Forward Estimation for Game-Tree Search *

Weixiong Zhang
Information Sciences Institute and
Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
Email: zhang@isi.edu

Abstract

It is known that bounds on the minimax values of nodes in a game tree can be used to reduce the computational complexity of minimax search for two-player games. We describe a very simple method to estimate bounds on the minimax values of interior nodes of a game tree, and use the bounds to improve minimax search. The new algorithm, called forward estimation, does not require additional domain knowledge other than a static node evaluation function, and has small constant overhead per node expansion. We also propose a variation of forward estimation, which provides a tradeoff between computational complexity and decision quality. Our experimental results show that forward estimation outperforms alpha-beta pruning on random game trees and the game of Othello.

1. Introduction

A game between two players, MIN and MAX, can be represented by a tree with MIN nodes followed by MAX nodes, and vice versa. It is usually infeasible to search to the end of a game and then make a move, due to the size of the game tree and the time limit on each move.

The prevailing idea to overcome this difficulty is full-width, fixed-depth Minimax, which takes the nodes at a fixed depth as terminal nodes and backs up their static evaluations to the root by minimax rules. In spite of the pathology of deep search on some game trees (Beal 1980; Nau 1982), searching deeper usually strengthens a play in practice. For example, there is almost a linear correlation between the search depth and the rating of a chess program, and each additional ply adds about 200 rating points to the playing strength (Hsu *et al.* 1990).

Algorithms that compute exact minimax value include alpha-beta pruning (Knuth & Moore 1975), SSS^* (Stockman 1979), Scout (Pearl 1984), and aspiration alpha-beta (Kaindl, Shams, & Horacek 1991). These algorithms all reduce the computational complexity or effective branching factor of full-width, fixed-depth Minimax. For example, on a random tree with branching factor b, the effective branching factor of alpha-beta pruning is roughly $b^{0.747}$ (Pearl 1984). Because of their simplicity and efficiency, full-width, fixed-depth Minimax algorithms, especially alpha-beta pruning and aspiration alpha-beta, are the dominating algorithms in practice.

In contrast, selective search methods selectively explore some promising avenues deeper than a fixed depth. A recent study (Smith & Nau 1994) suggests that selective search (called forward pruning in (Smith & Nau 1994)) may possibly be useful on games when there is a high correlation among the values of sibling nodes in the game tree. Many selective algorithms (Berliner 1979; McAllester 1988; Rivest 1987; Russell & Wefald 1989), however, are difficult to apply or have large overhead. Two selective search algorithms that have been used or tested on real games are singular extension (Anantharaman, Campbell, & Hsu 1990) and best-first minimax search (Korf & Chickering to appear). In order to be effective, these two algorithms run alpha-beta pruning first to some depth and then extend some lines of play further if they seem to have more or direct impact on the outcome. Research on selective search was inspired by long standing observation that heuristic information on interior nodes of a game tree can improve the efficiency of minimax search.

A heuristic node evaluation can either be a single value measuring the merit of a node, or a bound on the minimax value of a node. Among these two kinds of evaluation functions, the interval-valued function has attracted a great deal of attention. Beliner (Berliner 1979) first proposed to use an interval-valued evaluation and developed the B^* algorithm. Ibaraki (Ibaraki 1986) and Pijls and Bruin (Pijls & de Bruin Dec 1992) considered how to improve alpha-beta pruning and SSS^* algorithms with interval-valued evaluations. However, these algorithms have not been used in practice, because of large overhead or lack of appropriate interval-valued functions that can be computed efficiently. Aspiration alpha-beta is a simple algorithm using interval-valued evaluations. It first estimates a

*This research was partially supported by NFS Grant, IRI-9119825, and by ARPA Contract, MDA972-94-2-0010.

bound on the minimax value of a game tree, which is called the aspiration window. It then runs alpha-beta pruning using the aspiration window as an alpha-beta bound. One method to derive an aspiration window is iterative-deepening (Korf 1985): searching a tree in successively deeper and deeper iterations, and using information from the previous iteration to estimate an aspiration window for the current iteration. However, if the minimax value of the tree is not within the aspiration window, the tree needs to be re-searched with a modified aspiration window.

