Properties of Forward Pruning in Game-Tree Search

Yew Jin Lim and **Wee Sun Lee**
School of Computing
National University of Singapore
{limyewji,leews}@comp.nus.edu.sg

Abstract

Forward pruning, or selectively searching a subset of moves, is now commonly used in game-playing programs to reduce the number of nodes searched with manageable risk. Forward pruning techniques should consider how pruning errors in a game-tree search propagate to the root to minimize the risk of making errors. In this paper, we explore forward pruning using theoretical analyses and Monte Carlo simulations and report on two findings. Firstly, we find that pruning errors propagate differently depending on the player to move, and show that pruning errors on the opponent's moves are potentially more serious than pruning errors on the player's own moves. This suggests that pruning on the player's own move can be performed more aggressively compared to pruning on the opponent's move. Secondly, we examine the ability of the minimax search to filter away pruning errors and give bounds on the rate of error propagation to the root. We find that if the rate of pruning error is kept constant, the growth of errors with the depth of the tree dominates the filtering effect, therefore suggesting that pruning should be done more aggressively near the root and less aggressively near the leaves.

Introduction

The Alpha-Beta ($\alpha\beta$) algorithm (Knuth & Moore 1975) is the standard approach used in game-tree search to explore all combinations of moves to some fixed depth. However, even with $\alpha\beta$ pruning, search complexity still grows exponentially with increasing search depth. To further reduce the number of nodes searched, practical game-playing programs perform forward pruning (Marsland 1986; Buro 1995; Heinz 1999), where a node is discarded without searching beyond that node if it is believed that the node is unlikely to affect the final minimax value of the node. As all forward pruning techniques inevitably have a non-zero probability of making pruning errors, employing any forward pruning technique requires a compromise between accepting some risk of error and pruning more in order to search deeper.

Forward pruning techniques should therefore consider how pruning errors propagate in game-tree search. In this paper, two properties of forward pruning are reported. We first show, using theoretical analysis and Monte Carlo simulations, that the severity of forward pruning errors is asymmetric with respect to the player to move; the error is likely to be more severe when pruning on the opponent's moves rather than the player's own moves. This effect arises because pruning errors, when pruning exclusively on children of Max nodes, cannot cause a poor move to be deemed better than a good move whose subtree does not contain errors; however, this is not the case when pruning exclusively on children of Min nodes. This suggests that to do well, pruning should be done more aggressively on the player's own move and less aggressively on the opponent's move.

We also analyzed the error filtering effect of the minimax evaluation process, showing that the probability of a pruning error propagating to the root decreases exponentially with the depth of the pruned node. Unfortunately, with constant pruning error probability, the exponential increase in the number of nodes with depth introduces an exponential number of errors as well. We use Pearl's error propagation model (Pearl 1984) to derive the rate of error propagation, showing that with constant pruning error probability, the increase in the number of errors overwhelms the filtering effect. The theoretical analysis and Monte Carlo simulations suggests that to achieve a fixed probability of error at the root, the rate of pruning errors (and hence the aggressiveness of the pruning method) should decrease exponentially with the depth of the node.

To the best of our knowledge, the first effect has not been reported in the literature. The second effect is novel as an analysis of forward pruning although it builds on prior work of *minimax pathology*, or the property that minimaxing amplifies errors as search depth increases. In addition, the Monte Carlo simulations used game-trees with branch-dependent leaf values, which have been shown to be non-pathological (Nau 1982; Smith & Nau 1994), to simulate more realistic game-trees.

Preliminaries

Consider a perfect-information zero-sum game between two players, Max and Min. The minimax value of a node u is

$$score_*(u) = \begin{cases} utility(u) & u \text{ is a leaf node,} \\ \max\{score_*(child(u))\} & u \text{ is a Max node,} \\ \min\{score_*(child(u))\} & u \text{ is a Min node.} \end{cases}$$

where $child(u)$ returns the set of child nodes of u. We define forward pruning as eliminating children of u to be considered by returning a subset of $child(u)$. If forward pruning

discards all children of a node, then the node returns a value of $-\infty$ for a Max node and $+\infty$ for a Min node. We also assume game-trees have uniform branching factor b and height h, where b is the number of children each node has and h is the number of nodes in the longest path from the root node to a leaf node. The depth of a node u is one less the number of nodes in the path from the root node to u. The depths of the root node and a leaf node for a game-tree of height h are therefore 0 and $h-1$, respectively.

