Clobber Player using CGT

TryCatch: Abdul Wahab (wahab1), Muhammad Kamran Janjua (mjanjua), Saqib Ameen (saqib1)

Team: TryCatch

Outline

- Problem statement
- Literature Review
- Previous Work
- Enhancements
- Results
- Interesting Ideas
- Conclusion
- Future Work
- Acknowledgments

Problem statement

- Develop a strong 2D clobber player
- Integrate CGT enhancements to simplify the game
- Integrate CGT within MCTS to improve search and simulations
- Implement MCTS Solver

Literature Review

- Wieczorek et al. 2011 [1]
  - Four heuristic algorithms for playing the game
  - Based on the patterns of the stones
  - Little knowledge about what constitutes good play
  - Heuristics fail very often
- Griebel and Uiterwijk 2016 [2]
  - Used an end-game database of CGT values
  - Improve an NegaScout *solver* for Clobber
  - Little impact with regards to playing on Clobber positions that cannot be looked up in a database.

Literature Review (Cont.)

- Damhuis and Koster 2017 [3]
  - Application of a convolutional neural network to Clobber
  - Players using neural networks have yet to outperform a plain MCTS player in Clobber
- Cotarelo, A., García-Díaz, V., Núñez-Valdez 2021 [4]
  - Proposes a general and optimized implementation of MCTS with neural networks
  - The game of Dots-and-Boxes was used as the example

Previous Work: De Asis' Work (2017) [7]

- CGT-aware MCTS player for Clobber
- Subgames simplification:
  - Depth limited boolean minimax
  - Solved positions <= 16 positions
  - Removes 0 value games
- Uses CGT prior to calling MCTS (game simplification)
- Zobrist Hashing

Limitations of De Asis' Work [7]

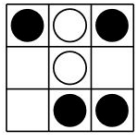
- Uses Alpha-Beta solver on the go to compute infinitesimals
- No endgame database used
- Does not use CGT enhancements within MCTS
- One-skip backpropagation in MCTS
- Mishandles edge cases (illegal moves)
    - When CGT simplification takes all the time allowed for move

Research Statement

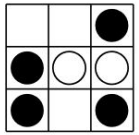
To build a strong CGT-aware Clobber player

Database

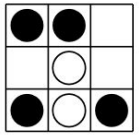
- Identify infinitesimals using sub-zero thermography (adopted from Fernando's work).
- Generated boards with at most 8 connected pieces.
- Based on most frequent infinitesimals.
- Exploit symmetries and inverses to cover more boards.
  - 90°, 180°, 270°, their symmetries and inverses.
- Total entries: 1,75,322 x (4+4+4)



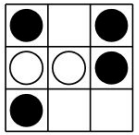
(a) Original position.



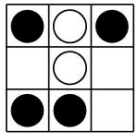
(b) 90° rotation.



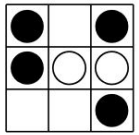
(c) 180° rotation.



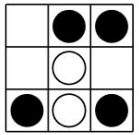
(d) 270° rotation.



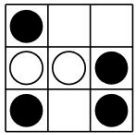
(e) Mirrored original position.



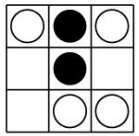
(f) Mirrored 90° rotation.



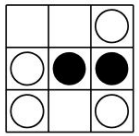
(g) Mirrored 180° rotation.



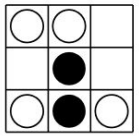
(h) Mirrored 270° rotation.



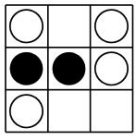
(i) Inverted original position.



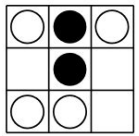
(j) Inverted 90° rotation.



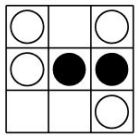
(k) Inverted 180° rotation.



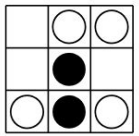
(l) Inverted 270° rotation.



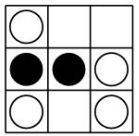
(m) Inverted and mirrored original position.



(n) Inverted and mirrored 90° rotation.



(o) Inverted and mirrored 180° rotation.

