Diminishing Returns for Additional Search in Chess

Andreas Junghanns, Jonathan Schaeffer,
Mark Brockington, Yngvi Bjornsson and Tony Marsland
University of Alberta
Dept. of Computing Science
Edmonton, Alberta
CANADA T6G 2H1
Email: {andreas,jonathan,brock,yngvi,tony}@cs.ualberta.ca

Abstract

Advances in technology allow for increasingly deeper searches in competitive chess programs. Several experiments with chess indicate a constant improvement in a program's performance for deeper searches; a program searching to depth $d+1$ scores roughly 80% of the possible points in a match with a program searching to depth d. In other board games, such as Othello and checkers, additional plies of search translated into decreasing benefits, giving rise to diminishing returns for deeper searching. This paper demonstrates that there are diminishing returns in chess. However, the high percentage of errors made by chess programs for search depths through 9 ply hides the effect.

1 Introduction

It is common knowledge that chess program performance is strongly correlated to the depth of the search tree explored by the program. Deeper searching can be achieved in several ways, including more efficient search algorithms (smaller search trees), forward pruning (with the possibility of introducing error into the search), search extensions (extending interesting lines) and by faster search engines. The latter point has had a tremendous impact on the history of computer chess. The drive for speed was the motivation for special-purpose chess machines (such as *Belle* [CT83], *Hitech* [BE90] and *Deep Blue* [HCN90a]) and parallel programs (such as *Zugzwang* [Fel93] and **Socrates* [Kus94]). Every chess programmer intenting on high performance devotes considerable effort to tuning and optimizing their code for maximum speed.

Ken Thompson's pioneering work with *Belle* dramatically illustrated the benefits of increasing the search depth [Tho82]. In his experiments, a program searching a fixed-depth tree to depth $d+1$ would play a 20-game match against a program searching to depth d.

Surprisingly, the deeper searching program scored 80% of the possible points in the match. Some researchers extrapolated this work to conclude that a program searching 12 ply would be able to achieve a rating that was higher than that of the World Champion ([HCN90b], for example).

Although it is enticing to think that every additional ply of search is worth the same number of rating points, simple logic suggests that diminishing returns for additional search must eventually appear. For example, can there possibly be a big difference between a 100-ply program and one doing 101 ply? Nevertheless, numerous self-play experiments in chess confirm that there seems to be a linear relationship between search depth and performance (for example, *Belle* [Tho82], *Hitech* [BE90] and *Zugzwang* [Mys94]). All of these results are remarkably consistent: a depth $d+1$ program scores 80% $\pm$ 10% of the points in a match with a depth d program. However, one experiment suggests the possibility of diminishing returns. A later repeat of the *Belle* experiment [CT83] gives weak evidence that there might be a slow decline in the benefits of additional search.

In other $\alpha-\beta$-based game-playing programs, diminishing returns have been demonstrated. For example, in checkers diminishing returns are evident in the *Chinook* program even at shallow search depths, eventually resulting in no/limited benefits for additional search (around 19 ply) [SLSL93]. In Othello, experiments with *Bill* show diminishing returns [LM90].

The results in other games raise a troubling question: Why doesn't chess show diminishing returns? This paper attempts to address this problem.

Our experiments confirm that there are diminishing returns in chess. The reason that these results are not evident in previous work is twofold:

1. Decision Quality: Our experiments show that chess programs have a high error rate in their searches. Plotting probability of an error as a function of search depth confirms that deeper searching reduces errors. Nevertheless, the error rates are still high enough that a deeper searching program has a significant advantage over a shallower searching program.

2. Game Length: Othello games are restricted to 30 moves aside, fewer if you consider the opening book moves and the endgame solvers. Checker games are usually decided in fewer than 30 moves aside; it's rare when a game goes over 50 moves. Chess games, on the other hand, last an average of 40 moves, and 60 move games are not uncommon. Computer self-play games tent to last even longer (>100). Since the deeper searching program has a smaller probability of error on a single move, the longer the game, the greater the chance of the shallow searching program making an error. Thus longer games increase the advantage of the deep searcher. These results are confirmed by playing matches where the game length (number of moves) is restricted.

With an improvement in speed of a factor of two in a chess program, the question arises how best to spend these resources. Traditionally, increased computing resources are used entirely for additional search. This has been an easy course to take since processor speed seems to double every 18 months. However, if diminishing returns do exist, then there will come a time when the performance gains for the additional search effort are small. Programmers will then have to resort to other means for improving performance, such as additional evaluation function knowledge. The strong correlation between program speed and performance has, unfortunately, resulted in a concentration of research effort into search. With diminishing returns for search, chess programmers will have to invest more time on their knowledge if they want their program to improve in a significant way.

2 Previous Work

Ken Thompson pioneered self-play chess program experiments, measuring the effect of an additional ply of search [Tho82]. His original results, roughly 80% winning advantage for the deeper-searching program, translate into a 200 Elo rating point advantage [Elo78]. A later repeat of the experiment show a similar result, however the advantage of the deeper searching program seemed to decrease slightly with increasing depth.

