Improving Local Decisions in Adversarial Search

Brandon Wilson\(^1\) and Inon Zuckerman\(^2\) and Austin Parker and Dana S. Nau\(^1,3\)

Abstract. Until recently, game-tree pathology (in which a deeper game-tree search results in worse play) has been thought to be quite rare. We provide an analysis that shows that every game should have some sections that are locally pathological, assuming that both players can potentially win the game.

We also modify the minimax algorithm to recognize local pathologies in arbitrary games, and cut off search accordingly (shallower search is more effective than deeper search when local pathologies occur). We show experimentally that our modified search procedure avoids local pathologies and consequently provides improved performance, in terms of decision accuracy, when compared with the ordinary minimax algorithm.

1 INTRODUCTION

Adversarial search, or game-tree search, is a technique for analyzing an adversarial game to determine what moves a player should make in order to win a game. We are interested in two-player, perfect information, zero-sum games, in which there are only wins and losses. Let \(G\) be a game tree where each node \(n\) has a set of moves \(m(n)\) for the player-to-move \(p(n)\). The terminal nodes are assigned a utility \(u(n)\), where 1 represents a win for player 1 and \(-1\) represents a win for player 2. Utilities can then be propagated using the standard minimax formula [11]. To determine which move is best, one simply computes the minimax values for all states reachable by one move from the current state, then pick a state with a maximal minimax value. When ambiguous, we will use the term correct minimax value to refer to the value of a node \(n\) computed according to minimax\((n)\).

In perfect information zero-sum games, minimax is known to return an optimal move when allowed to search the entire game tree [12]. Since many games have combinatorially large game trees that are far too large to permit exhaustive search in normal game-play, implementations of the minimax algorithm generally involve searching to a limited depth \(d\), applying a heuristic function called a static evaluation function that estimates the utility value of the nodes at that depth, and inserting these estimates into the minimax formula in place of the nodes’ true utility values. The values of nodes with depth less than \(d\) can then be computed as normal, giving estimated utility values for each possible move.

It is generally accepted that deeper search results in better gameplay. However, in the early 1980s, Nau [8, 9] discovered classes of games that exhibit a phenomenon known as game-tree pathology, in which deeper minimax search results in worse performance. In other words, in pathological game trees, searching deeper is consistently less likely to produce a move with maximal utility (hereafter we call such moves “correct moves”).

Although there have been several attempts to study this phenomenon [1–4, 7, 9, 13, 14, 16], it has generally been thought to be quite rare in real games. However, recent work [10] has shown that pathological situations can occur in Chess, Kalah and the 8-puzzle. In addition, their simulations show that the benefit provided by deeper search increases with the evaluation function’s granularity (the number of returned values), decreases with the game tree’s branching factor (the number of successors of each node), and increases with the tree’s local similarity (the similarity among the values of closely related nodes).

In this paper, we make the following contributions:

- We analyze and detect local pathologies, and show how they arise within a game tree, and that local pathologies are likely to occur in all interesting games.
- We show how to modify the minimax search procedure to recognize and overcome local pathologies. Our modified search algorithm is called error minimizing minimax (EMM), and it works by tracking both the minimax value of a node and the error associated with it. As the minimax value of a node is aggregated up the tree in a minimax fashion, the associated error is also aggregated up the tree.
- We provide experimental results showing that the EMM provides improved decision accuracy compared to ordinary minimax. We also show that EMM exhibits no pathology even in situations where minimax does exhibit pathology.

2 RELATED WORK

Since the discovery of minimax pathology thirty years ago [8], several explanations have been proposed for why it was not observed in most real-world games. Probably the most widely accepted explanation is that pathology is inhibited by similarity among different parts of the search tree [2, 9]. Pearl [13] suggested traps as an alternative explanation. Traps are moves that cause the game to end abruptly, introducing very accurate, if not perfect, heuristic values at some shallow nodes. Lustrek et al. [6] cited low granularity of the evaluation function, i.e. the number of possible values that an evaluation function can return with a non-zero probability, as a source of pathology.

