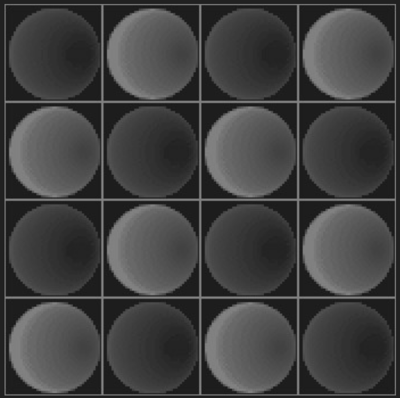
A CGT-Informed Clobber Player

CMPUT 655 Project



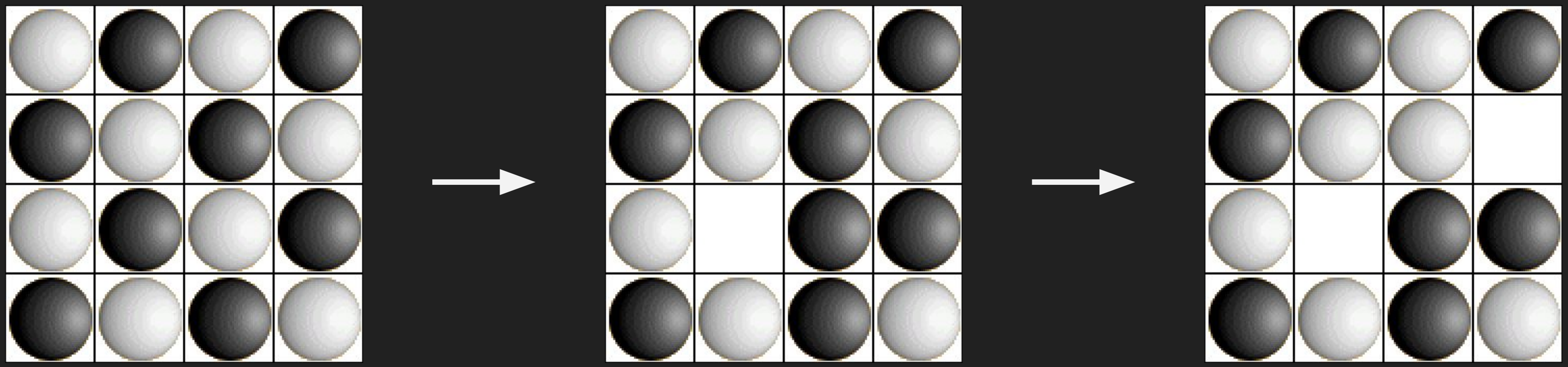
Kris De Asis

Project Objectives

- Create a strong Clobber playing program
- Investigate how CGT ideas can strengthen the player

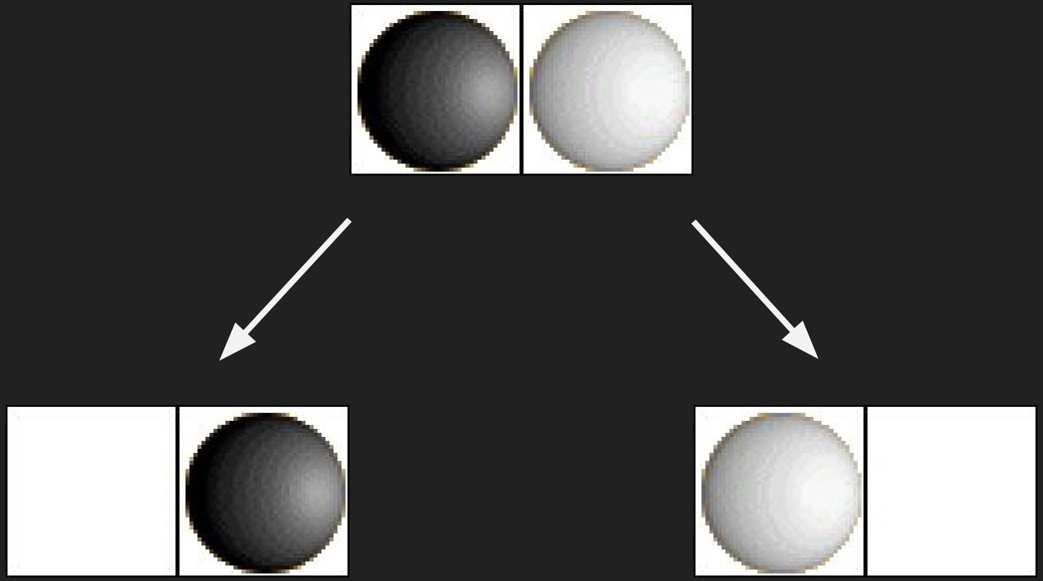
Clobber

- Typically starts on a rectangular grid of stones in a checkered pattern
- Moves consist of moving a stone of a player's color onto an adjacent stone of the opposing color, *clobbering* it