Effect of Player to Move

In this section, we show that forward pruning errors are propagated differently depending on the player to move. We first state a lemma showing how the different pruning errors affect minimax results:

Lemma 1. *Assume that we are performing forward pruning only on the children of Max nodes throughout the tree. Then, for any unpruned node u, $score_{Max}(u) \leq score_*(u)$, where $score_{Max}(u)$ is the score of the algorithm that only forward prunes children of Max nodes. Conversely, if we are performing forward pruning only on the children of Min nodes, then for any unpruned node u, $score_{Min}(u) \geq score_*(u)$, where $score_{Min}(u)$ is the score of the algorithm that only forward prunes children of Min nodes.*

The proof is by induction on the height of the tree.

The standard approach for game-playing is to use the result of the game-tree search with the root node representing the current board position. We assume without loss of generality that the root node is a Max node.

Theorem 1. *Assume there are b moves at the root node and that there is a strict ordering based on the utility of the moves such that $score_*(u_i) > score_*(u_j)$ if $1 \leq i < j \leq b$ and u_i is the i^{th} child.*

If forward pruning is applied only to children of Max nodes, then $\forall i$ such that no pruning error occurs in the subtree of u_i, i.e. $score_{Max}(u_i) = score_*(u_i)$, the new rank i' based on $score_{Max}$ has the property $i' \leq i$.

The converse holds if forward pruning is applied only to the children of Min's nodes, i.e., $\forall i$ such that no pruning error occurs in the subtree of u_i, i.e. $score_{Min}(u_i) = score_*(u_i)$, the new rank i' based on $score_{Min}$ has the property $i' \geq i$.

Proof. We will only prove the first statement, as the proof of the converse statement is similar. Assume, on the contrary, that $i' > i$. This implies that $\exists j > i$ such that $score_*(u_j) < score_*(u_i)$, but the new ordering j' based on $score_{Max}$ is $j' < i'$ and $score_{Max}(u_j) > score_{Max}(u_i)$. But $score_{Max}(u_j) \leq score_*(u_j)$ by Lemma 1, and $score_*(u_j) < score_*(u_i) = score_{Max}(u_i)$ which imply that $score_{Max}(u_j) < score_{Max}(u_i)$. Contradiction. □

Observing the Effect using Simulations

In order to show the significance of Theorem 1, we perform *Monte Carlo simulations* of random game-trees searched using algorithms that perform forward pruning.

In our simulations, we used game-trees of uniform branching factor 5. The values of the leaf nodes were chosen from the uniform distribution $[0, 1)$ and the root is a Max node. We fix the number of nodes to be forward pruned – in each instance, three randomly chosen children at either Max or Min nodes were forward pruned. Unfortunately, error filtering effects that reduce error propagation to the root makes comparison more difficult; the type of pruning errors that is filtered more depends on the height of the tree. To ensure that the observed effects are not because of the filtering effect, we experimented with trees of heights ranging from 4 to 6 so that there would instances of both cases: having fewer Max nodes and having fewer Min nodes.

We recorded the rank of the move at the root (as ranked by a search without forward pruning) that was eventually chosen by the search. For each experimental setup, we ran a simulation of 10^6 randomly generated game-trees.

Figure 1 shows the number of times the search with forward pruning chooses a move of a particular rank. We see that when children of Max nodes are pruned erroneously, the probability of choosing a ranked 4^{th} or 5^{th} move decreases sharply towards zero; when children of Min nodes are pruned wrongly, the probability of choosing a ranked 4^{th} or 5^{th} move only tapers gradually. In other words, if we have to choose between forward pruning only the children of Max or Min nodes, and the eventual rank of the move chosen is important, we should choose to forward prune the children of Max nodes.

