Avoiding Game-tree Pathology in Multi-player Games

Tal Shmueli
Department of Computer Science,
Bar-Ilan University, Ramat Gan, 5290002, Israel
tals@cs.biu.ac.il

Inon Zuckerman
Department of Industrial Engineering and Management,
Ariel University, Ariel, 40700, Israel
inonzu@ariel.ac.il

Abstract—Game-tree pathology is a phenomenon where searching a game-tree deeper gives results in worse decision. There are several solutions to dealing with game-tree pathology in two-players games, however there is no algorithmic solution available for coping with game-tree pathology in multi-player games. In this work we present the EMMN and $EMMN_v$, two algorithms that overcome pathology in multi-player games. Our presentation includes a theoretical analysis and an extensive empirical work on the multi-player version of the Nim board game.

Index Terms—Games; Search ; Game-tree pathology

I. INTRODUCTION

In Artificial Intelligence, game-tree search (also called “adversarial search”) is one of the oldest and most useful techniques for constructing an intelligent, automated game-playing agent [1].

The common assumption in game-tree search is that searching to deeper levels in the tree will result in better decisions, and consequently, stronger automated players. Nonetheless, in the early 1980’s, Nau [2] and Beal [3] independently discovered that there are infinitely many game that exhibit a phenomenon known as game-tree pathology, in which searching deeper gives worse decisions [4]. Moreover, recent work has shown that local pathologies can occur in all interesting games, and suggested the Error Minimizing Minimax (EMM) algorithm to recognize and overcome local pathologies [5].

In this research we will focus on game tree pathologies in Multi-player game trees. First we will show a trivial extension of the EMM algorithm for Multi-player games, denoted as Error Minimizing $Max^n$ (EMMN), and show that it can overcome local pathologies in Multi-player games. However, the extended algorithm described above is somewhat limited due to the assumption about uniform distribution among the players. Therefore, we will present a theoretical analysis regarding the basic and naive assumptions of the algorithm and overcome them in our improved algorithm, denoted as $EMMN_v$. In our $EMMN_v$ extension we improve the error equations by considering more factors from various features of the current game position, and avoid the assumption of uniform distribution in the heuristic function winner selection method.

Following the presentation of the algorithms we present an extended empirical evaluation on the Nim board game. We compare the performances of the $EMMN_v$ algorithm and show its strength both in the fraction of correct decision (thus, avoiding game-tree pathology) and in overall game-playing abilities by playing against and showing strong performances against the $Max^n$ and EMMN algorithm.

II. GAME-TREE PATHOLOGY

In the early 1980’s Dana Nau and others have discovered pathology theorems showing that searching deeper in the game tree causes the quality of the ultimate decision to become worse, not better [6], [3]. David Mutchler [7] extended their findings and proved that the $Max^n$ algorithm is also pathological in nature.

Nau also proved that P-Games, a class of board-splitting games invented by Judea Pearl [4] are pathological and defined the term degree of pathology, which is the ratio between the decision error when searching on two different lookahead depths. They defined $m(x)$ as the utility value for some node $x$ that will be obtained if both players play optimally, and $m_e(x,d)$ as the approximation of $m(x)$ that can be computed using the Minimax algorithm with max depth $d$, and also the following:

- $opt(x,d) = \{ y \in suc(x) : m_e(y, d-1) = m_e(x, d) \}$ - The set of successors of $x$ that look optimal according to the Minimax algorithm
- $P_{opt}(x,d) = \frac{|opt(x,d)|}{|opt(x,d)|}$ - The probability of making an optimal move if the player moves to one of the nodes at random
- $P_{err}(x,d) = 1 - P_{opt}(x,d) = 1 - \frac{|opt(x,d)|}{|opt(x,d)|}$ - The Minimax decision error, which is the probability of moving to successor $y$ of $x$ such that $m(y) \neq m(x)$.
- Finally, $p(x,i,j) = \frac{P_{err}(x,j)}{\mathcal{P}(x,j)}$ The degree of pathology.

A two-player game model is considered pathological if $p(x,i,j)$, averaged over $x$, is greater than 1, depending on the game being studied. Nau claims in his work that when
a game or model is pathological for some values of i and j, usually it will also be pathological for other values.

III. ANALYSIS

In this paper we extend upon a recent work by [5] showing that pathology is a local phenomena of various areas in the game-tree. We assume a static evaluation function that returns the correct utility value on any given node with probability 1−e (i.e., incorrect values will be returned with probability e).

In the Max^n case there are multiple incorrect values, one for each non-winning player, each of which are chosen with equal probability on an error. We will be looking at the evaluation error at nonterminal nodes. Evaluation error occurs when a node’s value is miscalculated by a depth-limited computation.

In games of branching factor two, there are four possible kinds of non-terminal nodes, as shown in Figure 1. At each node, it is player 1’s move, so the node’s Max^n value is the maximum of the Max^n values of its children (which are not terminal, but rather the search’s horizon). When examining the different node types, a value of 1 will correspond to a vector where the player’s evaluation will equal 1, while the other vector elements will be −1. However, a node value of −1 can represent any of the possible losing scenarios for the principle player. In our examples, a value 1 for a player will correspond to a player winning, in which case all other players will have value -1. In the Max^n search, these values are propagated up the tree in vectors. For instance, in a 4 player game, a node value of −1 for player 1 represents a loss and means that one of players 2, 3 or 4 have won the game.

