

Error Minimizing Minimax Avoiding Search Pathology in Game Trees

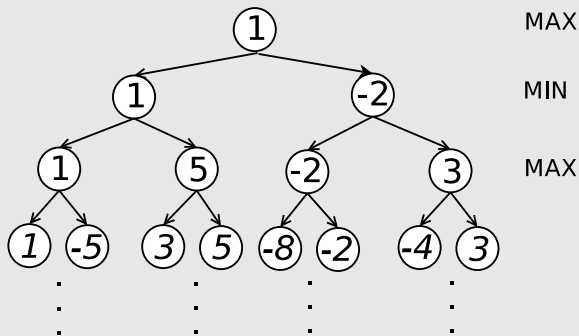
Brandon Wilson, Austin Parker, and Dana Nau

Department of Computer Science
University of Maryland, College Park

Motivation

- Traditionally minimax search proceeds to a limited depth frontier
- Nodes on search frontier are assigned a heuristic evaluation
- Values are propagated to the root to determine a course of action
- Traditional thought dictates that deeper search yields *better* decisions

Search example



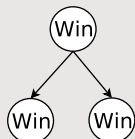
Motivation *cont'd*

- Discovered 30 years ago, game-tree pathology is a phenomenon where deeper minimax search *reduces* decision accuracy
 - This is counter-intuitive to conventional search wisdom
 - Games that exhibit this behavior are called *pathological games*
- Limited work has been done on searching pathological trees
 - This is because pathology has been thought not to occur in practice
- Our contribution is three-fold:
 - 1 Argue that even non-pathological games can have *local pathologies*
 - 2 Introduce a method of detecting local pathologies
 - 3 Overcome the negative effects of local pathologies on decision quality

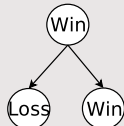
Subtree Pathology

- We restrict analysis to binary evaluation functions $[\pm 1]$
- Error of the evaluation function is the probability of incorrectly labeling a state (e.g. the evaluation function returns -1 when true value is 1)
- The error can be propagated based on branch type:

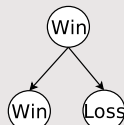
Types of branches (branching factor = 2):



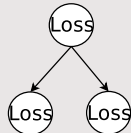
Type A - Forced Win



Type C - Critical Node



Type B - Critical Node

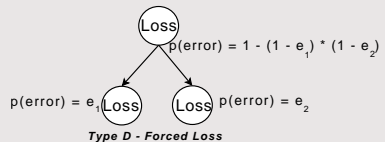
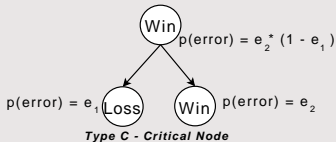
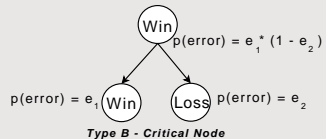
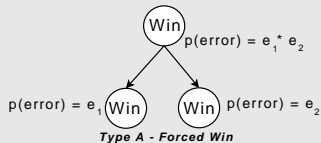


Type D - Forced Loss

Subtree Pathology

- We restrict analysis to binary evaluation functions $[\pm 1]$
- Error of the evaluation function is the probability of incorrectly labeling a state (e.g. the evaluation function returns -1 when true value is 1)
- The error can be propagated based on branch type:

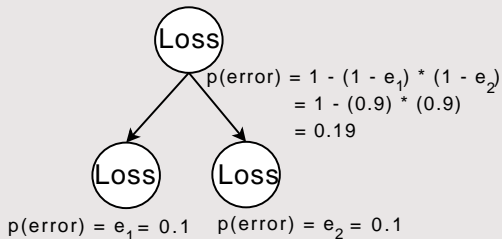
Types of branches (branching factor = 2):



Aggregating Error in a Generic Tree

Aggregating error examples (static evaluation error, $e = 0.15$):

If $e_1 = 0.1$ and $e_2 = 0.1$ then:

