this endgame: a win for White or a draw. While the KRK endgame is very simple, it still possesses all the interesting attributes: positions are of various difficulties (measured in the number of moves to mate), there surely exist dependencies between the values of sibling nodes in a game tree, and there is a possibility of blunders and early termination for both sides (stalemate or losing a Rook for White; premature mate for Black).

We are interested in the quality of play for White under different conditions. Therefore, unless stated otherwise, we always look at things from the White player's perspective. Also, White is our MIN player and Black is our MAX player.

For the KRK endgame we have at our disposal an absolutely correct evaluation function. It tells us how many moves are needed to reach mate in the case that both players play optimally and is measured in moves. It is in the form of a database that consists of all possible legal positions and their evaluations. The database can be obtained from UCI Machine Learning Repository (Blake and Merz, 1998). The positions in the database always assume it is Black's turn to move. There are two special cases: value 0 means Black is mated and value 255 means that Black has a draw (either the position is a stalemate or Black can capture the white Rook).

The database consists of 28,056 positions. There are actually over 200,000 legal KRK positions, however board symmetries allow for such a reduction. Detailed description of the database and board symmetries is given in Bain (1992). Our version of the database is implemented as an array of 28,056 cells and can be viewed as a sort of transposition table. Apart from positions having special evaluations of 0 or 255, there are 25,233 positions divided into 16 levels of difficulty. Positions from level 1 require one move (2 plies) to mate (assuming optimal play); positions from level 2 require two moves to mate, and so on. Positions from the most difficult level require 16 moves (32 plies) to mate. Different levels have different number of positions; for example, there are 4,553 positions of level 14 and only 390 positions of level 16. Figure 1 shows how many cases (positions) are left unsolved if we applied searches of various depths without any knowledge apart from the rules of the game. The term 'unsolved' in this context means that White has to make a move without knowing at that time the complete move-tree that guarantees a mate. The curve starts to fall significantly between depths of 14 and 20 plies and after ply 20 it steeply drops towards zero.

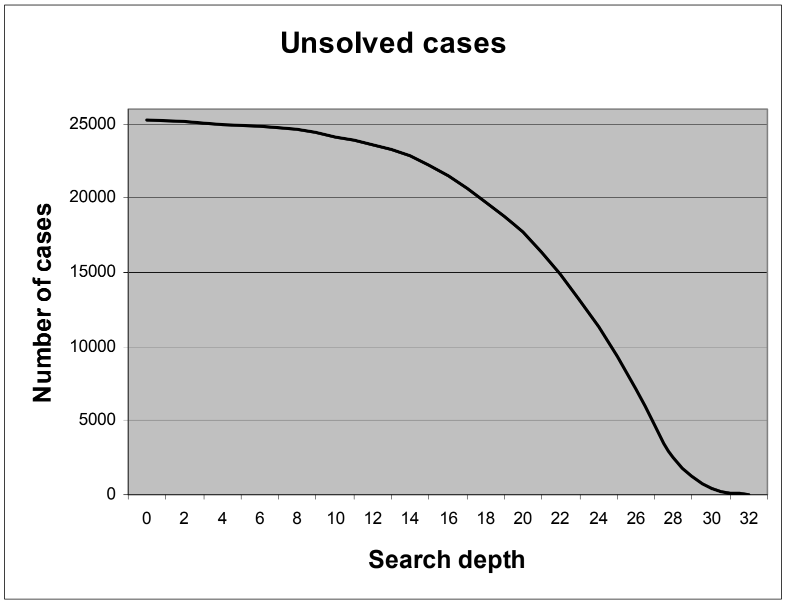
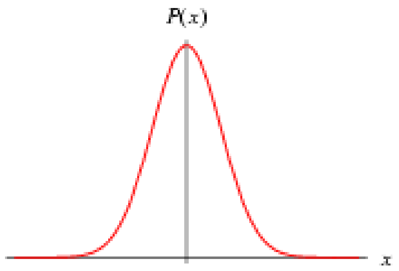


Figure 1. Number of unsolved cases as a function of search depth.

