Search Versus Knowledge in Game-Playing Programs Revisited*

Andreas Junghanns, Jonathan Schaeffer
University of Alberta
Dept. of Computing Science
Edmonton, Alberta
CANADA T6G 2H1
Email: {andreas,jonathan}@cs.ualberta.ca

Abstract

Perfect knowledge about a domain renders search unnecessary and, likewise, exhaustive search obviates heuristic knowledge. In practise, a tradeoff is found somewhere in the middle, since neither extreme is feasible for interesting domains.

During the last two decades, the focus for increasing the performance of two-player game-playing programs has been on enhanced search, usually by faster hardware and/or more efficient algorithms. This paper revisits the issue of the relative advantages of improved search and knowledge. It introduces a revised search-knowledge tradeoff graph that is supported by experimental evidence for three different games: chess, Othello and checkers, using a new metric: the "noisy oracle".

Previously published results in chess seem to contradict our model, postulating a linear increase in program strength with increasing search depth. We show that these results are misleading, and are due to properties of chess and chess-playing programs, not to the search-knowledge tradeoff.

1 Introduction

Many experiments have been performed in game-playing programs that measure the benefits of improved knowledge and/or deeper search. In particular, chess has been a popular application for these experiments. The explicit or implicit message of these works is that the results for chess are generalizable to other games. There have been few studies that examined the impact of improved knowledge on program performance [Schaeffer and Marsland, 1985; Mysliwietz, 1994]. In contrast, the benefits of additional search are well documented: deeper search provides immediate performance gains (for example, [Thompson, 1982]).

*This research was supported by the German Academic Exchange Service (DAAD), the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Killam Foundation.

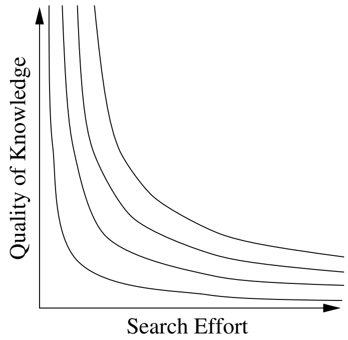

Figure 1: Proposed Search-Knowledge Relationship

Figure 1 has been hypothesized to represent the relationship between the quality of knowledge and search effort expended (first expressed in [Michie, 1977] and later refined in [Berliner *et al.*, 1990]). The curves represent various combinations of search and knowledge with equivalent performance. The figure illustrates that by increasing the search effort, less knowledge is required by the application to achieve the same level of performance, and vice versa.

Of the two dimensions, improvements in search are the easiest to address. Gains can often be achieved with little effort. One can redesign algorithms or rewrite code to execute faster or, even better, do nothing and just wait for a faster computer to become available.

The knowledge dimension, however, is nebulous. Whereas there are well-defined metrics for measuring search effort (such as search depth, execution time, and nodes examined), there is nothing comparable for knowledge.

This paper makes a number of contributions to our understanding of the relationship between search and knowledge in game-playing programs:

- Figure 1 is a hypothesis and has not been verified. In fact, it turns out to be misleading. Analytical and experimental data (from three game-playing programs: chess, Othello, and checkers) allows us to construct a new view of the search-knowledge tradeoff. This is shown in Section 2.
- To do the experiments, we needed a way of assessing

the quality of a program's knowledge. To do this, we introduce a new metric, the *noisy oracle*. Section 3 presents experimental data that yields new insights into the shapes of the curves.

- Figure 1 predicts decreasing benefits for increasing search effort—diminishing returns. This has been confirmed for both Othello and checkers by others. However, numerous papers suggest that for chess the relationship between search depth and performance is relatively constant and non-decreasing. In Section 4, we demonstrate that diminishing returns do indeed occur in chess, and that the reason for this discrepancy with the literature is rather surprising—the game length in combination with relatively high error rates.

2 Search Versus Knowledge: Theory

In this section, we use an idealized definition of knowledge. Knowledge is uniformly applicable throughout the search tree. This allows us to avoid thorny issues such as search pathology [Nau, 1983] and search-depth-dependent anomalies.

