Dominion AI Using Monte Carlo Tree Search

By: Jonathan Sarasua

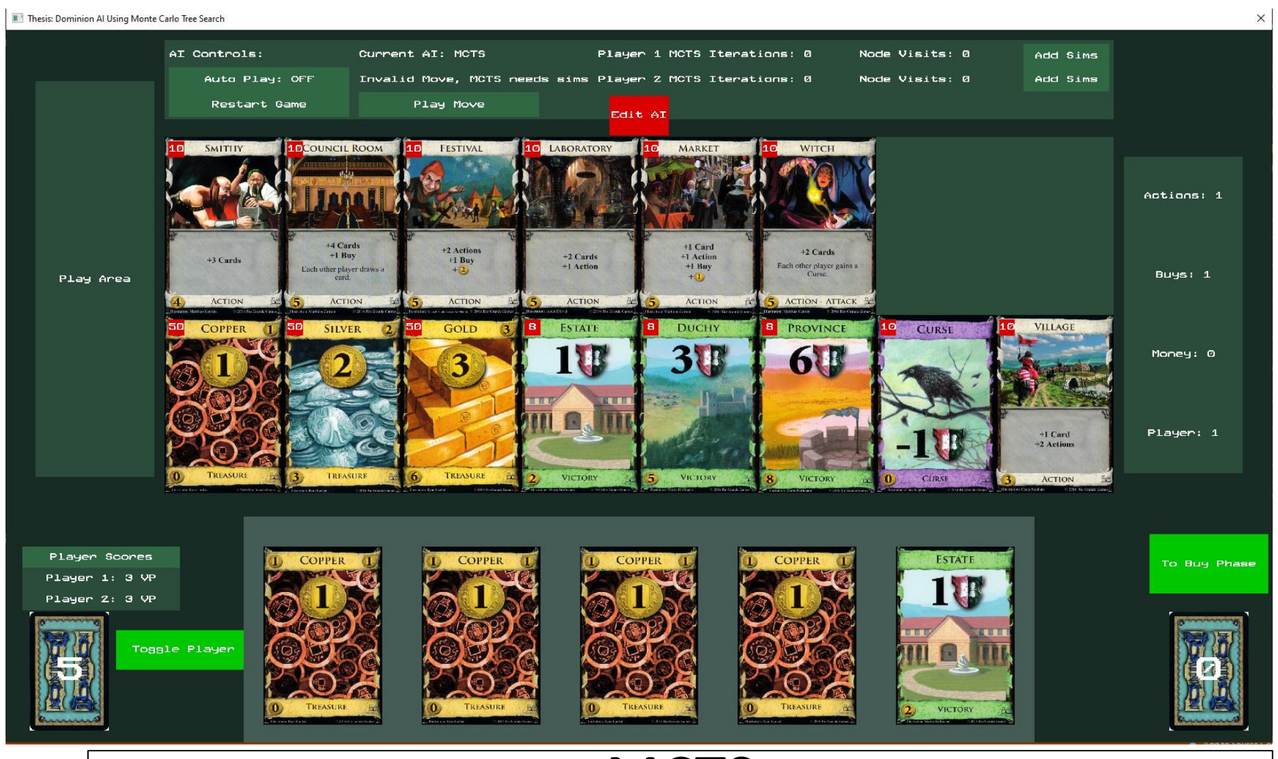
Dominion is a deckbuilding game of 2 to 4 players where each player takes turns playing and buying cards from a set of card piles

Goal: Have the most victory points from victory (green) cards

Game ends when: Province pile is empty OR three of any pile are empty

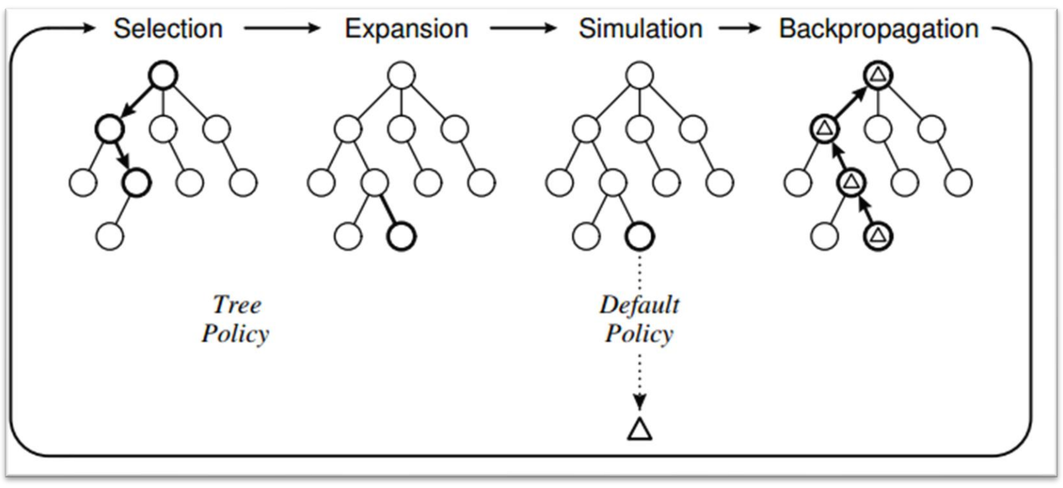
Each Turn:

1. Play 1 action
2. Buy 1 card from the available piles
3. End Turn, discard hand, and draw 5 new cards



Score = Wins/Simulations

$$UCT = Score + C^* \sqrt{\frac{\ln(\text{Parent Simulation})}{\text{Current Node Sims}}}$$



MCTS

MCTS is an AI strategy that tries to build the best incomplete tree that can always be asked what it believes is the best move.