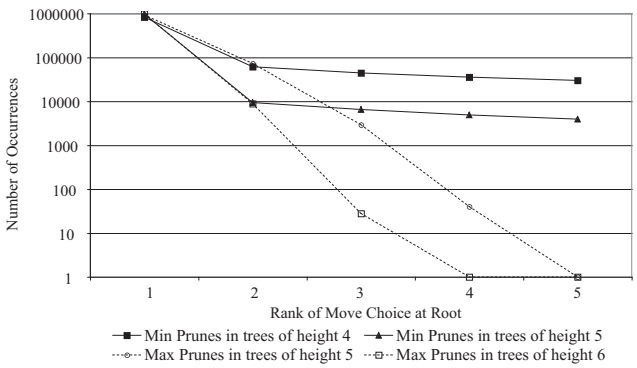

Figure 1: Log plot of the number of times ranked moves are chosen where either Max or Min nodes are forward pruned.

We also simulated more realistic game-trees, using the approach of Newborn (Newborn 1977). In his approach, every node in the tree receives a random number, and the value of a leaf node is the average of all random numbers in the nodes on the path from the root of the tree to the leaf node. This ensures some level of correlation between the minimax values of sibling nodes, which is known to be non-pathological (Nau 1982). The random number in each node is chosen from the uniform distribution $[0, 1)$. The results with such branch-dependent leaf valued game-trees were similar to Figure 1 but slightly less pronounced.

We may want to prune of both types of nodes in practice, if pruning on only children of Max nodes does not pro-

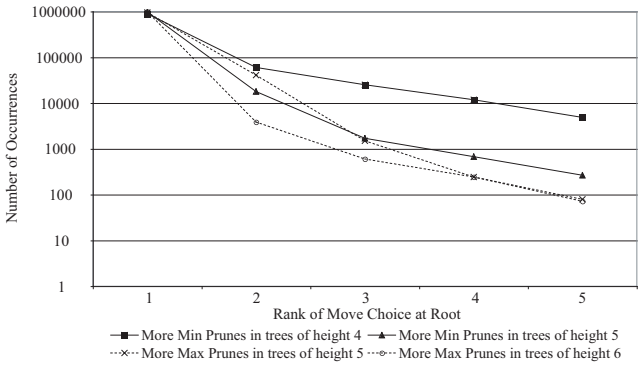

Figure 2: Log plot of the number of times ranked moves are chosen with unequal forward pruning on both Max and Min nodes.

vide enough computational savings. To simulate this, we ran experiments with branch-dependent leaf valued trees where three randomly chosen children at either Max or Min nodes and one randomly chosen child of the other node type were forward pruned. Figure 2 shows the results of these experiments for various search depths. We see that the asymmetric effects of the severity of errors are still present. Most of the errors that resulted in a rank 4^{th} or 5^{th} move being chosen are likely to have come from pruning children of Min nodes.

Discussion

In two-player perfect information games, the player can typically make only one move, so if the best move has been pruned, the game-tree search should then preferably return the second best move. Theorem 1 and our simulation results suggest that different pruning errors relative to the player at the root node have different effects on the move quality chosen by the game-tree search. Pruning errors in children of Max nodes will not decrease the rank of moves that are correctly evaluated. This means that if the second best move is correctly evaluated but the best move is incorrectly evaluated due to pruning errors, then the game-tree search will return the second best move as the move to play. On the other hand, pruning errors in children of Min nodes can incorrectly increase the rank of moves and the game-tree search could possibly return the worst move as the move to play, even if the best move is correctly evaluated.

This suggests the risk management strategy of forward pruning more aggressively on the children of Max nodes and more conservatively on the children of Min nodes. We have done initial experiments in chess: a small scale preliminary study suggests that pruning more aggressively on children of Max nodes results in a stronger chess program compared with pruning equally on all nodes and pruning more aggressively on children of Min nodes. However, a more detailed experimental study is required to confirm this.

Effect of Depth of Node

The minimax algorithm propagates scores from leaf nodes via a process of alternating between maximizing and minimizing. This process confers some measure of filtering for pruning errors. In this section, we build on Pearl's error propagation model (Pearl 1984), which constructs game-trees from the leaf nodes to the root. For ease of explanation we use the *height* of a node u, defined as one less the number of nodes in the longest path from u to a leaf node. Depth and height of a node are closely related; a node of depth d is at height $h - d - 1$ for a game-tree of height h. To obtain some insights, we first consider the case of a single pruning error.