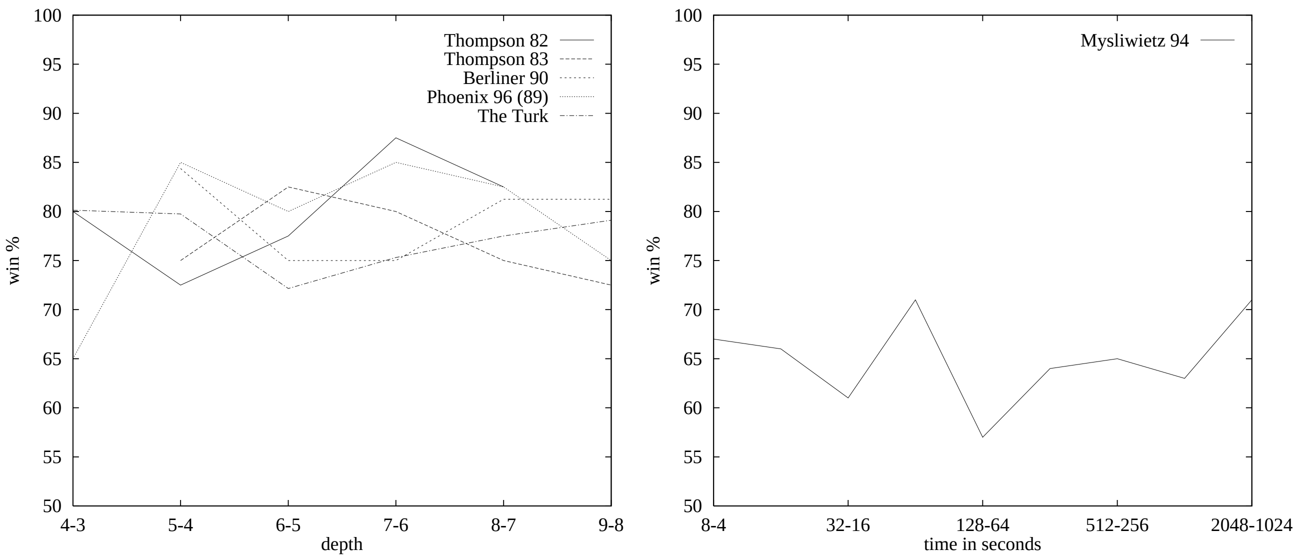


Figure 1: Chess: No Diminishing Returns

Figure 1 shows the results of comparing a number of self-play experiments, including Thompson (*Belle*) [CT83, Tho82], Berliner *et al.* (*Hitech*) [BE90], and Mysliwietz (*Zugzwang*) [Mys94]. Note that the *Zugzwang* results were measured by doubling the program's time. The winning percentage shown is less than those of the other programs, since the advantage of a factor of two in time is less than that of searching an additional ply (roughly 4- to 8-fold time increase in practice). None of these results seem to indicate that there are diminishing benefits for additional search.

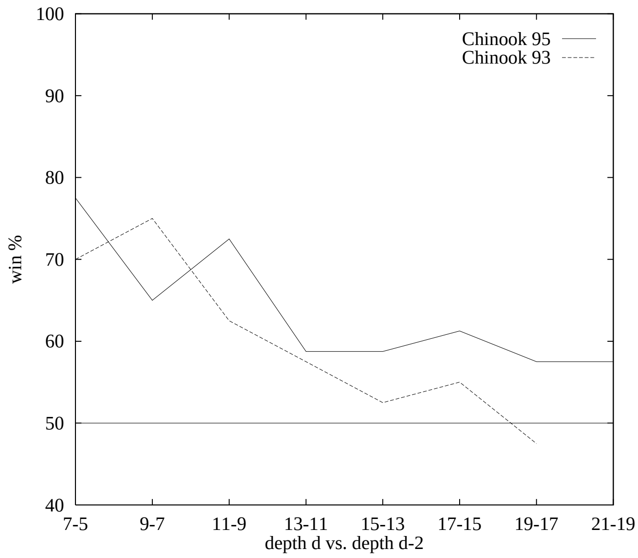


Figure 2: Checkers: Diminishing Returns

To confirm these results, we repeated the experiment in *The Turk*[1]. Figure 1 shows *The Turk*'s results, which are consistent with what every one else has reported: through 9 ply, the benefits for an additional ply of search are significant.

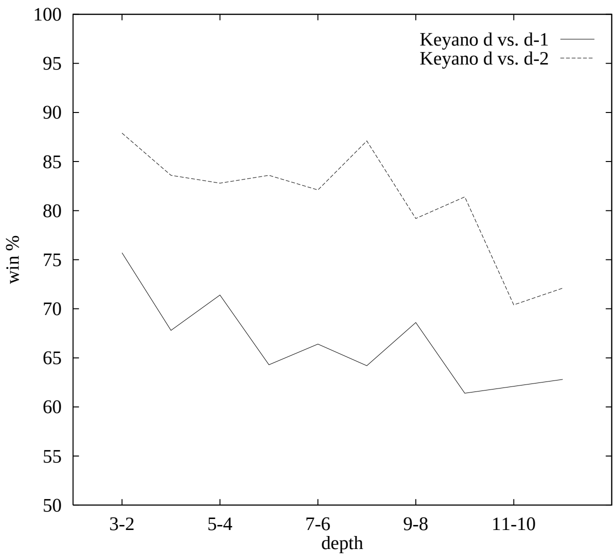
In contrast, results have been reported for other $\alpha-\beta$ search domains that show the benefits of additional search decrease with increased depth. For the checkers program *Chinook*, diminishing search returns were demonstrated even for shallow depths [SLSL93]. A re-run of this experiment with the latest version of *Chinook* is shown in Figure 2. Note that these experiments are the results of self-play between a d and $d+2$-ply searcher, since *Chinook* only searches an odd number of plies deep. These results suggest that *Chinook* is rapidly approaching the point where a faster machine should be used for other things, such as more knowledge, rather than just increasing the number of plies searched[2].