For the purpose of our experiments we corrupted the ideal evaluation function in a controlled manner. Our method of doing this is as follows. We take a position value and add to it a certain amount of Gaussian noise. The formula and a plot are as follows:

$$P(x) = e^{-(x-\mu)^2 / (2 \cdot \sigma^2)}$$



The formula gives the probability *P(x)dx* that given the correct evaluation μ and standard deviation σ the new (corrupted) evaluation x will take on a value in the range [x, $x + dx$], which is a real number. The error of new evaluation is $\mu - x$. We do this for all positions in the database, including the positions where Black is already mated (special value 0). The corruption is symmetrical, meaning that there is practically equal chance that the new evaluation will be optimistic or pessimistic. We allow x to take on a negative value – in this way we are able to preserve the symmetry for positions that have true values close or equal to 0.

The level of corruption is controlled by the parameter σ, which is in fact the standard deviation and which controls how dispersed are the corrupted values x around the correct values μ (the width of the hill on the plot above). The standard deviation is measured in moves. For example, if σ equals 0.5, this means that approximately two thirds of corrupted evaluations are within 0.5 moves around the true evaluation and over 95% of corrupted evaluations are within 1.0 move (two standard deviations) around the true evaluation.

To be able to compare the quality of initial knowledge (evaluation function) to the quality of knowledge after backing up the values with the minimax algorithm, we have to be able to calculate the standard deviation after minimaxing. This is easy, because our search algorithm returns the backed-up values from a fixed search depth for every unique KRK position in an array exactly the same as our initial database. This array is in fact our 'backed-up' evaluation function. We thus have one such array for every search depth from 0 (initial database) to 32 ply (our chosen final search depth). After obtaining such an array, we calculate σ with the formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu_i\right)^2}$$

where x_i is the backed-up corrupted value and μ_i is the true value for position i. N is the number of positions in the array. This gives us a tool to monitor directly how minimax affects the quality of evaluation function.

Our search engine is the standard fixed-depth minimax search. The only built-in knowledge it has is the ability to detect fatal errors for White (stalemates and losing a Rook); the ability to detect mates is not given. We were able to search to very high search depths of 32 plies and beyond (if desired) by exploiting the fact that the KRK endgame only has a comparatively small number of unique (under symmetries) positions (28,056) which we can all store in a sort of transposition table. We start at depth 0 by loading the values from a (corrupted) database, then move on to depth 2, perform a 2-ply minimax search and use the results of the previous depth as evaluations of the leaves, store results of depth 2 search, move on to depth 4 and so on. Such an implementation of the search algorithm allows it to have a linear time complexity instead of the usual and very constraining exponential time complexity.

3. The Results of the Experiments

Figure 2 shows what happens to the quality of the evaluation function when we change the search depth. The x-axis represents the search depth measured in plies and the y-axis represents the standard deviation σ measured in moves. Each curve in the graph represents a different evaluation function – they differ in the level of their corruption (the initial σ). The legend marks these different evaluation functions with the size of their initial σ. The best way to separate the curves is to look at their initial corruption. The last evaluation function with initial σ of 20 is off the scale and its corruption level never drops. We performed the experiments with several evaluation functions having the same initial σ, because the introduction of noise is a random process. We found out that the main characteristics remain the same for all evaluation functions with the same σ and have therefore plotted just one evaluation function with certain initial σ in all the figures.

It seems that we have to divide the evaluation functions into two groups: in the first group we have evaluation functions with a (relatively) low initial error of less than 3.0 and in the second group those with a high initial error. The first group is a realistic model of 'real-life' evaluation functions, while the second group contains evaluation functions with (almost) zero knowledge. We can observe that evaluation functions from the first group do not exhibit the tendency to drop towards 0 (perfect knowledge). They drop slightly or remain on the same level of corruption at best; some even increase. In contrast, evaluation functions from the second group only increase with increased search depth.