We present a very simple method to estimate a bound on the minimax value of a node in a game tree without additional domain knowledge other than a static node evaluation function. Specifically, this method estimates the *minimal* and *maximal* possible values of the minimax value of a node. This estimation can be applied not only to the root node, but also to the interior nodes of a game tree. We then introduce this estimation method into alpha-beta pruning. The new algorithm, called *forward estimation*, has only a small constant overhead per node expansion.

In order to explain the idea more clearly, we first present forward estimation on a random tree in which edges have costs and a node cost is computed as the sum of the edge costs on the path from the root to the node (Section 2). Due to space limit, we assume that the reader is familiar with Minimax search and alpha-beta pruning (Pearl 1984). Our experimental results show that the effective branching factor of forward estimation is smaller than that of alpha-beta pruning with perfect node ordering on random trees. We then discuss how to apply the new algorithm when there is no information on edge costs (Section 3). We further discuss how to extend the search horizon by searching more selectively using forward estimation (Section 4). Our experimental results show that forward estimation outperforms alpha-beta pruning on games on random trees and the game of Othello (Section 5). Finally, our conclusions appear in Section 6.

2. Forward estimation

To obtain the minimax value of a node in a game tree without search is generally infeasible. Rather than directly estimating the minimax value of a node, our idea is to estimate the minimal and maximal possible values of the minimax value. The idea is based on the following tree model. *An incremental random tree, or random tree for short, is a tree with depth d, finite random branching factor with mean b, and finite random edge costs. The root has cost 0, and the cost of a node is the sum of the edge costs from the root to that node.* One important feature of random trees is that they naturally introduce a correlation among the costs of the nodes that share common edges on the path from the root to them. One advantage of using random trees is that they are easily reproducible (Korf, Pemberton, & Zhang 1994). Random trees have been used as an analytical model and testbed for both two-agent search (Fuller, Gaschnig, & Gillogly 1973; Korf & Chickering to appear; Nau 1982) and single-agent search (Karp & Pearl 1983; Korf, Pemberton, & Zhang 1994; McDiarmid & Provan 1991; Zhang & Korf 1992; 1993; 1995).

Consider a subtree with depth d and root cost c. If the minimal and maximal edge costs are l and u, then the minimax value of the subtree must be in the range of $[c + l * d, c + u * d]$.

How can we use the estimated bound on a minimax value in alpha-beta pruning? Alpha-beta pruning uses two bounds, α and β, to search the subtree of a fixed depth under a node. The α bound is the maximum of the current minimax values of all MAX node ancestors of the node, and the β bound is the minimum of the current minimax values of all MIN node ancestors of the node. Search of a MIN node and its subtree can be abandoned if its current minimax value, obtained after searching some of its children, is less than or equal to α. This is simply because a MIN node can only reduce its minimax value, but its MAX node ancestors always want to increase their minimax values, so that the MAX player can always choose another line of play that leads to a better outcome for MAX. This is alpha pruning. Likewise, searching a MAX node can be abandoned if its current minimax value is greater than or equal to β. This is beta pruning.

For a subtree with a MIN root node, if the maximal possible value $c+u*d$ is less than or equal to its current alpha bound α, then searching this subtree does not affect the minimax value, and thus it does not need to be searched. This is because the minimax value will not exceed $c + u * d$ which is not greater than α. Similarly, for a subtree with a MAX root node, if the minimal possible value $c + l * d$ is greater than or equal to its current beta bound β, then this subtree does not need to be searched either. In other words, a subtree can be abandoned if

$$\begin{cases} c + u * d \leq \alpha; & \text{if a MIN root node} \\ c + l * d \geq \beta; & \text{if a MAX root node} \end{cases} \quad (1)$$

where c is the cost of the root node of the subtree, d is the depth of the subtree, and u and l are the minimal and maximal edge costs. We call alpha-beta pruning plus the new pruning conditions in (1) *forward estimation.*