Figure 1 shows various performance levels for different combinations of search effort and quality of knowledge (isocurves). This graph has been hypothesized, but never verified. The shape of the isocurves comes largely from two known data points in the graph: no search and perfect knowledge, as well as exhaustive search with no knowledge—both yield a perfect program[1]. However, there is nothing in this data that implies that the isocurves should be concave down, or even that they should be curves at all. In fact, [Michie, 1977] and [Berliner *et al.*, 1990] provide no justification for their shape. However, experience suggests this shape to be likely (unproven). For now, we assume that they are concave down, and we examine this issue in the next section.

What does it mean to do no search and have perfect knowledge? In fact, this implies a minimal amount of search (1 ply) to evaluate all the moves and then choose the one leading to the highest outcome. What does it mean to perform exhaustive search (to depth GL, the maximum game length) with no knowledge? The "no knowledge" is misleading, because to play perfectly one must have some knowledge—in this case being able to identify and correctly backup the scores for wins, losses, and draws. This suggests a scale for these two data points. We let WLD represent the knowledge about correctly backing up terminal nodes in the search. This is knowledge supplied by the application domain rules. The "perfect knowledge" program requires 100% of the domain-specific knowledge required to play flawlessly. The above distinction allows us to plot two data points on the y-axis. We can now plot the isocurve for perfect programs, a concave down curve between the points $(1,+100)$ and (GL,WLD).

Although we are interested in programs that have high performance, we can also consider the case where the quality of a program's knowledge is worse than WLD. The worst-case scenario is a program whose knowledge is -100: the program assesses positions inversely proportional to their worth. Thus, this *anti-perfect* program with a one-ply search will always choose the worst move on the board.

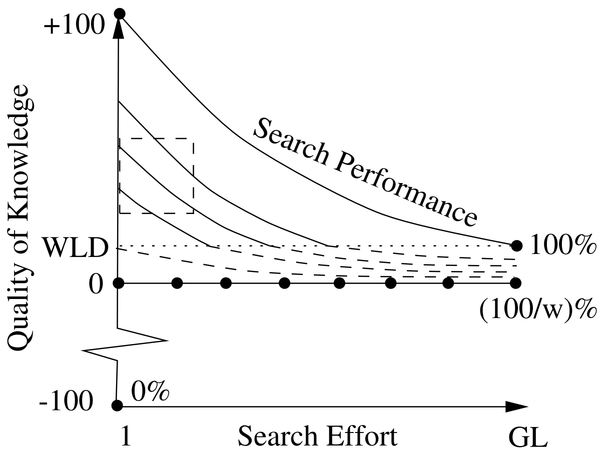

Figure 2: Search Versus Knowledge Revisited

One other data point of interest is the 0 knowledge program. Since the the program has no knowledge, its move choices are random. Given an average branching factor of w move choices in a position, the program will make the right move $1/w$ of the time ($w = 40$ for chess, while only 8 in non-capture checkers positions). Here there are no benefits of search; the program will play the correct move $1/w$ of the time regardless of the search depth[2]. This program is clearly better than the anti-perfect program, but must be worse than a WLD program. The latter follows since the WLD program has positive knowledge about wins, losses, and draws, which can only improve the likelihood of selecting the right move.

These arguments allow us to construct a revised version of Figure 1, as shown in Figure 2 (ignore the dashed box for now). The x-axis is search depth, starting at 1 and increasing. The y-axis is the quality of knowledge, ranging from perfect positive knowledge to perfect negative knowledge. Each isocurve represents a fixed level of performance. If we measure performance by the percentage of correct move decisions made by the program, then the perfect program is 100% correct, and the anti-perfect program scores 0%. The random program alway scores $100/w\%$ (we ignore the minor differences that occur if w varies during the game).

[1]Perfect is meant to imply that the program never makes a game-theoretic-value error. We do not consider the case where the program is also required to play the move that maximizes the chances for improving its expected outcome.

[2]We use the simplifying assumption of a uniform branching factor. As [Beal and Smith, 1994] showed, random evaluations can implicitly capture concepts like mobility in non-uniform trees.