More recently, a study was performed to examine the relationship between the degree of pathology in a game tree and three possible causes of pathology: branching factor, local node similarity, and evaluation function granularity (number of possible values returned) [10]. In that study, the authors defined the degree of pathology for a search of depth \(d\) as the fraction of correct decisions made by searching to depth \(d\) over the fraction of correct decisions made...
by searching one level deeper. Experimenting on synthetic trees, they discovered that, in general, pathology is more likely to occur, and have more severe effects, when searching with higher branching factor, lower evaluation function granularity, and lower local node similarity. Expanding their study to include real games, they showed that endgames database exhibit some degree of local pathology despite being overall non-pathological (5.5% – 9.2% of positions for chess were pathological). They also showed that the African game of Kalah (for sufficiently high branching factor) is the first real game to consistently exhibit pathology throughout the game. In addition, experiments on the single agent 8-puzzle showed that 19.7% of positions exhibit pathology.4

Sadikov et al. [16] differentiated between two types of accuracy affecting pathology: evaluation and decision accuracy. Evaluation accuracy refers to the difference between heuristic values and the backed up values. On the other hand, decision accuracy is a measure of how many correct decisions are made by a deeper search compared to a shallow one. Their experimental results on the King-Rook-King chess endgame show that although a heuristic evaluation may be increasingly inaccurate with deeper search, the decision accuracy may actually improve. The explanation for this unexpected result is that heuristic evaluators, by nature, introduce a bias into the evaluation values. The bias is similar among all nodes on the search frontier so the relative ordering among nodes is preserved. It is for this reason that we focus on decision accuracy as our measure of performance in our experiments.

All of the work above either suggests potential sources of pathology or classifies a set of games as being pathological. Based on that work, it is clear that identifying a single or even a handful of sources of pathology is a difficult task. Instead of isolating the cause of pathology, we propose to detect when it begins to manifest itself during the propagation process and truncate the pathological portions of search at a shallower depth.

3 ANALYSIS

As can be seen from the previous section, the source for search pathology is still a mystery. We still do not have a decision procedure to verify whether a certain game is pathological or not. In the following analysis we show that the question is not a binary one, rather, we claim that every game has pathological situations. We call these pathological situations, local pathologies. As a consequence (and in accordance with [10]), one can say that different games exhibit different degrees of local pathologies.

To simplify the presentation we start with a quick analysis of a game with branching factor two,5 showing that local pathologies are likely to occur in all interesting games. For this analysis, we will assume a static evaluation function that returns the correct utility value on any given node with probability 1 – e (similar to the model used in [4]), which also means that incorrect values will be returned with probability e.

We will be looking at the evaluation error at nonterminal nodes. Evaluation error occurs when a node’s minimax value is miscalculated by a depth-limited minimax computation. At one extreme, we can imagine a depth 0 minimax computation wherein a static evaluation function is applied to the node. In this case, the evaluation error will simply be that of the static evaluation function, e. When deeper minimax searches occur, we have different evaluation errors for different types of nodes. Here we examine only searches of depth one, as any search to depth d can be instead thought, for the sake of analysis, as many depth-one searches.

In games with a branching factor of two, there are four possible types of nonterminal nodes. These are shown in Figure 1 (nodes B and C are symmetric and are therefore considered together). At each node, it is player 1’s move, so the node’s minimax value is the maximum of the minimax values of its children (which are not terminal, but rather the search’s horizon).

Using an evaluation function with error e, we can calculate the probability that a depth one minimax search will return the wrong value for the root node in each type of node:

\[
\text{error}(A) = e^2 \\
\text{error}(B) = e(1 - e) \\
\text{error}(C) = \text{error}(B) \\
\text{error}(D) = 1 - (1 - e)^2
\]

When comparing these functions by simply applying the static evaluation function with error e to the root node, we get:

\[
\text{error}(D) \geq e \geq \text{error}(B) \geq \text{error}(A)
\]

for any error \( e \in [0, 0.5] \). That is, the error resulting from searching below type D nodes exceeds the error resulting from simply applying the static evaluation directly, while for types A, B, and C nodes, the error for depth-one search is less than that of simply applying the evaluation function. Figure 1 shows this relationship in a graph, where we plot the value of e against the error present at each type of node for simply evaluating the node \( f(e) = e \) and for searching below it.