A second observation obtained from the results is that the curves seem to be shifted from the 1993 to the 1995 results. The 1993 *Chinook* contained several bugs that were fixed in the later version. The shifted curve suggests that the better knowledge (because of fewer bugs) delays the diminishing returns for additional search depth.

In the game of Othello, diminishing returns were suggested by several experiments reported for the *Bill* program [LM90]. In Othello, there are two metrics one can use to demonstrate the benefits of additional search: the winning percentage (how often the deeper searcher defeats the shallow searcher), and disk differential (by how many disks a program

[1] *The Turk* was developed at the University of Alberta by Andreas Junghanns and Yngvi Bjornsson.

[2] The reader might notice the '93 19 vs 17 datapoint. In the original paper the authors argue that this data point has no statistical significance.



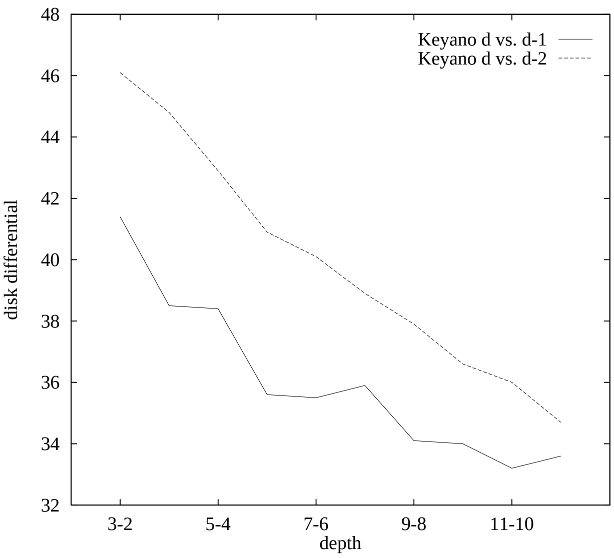


Figure 3: Othello: Diminishing Returns a) winning % b) disk differential

wins). Recent experiments with *Keyano*, one of the top programs in the world[3], demonstrates diminishing returns using both metrics (Figure 3).

Chess has many similarities with Othello and checkers, including an $\alpha - \beta$ program structure using all the same search enhancements. The most noticeable difference is the branching factor, roughly 30-35 in chess, 10 in Othello and 3 in checkers (1.1 in capture positions, 8 in non-capture positions) which translates into different search depths in the same amount of search time. Given similar solution methods, one must conclude the absence of diminishing returns in chess may have something to do with an application-dependent property. If so, then what differences are responsible for this phenomenon?

3 Decision Quality

Searching to depth $d+1$ pays off against a d-ply searcher only if the deeper search reveals new information that causes the program to switch its move choice. For each position, there must be a search depth d' for which additional search will not change the choice of best move. In some positions, this d' depth is quite shallow as, for example, in the case of an obvious best move. In many positions, however, d' is greater than the depths reached by chess programs under tournament time controls. If the combination of search and knowledge allows us to zero in on the best move, then additional search effort is not needed: it only confirms the choice of best move. This line of thought leads us to view the d-ply search as a predictor of the $d+1$-ply search. The better this prediction is, the smaller the gain or

[3] *Keyano* was developed at the University of Alberta by Mark Brockington.

benefit of searching an additional ply will be.

We can design an experiment to measure the quality of a d-ply search as a predictor of a $d+1$-ply search. This is easily done by recording how often the $d+1$-ply search leads to a different move choice than a d-ply search. If the quality of the d-ply predictions improves with additional search depth (deeper searches are better predictors), then the expected gain of a deeper search is necessarily less. Hence we will have demonstrated one form of diminishing returns for deeper searches.

To confirm the hypothesis, the following experiment was performed. *The Turk* was used to analyze roughly 1,000 positions from self-play games. Each position was searched to 9 ply using iterative deepening. The program recorded whether consecutive search depths agreed with the previous iteration's move choice or not. Figure 4 shows the frequency with which a $d+1$-ply search changes the move choice of the d-ply search. The results show that there are fewer move changes as the search depth increases. In other words, deeper searches are better move predictors and the expected benefits of search decrease with increased depth. Interestingly, a 9-ply search changes the best move of the 8-ply search an astonishing 27% of the time[4].

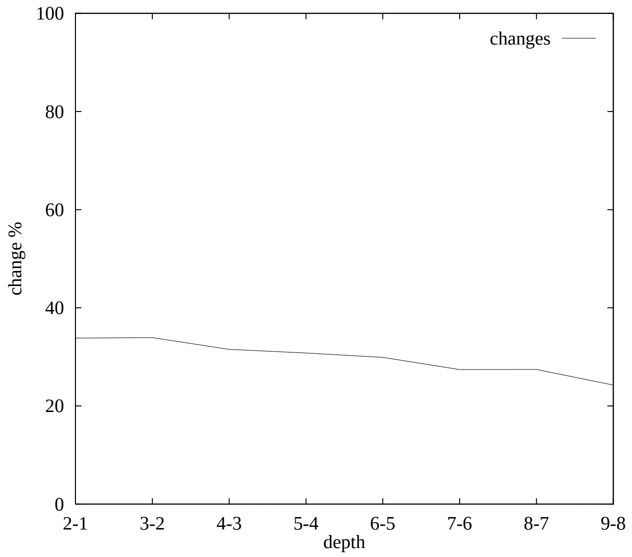


Figure 4: Self-Play Chess Move Changes

A move change may not be a significant event. There may be several equally good moves, and which move happens to be "best" may be the result of a few random factors in the evaluation function. We know that a 9-ply search changes the move of an 8-ply search

[4]A similar experiment was performed using *Phoenix* yielding results that were almost identical. This was surprising to us, since the two programs differ in 1) evaluation function, 2) search engine, 3) quiescence search, and 4) search extensions.