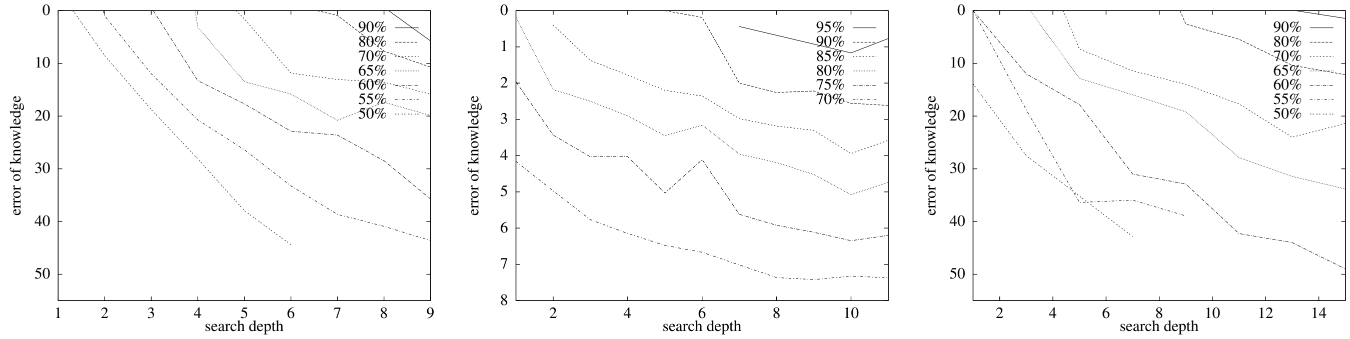

Figure 3: Search Knowledge Behavior in Chess (left), Othello (middle) and Checkers (right)

Note that we have not included any isocurves in the 0 to -100 range. Here the knowledge is worse than random, and one can expect to see search pathology [Nau, 1983]. Since this region is not of interest in practice, for reasons of brevity we ignore it. It is interesting to note that the anti-perfect program, which always makes the worst move with a 1-ply search, may play the right move given a 2-ply or larger search (albeit for the wrong reasons).

Consider the region in Figure 2 that is bounded by the perfect curve (100%) and the random line ($100/w\%$). All curves start at $depth = 1$, but are spaced out over a range from $+100$ to 0 on the knowledge axis. They end up at $depth = GL$, in the smaller range from WLD to 0. Therefore the curves move closer together as the search depth increases. In other words, the isocurves do not have the same slope. The lower the performance, the flatter the curve—the extreme being the flat random line. The higher the performance, the steeper the curve—the extreme being the perfect performance isocurve. Hence, as one moves to higher performance levels, the slope of the isocurves increase. This implies that for shallow search depths, more knowledge is required to move to a higher isocurve than for deeper search depths.

3 Search Versus Knowledge: Practise

The difficulty in experimentally verifying Figure 2 lies in quantifying the knowledge axis. Perfect knowledge assumes an oracle, which for most games we do not have. However, we can approximate an oracle by using a high-quality, game-playing program that performs deep searches. Although not perfect, it is the best approximation available. Using this, how can we measure the quality of knowledge in the program?

A heuristic evaluation function, as judged by an oracle, can be viewed as a combination of two things: oracle knowledge and noise. The oracle knowledge is beneficial and improves the program's play. The noise, on the other hand, represents the inaccuracies in the program's knowledge. It can be introduced by several things, including knowledge that is missing, over- or under-valued, and/or irrelevant. As the noise level increase, the beneficial contribution of the knowledge is overshadowed.

By definition, an oracle has no noise. We can measure the quality of the heuristic evaluation in a program by the amount of noise that is added into it. To measure this, we add a random number to each leaf node evaluation (N_L).