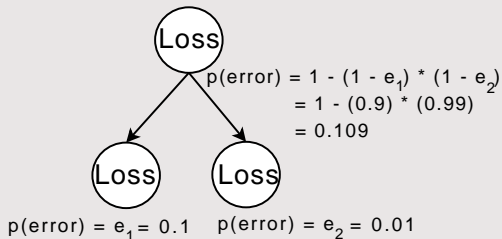


Local Pathology ($0.19 > 0.15$)

Aggregating Error in a Generic Tree

Aggregating error examples (static evaluation error, $e = 0.15$):

If $e_1 = 0.1$ and $e_2 = 0.01$ then:



No Local Pathology ($0.109 < 0.15$)

Error Minimizing Minimax (EMM)

General error aggregation:

$$\text{aggErr}(n) = \begin{cases} e_1 * e_2 & \text{if } \text{type}(n) = A, \\ e_1 * (1 - e_2) & \text{if } \text{type}(n) = B, \\ (1 - e_1) * e_2 & \text{if } \text{type}(n) = C, \\ 1 - (1 - e_1) * (1 - e_2) & \text{if } \text{type}(n) = D. \end{cases}$$

Error Minimizing Minimax (EMM)

General error aggregation:

$$\text{aggErr}(n) = \begin{cases} e_1 * e_2 & \text{if } \text{type}(n) = A, \\ e_1 * (1 - e_2) & \text{if } \text{type}(n) = B, \\ (1 - e_1) * e_2 & \text{if } \text{type}(n) = C, \\ 1 - (1 - e_1) * (1 - e_2) & \text{if } \text{type}(n) = D. \end{cases}$$

EMM: Propagate both value and probability of error, apply static evaluation when probability of error becomes *too high*

$$\text{EMM}(n) = \begin{cases} \text{eval}(n) & \text{if } d = 0, \\ u(n) & \text{if } n \text{ is terminal,} \\ \max_{n' \in m(n)} \text{EMM}(n') & \text{if p1's move,} \\ \min_{n' \in m(n)} \text{EMM}(n') & \text{if p2's move.} \end{cases}$$

- Start with traditional minimax, propagating only heuristic values

Error Minimizing Minimax (EMM)

General error aggregation:

$$\text{aggErr}(n) = \begin{cases} e_1 * e_2 & \text{if } \text{type}(n) = A, \\ e_1 * (1 - e_2) & \text{if } \text{type}(n) = B, \\ (1 - e_1) * e_2 & \text{if } \text{type}(n) = C, \\ 1 - (1 - e_1) * (1 - e_2) & \text{if } \text{type}(n) = D. \end{cases}$$

EMM: Propagate both value and probability of error, apply static evaluation when probability of error becomes *too high*

$$\text{EMM}(n) = \begin{cases} \text{eval}(n) \text{ with error } e & \text{if } d = 0, \\ u(n) \text{ with error } 0.0 & \text{if } n \text{ is terminal,} \\ \max_{n' \in m(n)} \text{EMM}(n') \text{ with error } \text{aggErr}(n) & \text{if p1's move,} \\ \min_{n' \in m(n)} \text{EMM}(n') \text{ with error } \text{aggErr}(n) & \text{if p2's move.} \end{cases}$$

- Start with traditional minimax, propagating only heuristic values
- Then propagate **error** with the respective heuristic values

Error Minimizing Minimax (EMM)

General error aggregation:

$$\text{aggErr}(n) = \begin{cases} e_1 * e_2 & \text{if } \text{type}(n) = A, \\ e_1 * (1 - e_2) & \text{if } \text{type}(n) = B, \\ (1 - e_1) * e_2 & \text{if } \text{type}(n) = C, \\ 1 - (1 - e_1) * (1 - e_2) & \text{if } \text{type}(n) = D. \end{cases}$$