Proposition 1. *Assume a complete b-ary tree of height at least 3. Then the probability of a change in value of a random node at height k, selected with uniform probability from all nodes at that height, affecting the minimax evaluation of its grandparent node at height $k + 2$ is no more than $\frac{1}{b}$.*

Proof. Consider the case where height k consists of Max nodes and a node at height k is chosen with uniform probability from all nodes at that height to have a change in value. We assume that the grandparent node g at height $k + 2$ has the true value of v. We consider the case where the error decreases the node value first. If more than one child of g has value v, a single value reduction at depth k will not change g's value, so we consider the case where only one child, say m, has value v. Note that the value of g can only change if the value reduction occurs in m's children and not anywhere else. The probability of this occurring is no more than $1/b$, since m has b children. Now, consider the case where the error increases the node value. Consider any one of g's children, say m. Let m have value v. The value of the node m can change only in the case where only one of its children has value v and that child is corrupted by error. Hence the number of locations at height k that can change g's value is no more than b out of the b^2 nodes at that height.

The cases for Min nodes are the same when we interchange the error type. □

Proposition 1 can be recursively applied to show that the probability of an error propagating through l depths is no more than $1/b^{\lceil l/2 \rceil}$. However, the number of leaves in a b-ary tree of height l is b^{l-1}. If the probability of a leaf being mistakenly pruned is constant, the faster growth of the leaves will swamp the filtering effect, as we show in the next section.

Theoretical model for the propagation of error

Theoretical models have found that the minimax algorithm can amplify the evaluation errors that occur at the leaf nodes. This is known as the *minimax pathology* and was independently discovered by Nau (Nau 1979) and Beal (Beal 1980). Pearl (Pearl 1984) used a probabilistic game model, which we reproduce here, to examine this distortion and quantify the amplification of errors. Let p_k be the probability of a WIN for a node at height k and consider a uniform binary game-tree where the leaf nodes are either WIN or LOSS with probability p_0 and $1 - p_0$, respectively. The leaf nodes also have an imperfect evaluation function that estimates the values with a bi-valued variable e, where $e = 1$ or $e = 0$ represent a winning and losing position, respectively. We

denote the minimax evaluation at position i as e_i and the WIN-LOSS status at position i as S_i. Positions where $i = 0$ represent leaf nodes. e_i and S_i are represented in negamax notation and refer to the player to move at position i. If node 3 has two children, 1 and 2, (Figure 3) we write

$$S_3 = \begin{cases} L, & \text{if } S_1 = W \text{ and } S_2 = W, \\ W, & \text{otherwise} \end{cases} \quad (1)$$

$$e_3 = \begin{cases} 0, & \text{if } e_1 = 1 \text{ and } e_2 = 1, \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

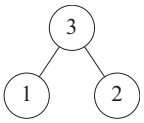

Figure 3: Node 3 has two children, 1 and 2.

Denote the probability of erroneously evaluating a LOSS position as winning and the probability of erroneously evaluating a WIN position as losing at position i as α_i and β_i respectively. In other words,

$$\alpha_0 = P(e = 1|S = L) \quad (3)$$

$$\beta_0 = P(e = 0|S = W) \quad (4)$$

The recurrence equations are:

$$\alpha_{k+1} = 1 - (1 - \beta_k)^2 \quad (5)$$

$$\beta_{k+1} = \frac{\alpha_k}{1 + p_k} \left[(1 - p_k)\alpha_k + 2p_k(1 - \beta_k)\right] \quad (6)$$

$$p_{k+1} = 1 - p_k^2 \quad (7)$$

Pearl also considered uniform *b-ary* game trees where each non-leaf node has b successors, and similarly we obtain:

$$\alpha_{k+1} = 1 - (1 - \beta_k)^b \quad (8)$$

$$\beta_{k+1} = \frac{\left\{[p_k(1 - \beta_k) + (1 - p_k)\alpha_k]^b - [p_k(1 - \beta_k)]^b\right\}}{1 - p_k^b} \quad (9)$$