27% of the time, but how often are these changes significant? In general, this is hard to evaluate. Sometimes it is obvious to see that a move change is significant because it results in the win of material. Other times, it can be more subtle: a move change results in a small positional gain that only manifests itself in the long term. Ideally, an oracle would tell us the significance of a move change.

We can attempt to estimate the effect of a move change resulting from deeper search by considering the immediate effect that the move change has on the search value. The data from the previous experiment was filtered to classify move changes based on the magnitude of the search score differential. If a depth d search prefers move m and a $d+1$ search prefers move n, then the score differential is the value of move n at depth $d+1$ compared to m at $d+1$. In other words, we want to quantify how much better the new move is over the previous best move. Figures 5 and 6 show the results for several differentials, where e.g. 25 means the value of a quarter pawn. The results are given for d versus $d+1$ ply (Figure 5, linear and logarithmic scales), and versus the 9-ply (oracle) results (Figure 6, linear and logarithmic scales).

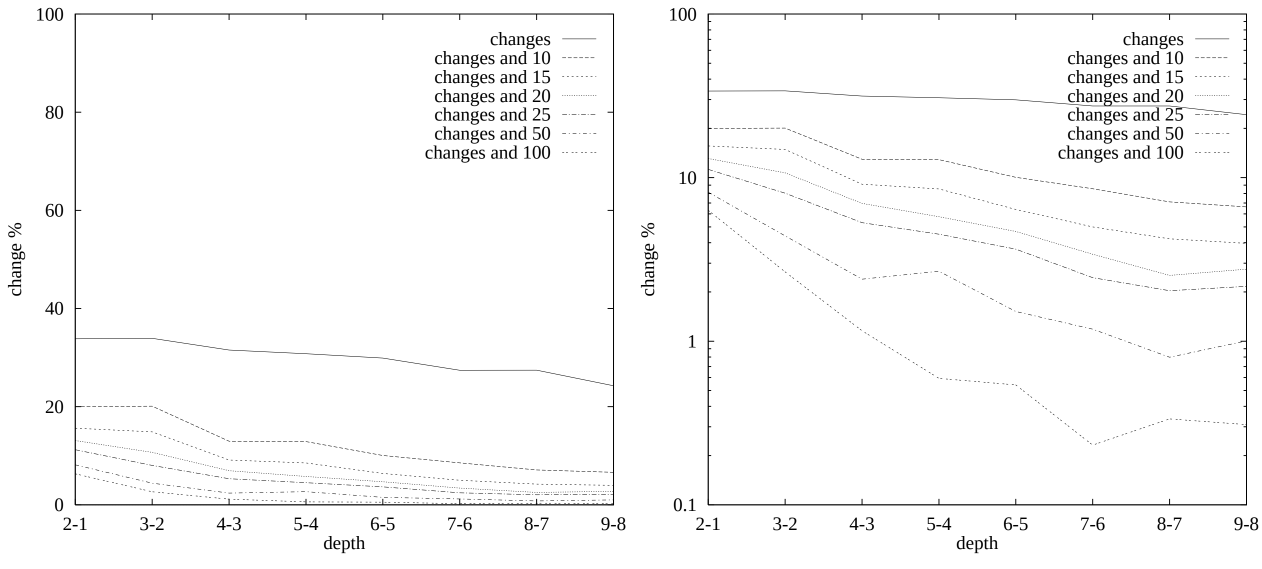


Figure 5: Self-Play Chess, Move and Value Changes d vs $d+1$ in Percentages a) Linear b) Logarithmic

A 9-ply program playing an 8-ply program may change its best move 27% of the time, but only 2.8% of the time is there a move change that results in at least a 25 point (quarter pawn) change in the search score. In other words, in a 100 move game, an 8-ply program will make an average of three moves that a 9-ply program views as being at least a quarter pawn mistake. Roughly 1.5% of the moves result in at least a 50 point advantage to the deeper searcher, and 0.3% result in at least a 100 point advantage.

These numbers might not seem to be high. However, they represent the minimum change