Clobber is All-Small

- For every move that a player has, the opposing player also has a move
- Under CGT, every game position has an infinitesimal value, and a temperature of zero

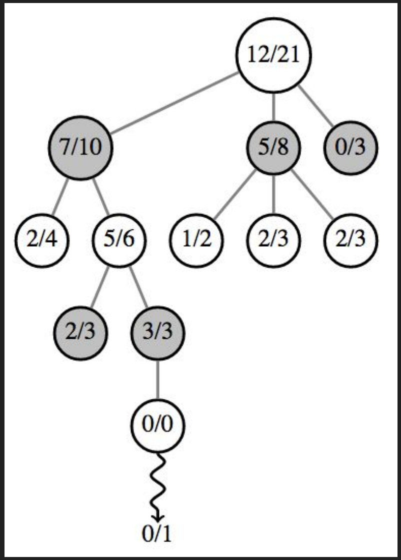


Clobber Framework

- Written in C++
- Uses bitboard representations
- Allows for incremental move updating
- Subgame extraction through computing bit masks

Baseline MCTS Player

- Implemented a Monte Carlo Tree Search (MCTS) player due to the difficulty in finding a good evaluation function
- Specifically used the UCT variant
- MCTS playout statistics were stored in a hashtable
- Used to evaluate performance of CGT-aware player



CGT - Outcome Classes

- Four outcome classes:
    - **N**: Next player wins, game is **confused with** zero
    - **P**: Previous player wins, game is **equal to** zero
    - **L**: Black player wins, game is **greater than** zero
    - **R**: White player wins, game is **less than** zero
- The **P** outcome class gives us the exact value of a game

CGT - Classifying Infinitesimals

- For a game G, the idea is to set up a sum of G and an infinitesimal such that the value of the sum game is zero when G is the infinitesimal of interest
- For example, if the outcome class of $G + \downarrow$ is computed to be a **P** position, the value of $G + \downarrow$ is **equal to** zero
- If $G + \downarrow = 0$, then $G = \uparrow$

- Adapted J. Fernando’s solver to compute the outcome classes for sum games, and made it an infinitesimal classifier

CGT - Zero Filtering

- If a subgame or a sum of subgames is equal to zero, they are a second player win
- Removing zeros or subgames that add to zero from the set of subgames does not change the outcome class of the original game

- Examples:
  - $* + * = 0$ → 2 games to remove
  - $\uparrow* + * + \downarrow = 0$ → 3 games to remove
  - $\uparrow\uparrow* + * + \downarrow + \downarrow = 0$ → 4 games to remove

CGT - Which Infinitesimals?

0	26177
*	21222
$\uparrow$	14793
$\uparrow*$	11150
$\uparrow\uparrow*$	4967
$\uparrow+\uparrow^2$	2563
$\uparrow\uparrow$	1979
$\uparrow\uparrow\uparrow$	1940
$\uparrow^2$	1891
$\uparrow\uparrow+\uparrow^2$	388
$\uparrow^3$	233
$\uparrow+\uparrow^3$	0
$\uparrow\uparrow+\uparrow^3$	0

- Frequency of select infinitesimals on boards that have up to eight pieces
- In addition to considering the frequency of an infinitesimal, have to also consider the odds of having the required infinitesimals to sum to zero