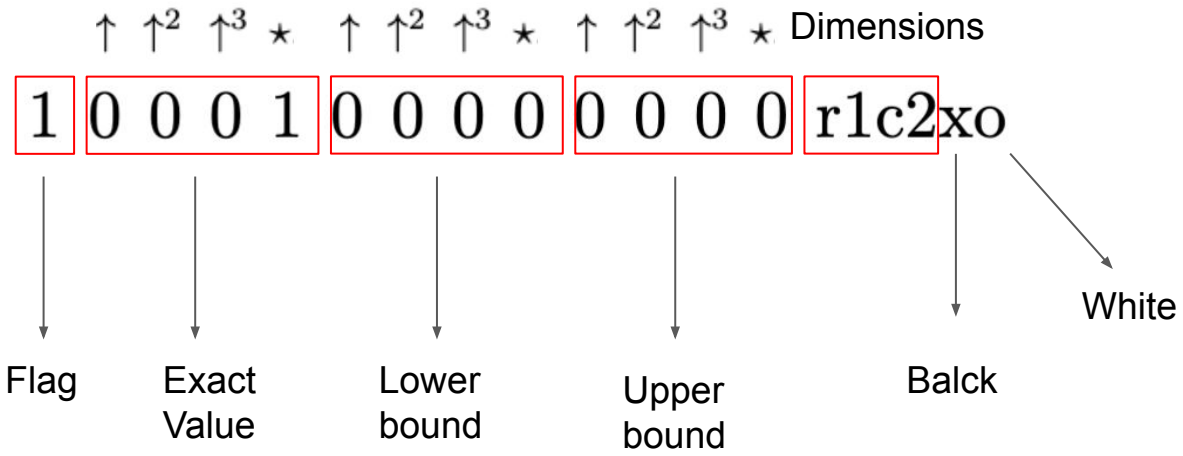


(p) Inverted and mirrored 270° rotation.

Picture Credits [6]

Database (Cont.)

- Sample database entry
- Values/bounds can be positive or negative to show inverse



Database (Cont.)

- Two scales used for comparison:

$$\uparrow\uparrow > \uparrow + \uparrow^2 > \uparrow + \uparrow^3 > \uparrow > \uparrow^2 > \uparrow^3 > 0 > \downarrow^3 > \downarrow^2 > \downarrow > \downarrow + \downarrow^3 > \downarrow + \downarrow^2 > \downarrow\downarrow$$

If * then move to second scale

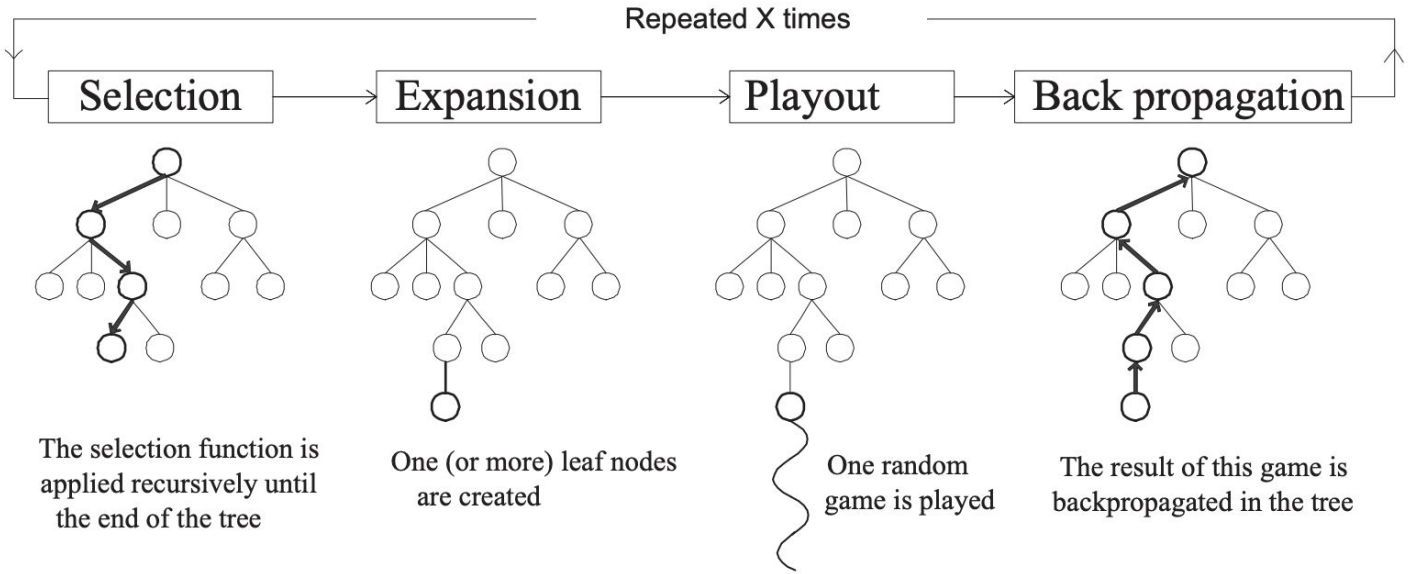
$$\uparrow\uparrow\star > \uparrow\star > \star > \downarrow\star > \downarrow\downarrow\star$$

- Starts comparison from first scale
- If fuzzy, moves to the second scale for better bounds/value estimates

Database (Cont.)