In most games of skill, the value of a parent node is strongly correlated with the values of its children. Hence, our noise model should reflect this. Following the previous work of [Iida *et al.*, 1995], we define the noise of a leaf node in a search to be $N_L = \sum_{i=1}^{d} r_i$, where $-R < r_i < R$, R is an adjustable parameter, and d is the depth in the tree of the leaf node. This simple representation comes closer to approximating the parent/child behavior. The resulting random numbers at the depth d leaf nodes have a normal distribution with mean 0 and a standard deviation of $\sqrt{d * (R^2/3)}$. One should be careful: simulating tree behavior is fraught with pitfalls [Plaat *et al.*, 1996].

The above discussion assumed we have a perfect oracle. For real games such as chess, Othello and checkers, the best we can do is use a high-quality, deep-searching program as our best approximation. In effect, this program is a *noisy oracle* with noise level N_O. We can now increase the noise level by increasing the distribution of random scores added to the evaluation ($N_O + N_L > N_O$).

To show the tradeoff between search and knowledge, we conducted experiments with chess, Othello, and checkers. The programs used were *TheTurk* (chess), *Keyano* (Othello) and Chinook (checkers)[3]. All three are well-known internationally. For each game, 256 positions from grandmaster play were selected. The noisy oracle would determine the best move in the position. Since the oracle is noisy, and evaluation functions differentiate positions by insignificant margins, all moves that were within 5 points (1/20th of a pawn/checker) in chess/checkers or 8 disc in Othello were considered as best moves. For each game, each position was searched to a variety of search depths with a variety of noise. The programs were searched with $R = 0, 5, 10, 15, 20, 25, 50, 100, 150$ for

[3] *TheTurk* is a tournament chess program developed at the University of Alberta by Andreas Junghanns and Yngvi Bjornsson. *Keyano* is one of the strongest Othello programs internationally and developed at the University of Alberta by Mark Brockington.

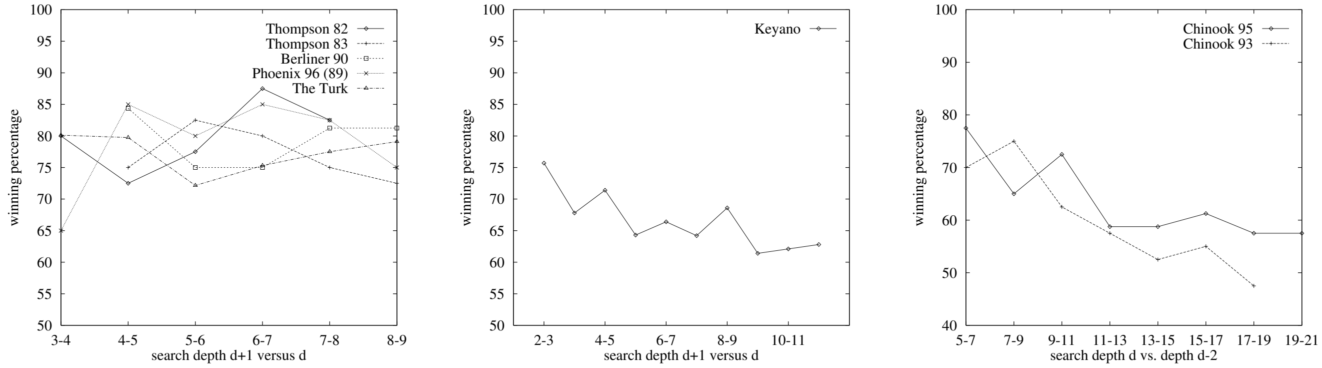

Figure 4: Self-Play Experiments in Chess (left), Othello (middle) and Checkers (right)

chess and checkers, and $R = 0, 1/8, 1/4, 1/2, 1, 2, 4, 8, 16$ for Othello.

Figure 3 shows the results for the three games (only some of the R values are shown). The x axis is the search depth, ranging from 1 to 9-15 depending on the game. The y axis measures the quality of the noisy oracle's knowledge, beginning at $R = 0$. The isocurves represent different levels of performance, where performance is measured as the percentage of times that the program makes the correct move selection in the test set.

All three programs exhibit similar behavior. The isocurves appear to be curved and concave down, although in many cases they are almost linear. The curves are not perfectly formed because of the statistical nature of the experiments. All three games show the curves leveling off, suggesting that for deeper searches, the benefits of additional knowledge (less noise) are more significant than for additional search.