Only in type D nodes is the error at the root greater than the error at the leaves, and, since any depth-d search can be seen as a combination of d depth-one searches, we can conclude that type D nodes

---

4 In single player search problems such as the 8-puzzle, we view the game tree as a min-min search tree as the player tries to minimize the number of moves required to solve the problem. More details can be found in [3, 14].
5 We hope it will be obvious how the analysis will extend to higher branching factors.
are the source of search pathology. This is not to say that any time one reaches a type \( D \) node, a shallower search should be preferred – it may be that each child of a type \( D \) node is a type \( A \) node, in which case the error at the root will be \( 1 - (1 - e^2)^2 \), which is less than \( e \). But if the entire tree consisted of nodes of types \( A \), \( B \), or \( C \), then there could not be evaluation pathology.\(^6\)

We expect all interesting games to contain nodes of type \( D \). This is especially true for zero-sum games as they are not interesting if one player always wins, and without type \( D \) nodes, it would be impossible for another player to win!

4 ERROR MINIMIZING MINIMAX

Our search algorithm tracks the error associated with the node value. The search computes the static evaluation function at any given node. If the static evaluation allows a tighter error bound than the propagated value then that value and error bound are substituted in the final return statement.

```
Algorithm EMM(s, eval, d): Error minimizing minimax search.
For game state \( s \), evaluation function \( eval \) (returning an evaluation of a board from the perspective of the player-to-move) with error \( e_s \), and search depth \( d \), returns a pair \( (a,e) \) where \( a \) is the valuation of the state \( s \) and \( e \) is the error associated with that valuation. \( \gamma(s, mv) \) is the state-transition function, returning the new state after making move \( mv \) from state \( s \).
```

```
Let \( curVal = eval(s) \), and \( curErr = e_s \).
if \( d = 0 \), \textbf{return} \( (curVal, e) \)
/* Determine values \( v_i \) and errors \( e_r \) for child nodes. */
Let \( mv_1, \ldots, mv_n \) be the moves from \( s \).
for \( i = 1, \ldots, n \) do
\( \langle v_{Tmp_i}, e_{r_i} \rangle = EMM(\gamma(s, mv_i), eval, d - 1) \)
end for
Let \( v_i = -v_{Tmp_i} \).
Let \( val = \max_i(v_i) \). /* This node’s value. */
/* Determine error for this node \( aggErr \). */
if \( val \) is a loss then
/* All children are losses. If any of them are wrong, this node is in error. */
\( aggErr = 1 - \prod_i (1 - e_{r_i}) \)
else
/* There is at least one win child. Error occurs if winning children are wrong and losing children are right. */
Let \( aggErr = 1 \)
for each \( (v_i, e_{r_i}) \) do
if \( v_i \) is a win then
\( aggErr = aggErr \times e_{r_i} \)
else
\( aggErr = aggErr \times (1 - e_{r_i}) \)
end if
end for
/* Flip values if \( aggErr \) is too big. */
if \( aggErr > 0.5 \) then \( (val, aggErr) = (-val, 1 - aggErr) \).
/* Check if static evaluation matches minimax value. */
if \( curVal = val \) then
/* Return the result with the stronger error guarantee. */
\textbf{return} \( (curVal, \min(curErr, aggErr)) \)
else if \( curErr \geq aggErr \) then
/* Non-pathological case: statically evaluated error is greater than search’s error. Use minimax results. */
\textbf{return} \( (val, aggErr) \)
else
/* Pathological case: the statically evaluated error is less than the search’s error. Use static results. */
\textbf{return} \( (curVal, curErr) \)
end if
/* Return the result with the stronger error guarantee. */
\textbf{return} \( (curVal, \min(curErr, aggErr)) \)
```

\( ^6 \) So long as the static evaluation function mislabels each node with independent probability \( e \).
5 EXPERIMENTS

Our experiments are performed on a board-splitting game developed by Judea Pearl [13]. In this perfect information game, two players take turns dividing a 2-D board, consisting of 1’s and 0’s, into \( b \) equal pieces and discarding all but one piece. Player one splits the board vertically and decides which half of the board to keep, then player two splits horizontally and decides which half to keep, and vice-versa. The game is over when only one square remains. If this square is a 1 then the last player to move is declared the winner, otherwise the other player wins.