At this point, one question we have to answer is how much can we gain from forward estimation? To answer the question, we first experimentally examined forward estimation on random trees. We chose edge costs independently and uniformly from integers $\{-2^{15}+1, -2^{15}+2, \cdots, 2^{15}\}$ in order to generate game trees impartial to the MIN and MAX players. Fig. 1 shows our results on uniform trees, averaged over 1000 random trials. The horizontal axis is the tree depth, and the vertical axis the number of nodes generated, in a logarithmic scale. We used the total number of

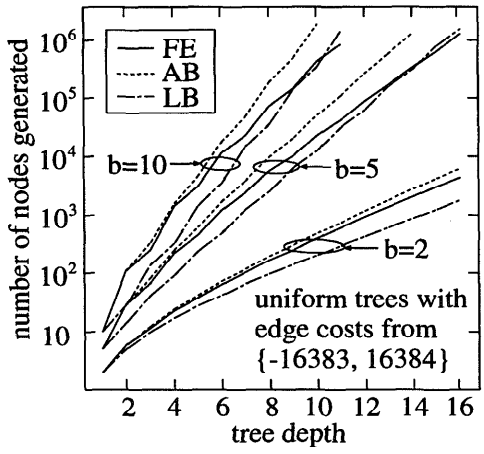

Figure 1: Forward estimation vs. alpha-beta pruning on random game trees.

node generations as a complexity measure, because the overhead of forward estimation is a small, negligible constant for each node generated. We compared forward estimation (FE) to alpha-beta pruning (AB). We also plotted the number of nodes generated by alpha-beta pruning with perfect node ordering (LB) (Knuth & Moore 1975) for comparison.

From Fig. 1, we can simply conclude that forward estimation reduces the effective branching factor of alpha-beta pruning. This reduced effective branching factor is even smaller than that of alpha-beta pruning with perfect node ordering. Therefore, forward estimation searches deeper with the same amount of computation as alpha-beta pruning, or runs faster than alpha-beta pruning on a fixed-depth tree. For instance, with one million node generations, alpha-beta pruning cannot reach depth 10 on a uniform tree with branching factor 10 on average, while forward estimation can search to depth 11 on average.

3. Learning edge costs

One may argue that the edges in a search tree do not have costs. The assumption that edges have costs is not a real restriction. With a static node evaluation function, we can compute edge costs. The cost of an edge is the difference between the cost of a child node and the cost of its parent which are connected by the edge.

In practice, there are three methods to obtain edge costs and to compute their minimal and maximal values. The first is to analyze the static evaluation function and further derive bounds on the minimal and maximal edge costs.[1] This method requires domain knowledge. The second is to learn the minimal and maximal edge costs by sampling a few game trees. We call this method off-line learning. The third method is to learn the minimal and maximal edge costs on-line or by sampling during the search. Learning the minimal and maximal edge costs can be simply done by keeping track of the minimum and maximum of all edge costs that have been encountered during a search. Let l' and u' be the learned minimal and maximal edge costs. We can use l' and u' in the conditions of (1). Unfortunately, this may affect decision quality. If the exact minimal and maximal edge costs, l and u, are known *a priori*, it is guaranteed that forward estimation produces exactly the same minimax value as alpha-beta pruning. Since the learned l' is usually greater than l and u' is usually less than u, the conditions in (1) are easier to satisfy. Consequently, the node that has the true minimax value may be pruned, and a different minimax value may be obtained.

To understand how sensitive the decision quality is to the on-line and off-line learning schemes, we tested them on random trees. We used trees of many different depths, with uniform and random branching factors. Our experiments were done as follows. On a tree with depth d, we first searched to the end using alpha-beta pruning, and recorded the first move at the root that alpha-beta pruning would take. Call this move *move*. We then searched the tree to different depths up to d, using forward estimation with exact edge-cost bounds, assuming that they were known, and forward estimation with learned edge-cost bounds. We then compared their first moves at the root to *move*. We generated 1000 random trees, and computed the percentage of times that they made the same first moves as a full depth alpha-beta pruning. The decision qualities of forward estimation with the exact bounds and learned bounds are the same most of the time, and forward estimation with learned bounds generates a few nodes less than forward estimation using exact bounds.