For a game with three players, a depth-one Max^n search at each node using an evaluation function with error e will result in a new aggregate error at the node (assuming that when an error occurs, the probability that any other player wins is uniformly distributed among the remaining N−1 players):

\[
\begin{align*}
\text{error}(A) &= e^2 \\
\text{error}(B) &= (1−e) + e^2(N−2)/(N−1) \\
\text{error}(C) &= \text{error}(B) \\
\text{error}(D) &= 1−(1−e)(1−e + e(N−2)/(N−1)),
\end{align*}
\]

\[N\text{ is the number of players in the game.}\]

Comparing these functions to 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 ∈ [0, 1]. 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 show this relationship in a graph Max^n 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 are the source for search pathology. As noted in [5] we can expect all interesting games to contain nodes of type D.

IV. THE EMMN ALGORITHM

We now describe the naive extension of the EMM algorithm (that was presented in [5]) to the Multi-player version, denoted as EMMN (see Algorithm 1). As before, our search algorithm tracks the error associated with the node value, which is now in the form of a vector of size n, where n is the number of players. The search computes the static evaluation function at any given node. If the static evaluation allows a tighter error bound than the propagated vector then that value and error bound are substituted in the final return statement. By keeping track of both the error from searching and the error from evaluating , the algorithm naturally distinguishes between pathological nodes (type D) and non-pathological nodes (types A,B and C). Further, notice that neither algorithm is limited to branching factor two. EMMN also suffers from EMM weaknesses, the assumption of a particular form of static evaluation function with a constant error rate.

V. EXTENDING EMMN TO OVERCOME ASSUMPTIONS

The EMMN algorithm is based on uniform winning distribution, and by four different node types. However, for the purposes of our research we propose a new set of probability equations. We will focus on two different node types, winning node type, and non-winning node type, which we will name D and D respectively. Our model specifies EMMN probability functions based on these types. First, we will determine the following basic definitions:

1. For multi-player game with set of players P (size N), set of possible game states S for each evaluation function \(f_x : S \times P \rightarrow \mathbb{R}^n\) and for every \(i \in \{1...N\}\) we will define \(e(f, i)\) as the probability that \(f\) incorrectly evaluated player’s i utility value (in vector notation, \(e_i^2\)).
2. For some node \(m\) in a multi-player game tree with N players, where player \(i \in \{1...N\}\) plays the current turn, and depth d, we will define \(p(m, d, i)\) as the propagated error vector that contains the data about how \(f\) incorrectly evaluated each player’s utility value.
3. \(opt(m, d) = \{y \in suc(m) : m_e(y, d-1) = m_e(m, d)\}\)
   The set of successors of \(m\) that look optimal according to the Max^n algorithm [2].

Second, the propagated error value will be calculated recursively by our algorithm in the following manner, dividing into two different cases :

For \(D\) node type:

\[
p^t(m, d, i) = 
\begin{cases} 
\min(p_{e \in opt(m, d)} \min\{p^t(x, d-1, (i+1) mod N),\} & |i = j| \\
p_{e \in opt(m, d)} (1−p^t(x, d-1, (i+1) mod N)) & |i ≠ j|
\end{cases}
\]

1Generalizing to more players would only increase the set of possibilities at the losing nodes.
Algorithm 1 EMMN($s, \text{eval}, d, p$): Error minimizing $Max^n$ search. From the perspective of player $p$, for game state $s$, and search depth $d$, this function returns a pair $(a, e)$ where $a$ is the valuation of the state $s$ and $e$ is the error associated with that valuation. $\text{eval}$ is the evaluation function (with error $e$), returning a vector such that $\text{eval}(s)[i]$ is the utility for player $i$. $N$ is the number of players in the game.

Let $\text{curVal} = \text{eval}(s)$, and $\text{curErr} = e$.
If $d$ is 0, return $(\text{curVal}, e)$
// Determine values $v_i$ and errors $e_r_i$ for children nodes.
Let $mv_1, \ldots, mv_n$ be the children of $s$.
for $i = 1, \ldots, n$ do
$(v_i, e_r_i) = \text{EMMN}(mv_i, \text{eval}, d - 1, (p + 1) \mod N)$
end for
Let $val = \max_x(v_i[p])$, and $w = \arg \max_x(v_i[p])$.
// Determine error for this node $aggErr$.
if $val$ is a loss then
$aggErr = 1 - (1 - e_r_w) \prod_{i \neq w} 1 - e_r_i + e_r_i \frac{N - 2}{N - 1}$
else
Let $aggErr = 1 - e_r_w$
for each $(v_i, e_r_i)$ where $i \neq w$ do
if $v_i$ is a win then
$aggErr = aggErr \times e_r_i$
else
$aggErr = aggErr \times \left((1 - e_r_i) + e_r_i \frac{N - 2}{N - 1}\right)$
end if
end for
// Check if static evaluation matches the $Max^n$ value.
if $\text{curVal} = \text{eval}$ then
// Return the result with the stronger error guarantee.
return $(\text{curVal}, \min(\text{curErr}, aggErr))$.
else if $\text{curErr} \geq aggErr$ then
// Non-pathological case: use $Max^n$ results.
return $(\text{val}, aggErr)$.
else
// Pathological case: use static results.
return $(\text{curVal}, \text{curErr})$
end if