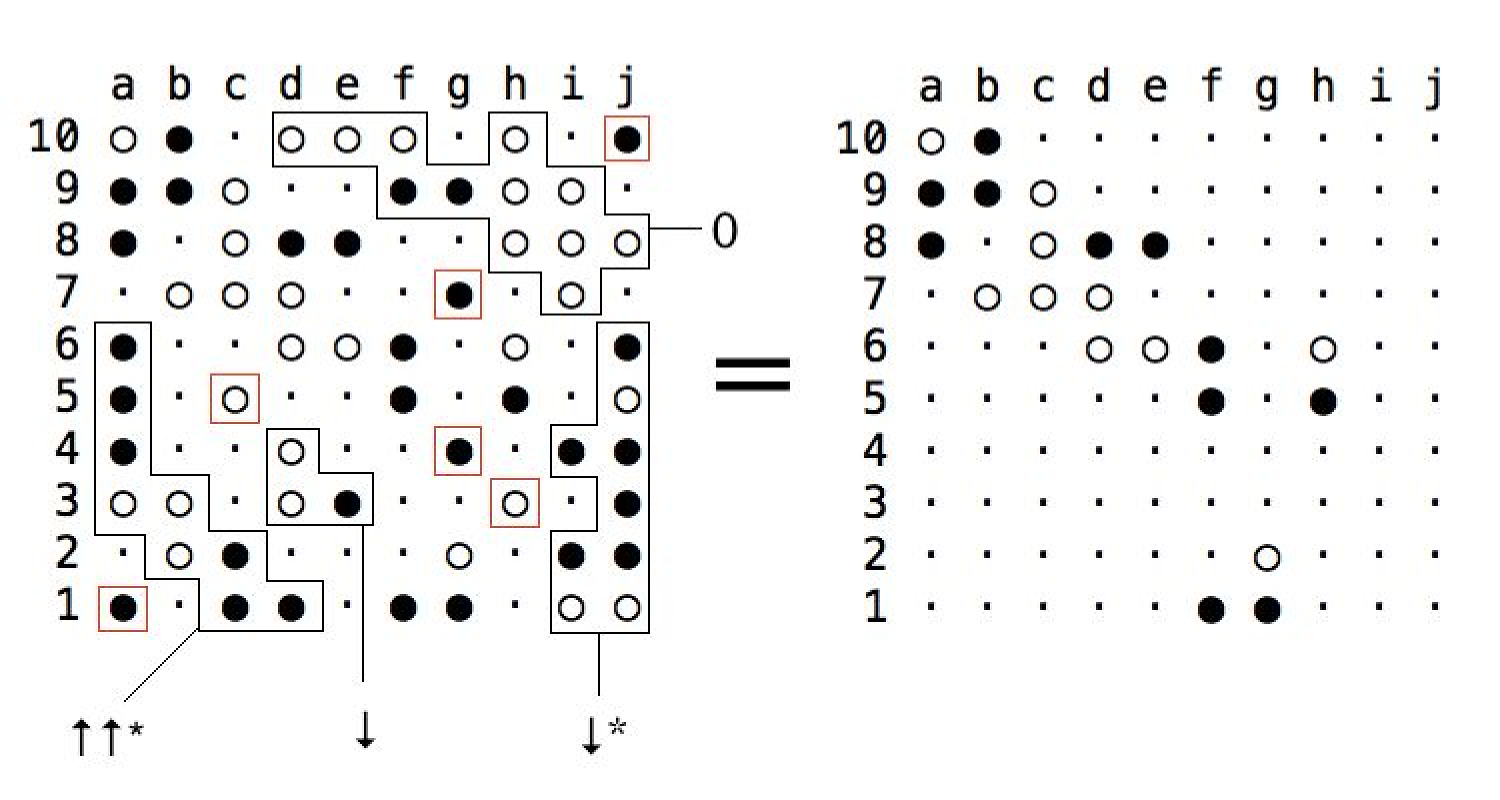
- Counted from J. Fernando’s database

CGT-Aware MCTS Player

- Built on top of baseline MCTS player
- When selecting a move, all subgames under a specified size are solved to try and identify the following infinitesimals:
  - 0, *, ↑, ↓, ↑*, ↓*, ↑↑*, ↓↓*
- Zeros are filtered from the board, as well as sums of subgames that equal to zero
- Have to ensure at least one subgame remains to play in

CGT-Aware MCTS Player

- Example of CGT zero-filtered board:



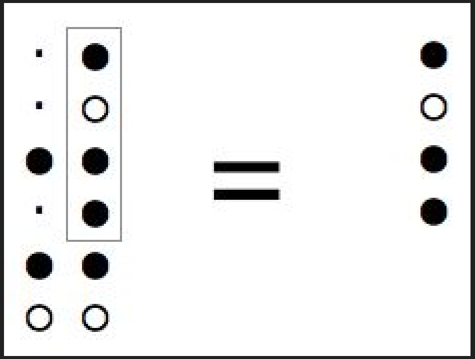
Results - CGT-Aware Player vs Baseline Player

