Expectiminimax for Othello RNG Variant

Andrew Mathies
Mathew Denney
Nick Irmscher

April 28, 2018

1 Introduction

Othello is a board game for two players played on an 8x8 board. Players are assigned either black or white tiles to put down on the board. When players place down a tile of their color, they flip their opponent's tiles adjacent to it. The objective of the game is to have as many of one color of tile on the board as possible. In order to increase the difficulty of solving a conventional Othello board, we have added the possibility for a random tile to flip every few rounds. Generally, the Minimax algorithm is used to solve Othello, but this calls for Expectiminimax, a specialized version of Minimax used for games with chance elements. Implementing IDS with a transposition table was definitely one of the more difficult parts of this project.

2 Algorithms

We used Memoization, IDS, and Expectiminimax in our project. Usually, instead of Expectiminimax, Othello AI's will use a Minimax or Negamax implementation. Our version of Othello calls for Expectiniminax because of its random element.

2.1 Memoization

Memoization is a technique used to optimize searches by caching results of the search in order to prevent the same search from being performed multiple times. We implemented a transposition table to store the results of IDS searches and optimize our program.

2.2 Expectiminimax

Expectiminimax is a version of Minimax used in games concerning RNG, or pure chance. It works just like a minimax tree, but with every level of nodes containing a ”chance” node equal to the projected value of an event taking place. One of Expectiminimax’s biggest pitfalls is that the tree grows much faster and as a result less accurate over time.

Time complexity: *unknown*
Space complexity: *unknown*

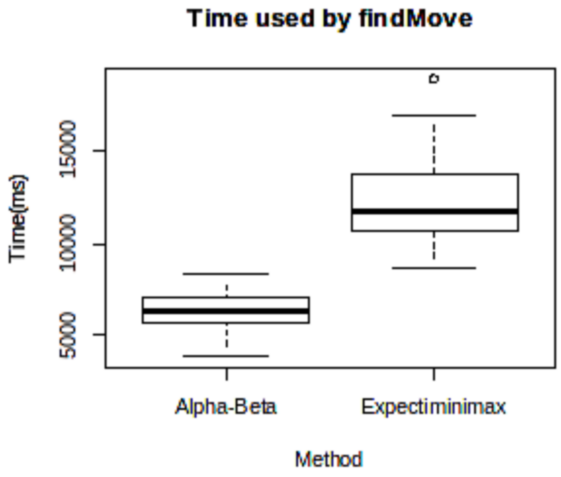
2.3 Iterative deepening depth-first search (IDDFS)

Iterative deepening search is a search method in which depth-first search is run repeatedly with increasing depths. We implemented this in conjunction with a transposition table to prevent our findMove() function from being run with the same inputs multiple times.

Time complexity: $O(b^d)$
Space complexity: $O(d)$

2.4 Results

These results were obtained using Python’s cProfiler profile. Depicted are the average time of running the findMove() method in a game with players of depth two versus depth three. These were repeated 30 times for accuracy.



Expectiminimax is much more efficient than a plain Minimax implementation. This is because it accounts for the random aspect of our Othello implementation.

3 Conclusion

We learned that IDS is actually only productive when used in conjunction with a transposition table.