For $D$ node type ($\text{opt}(m, d) = \{\}$):

$$p^j(m, d, i) =$$

$$\begin{cases} \min((1 - \prod_{j \in \text{suc}(m)} (1 - p^j(x, d - 1, (i + 1) \mod N)), e(f, i))) & |i = j | \\ p^j(x, d - 1, (i + 1) \mod N) & x \in \text{suc}(m) \end{cases}$$

The stopping condition for the recursion condition occurs if node $m$ is a leaf ($d=0$), then we will assign $p(m, d, i) = [e(f, j)]_{j=0}^d$

Finally, we will observe local pathology in cases where $p^j(m, d, i) > e(f, i)$ for some static evaluation function $f$, node $m$ in depth $d$, and error vector $e_r$ of function $f$. In these cases, the error resulting from searching below these nodes exceeds the error resulting from simply applying the static evaluation directly.

VI. EXPERIMENTS

The goal of our experiments was to examine and study the performance of both EMMN and $EMMN_w$ algorithms depending on the search depth in real multi-player games compared to the state of the art algorithm $Max^n$. Our experiments assume that the heuristic function error rate is known and therefore we used artificial static evaluation functions.

We will be using the game of Nim. Nim is a simple combinatorial game with finite set of possibilities. The game originally designed to be played by two players, A and B, but can easily be extended to its multi-player version. Upon a table are placed $n$ piles of objects of any kind, let us say counters. The number in each pile is quite arbitrary (but is larger than the number of players). A play is made as follows: The current player selects one of the piles, and from it takes as many counters as he chooses; one, two . . . or the whole pile. The only essential things about a play are that the counters shall be taken from a single pile, and that at least one shall be taken. The players play alternately, and the player who takes up the last counter or counters from the table wins.

A. Evaluating Fraction of Correct Decisions

A pathology is characterized by a decrease in correct decisions with an increase in search depth. Therefore, Figure 2 shows the fraction of correct decisions made by all tested algorithms (depth limited $Max^n$, EMMN, and $EMMN_w$)
averaged over 10 randomly selected Nim boards as the root node (due to performance issues). Both Max and the EMMN algorithms are using an artificial evaluator with a constant error rate (e = 0.3), EMMN \(_n\) algorithm error vector returned from the learning phase function which iterates over vectors in the corresponding vector space and returns the first vector that wins EMMN algorithm. Each game play contains two truly-random players, and one Max player (not depth limited). For each Max decision point we forced all other depth-limited algorithms to make a decision as well, and their decision is compared to the set of correct decisions returned by Max, a decision is considered correct if it’s contained in Max returned set of nodes \(opt(x, d)\) defined previously in this work. From the results in Figure 2 we can see that both EMMN and EMMN \(_n\) have better fraction of correct decisions in most of the different depths. In fact, at a search depth of 5, EMMN \(_n\) is making 20% more correct decisions than Max.

B. Evaluating Performance in Games

Figure 3 shows the fraction of game wins per for each tested algorithm, EMMN \(_n\), and two Max players over 40 randomly selected 3-columns Nim boards as the root node. Both Max algorithms are using an artificial evaluator with a constant error rate (e = 0.3), EMMN \(_n\), algorithm error vector returned from the learning phase function which iterates over vectors in the corresponding vector space (players count size) and returns the first vector that wins EMMN algorithm. The interesting comparison is actually between EMMN \(_n\) and one of the other tested Max players. We can see that in the lower depths, the EMMN \(_n\) algorithm wins fraction is at least the maximum between both other Max \(_n\) players, which is not true in higher depths. One of the explanation we suggest is traps as presented by Pearl [4]. Traps are moves that cause the game to end abruptly. In our examples traps can be observed after depth 3, the minimal depth in which a winner can be declared in our test game environment. We performed another set of experiments to validate our assumptions, by artificially limiting the minimum winning depth to be much higher (by enlarging the amount of Nim columns), and therefore avoid any kind of traps in lower depths.

VII. CONCLUSIONS AND FUTURE WORK

We introduced new version of pathology-aware algorithm for multi-player games, and performed several experiments on the Nim board in various settings and sizes which confirms that our EMMN \(_n\) algorithm using a static artificial evaluating function with known (or artificially learned) error rate vector performs better at almost every look ahead depths in Max \(_n\) players environment. On the other hand, we noticed that in cases when look ahead depth reaches the minimum game tree leaves depth that caused the games to end abruptly, we noticed a reduction in our algorithm performance.

There are numerous extensions to the model presented in this work, for which our results can establish the foundations for further research in this important area of game-tree pathology in multi-player game-trees. For example, expanding the evaluation function granularity to be non-binary.

REFERENCES


