Simon Fraser University

Computing Science 310 -
Artificial Intelligence:

Reversi with Monte-Carlo Tree Search

Instructor: Toby Donaldson

TA: Vincent Huang

Author: Jeff Wang

August 10, 2020

Table of Content

Pure MCTS Implementation

AI Design Phase

Designing Reversi AI using Monte-Carlo Tree Search (MCTS) requires an efficient programming language to iterate through multiple random playouts in order to compute the most optimal move. The performance of a programming language such as Python is less efficient at computing large iterations. However, Python is still efficient enough to implement a Reversi AI that can beat most average players.

Implementation

The Reversi AI takes the given board board state and performs a number of random playouts to select the most optimal move. A playout ends when no available moves are left and the score is tallied up. The piece with the most tiles wins and the initial move that led to the win is given a point. After a number of random playouts from the given board state, the move that led to the most number of wins is selected.

Performance

Doing a performance measure, the Python program can compute 300 random playouts in roughly 5 seconds on a new board state. Each playout consists of roughly 64 random moves which can be viewed as 64 board iterations. These iterations all generate a list of available moves which the Reversi AI randomly selects to proceed. For every real move, the AI performs 300 random playouts and selects the move that led to the most number of wins. The average player with knowledge regarding the rules of Reversi was not able to win once over 20 games.

Improving MCTS with Heuristics

Initially a greedy algorithm was implemented to capture the most number of pieces at any given point of the game. However, this technique proved to be ineffective against pure Monte-Carlo Tree Search (MCTS) since having the most pieces initially does not translate to having the most pieces at the end. This led to the idea of adding additional heuristics to MCTS which included *Corner Capturing*, *Piece Mobility* and a different *Scoring Heuristic*.

Corner Capturing

Unlike most board positions, the 4 corners are known to be the most essential positions. This is because the corners are a *stable position*; meaning it can not be captured by another piece. The second version of the Reversi AI prioritizes the corners and attempts to capture it at any available moment.

Piece Mobility

Another important aspect of Reversi is the number of available moves. A move is considered strong if it opens up more available moves for the player. The second version of the Reversi AI puts a small emphasis on the number of available moves. Given the board state, each available move is given a mobility score based on the number of moves it opens up for future board states.

Scoring Heuristics

Instead of just tracking the wins and loses for the 300 random playouts, the second version of the Reversi AI implements a scoring heuristic. For each random playout, the heuristic computes the difference between the two pieces and the priotizes on the moves that end in the largest number of pieces captured. The scoring heuristic also factors in the mobility score and returns the move with the highest overall score.