In our experimental setting, we are restricted to a small range of possible values on the x and y axis. From the shape of the curves in Figure 3, we can approximate where this graph fits into the Figure 2 framework (shown by the dashed box).

When comparing the graphs for the different games, the reader should keep in mind that neither the search nor the knowledge axis are comparable, since it is not clear how close we are to perfect knowledge and exhaustive search depth. Although it is well-defined what it means to search an additional ply of search, it is not clear what it means to reduce the noise from, say, 20 to 10. In other words, although the y axis is shown as a linear scale, the effort required to improve the program along this axis may not be linear.

4 The Chess Anomaly

The results from Sections 2 and 3 suggest that the benefits of additional search decline as the search depth increases—so-called *diminishing returns*. A number of papers have experimentally addressed this question. Figure 4 graphs some of those results. These graphs are the result of self-play experiments, where a program searching to depth d plays matches against the same program searching to depth $d + \delta$, where $\delta = 1$ for chess and Othello, and $\delta = 2$ for checkers. The idea is that, for example, the winning percentage of a 3-ply program playing against a 2-ply program should be higher than for a 13-ply program playing a 12-ply program. At least in Othello (experiments with *Keyano*, and supported by data in [Lee and Mahajan, 1990]) and checkers [Schaeffer *et al.*, 1993], this seems to be borne out.

However, the results for the game of chess are perplexing because, even though there is a logical argument for stating that the benefits obtained by deeper searching will gradually reduce, the experimental evidence does not substantiate this. Many publications consistently show a linear relationship between search depth and performance (for example, [Newborn, 1979; Thompson, 1982; Condon and Thompson, 1983; Newborn, 1985; Berliner *et al.*, 1990; Mysliwietz, 1994]). Only [Condon and Thompson, 1983] shows a slight decline in performance with increased search depth, however this trend is still within the range of statistical noise. Intuitively, diminishing returns must exist, since eventually exhaustive search solves the problem and additional search effort would be entirely wasted.

Our new experiments with chess show that there are diminishing returns, further confirming the general shape of Figure 2. The reason that these results were not evident in previous work is twofold; one reason having to do with the quality of the program's knowledge, and the other having to do with a characteristic of the game.

Decision Quality

Searching to depth $(d + 1)$ pays off only if the deeper search results in a better move choice than is possible with a d-ply search. The smaller the probability that this happens, the better the d-ply search is a predictor of the $(d + 1)$-ply search. Note that the value of the search is irrelevant; only the move selection influences the game result (even if the right move is played for the wrong reasons).

We conducted an experiment to measure how the move choice changes as a function of search depth (similar to [Newborn, 1985]). One thousand opening positions were searched by a deep (9-ply) version of CHESS (a noisy oracle) to determine the best move and value. Figure 5 shows the percentage of move changes in the top-most curve. A move change might not be a significant event if the value difference between the moves is small, as judged by the noisy oracle. The additional curves in the figure represent the percentage of *significant* move changes, according to the difference in move values (at least 10, 15, 20, 25, 50, 100 points), where 100 points is the equivalent of a pawn.

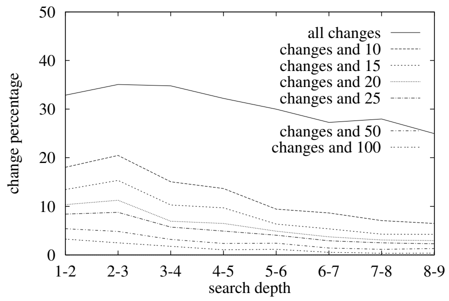

Figure 5: Move Changes from d to $(d+1)$ Ply (Chess)

The graph shows a reduction in error (or alternatively, an increase in prediction accuracy) with increasing search depth, but the error reduction slows down with deeper searches. Figure 6 shows a different view of the data. Here the change in value in going from d to $d+1$ ply is plotted versus depth. The curves represent the percentage of moves that achieve a certain level of performance. For example, the top curve shows that 1% of the moves result in value changes of roughly 100 points (a pawn) when you search from 8 to 9 ply. The curves show a dramatic decrease in expected error and, again, exhibits a tapering off with deeper searches—an indication of diminishing returns.