To build the tree four steps are repeated as many times as possible: selection, expansion, simulation, and backpropagation

Selection: Start at the top of the tree and ask the current node is there a move that has not been explored at least once. If yes, move to expand step. If no, use the Upper Confidence Bounds Applied to Trees (UCT) formula to determine which child should repeat this question.

Expand: Add a random not explored child node to the given node.

Simulation: Run a game of random moves as a simple test of how good the game state is

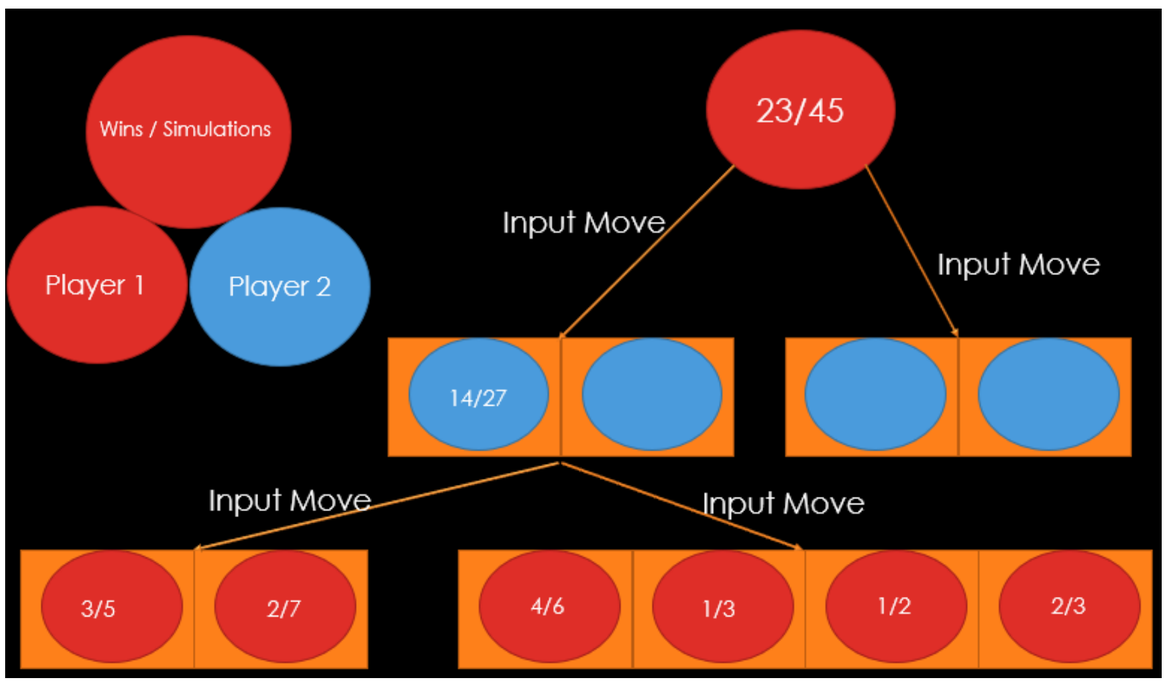
Backpropagation: Take the result from simulation and pass it back up the tree. This meta data will be used by the selection step to decide which node to expand

MCTS in Dominion

Problem: Each Move may have multiple outcomes, so how should MCTS decide what is the best node to select?

Solution: When moving down the tree, pass the game state and the move to make to the Game, take the resulting game state and compare it against the child game states. If it does not exist, the new game state is the node to expand. Change the UCT formula to use the sum of the possible outcomes instead of a single node.

$$UCT = Score + C^* \sqrt{\frac{\ln(\text{Parent Simulation})}{\sum \text{Outcome sims}}}$$



AI	Wins vs Big Money	Games Played	Win%
Random Moves	0	200	0%
MCTS 100,000 iterations per move using Random	4	20	20%
MCTS 20,000 iterations per move using RandomPLUS	61.5	100	61.5%

C Value	Wins vs C=√2	Games played	Win %
0.5	9.5	20	47.5%
5	9	20	45%

MCTS Using Big Money	Wins vs Big Money	Games played	Win %
1,000 iterations per move	85	200	42.5%
10,000 iterations per move	162	200	81%

Chaos Chance	Wins vs 0% Chaos Chance	Games Played	Win%
15%	27	50	54%

Results

My test AI is common AI strategy in Dominion called Big Money. It ignores action cards and focuses only on buying money and provinces. Random moves was shown to have a 0% win rate against Big Money.

Pure MCTS using random moves was found to perform poorly against Big Money due to order of cards played mattering. A modified version of random that plays actions giving more actions first showed a massive win rate improvement even on a lower number of iterations per move.

The theoretical value of C is √2, so other values were tried. The changes showed almost no difference, so they were deemed unimportant compared to changes in other areas.

A powerful ability of MCTS is making other AIs better by using the specified AI as the simulation step. What is interesting is 1,000 iterations per move using Big Money performed worse than just Big Money on its own. This shows there needs to be a minimum number before MCTS can have any confidence in its moves. The win rate doubling to 81% using 10,000 iterations shows that MCTS can make an AI significantly better.

Because using an AI that is not random moves as the simulation step will not try all moves, a chaos chance can be added that gives a percent chance that a random move is done instead of the given simulation AI. This change shows a minor improvement in results.