Comparison

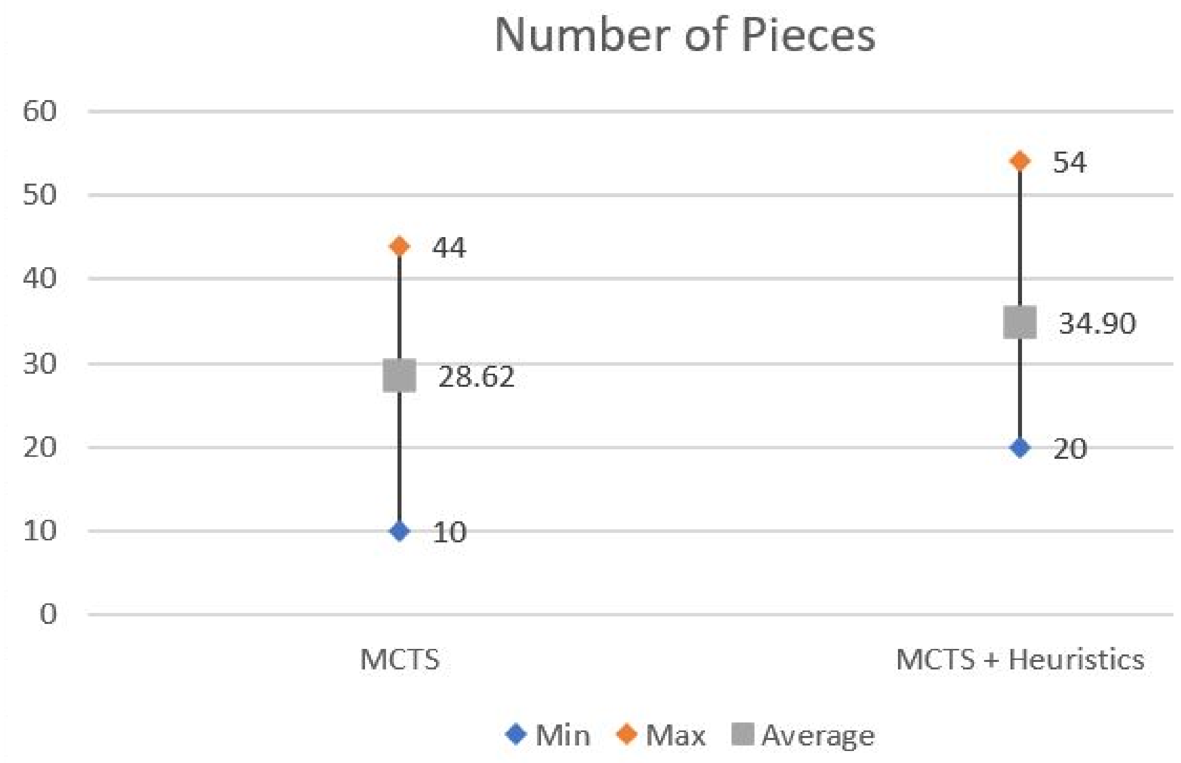
Data Analysis

A comparison was done between Reversi AI using pure Monte-Carlo Tree Search (MCTS) versus MCTS with heuristics. Over the 100 games, MCTS with Heuristics won 55 times, lost 43 times and tied 2 times; it had a 55% win rate over the pure MCTS Reversi AI. The additional heuristic to MCTS did not have a huge impact on its performance. However, it is quite noteworthy that applying human techniques such as corner capturing did in fact increase the win rate by apoximatly 5%.

	Total	Min	Max	Average	Median
MCTS					
Total Pieces	2862	10	44	28.62	31
Win	43				
Loss	55				
Tie	2				
MCTS + Heuristics					
Total Pieces	3490	20	54	34.90	33
Win	55				
Loss	43				
Tie	2				

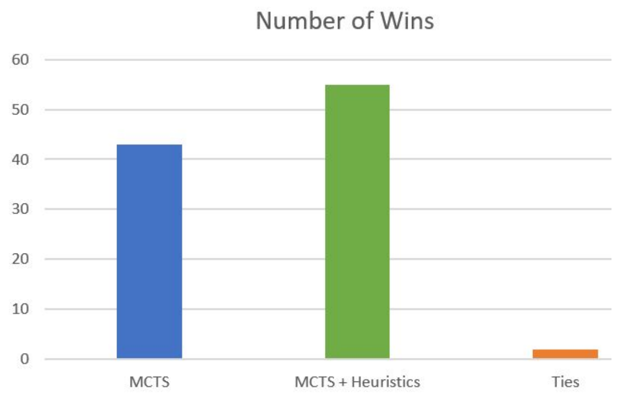
Table 1. Summary of the data over 100 games

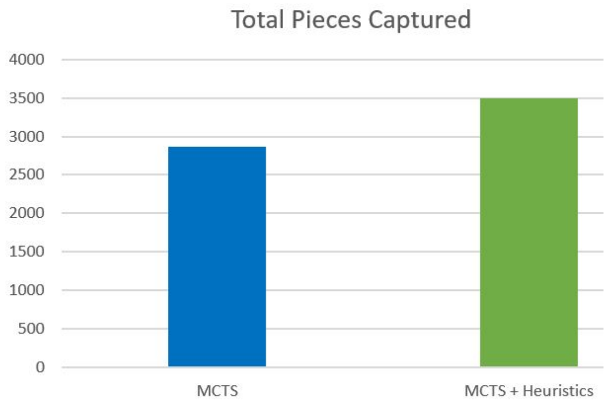
An interesting observation is that the AI using MCTS with additional heuristic plays 6 more pieces on average when compared to pure MCTS AI as in shown in *Graph 1*. Another interesting observation is that the minimum number of pieces captured by the AI using MCTS with heuristics is double that of pure MCTS (10 vs 20).



Graph 1. Min/Max/Avg number of pieces over 100 games

Finally, the total number of wins between the 2 Reversi AI is roughly proportional to the ratio of pieces captured by the two algorithms as seen in Graphs 2 & 3.





Graph 2. Number of Wins **Graph 3. Total Pieces Captured**

References

"Othello evaluation function", *Stack Overflow*, September, 2012 [Online], Available: https://stackoverflow.com/questions/13314288/need-heuristic-function-for-reversiothello-ideas [Accessed August 9, 2020]

iacta, "How to Win at Reversi", *Net4TV*, May 13, 2014. [Online], Available: https://guides.net4tv.com/games/how-win-reversi [Accessed August 9 , 2020]

"Strategy Guide for Reversi & Reversed Reversi", Samsoft, March 1, 2012 [Online], Available: http://samsoft.org.uk/reversi/strategy.htm [Accessed August 10, 2020]