The surprising feature of Figure 6 is the magnitude of the errors. In going from 8 to 9 ply, 10% of the moves result in at least a 25-point differential; usually a significant score swing. In other words, the error rates of even an 8-ply search in CHESS are extremely high.

This data can be dramatically put into perspective by comparing it with the results of a similar experiment with Chinook. Chinook is the world's strongest checkers playing entity (man or machine). With its massive endgame databases (444 billion positions), the program is close to being an oracle. Figure 7 shows the percentage of move changes for checkers. The difference is clear: the error rates are much lower, an indication of how much better the evaluation quality of *Chinook* is as compared to CHESS. With such low error rates, searching deeper in *Chinook* yields little benefits. In CHESS, the error rates are still high enough to allow for significant improvements as search depth increases, which in return obscures the effect of diminishing returns in self-play games.

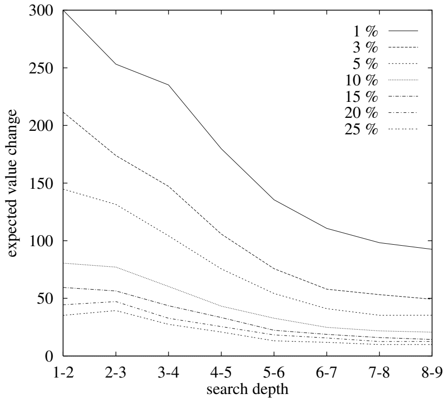

Figure 6: Value Changes from d to $(d+1)$ Ply (Chess)

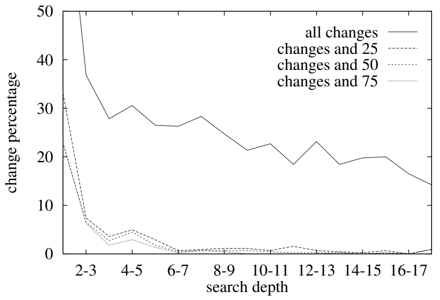

Figure 7: Move Changes from d to $(d+1)$ Ply (Checkers)

Game Length

The above suggests that the decision quality in chess is not as good as one would like (i.e. the noisy oracle is too noisy). Each move played by the d-ply program against the $d+1$-ply program is fraught with danger, since the deeper searching program has less probability of making a mistake. This suggests that the longer the games lasts, the greater the winning chances of the $d+1$-ply program.

To test this hypothesis, we conducted two experiments. First, we measured the average length of self-play games played by CHESS. As the search depth of the programs increased, so did the length of the game. In other words, for shallow searches, the games tended to be shorter because the probability of an error was higher. As the search depth increased, the error probability dropped and, hence, the games lasted longer because the opponents were more evenly matched. Games played between 8- and 9-ply programs averaged out to be 29% longer than games between 3- and 4-ply programs.

The above suggests that game length has something to do with chess self-play results. To test this hypothesis, we played a series of 80 self-play games where the game length was restricted. After a specified number of moves, the game was adjudicated.

Figure 8 shows a constant winning percentage for games of unrestricted length (top line) that would lead to the conclusion that diminishing returns do not exist in chess. However, if we restrict the length of the games (to 10 through 45 moves), a decline in the winning percentage is visible, leading to the conclusion that diminishing returns exist in chess for truncated games.

