

Reversi

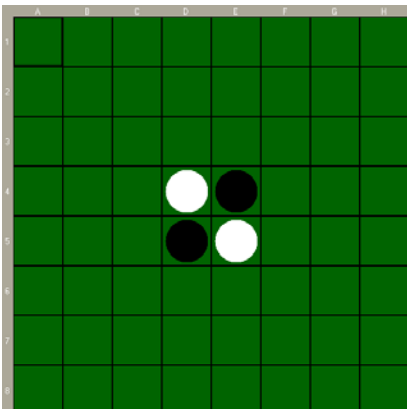
Meng Tran – tranm@seas.upenn.edu
Faculty Advisor: Dr. Barry Silverman

Abstract:

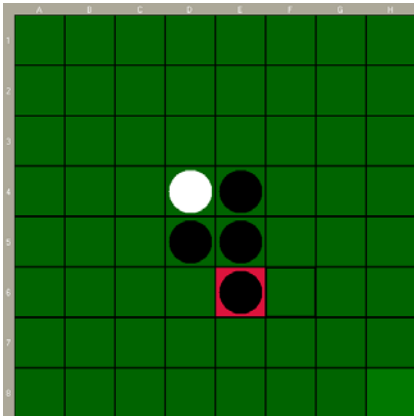
The game of Reversi was invented around 1880 by two Englishmen, Lewis Waterman and John W. Mollett. It later became incredibly popular when Mattel produced the game Reversi under the name Othello. Now the game of Reversi is also known as Othello. It took its name from Shakespeare's play Othello probably referencing the conflict between the Moor Othello and Iago. Since 1978 there has been a World Othello Championship. The current reigning champion is Hideshi Tamenori from Japan.

Rules:

The game is played on an 8 by 8 board. The board can be bigger but it is usually 8 by 8 for a standard game. The game starts with 4 pieces of each color already on the board.



White pieces are placed on D4 and E5 while black pieces are on E4 and D5. Black always goes first. It is up to the players to decide who will be black.



Here Black plays on E6, which flanks the original white piece on E5 making it switch colors. Each player alternately takes turns. Each player must make a move whether they want to or not so each player is forced to move if there are any valid moves. Every piece that is flanked by the opponent's piece on both sides switches colors. This is in every direction, horizontal, vertical and diagonal. This continues until all the places have been filled or neither player can make a move. The winner is the player who has a higher piece count than the other person. Therefore there are three outcomes to the game, you can win, lose or draw.

Strategies:

Reversi is a zero-sum game with perfect information and ends in a finite number of moves. Since all of the pieces are on the board and

you can perform a look-a-head to figure out if you win or lose, this is the perfect game for a computer to play. The rules are fairly simple, much simpler than chess and Go. Therefore a computer performing look-a-heads on every move can theoretically play perfectly all the time.

Corners:

Probably the most important strategy in the game is to capture the corners. With the corners, you can flank your opponent's pieces on the edges or through the whole diagonal of the board. Plus you gain the advantage of staying alive as these corners can not be flanked. Therefore the squares around the corner are dangerous if you play into them. This gives the opponent an opportunity to grab a corner gaining the advantage.

Edges:

Edges are rows 1 and 8 and columns A and H that are not corners. Edges are important as well as it gives you the most potential in gaining more pieces across the board. You can use an edge piece to flank a whole row gaining 8 more pieces on the board. Also edge pieces can only flank other pieces therefore it is the second safest spots behind corners.

Mobility:

One of the best strategies that are hard to pull off is to limit your opponents move until they have no moves left and must forfeit a turn. This requires extensive look-a-head and usually requires you to

dominate one side of the board along the edge. Once this is achieved you have gained an extra turn therefore more opportunities to flip more pieces to your advantage.

Endgame:

This is the end of the game when there are about 8 to 12 pieces left to play. This is probably the most important part of the game as a good play can result in a win even though the player has been losing throughout the game. Looking ahead is very simple at this point as there are fewer squares to consider. Anybody can look-a-head a few pieces quite easily. Look ahead is a strategy itself as it is the best known strategy to find the best move for a perfect game.