$$p_{k+1} = 1 - p_k^b \quad (10)$$

Furthermore, p_k has three limit points (Pearl 1984):

$$\lim_{k\to\infty} p_{2k} = \begin{cases} 1 & \text{if } p_0 > \xi, \\ \xi & \text{if } p_0 = \xi, \\ 0 & \text{if } p_0 < \xi, \end{cases} \quad (11)$$

where ξ is the solution to $x^b + x - 1 = 0$. The limit points show that when the height of the tree is large enough, under this model, the outcome at the root is uncertain only for $p_0 = \xi$. Hence, we are mostly interested in the behavior at these three limit points.

While Pearl's model assumed an imperfect evaluation function at the leaf nodes, we assume that the evaluation of leaf nodes are reliable, since we are considering only pruning errors. We adapt Pearl's model to understand the effects of forward pruning by considering α_0 and β_0 as the probabilities of pruning errors made at the frontier nodes (nodes that have leaf nodes as children). We can now demonstrate the effects of the depth of the node in forward pruning:

Theorem 2. *Assume that errors exists only at the leaves of uniform b-ary game trees. Then $\beta_{k+2}/\beta_k \leq b$ with $\beta_{k+2}/\beta_k = b$ for some cases. Similarly, $\alpha_{k+2}/\alpha_k \leq b$ with $\alpha_{k+2}/\alpha_k = b$ for some cases.*

Proof. We recurse equations (8) and (9) once to get

$$\alpha_{k+2} = 1 - (1 - \frac{1}{1 - p_k^b} \times \{[p_k(1 - \beta_k) + (1 - p_k)\alpha_k]^b - [p_k(1 - \beta_k)]^b\})^b \quad (12)$$

$$\beta_{k+2} = \frac{1}{1 - p_{k+1}^b} \{\left[(1 - p_k^b)(1 - \beta_{k+1}) + p_k^b\alpha_{k+1}\right]^b - \left[(1 - p_k^b)(1 - \beta_{k+1})\right]^b\} \quad (13)$$

The value of α_{k+2} in equation (12) reaches its maximum value when $\beta_k = 0$. We also see that $\beta_{k+1} = 0$ when $\alpha_k = 0$ and therefore β_{k+2} in equation (13) reaches its maximum value when $\alpha_k = 0$.

We denote α_k when $\beta_0 = 0$ by α'_k. When $\beta_0 = 0$, we have $\beta_k = 0$ when k is even. α'_{k+2}/α'_k gives us the rate of increase in error propagation based on the value of α'_k:

$$\frac{\alpha'_{k+2}}{\alpha'_k} = \begin{cases} \frac{1-\left[1-(\alpha'_k)^b\right]^b}{\alpha'_k} & \text{if } p_k \to 0, \\ \frac{1-(1-\alpha'_k)^b}{\alpha'_k} & \text{if } p_k \to 1, \end{cases} \quad (14)$$

$$\lim_{\alpha_k\to 0} \frac{\alpha'_{k+2}}{\alpha'_k} = \begin{cases} 0 & \text{if } p_k \to 0, \\ b & \text{if } p_k \to 1, \end{cases} \quad (15)$$

Similarly, we denote β_k when $\alpha_0 = 0$ as β'_k:

$$\frac{\beta'_{k+2}}{\beta'_k} = \begin{cases} \frac{\left[1-(1-\beta'_k)^b\right]}{\beta'_k} & \text{if } p_k \to 0, \\ \frac{\left[1-(1-\beta'_k)^b\right]^b}{\beta'_k} & \text{if } p_k \to 1, \end{cases} \quad (16)$$

$$\lim_{\beta_k\to 0} \frac{\beta'_{k+2}}{\beta'_k} = \begin{cases} b & \text{if } p_k \to 0, \\ 0 & \text{if } p_k \to 1, \end{cases} \quad (17)$$

Lastly, the proof in Proposition 1 can be modified to show that $\beta_{k+2}/\beta_k \leq b$ and $\alpha_{k+2}/\alpha_k \leq b$ by considering one type of error instead of a single error. □