The main reason that forward estimation is not very sensitive to learned bounds on random trees may be that edge costs are independently chosen from a common distribution. In addition, forward estimation only uses the conservative minimal and maximal edge costs so that the additional pruning power introduced is limited. Furthermore, at the beginning of the search with the on-line learning scheme, although the learned edge-cost bounds are not very accurate, the alpha-beta bound is large enough such that the chance to satisfy the conditions in (1) may be small.

4. Forward estimation as selective search

Alpha-beta pruning is still not very efficient, in terms of number of node generations, because it finds the *exact* minimax value of a node. It spends a lot of computation to find the leaf node that has the exact minimax value among the leaf nodes whose costs are close to the exact minimax value. On an incremental tree, it is most likely that the values of leaf nodes are congregated, meaning that the cost of a leaf node

[1] This was suggested by Rich Korf.

and the costs of its siblings or even the costs of the children of its grandparent are not too different from each other, because they share many common edges on their paths to the root. This suggests that some subtrees have more impact on the decision at the root, and thus they deserve more computation.

Forward estimation can be viewed as a selective search algorithm, since it may prune a node before searching it. Can we further improve its pruning power? In conditions (1), we used the maximal possible increment, $u * d$, and the maximal possible decrement, $l * d$, of the costs of leaf nodes relative to the root cost of a subtree, which were very conservative. Instead of using the maximal and minimal edge costs, we may use the most likely increment, u'', and the most likely decrement, l'', of edge costs. These two quantities can also be learned by the learning schemes discussed in Section 3. Therefore, conditions (1) become

$$\begin{cases} c + u'' * d \leq \alpha; & \text{if a MIN root node} \\ c + l'' * d \geq \beta; & \text{if a MAX root node} \end{cases} \quad (2)$$

where c is the cost of the root node of a subtree, and d is the depth of the subtree.

One "quick and dirty" way to estimate u'' and l'' is to introduce a parameter $\delta \in [0, 1]$. We can simply use $u'' = u * \delta$ and $l'' = l * \delta$ in conditions (2). Then (2) becomes

$$\begin{cases} c + (u * \delta) * d \leq \alpha; & \text{if a MIN root node} \\ c + (l * \delta) * d \geq \beta; & \text{if a MAX root node} \end{cases} \quad (3)$$

where c is the cost of the root node of a subtree, d is the depth of the subtree, and u and l are the minimal and maximal edge costs. Conditions in (3) are satisfied when those in (1) are satisfied, since $\delta \leq 1$. The first pruning condition of (3) can be explained as follows. If the maximal possible minimax value of a MIN node is close enough to its α bound from above, it is then unlikely that the exact minimax value will be greater than α. Similarly, the second condition can be explained as follows. If the minimal possible minimax value of a MAX node is close enough to β from below, it is unlikely that the exact minimax value will be smaller than β. With a smaller δ, we can prune more nodes and search deeper with the same amount of computation. However, by introducing δ, we also run the risk of making a wrong move. There is a tradeoff between making a decision based on the exact minimax value from a particular depth and that based on a near exact minimax value from a deeper depth. This tradeoff can be tuned by parameter δ.

Fig. 2 shows our experimental results on random trees with depth 10, random branching factor of mean 5, and edge costs from $\{-2^{15}+1, -2^{15}+2, \cdots, 2^{15}\}$. In the experiments, the on-line learning method and pruning conditions of (3) were used. The horizontal axis of Fig. 2 is the total number of node generations, in a logarithmic scale, and the vertical axis the decision