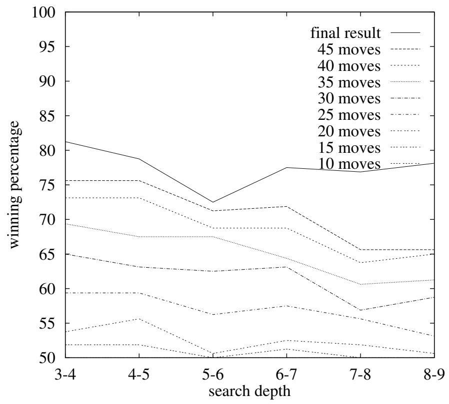

Figure 8: Winning % for Truncated Games (Chess)

Both Othello and checkers games are limited in the number of moves. Othello games are constrained to a maximum of 30 moves aside. Checkers, with its forced capture rule, tends to have similarly short games. In contrast, chess has no such limitations. Given that the $d + 1$-ply searching program has an advantage over the d-ply searcher, the longer the game, the greater the likelihood that the advantage will manifest itself. Essentially, self-play experiments in chess suffer from the *gambler's ruin*, the reason why diminishing returns have remained hidden for almost 20 years.

5 Conclusion and Future Work

A new graph for the search-knowledge tradeoff was proposed and experimentally verified. This graph suggests diminishing returns for both the knowledge and search axis. We show that, contrary to the previous literature, there are diminishing returns in chess. This is due to two reasons. The first, decision quality, is not a surprise. The second, game length, is a new result that illustrates how sensitive experimental data can be to hidden properties of the search domain.

Diminishing returns for both increasing knowledge and search raises the question as to what the best way is for improving program performance. The answer depends on several factors including, for example, the application domain, the quality of the evaluation function, and the computational resources available. Future research is needed to understand the role played by each of these factors in program performance.

The designers of the current best chess program, *Deep Blue*, have concentrated their efforts on the search axis. In a typical search, 50 billion positions are considered. The *Deep Blue* chess knowledge is limited because it is implemented in silicon. Our results suggest that small improvements in their knowledge, even at the expense of some search effort, could greatly improve their performance.

6 Acknowledgements

This paper benefited from interactions with Yngvi Björnsson, Tony Marsland, Aske Plaat and Manuela Schöne, special thanks are due to Mark Brockington for providing the data for Othello.

References

[Beal and Smith, 1994] D. Beal and M. Smith. Random evaluations in chess. *ICCA Journal*, 17(1):3–9, 1994.

[Berliner *et al.*, 1990] H. Berliner, G. Goetsch, M. Campbell, and C. Ebeling. Measuring the performance potential of chess programs. *Artificial Intelligence*, 43(1):7–21, April 1990.

[Condon and Thompson, 1983] J. Condon and K. Thompson. Belle. In P. Frey, editor, *Chess Skill in Man and Machine*, pages 201–210. Springer-Verlag, 1983.

[Iida *et al.*, 1995] H. Iida, K.-I. Handa, and J.W.H.M. Uiterwijk. Tutoring strategies in game-tree search. *ICCA Journal*, 18(4):191–204, 1995.

[Lee and Mahajan, 1990] K.-F. Lee and S. Mahajan. The Development of a World Class Othello Program. *Artificial Intelligence*, 43(1):21–36, 1990.

[Michie, 1977] D. Michie. A theory of advice. *Machine Intelligence 8*, pages 151–170, 1977.

[Mysliwietz, 1994] P. Mysliwietz. *Konstruktion und Optimierung von Bewertungsfunktionen beim Schach*. PhD thesis, University of Paderborn, 1994.

[Nau, 1983] D.S. Nau. Pathology on game trees revisited, and an alternative to minimaxing. *Artificial Intelligence*, 21(1–2):221–244, March 1983.

[Newborn, 1979] M. Newborn. Recent progress in computer chess. In M. Yovits, editor, *Advances in Computers*, volume 18, pages 59–117. Academic Press, 1979.

[Newborn, 1985] M. Newborn. A hypothesis concerning the strength of chess programs. *ICCA Journal*, 8(4):209–215, 1985.

[Plaat *et al.*, 1996] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, 87(1–2):255–293, November 1996.

[Schaeffer and Marsland, 1985] J. Schaeffer and T. Marsland. The utility of expert knowledge. In *IJCAI'85*, pages 585–587, 1985.

[Schaeffer *et al.*, 1993] J. Schaeffer, P. Lu, D. Szafron, and R. Lake. A re-examination of brute-force search. In *Games: Planning and Learning*, pages 51–58. AAAI, 1993. 1993 Fall Symposium, Report FS9302.

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