To help us gain more insight into the rate of error propagation, we simplify the analysis by considering uniform binary game trees. Setting $\beta_0 = 0$ and reapplying the recurrence equations for $b = 2$, we get

$$\alpha'_{k+2} = \frac{\alpha'_k}{1 + p_k} \left[(1 - p_k)\alpha'_k + 2p_k\right] \times \left\{2 - \frac{\alpha'_k}{1 + p_k} \left[(1 - p_k)\alpha'_k + 2p_k\right]\right\}. \quad (18)$$

Similarly, we can set $\alpha_0 = 0$ to get:

$$\beta'_{k+2} = \frac{\beta'_k(2 - \beta'_k)}{2 - p_k^2} \left\{p_k^2 \left[1 - (1 - \beta'_k)^2\right] + 2(1 - p_k^2)\right\} \quad (19)$$

Several interesting observations can be made from Figure 4, which shows the plot of α'_{k+2}/α'_k and β'_{k+2}/β'_k for the

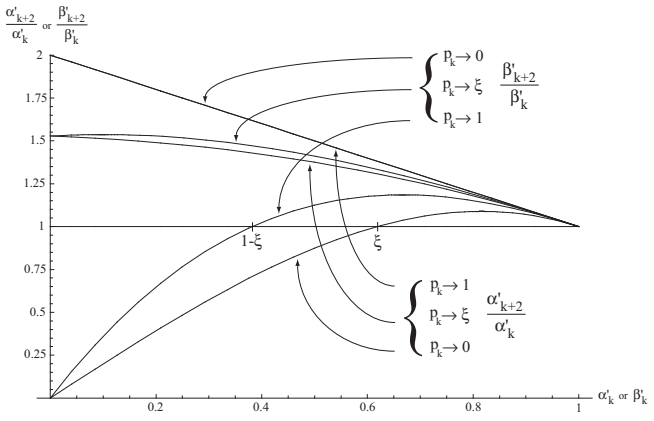

Figure 4: Rates of change in error propagation for $b = 2$

limit points of p_k. If $p_k \to 1$, errors are being filtered out when $\beta'_k < 1 - \xi$, giving $\lim_{k\to\infty} \beta'_{2k} = 0$. However, when $\beta'_k > 1 - \xi$, error rate will increase with the height giving $\lim_{k\to\infty} \beta'_{2k} = 1$. If $\beta'_k = 1 - \xi$, the rate of error propagation is constant, $\lim_{k\to\infty} \beta'_{2k} = 1 - \xi$. Similarly, when $p_k \to 0$, $\lim_{k\to\infty} \alpha'_{2k}$ is 0 when $\alpha'_k < \xi$, is 1 when $\alpha'_k > \xi$, and is ξ when $\alpha'_k = \xi$. For $p_k = \xi$, both types of errors grow with the height. These results are also given in (Pearl 1984) for errors caused by imperfect evaluation functions.

The upper bound on the rates of change in error propagation to the root of b by Theorem 2 is clearly too conservative as shown in Figure 5. For example, when $b = 2$, the maximum of β'_{k+2}/β'_k for $p_0 = \xi$ occurs when $\beta'_k \approx 0.099$ where $\beta'_{k+2}/\beta'_k \approx 1.537$, which is less than the bound of 2 that the theorem suggests.