The experiment demonstrates that for the evaluation functions tested searching deeper does not improve the quality of the evaluation function for playing the KRK endgame, not even for those with a small initial level of corruption. The endgame undoubtedly contains dependencies between the values of sibling nodes in a game tree. It is also full of possibilities for blunders on the part of white player. White can, after all, lose the Rook in at most two moves if Black plays normally. This means that the two reasons why minimax is believed effective in practice are present and yet the pathology is present as well. How can this be explained?

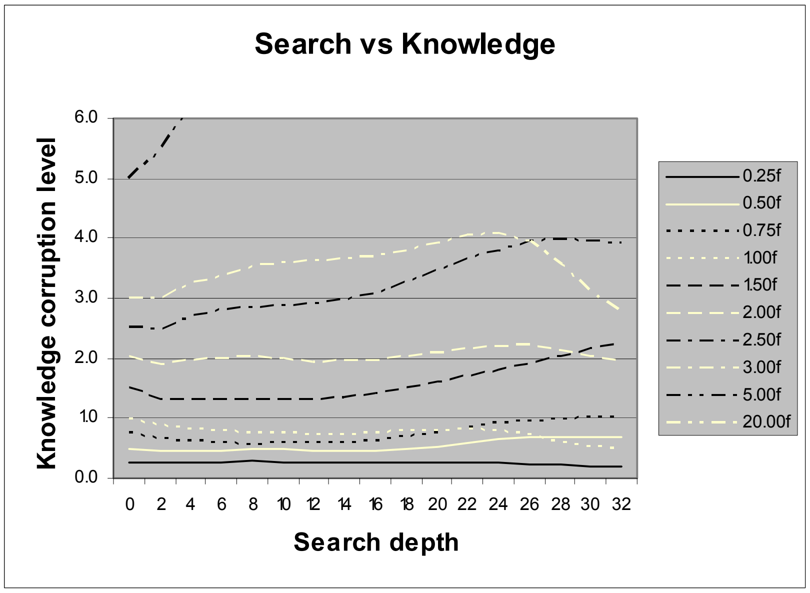


Figure 2. Influence of search depth on quality of evaluation function.

One thing we were interested in was how the backed-up evaluations are corrupted. We were curious whether backed-up evaluations are excessively optimistic or excessively pessimistic. To this end we calculated the bias of a backed-up evaluation function. Bias is defined as:

$$bias = \frac{1}{N}\sum_{i=1}^{N}(\mu_i - x_i)$$

where μ_i and x_i are again the true and backed-up value of position i, respectively. If bias is highly negative then the backed-up values are generally overly pessimistic and if bias is highly positive then the backed-up values are generally overly optimistic. Since the noise introduced into the various evaluation functions was symmetrical we expected the bias to be close to zero, meaning some backed-up evaluations are too optimistic and others too pessimistic. Figure 3 charts how biased various evaluation functions are with respect to search depth. All curves start in close proximity of zero and then without exception they all exhibit a highly positive bias. The higher the level of initial corruption the higher the bias gets. Most of the bias is acquired in transition from search depth 0 to search depth 2. These two levels differ the most of any two consecutive levels – depth 0 means the algorithm goes directly to the lookup table, while on level 2 it performs minimaxing for the first time. Other transitions just increase the depth of minimaxing.