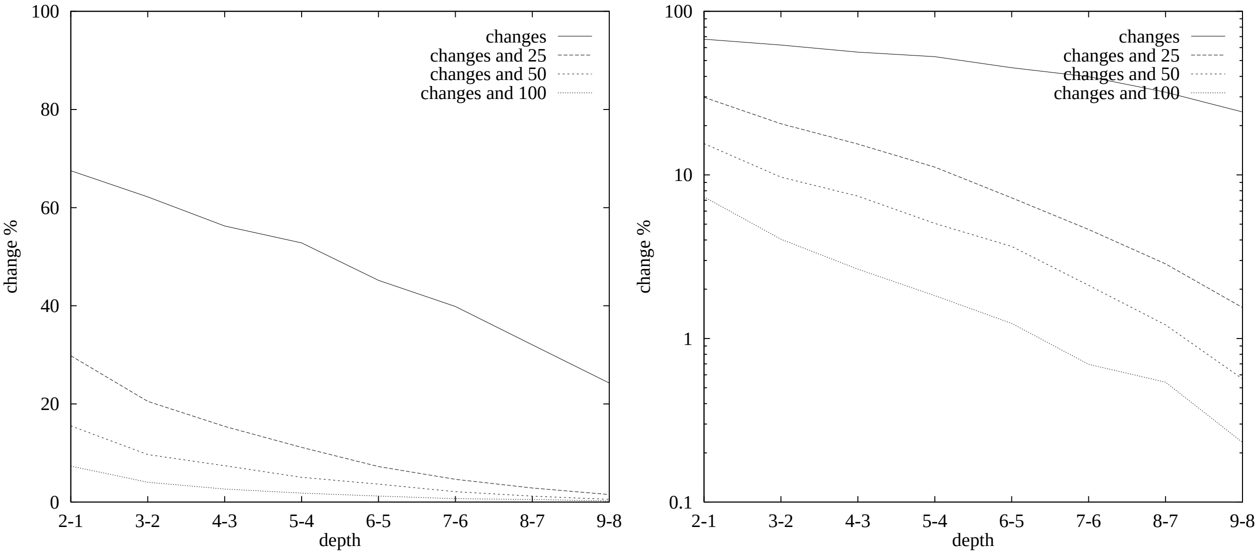


Figure 6: Self-Play Chess, Move and Value Changes d vs 9 ply (Oracle) in Percentages a) Linear b) Logarithmic

for a certain percentage of moves (e.g. 25 points for 3%). If one calculates the expected error for these percentages, the magnitude of these errors gets more apparent. Figure 7 shows what the expected value change for a certain percentage of (the worst) moves is. The expected error for the 3 worst percent of the moves of an 8-ply program judged by a 9-ply program is now almost 37 points (25 before), 1% have 74 points expected change.

Since mistakes in a already lost positions are not as significant as mistakes in an approximatly even position, we filtered the data further to exclude move changes in already lost positions. Figure 8 shows the results. Now the worst 3% of the moves have an expected change 25 points, 1% have 40 points expected change.

Note that these numbers must be properly interpreted. The score differential is based on the score difference of the d-ply and $d+1$-ply programs, as measured *by a d+1-ply search*. Consider, for example, an 8-ply program that chooses move m with a score of +10, and a 9-ply program that chooses move n with a score of 150. The differential is measured by subtracting the 9-ply score of move m from 150.

Are these numbers high or low? One measure can be obtained from the game of checkers, where the *Chinook* program is the World Man-Machine Champion [SLLB96]. Since the program is better than all humans, one would expect its error rate to be lower than for chess. Figure 9 shows that the overall trend in the graphs is the same, however, the curves are shifted, because the error rates are much smaller than in the chess experiments. Although the checker graphs go to greater search depths than for chess, it is important to realize that because of the lower branching factor, searching 20 ply in checkers is comparable computationally to searching 8 ply in chess. The data was obtained from almost 1,000 searches.

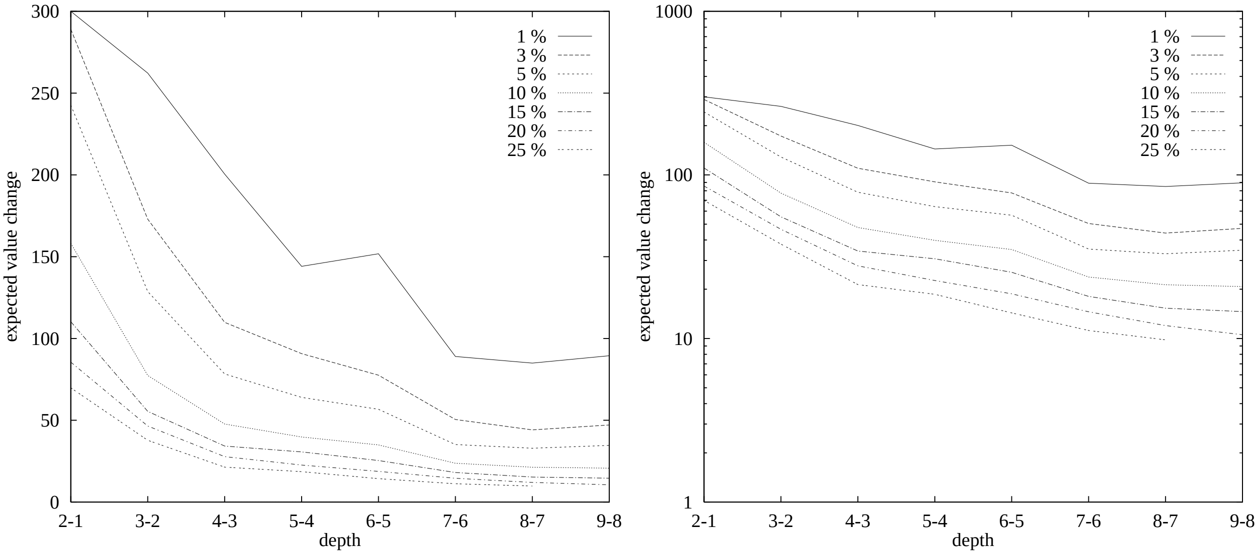


Figure 7: Self-Play Chess, Move and Expected Value Changes a) Linear b) Logarithmic

Surprisingly, the 25/50/75 point differential curves all eventually reach an error rate of 0 (a checker is worth 100 points). The lower error rate may be due to properties of checkers, including the lower branching factor, a "better" program and endgame databases.