We focus on two versions of the game that differ only in the construction of the initial board. The first version is referred to as a P-game\(^7\). The initial board for each P-game is generated so that each square is randomly and independently assigned a value of 1 with probability \( p \) and a 0 with probability \( 1 - p \). The board size itself is \( b \times \lceil \frac{d}{2} \rceil \times b \times \lceil \frac{d}{2} \rceil \) where \( b \) and \( d \) are the desired branching factor and depth of the game tree respectively. Minimax has been shown to be pathological on P-games using a natural evaluator.

The second version is referred to as an N-game. This construction was introduced by Nau [9] to emulate the dependence of heuristic values among siblings, in order to create non-pathological instances of the game. For an initial board of size, \( b \times \lceil \frac{d}{2} \rceil \times b \times \lceil \frac{d}{2} \rceil \), a value of 1 is assigned to each edge of the game tree with probability \( p \) and \(-1\) with probability \( 1 - p \). Each leaf of the game tree represents a single square on the board and its value is determined by summing the edge values from the root to that leaf, giving the leaf a value of 1 if the sum is positive and 0 otherwise.

Since these two versions of the game are considered to be on opposite ends of a spectrum (in terms of degree of pathology), we also experiment on games that fall in between, where we suspect the game tree might be more similar to that of a real game. These games are constructed and classified by an additional parameter we refer to as the mixing factor, \( m \in [0.0, 1.0] \). After constructing a standard N-game, there is a probability \( m \) that each square is randomly perturbed and assigned a new value according to the P-game construction method. A game with a value of \( m = 0.0 \) is a pure N-game and similarly a game constructed with a value of \( m = 1.0 \) is a pure P-game.

The mixing factor is similar to the local similarity parameter that Nau et al. [10] used to generate synthetic game trees with varying local similarity. Nau et al. showed that this similarity measure is inversely correlated with the degree of pathology. Therefore, we expect that our analogous game construction will generate games with a greater amount of local pathology as the mixing factor varies from 0.0 to 1.0.

Our experiments compare the performance of minimax and EMM. We also use two different static evaluation functions:

1. An artificial static evaluation function. This is a binary function that returns the true minimax value of a state with probability \((1 - e)\) and the incorrect value with probability \( e \), where \( e \) is a predetermined error rate.
2. A natural static evaluation function based on the percentage of winning squares on the remaining board. To make this a binary evaluation (required by EMM), the player with the largest number of winning positions will be evaluated as the winner. For this evaluation function, we estimated the associated error (used by the error-minimizing search) as the fraction of the board that is not associated with the estimated winner.

A pathology is characterized by a decrease in correct decisions with an increase in search depth. Therefore, we measure performance in terms of the fraction of correct decisions made at the root node, where a returned move is “correct” when its true minimax value is maximal among moves at that node. Scenarios with different branching factors produced similar results.

Fig. 3 shows the fraction of correct decisions made by each algorithm using the artificial evaluator (\( e = 0.2 \)) on 5,000 non-trivial P-games with 11 turns (i.e., full game tree of height 11) and a branching factor of 2. EMM clearly outperforms minimax as the search depth increases. Both games achieve a perfect decision rate of 1.0 at search depth 11 since this equates to searching the complete game tree. We can also see that EMM does not exhibit pathological characteristics, while minimax does.\(^8\) In fact, at a search depth of 7, EMM is making over 20% more correct decisions than minimax.

Fig. 4 shows the performance of the algorithms, but this time the natural evaluator is used. Here EMM is still non-pathological, whereas minimax search is pathological and loses approximately 10% accuracy by searching ahead just 3 moves to depth 7. Here we can see that even using a more realistic evaluation function, with an estimated error, EMM still outperforms minimax with increasing search depth.

\(^{8}\) The slight drop from depth 7 to depth 9 is due to the fact that EMM does not always identify the correct node type to work on. In another series of experiments with the true node types we obtained better results and the slight drop had vanished.
more towards P-games \((m = 1.0)\) where it makes 26% more correct decisions. With respect to the natural evaluator (figure 6), we see that around \(m = 0.5\) and higher is where EMM begins to outperform minimax. This indicates that EMM is not only better in strongly pathological games (P-games), but also in games with smaller degrees of pathology.

