From Calculating Areas to AI

Applications of Monte Carlo Methods

Qisheng Li, Zhaoqi Li, Zun Yin
Macalester College

MACALESTER

Introduction - Monte Carlo Method

Monte Carlo Methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
Monte Carlo Methods are applied in various fields of studies:

- ❏ Risk analysis in decision making
- ❏ Game designing
- ❏ Finance and insurance
- ❏ .....

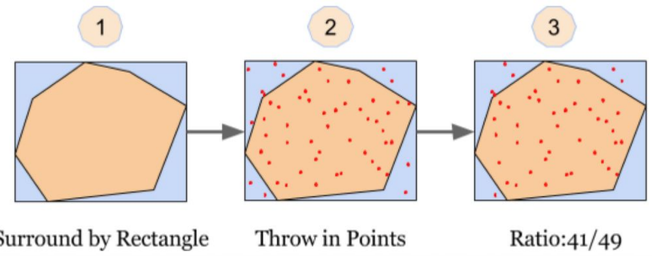
They are especially useful for simulating phenomena with *significant uncertainty in inputs* and *systems with a large number of coupled degrees of freedom*.

In our project, we employ Monte Carlo Methods in two different applications:

- ❏ Calculating Areas for Convex Polygons
- ❏ Using Monte Carlo Tree Search (MCTS) to improve the moving strategy of a popular board game — Reversi

Calculating Areas

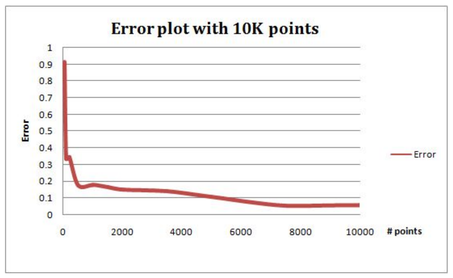
Algorithm for Area Calculation

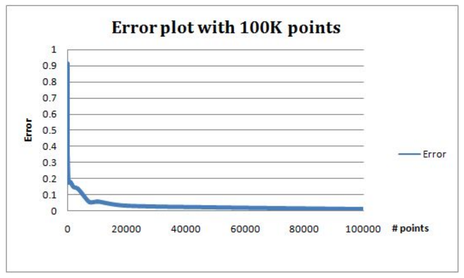


Given a polygon in random shape, we surround it with the smallest rectangle possible, and compute its area. Then we throw in points at random locations inside the rectangle and calculate the ratio of points lie inside the polygon. We use Convex Hull to check containment. It would be trivial then to compute the area of the polygon.

Further Analysis

We analyze the performance of our algorithm by testing on a hexagon with coordinates [(1,0),(2,0),(3,2),(2,4),(1,4),(0,2)]. The actual area of the hexagon is 8. We calculate the absolute error of the area calculated by throwing different number of points.





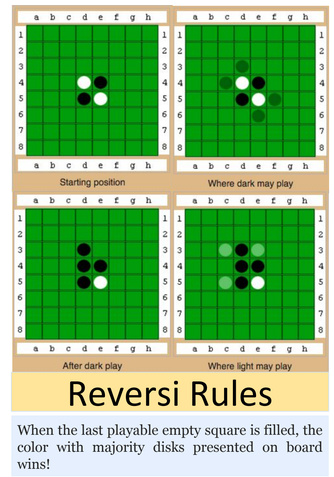
Reference

[1] Reversi Source Code: https://inventwithpython.com/chapter15.html
[2] Poster Template: http://www.makesigns.com/SciPosters_Templates_UOM.aspx
[3] Wikipedia: Reversi, Monte Carlo Method, Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) on Reversi

Reversi is a strategy board game for two players, played on an 8x8 board. When improving an existing Reversi program, we use Monte Carlo Tree Search (MCTS) method to determine computer's move by randomly simulating the games 100 times.

The basic procedure of MCTS can be applied to any game whose positions necessarily have a finite number of moves and finite length. For each position, all feasible moves are determined: k random games are played out to the very end, and the scores are recorded. The move leading to the best score is chosen.

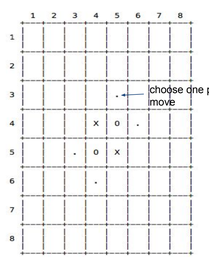


When the last playable empty square is filled, the color with majority disks presented on board wins!

MCTS Algorithm

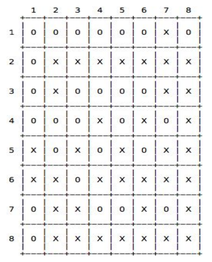
Algorithm for Reversi

Step1



For each step, choose one possible move among all possible moves.

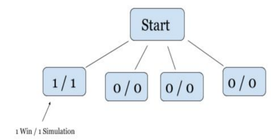
Step 2



X: 37
O: 27
Win!

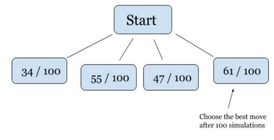
After making this move, randomly select each move until this game finishes.

Step3



Add the simulation result to the game tree. The result is: the number of wins / simulations.

Step 4



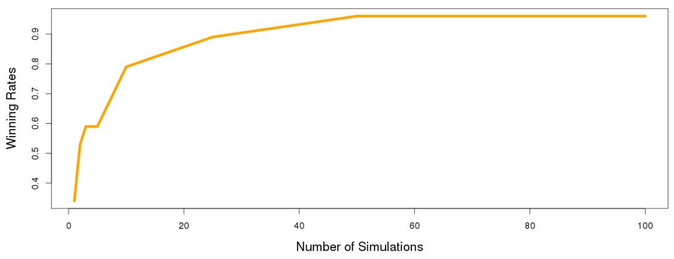
Simulate each possible move for 100 times. Pick the one with the highest winning probability.

Comparison Between AI

Original AI: Using scoring methods to find the optimal move. For each possible move, it counts the number of reversi pieces that are being turned over with this move. Then find the optimal move, which has the maximum number.
Our AI: Using Monte Carlo Tree Search to find the optimal move. It plays on the intermediate human level. (It beats all of our group members at least once.)
Results: 4 (Original AI) : 96 (Our AI)

Further Analysis

In our AI, an increasing number of simulations for each move results in better performance. Since 100 simulations already guarantee high enough winning rates against the original AI, we explore the cases when decreasing the number of simulations. The result shows that our AI is as good as the original one only with 2 simulations.



Acknowledgement

We would like to express our gratitude to Professor Shilad Sen for helping us understand the concept of Monte Carlo Tree Search, and giving suggestions to our topic.