- Half of the games are played with one player going first, and the other half has the other player going first
- Each player is allotted 30 seconds per move

Board size	**Win Rate (CGT)**
6 x 6	57/100 = 57%
8 x 8	73/100 = 73%
10 x 10	89/100 = 89%

Other Ideas Explored

- If an infinitesimal is identified and not removed, remove redundant stones while maintaining the subgame value
  - It was difficult to come up with rules for identifying redundant stones
  - More often than not, identified infinitesimals were already in or close to their minimum number of stones

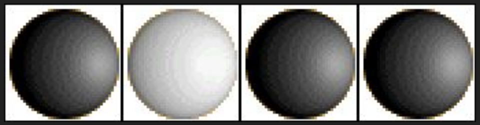


Other Ideas Explored

- MCTS solver, allowing nodes to be flagged as proven wins or losses when the tree reaches the end of the game
    - When the tree is able to reach the end of the game, a winner is typically already known from the statistics
    - In Clobber specifically, it did not appear worth the extra computation in the tree search

Ideas Moving Forward

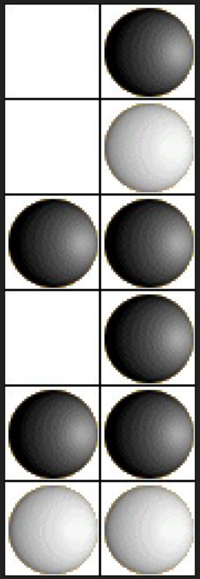
- Some evaluation function or heuristic for identifying “hot” subgames to play in
- While all temperatures in Clobber are zero, some games might be more favorable to play in than others
- Perhaps an approximation to incentives or an all-small scale temperature exists?



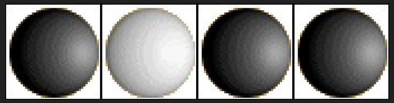
↓* - Parity Control for White

Ideas Moving Forward

- A clever way of selecting which infinitesimals to filter
- Currently has rules that aim to maximize the total number of subgames removed, ignoring the subgame sizes
- Would be better to choose subgames that minimizes the search space of the filtered board



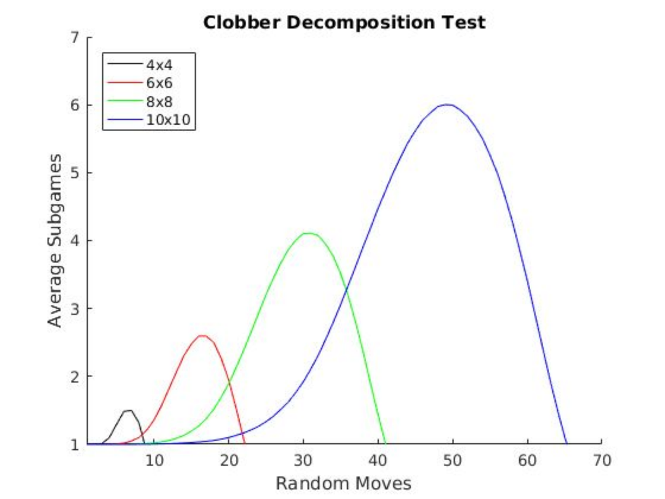
vs



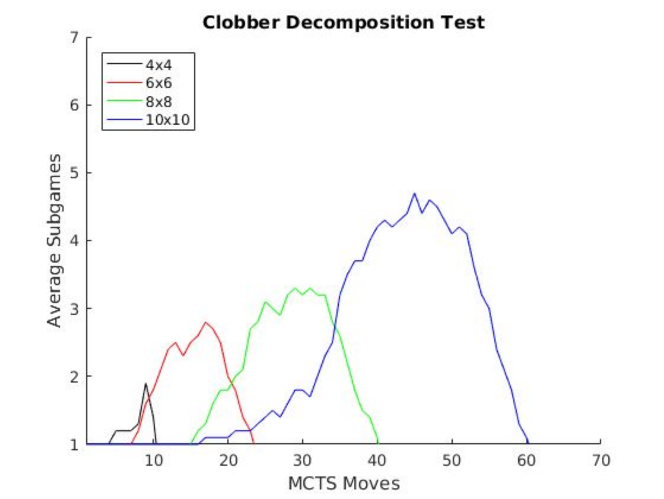
Thanks!

Decomposition Tests

- Played a bunch of games and computed the average number of subgames at each point in the game



Under Random Players



Under MCTS Players