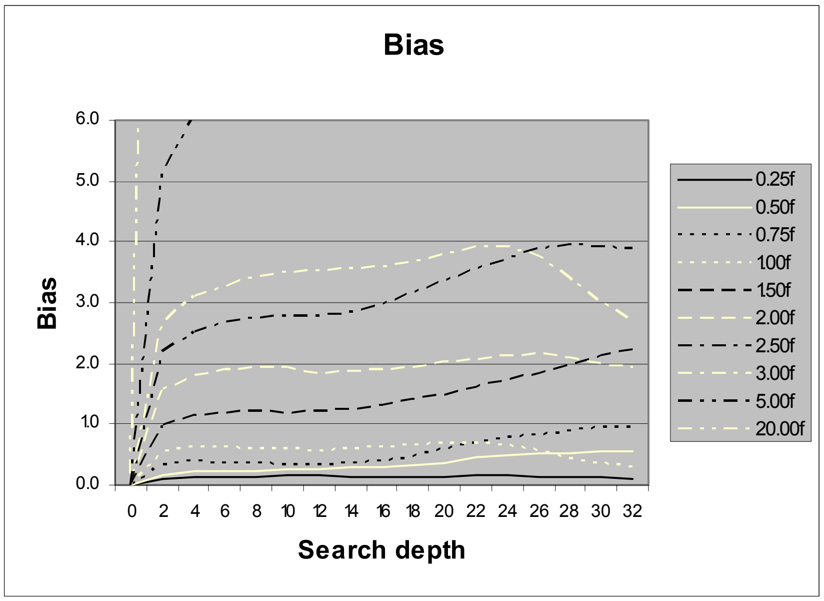


Figure 3. Influence of search depth on bias of the evaluation function.

If we look closely at what is happening at the last level of minimaxing we can come up with an explanation why the bias occurs. On the last level we either have a max or a min operation. In our case we always had White to choose on the last level which meant a min operation. In presence of noise the operation of choosing a minimum value will be biased towards lower values. This is not saying that the value chosen will always be lower than it would be without noise, but in general this will be so much more often than not. We can thus see that the lowermost operation of the minimax algorithm introduces a bias. But surely the opposite operation, finding a maximum, which follows on the next higher level should (partly) negate this bias? It does not, however, because the majority of the values it operates on are already biased and all it does is selecting one of them. The bias is actually introduced on that lowest level of minimaxing.

If we look at Figures 2 and 3 simultaneously we find out that the corruption level and the level of bias are highly correlated. For evaluation functions with a lower initial level of corruption (0.25 and 0.50) this correlation begins to manifest with the higher search depths of 16 to 20, while for others it begins much sooner, from a search depth of 10 for curve 0.75 and from a search depth of 4 for curve 1.0. For curves with initial corruption higher than 1.0 the correlation is strong immediately from search depth 2 onward. It is no wonder then that backed-up evaluation functions could not get any better – they were prevented from doing so by the bias. However, bias, at least in general, equally affects all evaluations. It resembles adding a constant to all evaluations. This in turn means that we do not change the order of the available moves relatively to one another in the position we are trying to evaluate. If this is true, then minimax actually does improve the evaluation, but on the surface it is not seen, because of the bias.

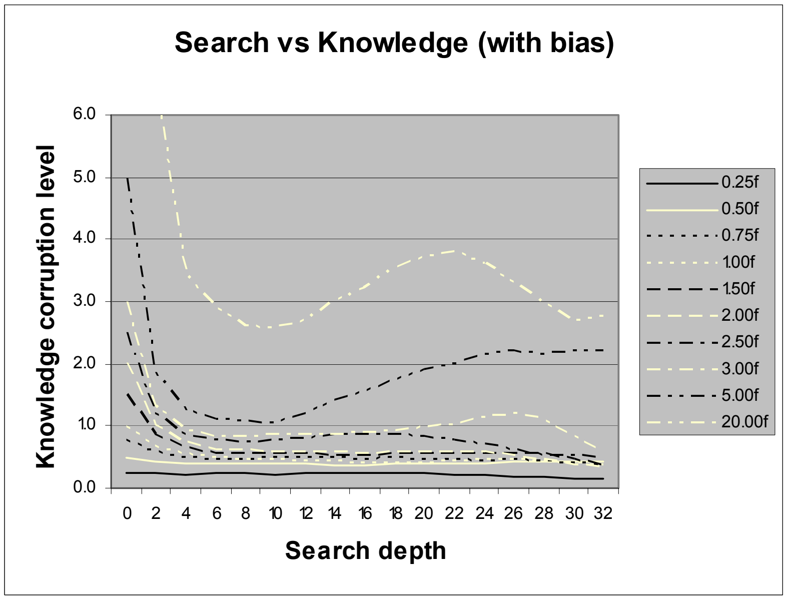


Figure 4. Influence of search depth on quality of evaluation function with bias accounted for.