Related Work:

There have been many computer programs that play Reversi ever since its popularity in the 1980s. The best known program to date is Logistello which beat the reigning human champion, Takeshi Murakami in Reversi in 1997 in a 6 to 0 sweeping win. Although the look-ahead strategy can almost guarantee a win in Reversi, the computations that are required for checking the moves are intense. Reversi has also not been mathematical solved making it a non-trivial game. There have been numerous others both for research purposes in human decision making and tools for teaching. Games are fairly popular as tools for teaching students how to program. Chess rules

are simply too complicated to program and the artificial intelligence for even a decent program requires many strategies and tactics. Each chess piece also has a different type of move and each play has a different weight depending on the piece that occupies it. Tic-tac-toe is simply too easy. Although it is a fairly good game to start, you can not extend the game beyond a beginner's course. Reversi is just the right game as a teaching tool for anybody learning a new programming language or learning about artificial intelligence in general.

Technical Approach:

I have used Microsoft's .NET C# programming language for Reversi's game rules and framework. I chose C# because it is fairly easy to use and it wouldn't take up much time creating the environment. Plus the graphics are the easiest to work with out of all the other programming languages. Not to mention the programming environment is probably the best. This gave me the advantage to quickly setup the game and work on the artificial intelligence side.

For the artificial intelligence engine, I considered using Python since Python has been used fairly extensively in artificial intelligence research. Using Python in conjunction with C# would've been a fairly easy task. Although it would've been simple to use Python, there was always the risk of compatibility. I was also not as skillful in

programming with Python. So I made the decision to make everything in C#. This would allow for smoother transitions between player and computer. Since I didn't have to worry about converting data objects between the two languages, the program would run a bit faster.

Representation:

Representing the Reversi board was a fairly easy task as well. The game is already a matrix and all the information was available right on the board. It was either empty or had a stone of either color. I just used a matrix and represented the white stones as 1 and the black stones as -1. The empty positions were 0.

Rules:

The rules were fairly complicated to program, but it was doable. First it must be an empty position or else it's not a valid move since each move must flank an opponent's piece. Next the position must flank at least one of the opponent's pieces. This is easily checked by checking all directions for a different color and also ends with the player's color. Each direction can be represented as two for loops from -1 to 1, excluding no movement (0,0). This represented 8 directions from the current position. Checking the opposite color and within bounds was fairly easy.

Endgame:

The game ended when the player either forfeited the game with an option or the board was filled or there were no more moves for either

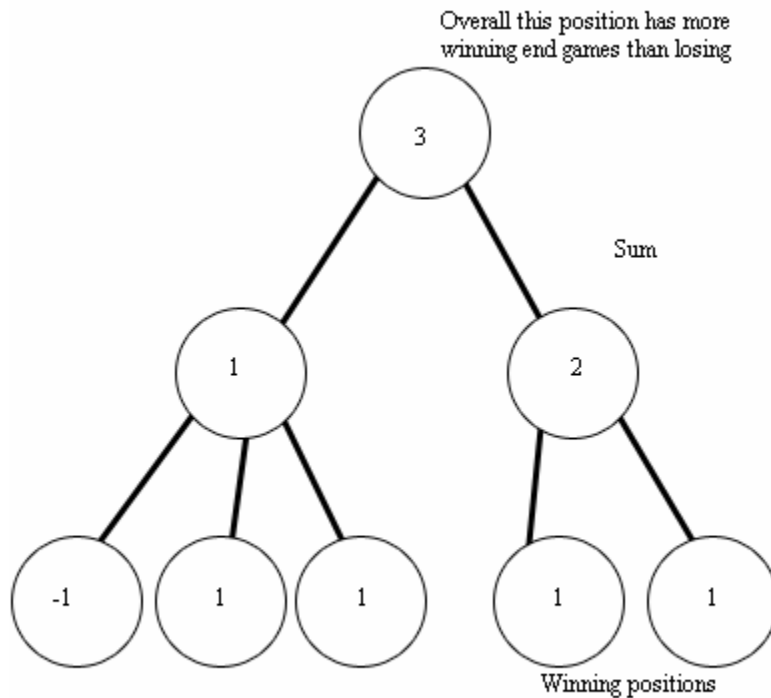
player. Checking whether it was a forfeit was a simple flag. Checking whether the board was filled was simple too. It was just checking if there existed an empty space on any position. Now checking if there was a valid move for either player was a bit more challenging, but simple. Just check every position on the board and if it is empty, then see if either player has a valid move on this position. If there is a valid move then the game has not ended. If all positions are checked and there are no valid moves then the game has ended. Count the number of each color and display the numbers and declare a winner or draw.