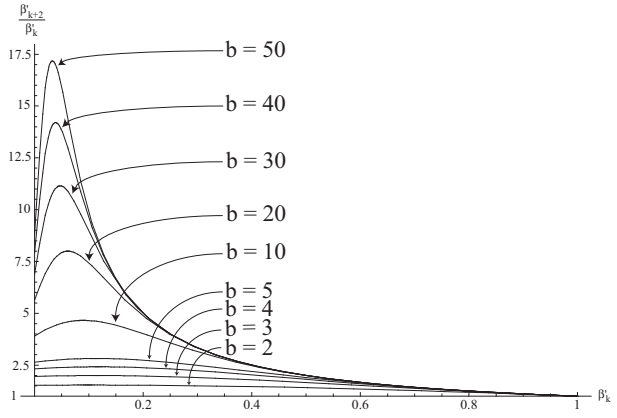

Figure 5: Plot of $\frac{\beta'_{k+2}}{\beta'_k}$ when $p_0 = \xi$ for various b

Observing the Effect using Simulations

To illustrate the implications of our results, we once again perform a number of Monte Carlo simulations. In our experiments, we use game-trees with uniform branching factor 5 and branch-dependent leaf values to simulate actual game-trees. Each node is assigned a pruning probability q_i: during search, a Bernoulli trial with probability q_i of pruning each child is performed, where i is the depth of the node. We test two different pruning reduction schemes – Multiplicative and Linear pruning reduction. The multiplicative pruning reduction schemes multiply the pruning probability by a constant factor for every additional 2 depths, or $q_{i+2} = q_i \times c$, and $q_1 = q_0 \times c$, where c is the multiplicative factor. A multiplicative factor of 1.0 is equivalent to a Constant Pruning scheme. Linear pruning reduction schemes reduce q_i for each depth by subtracting a constant c from the previous depth, or $q_{i+1} = q_i - c$. Figure 6 shows the proportion of correct minimax evaluations for various pruning reduction schemes with starting pruning probability $q_0 = 0.1$. We see that the linear pruning reduction schemes are clearly inadequate to prevent amplification of pruning errors propagating to the root, even though the linear pruning scheme of $c = 0.02$ reduces q_i to zero when at search depth 6.

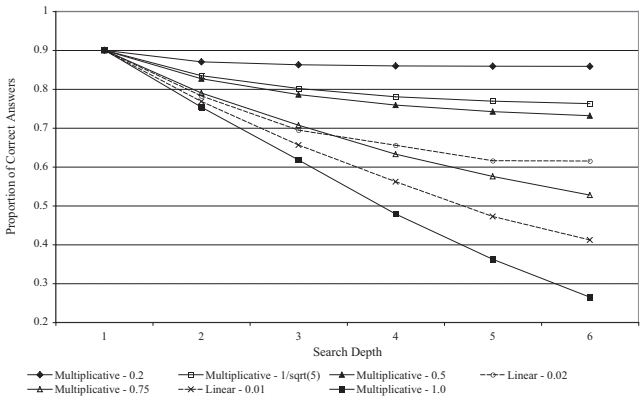

Figure 6: Comparison of various pruning reduction schemes

While the experiments have shown that multiplicative pruning reduction schemes can prevent the amplification of pruning errors propagating to the root, it might be possible that multiplicative pruning reduction schemes are not pruning enough to justify forward pruning at all. It is more interesting to consider the question "Given a fixed time limit, what is the best pruning reduction scheme that allows the deepest search while making less than a pre-defined threshold of errors?".

We set the error threshold at 0.25, which means that the search should return a back-up evaluation equal to the true minimax value of the tree at least 75% of the time[1]. To simulate a fixed time limit, we used $\alpha\beta$ search and iterative deepening to search until the maximum number of nodes, which we set at 1000, were searched. We tested two additional pruning reduction schemes – Root Pruning and Leaf Pruning. In the Root pruning scheme, only the root node forward prunes, or $q_0 = c > 0$ and $q_i = 0$, for $i > 0$. The Leaf pruning scheme only forward prunes leaf nodes, or $q_i = 0$, for $i < d$ and $q_d = c > 0$, where d is the search depth.

[1] In a real game, we would be able to use domain dependent information to decide when to prune. This is likely to result in a smaller pruning error rate for the same aggressiveness in pruning

We first used 8 iterations of binary searches with 10^5 simulated game-trees each time to find pruning probabilities p_0 for the various pruning schemes that return correct answers at least 75% of the time. Next, we ran a simulation of 10^6 game-trees and, for each generated game-tree, we performed every pruning scheme with the pruning probabilities found using binary search.

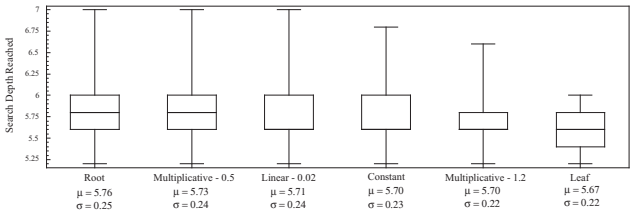

Figure 7: Box plot showing the search depths reached with correct answers by each pruning scheme. A box plot gives a five-number summary of: minimum data point, first quartile, median, third quartile, and maximum data point. The mean and standard deviation of the search depths reached for each pruning scheme are also given.