6 DISCUSSION AND FUTURE WORK

Error minimizing minimax bears some resemblance to the product rule [18]. The product rule computes the probability that a given node is a win for player one, then aggregates those probabilities up the tree in a method similar to the one used by EMM. The major difference between EMM and product rule search is in the shortcutting of the aggregation up the tree when the static evaluation function is less erroneous than the minimaxed value. This limits the search below nodes with pathological characteristics: when searching below a node produces more erroneous values, then the error associated with that search will be higher and the results of the search will be more likely to be thrown away. In this fashion, EMM can be said to “recognize” the pathological portions of a game tree, avoiding them, while doing full-depth search on non-pathological portions of the tree.

Despite the positive results we have seen, there are several potential weaknesses present in EMM. The first is the assumption of a particular form of static evaluation function. Generally, if one finds a static evaluation function that is wrong 10% of the time, those errors do not occur independently at random (as we assume in our error propagation equations). Instead, for many natural static evaluation functions, when they are wrong about one game state, they are likely to also be wrong about children of that game state. Incorporating the dependence among sibling nodes is an important next step as that is the primary difference between the performance of the artificial and natural evaluators. However, even with an independent assumption among nodes, we saw that EMM performed better than minimax in games where node values were not completely independent (i.e., games with degrees of pathology between P-games and N-games).

Second, it is not clear that estimating error characteristics for natural static evaluation functions the way we did for the board-splitting game (i.e., scaling the evaluation function to a range of \([0.0, 1.0]\) and treating them as probabilities) will generalize well for real games. Understanding how the error characteristics are affected by parameters of the search, such as depth and branching factor, is another key to making EMM effective in a larger set of games.

The algorithm is also limited to two-player games. We plan to extend the work to multi-player domains by building upon the multi-player extension of minimax, the Max^n algorithm [5], where pathology has also been shown to exist [7]. We already have preliminary results in this area that look promising (although the mathematical equations of the node type analysis are much more complex).

Finally, alpha-beta pruning presents a challenge for EMM, because EMM cannot calculate the errors unless it visits the nodes alpha-beta would prune. Consequently, EMM will be at a serious disadvantage if a game tree does not contain pathological nodes—but if it does contain pathological nodes, then the deeper searches performed by minimax with alpha-beta can actually degrade performance! A pruning procedure should look both at the heuristic value and the propagated error and try to approximate when to prune. A good starting point should be somewhat similar to the algorithms found in [15].

However, it is important to note that the pruning limitation does not exist when talking about multi-player games. It was shown in...
[17] that the most important pruning procedure, deep pruning, is not applicable in the $Max^n$ algorithm, leaving the algorithm with only immediate pruning and shallow pruning at its disposal. Consequently, we hypothesise that the pruning procedure will be a non-issue in multi-player games, and improve the motivation of applying our suggested technique.

7 CONCLUSION

We have shown that, of the four possible types of nodes, only one kind of node (i.e., type D nodes) increases evaluation error and therefore causes local pathologies in game trees. We also present a probabilistic approach to propagating the evaluation error based on the type of node. Using these rules, we have argued that such nodes exist in all interesting games, even those not known to be pathological.

We have presented a new algorithm, based on minimax, that propagates both heuristic values and error estimates on those values. The algorithm uses the error estimates to recognize and avoid searching pathological portions of a game tree, while still searching non-pathological portions of the tree. In this way the algorithm can adapt to the individual game tree and the degree of local pathology present.

In experiments performed on a board-splitting game, the algorithm performed well: it always performed best or nearly identical to minimax. The results show that the performance of EMM varies as the degree of local pathology in the game changes. This leads us to conclude that EMM will be most beneficial when used in games with a medium to high degree of local pathology, not just purely pathological games, such as P-games.

In conclusion, we can say that by incorporating the error of the static evaluation function in the search, we were able to improve upon the abilities of minimax in situations where such search previously performed badly. We think this may be a generally applicable lesson: when heuristic values exist in an algorithm, it may be advantageous to treat those values as probabilistically valid rather than blithely assuming them accurate.

ACKNOWLEDGEMENTS

This work was supported in part by AFOSR grant FA95501210021 and Army Research Office grant W911NF1110344. The opinions in this paper are those of the authors and not necessarily those of the funders.

REFERENCES