To confirm this claim, we calculated another statistic, a standard deviation as before. However, this time we have taken into account that all the values were shifted away from the true values by the bias. The formula for this statistic, *σ'*, is:

$$\sigma' = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \left(\mu_i + bias\right)\right)^2}$$

where μ_i and x_i are again the true and backed-up value of position *i*, respectively. How this new standard deviation changes in relation with search depth is shown in Figure 4. Here we can see that it drastically falls with the increase of search depth. Again, the evaluation functions from the two groups defined earlier behave differently. Evaluation functions from group one drop and then stay more or less on the same level, while evaluation functions from group two first drop and then start to rise back up again. This positive effect of the minimax principle with increasing search depth on the evaluation functions from group one is exactly the result that

Beal was expecting from his model in 1980, but he was unable to prove it because the model did not account for the introduced bias.

Up to this point, we did not say anything about how well a computer program using one of our corrupted evaluation functions would actually play. The answer is given in Figure 5. We have played out all unique KRK positions except the ones with special values of 0 or 255, in total 25,233 positions. White was guided by a corrupted evaluation function. Additionally, White was allowed to use a simple mechanism to avoid repeating the same position over and over again. The mechanism kept a list of all positions that already occurred in the game and if the position was to be repeated a different move was selected (the next best move according to the evaluation function). Black was always playing optimally. We measured the quality of play as the average number of moves above what an optimal white player (using a non-corrupted database) would need. This statistic is computed as the difference between the number of moves spent by White for all positions and the number of moves needed for all positions using optimal play, divided by the number of positions (25,233). The curves representing play using evaluation functions with initial corruption level of 5 and 20 are off the scale and result in play that is not even able to mate the opponent within the required 50 moves.

In Figure 5 we can see that an evaluation function with initial corruption level of 0.25 moves provides practically optimal play starting already at search depth 0. The quality of play using other evaluation functions gradually increases with deeper searches until it reaches a sort of threshold for a given evaluation function. From that point onward the quality of play remains more or less on the same level. This is true for evaluation functions with initial corruption level below 3. Those evaluation functions with initial corruption level higher than 3 result in a play that is not even good enough to mate the opponent within the required 50 moves. We can observe a correlation between the quality of play in Figure 5 and the knowledge corruption level of the evaluation functions in Figure 4.

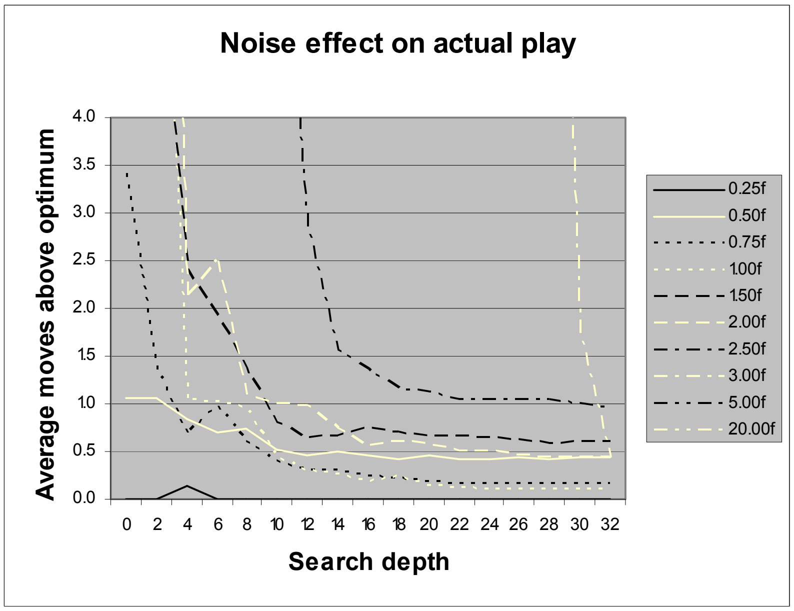


Figure 5. Quality of play using corrupted evaluation functions.