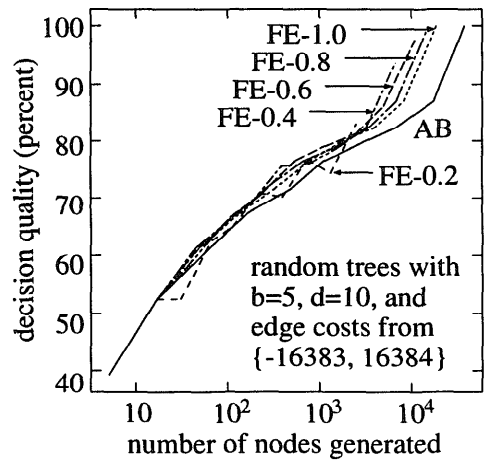

Figure 2: Forward estimation as selective search.

quality in terms of percentage of times that a search makes the same first moves as full-depth Minimax. The results are averaged over 1000 trials. The curve labeled AB is for alpha-beta pruning, and the curves labeled $FE-1.0$, $FE-0.8$, $FE-0.6$, $FE-0.4$, and $FE-0.2$ correspond to forward estimation with $\delta = 1.0$, $\delta = 0.8$, $\delta = 0.6$, $\delta = 0.4$, and $\delta = 0.2$, respectively.

Fig. 2 indicates that forward estimation with $\delta = 1$ expands fewer nodes than alpha-beta pruning and makes optimal decisions when searching to the end of the tree. However, forward estimation with $\delta < 1$ may not make optimal decisions because of the decision error introduced by δ, as predicted. Furthermore, when δ is too small, for instance $\delta = 0.2$ in Fig. 2, the saving in computation for a deep search cannot pay off the loss in decision quality. For random trees with uniformly distributed edge costs, a median δ, such as $\delta = 0.5$, will be a good choice. Of course, the value of δ depends upon problem domains.

5. Playing games

To better understand forward estimation, we played it against alpha-beta pruning on random trees and the game of Othello.

We adopt the game playing rules suggested in (Korf & Chickering to appear): Every game is played twice, with each player alternately playing MAX. A play of the game consists of a sequence of alternating moves by each player until a leaf node is reached. The static value of this leaf is the outcome of the game. The winner of a pair of games is the player that played MAX when the larger outcome was obtained. If the outcome is the same in the two games, the pair of games is declared a tie. A tournament consisted of a number of pairs of games. We played forward estimation against alpha-beta pruning, with each algorithm initially searching to depth one. Whichever algorithm generated the fewest total nodes in the last tourna-

ment had its search horizon incremented by one in the next tournament.

5.1. Random game trees

On random trees, a tournament consisted of 100 pairs of random games. These trees have random branching factors with different mean, and edge costs uniformly and independently chosen from $\{-2^{15}+1, -2^{15}+2, \cdots, 2^{15}\}$. We used the on-line learning scheme to obtain the minimal and maximal edge costs, and pruning conditions of (3). Fig. 3(a) shows the results on random trees with mean branching factor 5. The horizontal axis is the lookahead depth of alpha-beta pruning. The vertical axis is the rate that forward estimation wins over alpha-beta pruning, as a percentage. We included the results when $\delta = 1.0$ and $\delta = 0.5$. Forward estimation with $\delta = 1.0$ (curve FE-1.0 in Fig. 3(a)) can search only one level deeper and generates more nodes than alpha-beta pruning. For forward estimation with $\delta = 0.5$, we report two results. One is when forward estimation searches deeper but generates more nodes than alpha-beta pruning the first time, as shown by curve FE-0.5(MORE) in Fig. 3(a). The other is when forward estimation searches to a depth one level shallower than the depth at which forward estimation generates more nodes than alpha-beta pruning, as shown by curve FE-0.5(LESS). Forward estimation with $\delta = 0.5$ searches to the same depth as alpha-beta but with less node generations than alpha-beta when the alpha-beta search horizon is less than or equal to 3. Since δ introduces decision errors, forward estimation loses to or does not win with a large margin over alpha-beta pruning when both algorithms search to the same depth. However, forward estimation can search one level deeper with less node generations than alpha-beta when the alpha-beta search horizon is greater than 3. Fig. 3(a) indicates that forward estimation is superior to alpha-beta in deep search.