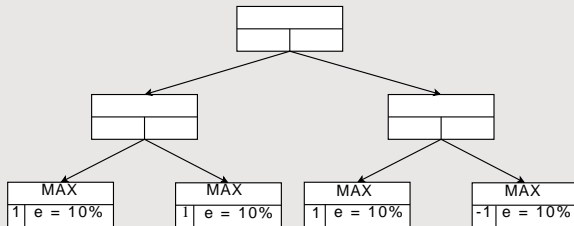
EMM: Propagate both value and probability of error, apply static evaluation when probability of error becomes *too high*

$$\text{EMM}(n) = \begin{cases} \text{eval}(n) \text{ with error } \mathbf{e} & \text{if } d = 0, \\ u(n) \text{ with error } \mathbf{0.0} & \text{if } n \text{ is terminal,} \\ \max_{n' \in m(n)} \text{EMM}(n') \text{ with error } \text{aggErr}(n) & \text{if p1's move and } \text{aggErr}(n) \leq \mathbf{e}, \\ \mathbf{eval}(n) \text{ with error } \mathbf{e} & \text{if p1's move and } \text{aggErr}(n) > \mathbf{e}, \\ \min_{n' \in m(n)} \text{EMM}(n') \text{ with error } \text{aggErr}(n) & \text{if p2's move and } \text{aggErr}(n) \leq \mathbf{e}, \\ \mathbf{eval}(n) \text{ with error } \mathbf{e} & \text{if p2's move and } \text{aggErr}(n) > \mathbf{e}. \end{cases}$$

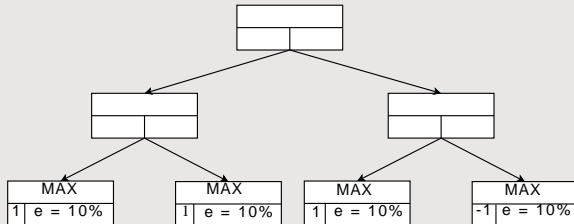
- Start with traditional minimax, propagating only heuristic values
- Then propagate **error** with the respective heuristic values
- Finally, when the **aggregate error exceeds the static evaluation error**, apply the static evaluation function

Search Example ($e = 10\%$)

Example: Level 3 Search by Minimax

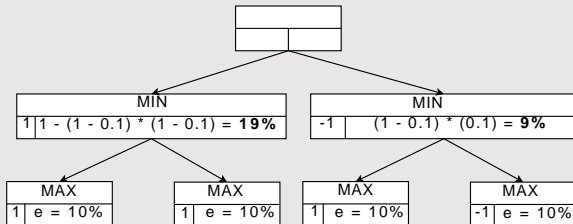


Example: Level 3 Search by EMM



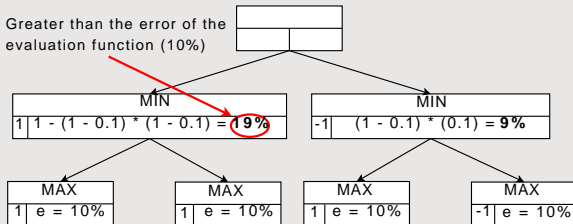
Search Example ($e = 10\%$)

Example: Level 3 Search by Minimax



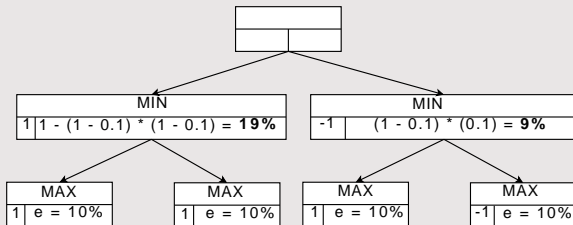
Example: Level 3 Search by EMM

Greater than the error of the evaluation function (10%)