The conclusion is that a chess (and checker) program makes fewer errors, as judged by a deeper searching program, as the depth of search increases. Hence, by this metric there are diminishing returns in chess. So, why does this (apparently) not manifest itself in a series of self-play games?

4 Game Length

Games of checkers rarely last longer than 50 moves each side. Othello games are restricted to 30 moves for each side. If you deduct the moves played out of the opening book and those that are solved in the endgame, far fewer moves have to be searched. In contrast, an average chess game lasts 40 moves, and it is not uncommon to see games over 60 moves. In computer self-play games, the games tend to last even longer (> 100).

The decision quality experiment shows that the error rates in chess decrease with search depth. This indicates that the probability of a major error occurring in, for example, a 4-ply versus 3-ply game is much higher than a 9-ply versus 8-ply game. In other words, with additional search depth the error rate reduces and the programs become stronger. Hence, one would expect that games between deeper searching programs would take longer to reach a conclusion than games between shallow searching programs.

The Turk was used to play 80 self-play games for each depth pairing using 40 roughly

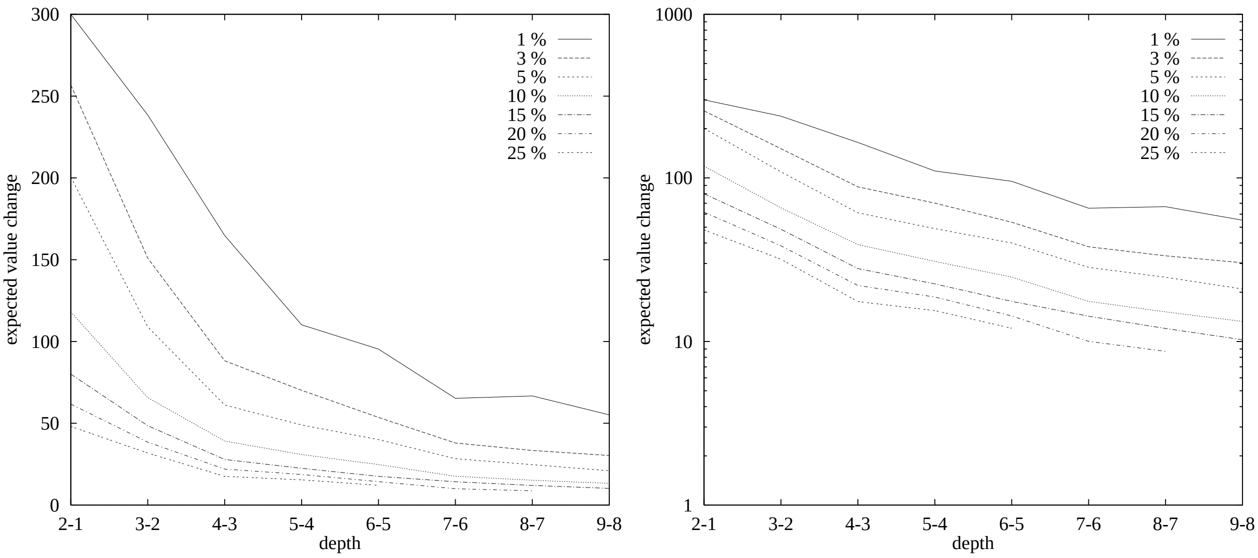


Figure 8: Self-Play Chess, Move and Expected Value Changes for Critical Positions a) Linear b) Logarithmic

even opening positions, each side playing the black and white pieces. The experiments were run up to 9-ply versus 8-ply. Games were adjudicated if one side had a significant advantage (the equivalent of 3 pawns) or move 100 was reached[5]. *The Turk* was modified to do fixed-depth searching (however, leaf node evaluation contained variable-depth capture/checking move quiescence search extensions).

Figure 10 shows the average length of the games played. Games between the 3-ply versus the 2-ply program lasted an average of 72 moves, while games between the 9-ply and 8-ply programs lasted an average of 109 moves[6]. The better the programs, the smaller the chance of an error and, thus, the longer the games will last. This figure suggests diminishing returns since, with deeper search depths, it gets more difficult to win.

The longer the game, the greater the chance that the shallower searching program will make an error. Recall from Section 3 that an 8-ply program playing a 9-ply program makes an error of at least 100 points in 0.3% of the moves, an error of at least 50 points in 1.5% of the moves, and an error of at least 25 points in 2.8% of the moves. In a short game, chances are that the 8-ply program will not make any mistakes, as judged by the 9-ply program. The

[5]Exceptions to this rule were made if one side had a considerable advantage and was expected to win within the next few moves. The following exact parameters were used: If the advantage of the stronger side was less than 50 the game was adjudicated at move 100, otherwise continued. If the advantage of the stronger side was smaller than 100 at move 115, or less than 150 at move 130, or move 150 was reached, the game was terminated.

[6]The game length is unusually long by human standards. This is due, in part to the self-play nature of the experiments: both opponents have the exact same knowledge, allowing each to usually anticipate what the other will play.