Artificial Intelligence:

This part was fairly complicated. First I needed a very simple move decision that wasn't just random. So I used a priority matrix that Peter Frey came up with in his research on human decision making [VALENTINE]. I checked to make sure that the weights made sense. The top priority was obviously the corners and the lowest would be the ones around them. All of the other weights had to be distributed with most of the higher ones on the edges along the rows 1 and 8 and columns A and H. Frey's matrix represented this perfectly. This was implemented fairly easily as each position had a weight and it just made sense to put them in a sorted list. As each one was taken, remove it from the list. The list has only positions of empty ones and

since the list is sorted, we can check for the highest priority empty position and take it. This is the default decision maker for the computer.

The smarter version of the computer AI is actually not that much smarter. It is still using the priority matrix except it does a look ahead on the endgame before a certain amount of moves usually 12. It duplicates the board and simulates the rest of the game. If the game ends in a win, 1 is returned or else if it loses then a -1 is returned and finally 0 for a draw. These numbers are used as a heuristic to determine which the best move to make is. The position before the final play gathers all of its children's heuristic and sums it up. It compares it with its neighbor and the higher heuristic position is filtered upward. The root then has the best position which gives the computer more winning chances. It returns a heuristic only when the opponent cannot move meaning that he must forfeit a move or its endgame, both of which are beneficial if you are in the lead. This limits the time it takes for the AI to consider a position, since it doesn't have to simulate until the end, only until the opponent cannot move.



The diagram shows how each parent sums up the children giving the heuristic of the win-lose ratio. If it is positive then there are more winning end games and if it is negative than the opponent has a better chance of winning. This method does not take into account certain positions that can guarantee a win say for example one definite route that ends in a win, the parent will always be 1. The AI will not choose this position as there are other positions that have a higher heuristic. This method also does not take into account the weight of the position say for example corners. Corners are crucial in endgame. Adding in the weights seems to ruin the system since the weight of the position will throw off the value of the end games.

Conclusion:

The priority matrix works surprisingly well and its performance is much better than a random guess from the computer. Against a new player using random guesses it can definitely win. Unfortunately it cannot learn and over time, the human player gets used to the game and defeats the AI easily. The modified look-ahead with the priority does not do as well but with the help of the priority matrix weights I believe it can be modified to be even better. Although I haven't tried using the smarter AI during the middle of the game due to time constraints, I am confident it can do well. There are many more ways of making it more efficient and cutting down the processing time. Alpha beta alone would make it feasible enough to play half the game. Although this would require an intense remodeling of the heuristic system in place, it would be much faster and more powerful. Artificial neural networking can be used as well as other learning methods to teach the computer how to play and get better. There are many advantages on using these techniques on Reversi instead of other games. The world seems to be focused on Go, but Reversi paved the way.

References:

Russel, Stuart J. Artificial intelligence : A Modern Approach. Upper Saddle River, N.J.: Prentice Hall/Pearson Education, c2003.

Rosen, Kenneth H, ed. Introducing Game Theory and Its Applications. Boca Raton: A CRC Press Company, 2004.

Valentine, David W. "Playing Around in the CS Curriculum: Reversi as a Teaching Tool." Consortium for Computing Sciences in Colleges. Online. ACM. 2005.
<<http://portal.acm.org/citation.cfm?id=1059888.1059948>>

Wikipedia. Online. 10 April 2007.
<<http://en.wikipedia.org/wiki/Reversi>>