The Root pruning scheme is the best pruning scheme as it achieves, on average, the deepest search depths among all pruning schemes as shown in Figure 7. It remains to be seen if this is also true for trees from real games, using practical pruning schemes. However, the results suggest that pruning rate should decrease with the depth of the nodes.

Discussion

There is anecdotal evidence that supports our analysis of the effect of the depth of nodes in forward pruning. In developing adaptive null-move pruning (Heinz 1999), it was initially expected that the best performance in an adaptive form of null-move pruning would come from pruning more when near the leaf nodes and less when closer to the root. However, Heinz found that the opposite is true: adaptive null-move pruning works better by pruning less near the leaf nodes, which agrees with our results.

The theoretical analysis agrees with an intuitive forward pruning strategy: prune more on your own move and less on your opponent's when you are winning. The player who is winning will generally have several good moves, and pruning some of them would not affect the possibility of finding a winning move, but pruning on the opponent's move may cause a refutation to be missed. Figure 4 similarly suggests that pruning errors in Max nodes are tolerated best when the leaf nodes are mostly wins (and are actually filtered out when the error is small), and tolerated least when the leaf nodes are mostly losses; and vice versa for Min nodes.

Lastly, existing literature had painted the pessimistic picture that the minimax algorithm corrupts the back-up values in the presence of errors. While there are alternative explanations for minimax pathology, including but not limited to, (Smith & Nau 1994; Sadikov, Bratko, & Kononenko 2005; Lustrek, Gams, & Bratko 2005), our analysis show that the minimax algorithm is filtering pruning errors, but at a slower rate than the rate of growth of leaf nodes.

Conclusion

We have shown theoretical and simulation results that suggest that the player to move and the depth of a node affect how forward pruning errors propagate to the root node. In particular, pruning errors in Max nodes generally result in better quality moves than if the same amount of pruning errors were made in Min nodes. This suggests that forward pruning techniques should adjust how much they forward prune based on the player to move. We also showed that the probability of errors propagating to the root node decreases as depth of the error location increases. Unfortunately, the rate at which the minimax algorithm can filter out errors is smaller than the rate at which leaf nodes are introduced for each additional search depth. This suggests that forward pruning techniques should prune less as search depth increases. While the simulations used non-pathological game-trees with correlated leaf values, experiments with actual game-trees are needed to verify the results.

Acknowledgement

This work is supported by an A*Star-NUS Graduate Fellowship to Y. J. Lim. We thank Jürg Nievergelt, Oon Wee Chong and the anonymous referees for their helpful comments.

References

Beal, D. F. 1980. An analysis of minimax. In *Advances in Computer Chess 2*, 103–109. Edinburgh University Press.

Buro, M. 1995. ProbCut: An effective selective extension of the $\alpha - \beta$ algorithm. *International Computer Chess Association Journal* 18(2):71–76.

Heinz, E. A. 1999. Adaptive null-move pruning. *International Computer Chess Association Journal* 22(3):123–132.

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

Lustrek, M.; Gams, M.; and Bratko, I. 2005. Why minimax works: An alternative explanation. In *IJCAI*, 212–217.

Marsland, T. A. 1986. A review of game-tree pruning. *International Computer Chess Association Journal* 9(1):3–19.

Nau, D. S. 1979. *Quality of decision versus depth of search on game trees*. Ph.D. Dissertation, Duke University.

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

Newborn, M. M. 1977. The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores. *Artificial Intelligence* 8:137–153.

Pearl, J. 1984. *Heuristics – Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley Publishing Co.

Sadikov, A.; Bratko, I.; and Kononenko, I. 2005. Bias and pathology in minimax search. *Theoretical Computer Science* 349(2):268–281.

Smith, S. J. J., and Nau, D. S. 1994. An analysis of forward pruning. In *Proceedings of 12th National Conference on Artificial Intelligence (AAAI-94)*, 1386–1391.