5.2. The game of Othello

In the experiments on the game of Othello, we used a static evaluation function from the software Bill, which won the first place in the 1989 North American Computer Othello Championship (Lee & Mahajan 1990).

We used both on-line and off-line learning schemes to obtain the minimal and maximal edge costs. The on-line scheme did not perform adequately and needs further investigation. We conjecture that this is mostly due to the locality feature of on-line learning, plus the correlations among node costs.

We sampled 10 games in the off-line learning scheme. The maximal and minimal edge costs learned were $u = 2993448$ and $l = -2680916$, and the most likely edge-cost increment and decrement were around $u' = 59869$ and $l' = -53618$. We used these two sets of edge costs, along with pruning conditions of (1) and (2), in a tournament consisted of 244 pairs of games that were generated by making all possible first four moves. When

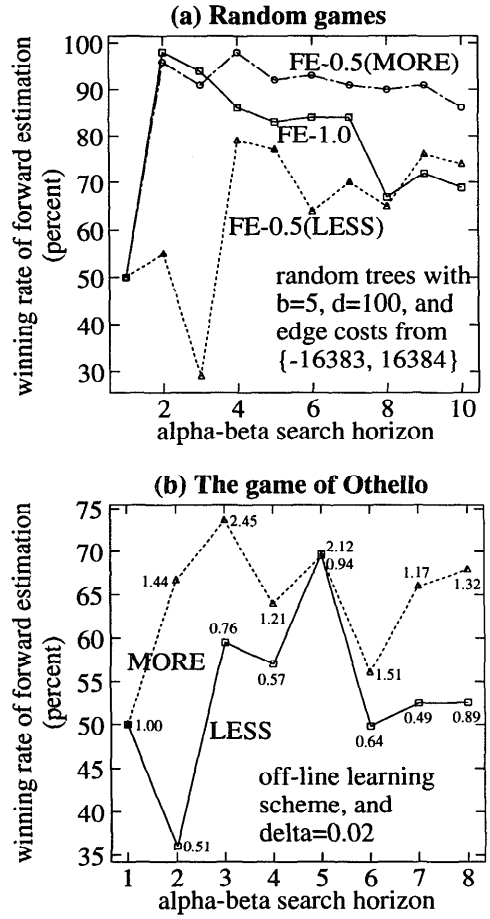

Figure 3: Forward estimation plays against alpha-beta pruning.

the maximal and minimal edge costs and conditions (1) were used, forward estimation played almost the same as alpha-beta pruning. The reason is that these minimal and maximal learned edge costs were extreme values that rarely occur.

Fig. 3(b) shows the results when the most likely edge-cost increment and decrement plus conditions (2) were used. The horizontal axis is the lookahead depth of alpha-beta pruning, and the vertical axis is the winning rate of forward estimation, as a percentage. The complexity measure for the game of Othello is the average CPU time per game on a Sun Sparc 10 machine. We report two sets of experimental results. One is when forward estimation searches deeper but generates more nodes than alpha-beta pruning the first time, as shown by the dashed curve MORE in Fig. 3(b), and the other when forward estimation searches to a depth one level shallower than the depth at which forward estimation generates more nodes than alpha-beta pruning, as shown by the solid curve LESS in Fig. 3(b). Forward estimation searches to the same depth as alpha-beta

pruning with fewer node generations when the alpha-beta lookahead depth is less than three, but can reach three levels deeper than alpha-beta pruning when the alpha-beta lookahead depth is eight. In Fig. 3(b), a number next to a data point is the ratio of the average CPU time per game using forward estimation to that of alpha-beta pruning. Fig. 3(b) shows that forward estimation outperforms alpha-beta pruning with less computation when the alpha-beta lookahead depth is greater than two.

6. Conclusions

We presented a very simple method to estimate a bound of the minimax value of a node in a game tree without additional domain knowledge other than a static node evaluation function. We introduced this estimation method into alpha-beta pruning and developed a new algorithm called forward estimation. Forward estimation has only a small constant overhead per node expansion. We also proposed a variation of the original algorithm, which provides a tradeoff between computational complexity and decision quality. Finally, we tested forward estimation on games on random trees and the game of Othello. The experimental results show that forward estimation outperforms alpha-beta pruning on both of these games.