4. Conclusions and Further Work

Some theoretical studies of the minimax principle in the past have shown that it has a negative effect on the quality of the evaluation function. As the answer why it is nevertheless successful in practice they suggested two reasons: (a) dependencies between the true values of sibling nodes in a game tree, and (b) existence of traps that cause early terminations of the game.

We have taken the opposite approach to the problem; we tried to check empirically these conclusions using the KRK chess endgame. We can confirm that the minimax algorithm appears to be a poor preserver of the knowledge built into the evaluation. Yet, regardless of that, it proved to be still successful in actual play (Figure 5). It turns out that even with dependencies between evaluations of sibling nodes in a game tree and an abundance of possibilities to commit blunders present in our endgame, the anomaly still existed.

However, we claim that the minimax principle in combination with noisy evaluation functions introduces a bias into backed-up evaluations. This bias is the culprit why mathematical models could not prove the effectiveness of the minimax that was observed in practice. Once bias is properly accounted for, the positive influence of the minimax principle with increasing search depth is unmasked. The main problem is that bias moves the backed-up evaluations away from the true values, hence causing the illusion that they are more corrupted. Yet, since it more or less affects all the evaluations equally it does not affect the relative ordering of the available moves with respect to their quality. So, if we look at the evaluations in absolute terms they are increasingly corrupted with a growing bias, but if we look at them in relative terms they become progressively better with higher search depths.

In view of the presented results it would be very interesting to recheck our results using a more complex game – perhaps a KQKR or KRKN chess endgame, or some artificially designed game. A further study of how the bias behaves and what affects it is also necessary. Especially interesting would be to investigate what happens with the bias if we mix the functions (min and max) at the lowest level of minimaxing – some branches we search to even depths, others to odd depths.

References

Althöfer, I. (1989). Generalized minimax algorithms are no better error correctors than minimax itself. *Advances in Computer Chess 5* (ed. D.F. Beal), pp. 265-282. Elsevier Science Publishers, Amsterdam, The Netherlands.

Bain, M. (1992). Learning optimal chess strategies. *Proc. Intl. Workshop on Inductive Logic Programming* (ed. S. Muggleton), Institute for New Generation Computer Technology, Tokyo, Japan.

Beal, D.F. (1980). An analysis of minimax. *Advances in Computer Chess 2* (ed. M.R.B. Clarke), pp. 103-109. Edinburgh University Press, Edinburgh, UK.

Beal, D.F. (1982). Benefits of minimax search. *Advances in Computer Chess 3* (ed. M.R.B. Clarke), pp. 1-15. Pergamon Press, Oxford, UK.

Blake, C.L. and Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA.

Bratko, I. and Gams, M. (1982). Error analysis of the minimax principle. *Advances in Computer Chess 3* (ed. M.R.B. Clarke), pp. 1-15. Pergamon Press, Oxford, UK.

Nau, D.S. (1983). Pathology on Game Trees Revisited, and an Alternative to Minimaxing. *Artificial Intelligence*, 21(1, 2), pp. 221-244.

Nau, D.S. (1982). An Investigation of the Causes of Pathology in Games. *Artificial Intelligence*, 19, pp. 257-278.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Publishing Company, Reading, MA.

Schrüfer, G. (1986) Presence and absence of pathology on game trees. *Advances in Computer Chess 4* (ed. D.F. Beal), pp. 101-112. Pergamon Press, Oxford, UK.