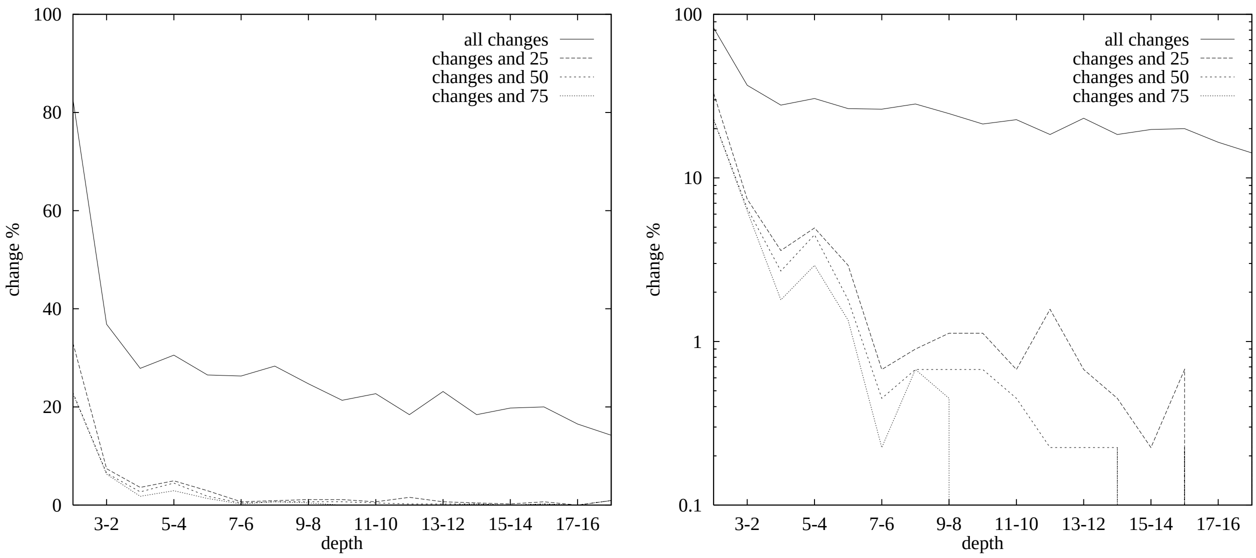


Figure 9: Self-Play Checkers, Move and Value Changes

longer the game lasts, the greater the likelihood that the shallower searching program will make the decisive mistake.

This suggests an interesting experiment: what if the length of the games were artificially limited? Figure 11 shows the results of the self-play matches as judged at various points in the game (10 through 50 and the final result). To make the overall trend more visible, the right graph shows the results smoothed for the same data as in the left (raw data) graph. The results show that when the length of the game is limited, diminishing returns can be clearly seen. The shorter the game, the smaller the chance of a mistake occurring. If the game length is unrestricted (200 moves in our case), the chances of a decisive mistake are almost a certainty. Thus, in Figure 11, the 200 move data point shows no diminishing returns - the likelihood of an error is too great. With shorter games, the probability of an error is reduced, showing smaller gains for additional search depth.

Throughout the discussion, it has been assumed that deeper searching is always beneficial (ignoring search pathology which does not seem to occur in practice [Nau80]). In the self-play games, we occasionally see the unusual result of the shallower searching program defeating a deeper searching program. Often this is the result of the deeper searching program making a commital tactical move, the refutation of which is just beyond the search horizon. Clearly this type of problem occurs more frequently the shallower the search, since deeper searches increase the likelihood of finding the refutation before it is too late. Since our experiments confirm that errors for the deeper searching program are far less likely than for the shallow searcher, we have not made any attempt to differentiate between them.

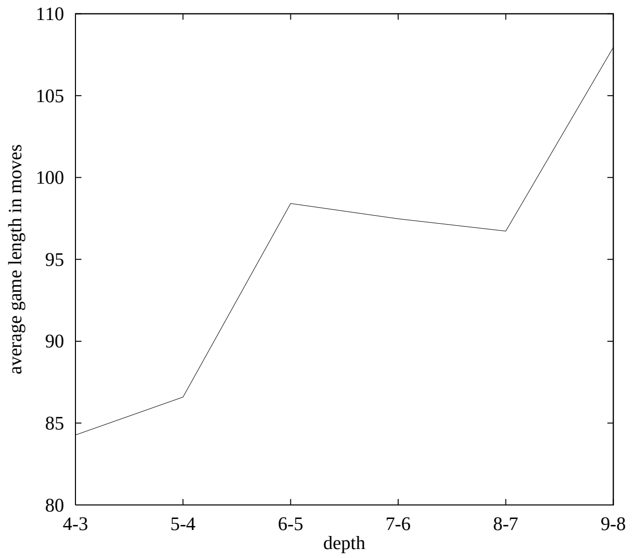


Figure 10: Average Length of Games

5 Conclusion

Contrary to what the literature portrays, chess does exhibit diminishing returns with additional depth of search. Our results suggest that previous work has inadvertently hidden the diminishing returns due to two factors:

- Search Quality: The probability of a program making a serious error is quite high. Our results show that even 8-ply chess programs make many errors, as measured by a 9-ply program. The frequency of these errors indicate that the deeper searching program has a distinct advantage over the shallow searching program.

- Game Length: Given the relatively high error rates, the longer the game the greater the advantage that accrues to the deeper searching program. In effect, every move increases the likelihood of a serious error by the shallow searching program. By limiting the length of the game, diminishing returns can be seen in chess.