Acknowledgements

The author is grateful to David Chickering for the code for the game of Othello, to Sheila Coyazo for editorial help, to Kai-Fu Lee for the static evaluation function of Bill, to Ramesh Patil for comments on a draft, and to Armand Prieditis for discussions. Special thanks to Rich Korf for encouragement, discussions, comments, and his game-playing code on random trees and the game of Othello.

References

Anantharaman, T.; Campbell, M.; and Hsu, F.-H. 1990. Singular extensions: Adding selectivity to brute-force searching. *Artificial Intelligence* 43:99–109.

Beal, D. 1980. An analysis of minimax. In *Advances in Computer Chess*. Edinburgh: Edinburgh University Press.

Berliner, H. 1979. The B^* tree search algorithm: A best-first proof procedure. *Artificial Intelligence* 12:23–40.

Fuller, S.; Gaschnig, J.; and Gillogly, J. 1973. Analysis of the alpha-beta pruning algorithm. Technical report, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA.

Hsu, F.-H.; Anantharaman, T.; Campbell, M.; and Nowatzyk, A. 1990. A grandmaster chess machine. *Scientific American* 263:44–50.

Ibaraki, T. 1986. Generalization of alpha-beta and SSS^* search problems. *Artificial Intelligence* 29:73–117.

Kaindl, H.; Shams, R.; and Horacek, H. 1991. Minimax search algorithms with and without aspiration windows. *IEEE Trans. Pattern Analysis and Machine Intelligence* 13:1225–1235.

Karp, R., and Pearl, J. 1983. Searching for an optimal path in a tree with random costs. *Artificial Intelligence* 21:99–117.

Knuth, D., and Moore, R. 1975. An analysis of alpha-beta pruning. *Artificial Intelligence* 6:293–326.

Korf, R., and Chickering, D. to appear. Best-first minimax search. *Artificial Intelligence*.

Korf, R.; Pemberton, J.; and Zhang, W. 1994. Incremental random search trees. In *Working Notes of AAAI 1994 Workshop on Experimental Evaluation of Reasoning and Search Methods.*

Korf, R. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27:97–109.

Lee, K.-F., and Mahajan, S. 1990. The development of a world-class Othello program. *Artificial Intelligence* 43:21–36.

McAllester, D. 1988. Conspiracy numbers for minmax search. *Artificial Intelligence* 35:287–310.

McDiarmid, C., and Provan, G. 1991. An expected-cost analysis of backtracking and non-backtracking algorithms. In *Proc. IJCAI-91*, 172–177.

Nau, D. 1982. An investigation of the causes of pathology in games. *Artificial Intelligence* 19:257–278.

Pearl, J. 1984. *Heuristics*. Reading, MA: Addison-Wesley.

Pijls, W., and de Bruin, A. Dec. 1992. Searching informed game trees. In *Proceedings of the* 3^{rd} *Intern. Symp. on Algorithms and Computation*, 332–341.

Rivest, R. 1987. Game tree searching by min/max approximation. *Artificial Intelligence* 34:77–96.

Russell, S., and Wefald, E. 1989. On optimal game-tree search using rational meta-reasoning. In *Proc. IJCAI-89*, 334–340.

Smith, S., and Nau, D. 1994. An analysis of forward pruning. In *Proc. AAAI-94*, 1386–1391.

Stockman, G. 1979. A minimax algorithm better than alpha-beta. *Artificial Intelligence* 12:179–196.

Zhang, W., and Korf, R. 1992. An average-case analysis of branch-and-bound with applications: Summary of results. In *Proc. AAAI-92*, 545–550.

Zhang, W., and Korf, R. 1993. Depth-first vs. best-first search: New results. In *Proc. AAAI-93*, 769–775.

Zhang, W., and Korf, R. 1995. Performance of linear-space search algorithms. *Artificial Intelligence* 79:241–292.