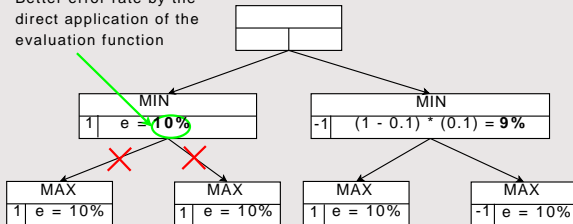
Search Example ($e = 10\%$)

Example: Level 3 Search by Minimax



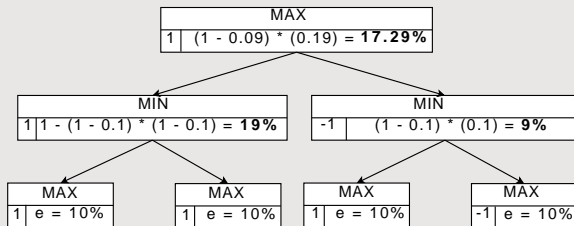
Example: Level 3 Search by EMM

Better error rate by the direct application of the evaluation function

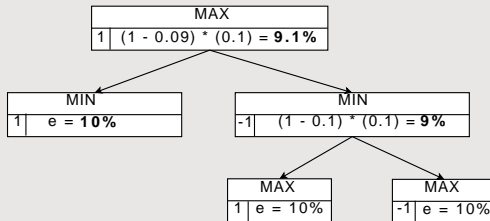


Search Example ($e = 10\%$)

Example: Level 3 Search by Minimax



Example: Level 3 Search by EMM



Experimental Methodology and Setup

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)

1	0	0	0	0	1	1	1
1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0
1	0	1	1	1	0	1	0
1	0	1	0	0	0	1	1
0	1	1	1	1	1	1	0
0	1	1	0	1	0	0	1

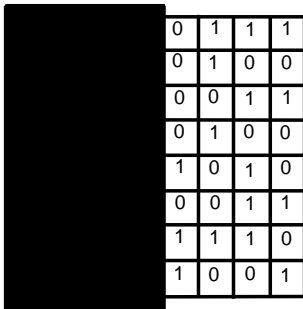
Board Splitting Game Example

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Initial board:

1	0	0	0	0	1	1	1
1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0
1	0	1	1	1	0	1	0
1	0	1	0	0	0	1	1
0	1	1	1	1	1	1	0
0	1	1	0	1	0	0	1

Board Splitting Game Example

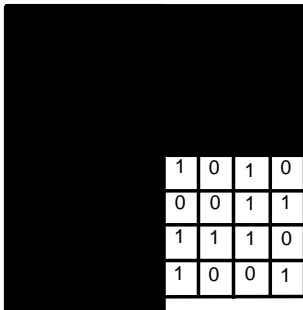
- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 1 splits the board vertically, throwing away half:



0	1	1	1
0	1	0	0
0	0	1	1
0	1	0	0
1	0	1	0
0	0	1	1
1	1	1	0
1	0	0	1

Board Splitting Game Example

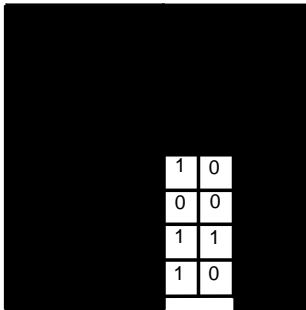
- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 2 splits the board horizontally, throwing away half:



1	0	1	0
0	0	1	1
1	1	1	0
1	0	0	1

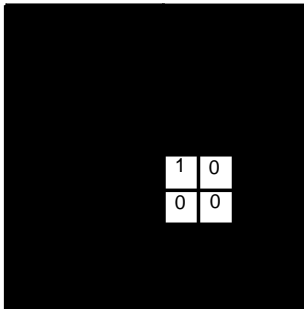
Board Splitting Game Example

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 1 splits the board vertically, throwing away half:



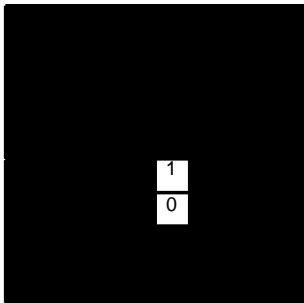
Board Splitting Game Example

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 2 splits the board horizontally, throwing away half:



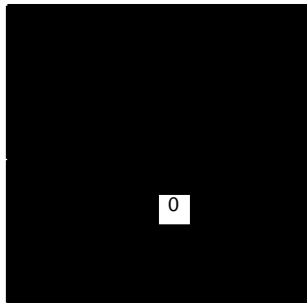
Board Splitting Game Example

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 1 splits the board vertically, throwing away half:



Board Splitting Game Example

- Domain: board-splitting game played on a grid of 1's and 0's (called P-games and N-games)
 - Player 1 splits the board vertically
 - Player 2 splits the board horizontally
 - The game ends when only one square remains:
 - If the square is a 1 then the last player to move wins
 - If the square is a 0 then the last player to move loses
 - Difference between P-games and N-games is how the initial board is set up (see paper for details)
- Player 1 wins because player 2 splits horizontally leaving a single 0:



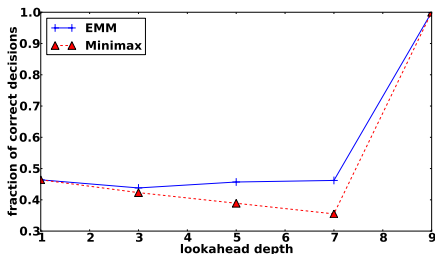
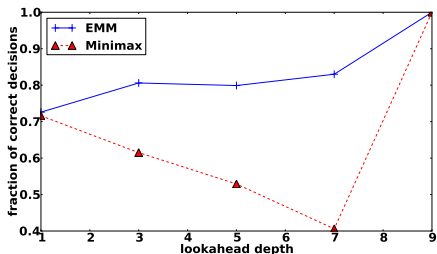
Evaluation Functions

Two binary evaluation functions are examined:

- *Artificial* – given an error rate e , returns the true Minimax value with probability $1 - e$
- *Natural* – a binary approximation to a standard natural evaluation function for the board-splitting
 - Binary evaluation – a board with more wins is evaluated as a win and a board with more losses is evaluated as a loss

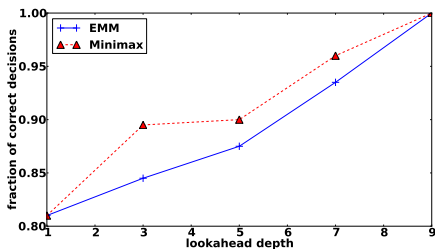
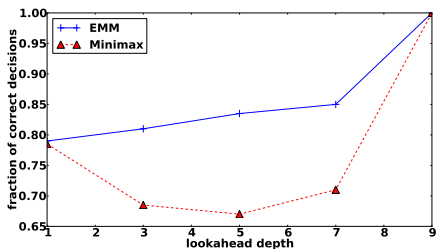
1	0	0	0	0	1	1	1
1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0
1	0	1	1	1	0	1	0
1	0	1	0	0	0	1	1
0	1	1	1	1	1	1	0
0	1	1	0	1	0	0	1

P-game Experimental Results (Artificial evaluator, $e = 0.2$)



- EMM is not pathological with either evaluator in P-games (known pathological game for Minimax)

N-game Experimental Results (Artificial evaluator, $e = 0.2$)



- EMM accuracy improves significantly faster than Minimax in N-games
- Evidence that detecting and accounting for local pathologies in non-pathological games can improve decision accuracy

Conclusions

- Defined *local pathology* in the context of an individual game tree
- We believe that local pathologies can occur in all interesting games
- Introduced EMM to identify such pathologies and reduce search depth appropriately
- EMM consistently outperformed Minimax

Future Work

- Investigate pruning method similar to alpha-beta
- Extend the EMM algorithm to incorporate non-binary evaluation functions
- Apply EMM to a real game such as chess or checkers