In February, 1996, the *Deep Blue* chess machine played the human World Champion, Garry Kasparov. *Deep Blue* was searching roughly 250 million positions per second, at least a factor of 25 in speed beyond any previous chess program. Conventional wisdom says that a factor of 25 translates into an additional two ply of search (perhaps less in *Deep Blue*'s case because of parallel search inefficiency). Previous estimates had the predecessor program, *Deep Thought*, playing somewhere near the 2500-2550 Elo level. *Deep Blue's* performance in the match with Kasparov translates into a roughly 2650 Elo performance rating for the machine. Given the additional computational power of the machine and the small increment

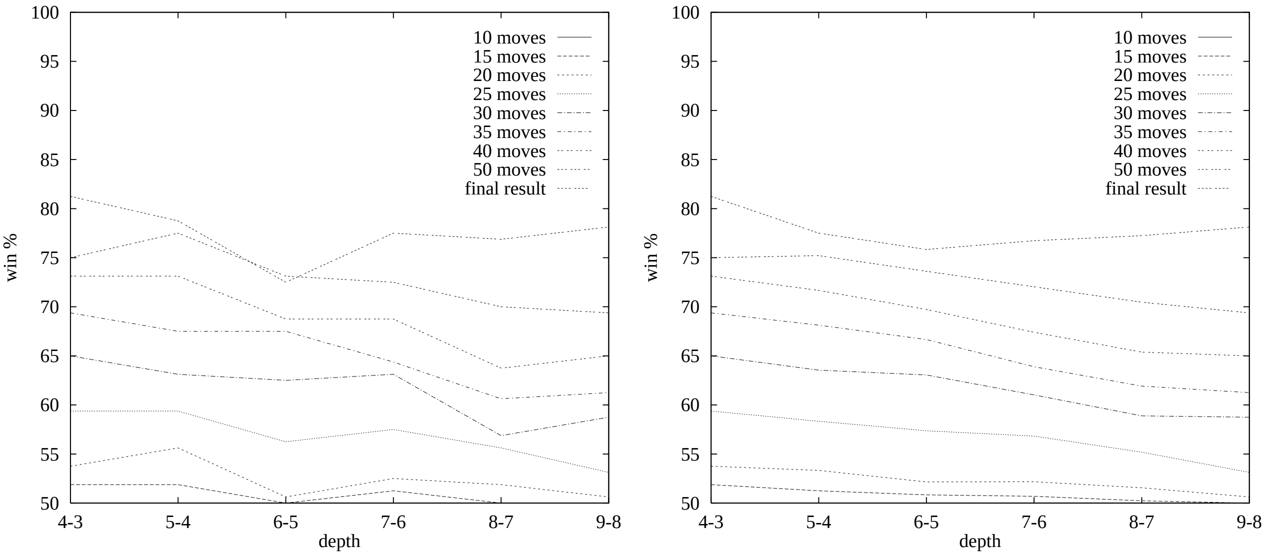


Figure 11: Winning Percentage for Restricted Length Games a) raw b) smoothed

in the Elo rating, one could argue that *Deep Blue* is already exhibiting diminishing returns for the additional search effort.

References

[BE90] H.J. Berliner and Carl Ebeling. Hitech. In T.A. Marsland and J. Schaeffer, editors, *Computers, Chess, and Cognition*, pages 79–109, New York, 1990. Springer-Verlag.

[CT83] J.H. Condon and K. Thompson. Belle. In P.W. Frey, editor, *Chess Skill in Man and Machine*, pages 201–210, New York, 1983. Springer-Verlag.

[Elo78] A. Elo. *The Rating of Chess Players, Past and Present.* Batsford Ltd., London, 1978.

[Fel93] R. Feldmann. *Game Tree Search on Massively Parallel Systems.* PhD thesis, University of Paderborn, Paderborn, Germany, August 1993.

[HCN90a] F. Hsu, T.S. Anantharaman M.S. Campbell, and A. Nowatzyk. Deep Thought. In T.A. Marsland and J. Schaeffer, editors, *Computers, Chess, and Cognition*, pages 55–78, New York, 1990. Springer-Verlag.

[HCN90b] F. Hsu, T.S. Anantharaman M.S. Campbell, and A. Nowatzyk. A Grandmaster Chess Machine. *Scientific American*, 263(4):44–50, October 1990.

[Kus94] B.C. Kuszmaul. *Synchronized MIMD Computing.* PhD thesis, Massachusetts Institute of Technology, May 1994.

[LM90] K.-F. Lee and S. Mahajan. The Development of a World Class Othello Program. *Artificial Intelligence*, 43(1):21–36, 1990.

[Mys94] P. Mysliwietz. *Konstruktion und Optimierung von Bewertungsfunktionen beim Schach.* PhD thesis, University of Paderborn, Paderborn, Germany, January 1994.

[Nau80] D.S. Nau. Pathology on Game Trees: A Summary of Results. In *Proc. First National Conf. AI*, pages 102–104, Stanford, CA, 1980.

[Sch86] J. Schaeffer. *Experiments in Search and Knowledge.* PhD thesis, Univ. of Waterloo, Canada, 1986.

[SLLB96] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook: The World Man-Machine Checkers Champion. *AI Magazine*, 17(1):21–29, 1996.

[SLSL93] J. Schaeffer, P. Lu, D. Szafron, and R. Lake. A Re-examination of Brute-force Search. In *Games: Planning and Learning: AAAI 1993 Fall Symposium, Report FS9302*, pages 51–58, Chapel Hill, N.C., 1993.

[Tho82] K. Thompson. Computer Chess Strength. In M.R.B. Clarke, editor, *Advances in Computer Chess 3*, pages 55–56, Oxford, UK, 1982. Pergamon Press.