- Evaluation of games:
  - If the thermograph of $G + h$ is 0, then G is the inverse of h.
  - If the thermograph is positive, then G > h.
  - If the thermograph is negative, then G < h.
  - If the thermograph is fuzzy, then G is incomparable with h.

MCTS Player



Picture Credits [6]

Sub-games Simplification

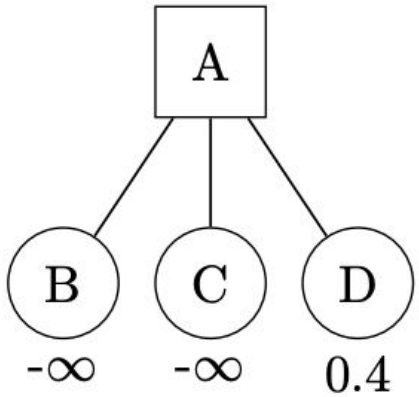
- Identify sub-game values using an end-game database.
- Remove the sub-games which add up to zero.
  - n* = * (if n = odd)
  - n* = 0 (if n = even)
  - G+H = 0
  - Other which add up to zero (e.g., ↑ + ↓= 0 and other multiples upto ±↑↑*)
- Replace the board with the equivalent board.
  - L + L + L = L (simple equivalent board; for the whole game)
- All done before the MCTS Step

MCTS-Solver

- Utilize proven loss/win information to improve MCTS
- Set flags corresponding to proven nodes
- Propagate infinities in MiniMax fashion
- If a node is a proven loss, it can be ignored
- If a node is a prove win, select that node
- Advantages:
  - Reduces the number of nodes in selection
  - Leads towards a better path in selection
  - A proven win node leads to a guaranteed win

MCTS-Solver (Cont.)

- Skip simulations for proven nodes
  - Propagate proven values instead of random simulations for value estimates
- Early termination of MCTS
  - If node is proven win and a child of the root node then return that action
- We use the CGT database to check if a node is a proven win or a loss
- We only query the database when:
  - There are multiple subgames
  - The maximum subgame size is present in the database
- Evaluation of board value returns:
  - 0 (Black to win)
  - 1 (White to win)
  - -1 (Fuzzy)



Picture Credits [6]

MCTS-Solver (Cont.)

- In standard MCTS, the final move selects the child with highest score v/n.
- SecureChild (SC) is a move selection criteria to account for proven nodes.
- Selection criteria:

$$v + (A)/(\sqrt{n})$$

- Improved selection even when used without solver.

Improved Backprop

- De Asis [7] implemented a one-skip backpropagation
- Backprop only affected the player whose turn it was to play
- Updated to do backprop in a minimax fashion
- Sequentially traverse through the tree alternating inversing the value
- Consistently performed better

Evaluation

- Comparison with:
  - CGT + MCTS → Kristopher’s work
- Win rate comparison
- Impact of different enhancements
- More effective on larger boards

Results (1): Modified BackProp (MBProp)

Board Size	MBProp Player (%)	De Asis' CGT Player (%)
6x6	73.91	26.08
8x8	93.24	6.75
10x10	85.36	14.63

Results (2): MBProp + SecureChild (SC)

Board Size	MBProp + SC Player (%)	De Asis' CGT Player (%)
6x6	80.61	19.38
8x8	90.0	10.0
10x10	92.0	8.0

Results (3): ZeroDB + MBProp

Board Size	ZeroDB + MBProp Player (%)	De Asis' CGT Player (%)
6x6	84.33	15.66
8x8	84.33	15.66
10x10	92.55	7.44

Results (4): ZeroDB (ZDB) + MBProp + SecureChild (SC)

Board Size	ZDB + MBProp + SC Player (%)	De Asis' CGT Player (%)
6x6	84.55	15.44
8x8	81.11	18.88
10x10	94.18	5.81

Results (5): SkipSimulations (SS) + MBProp + SecureChild (SC)

Board Size	SS + MBProp + SC Player (%)	De Asis' CGT Player (%)
6x6	73.91	26.08
8x8	81.25	18.75
10x10	90.69	9.30

Results (6): SkipSimulations (SS) + ZeroDB + MBProp + SecureChild (SC)

Board Size	SS + ZDB + MBProp + SC Player (%)	De Asis' CGT Player (%)
6x6	78.57	21.42
8x8	89.13	10.86
10x10	90.42	9.574

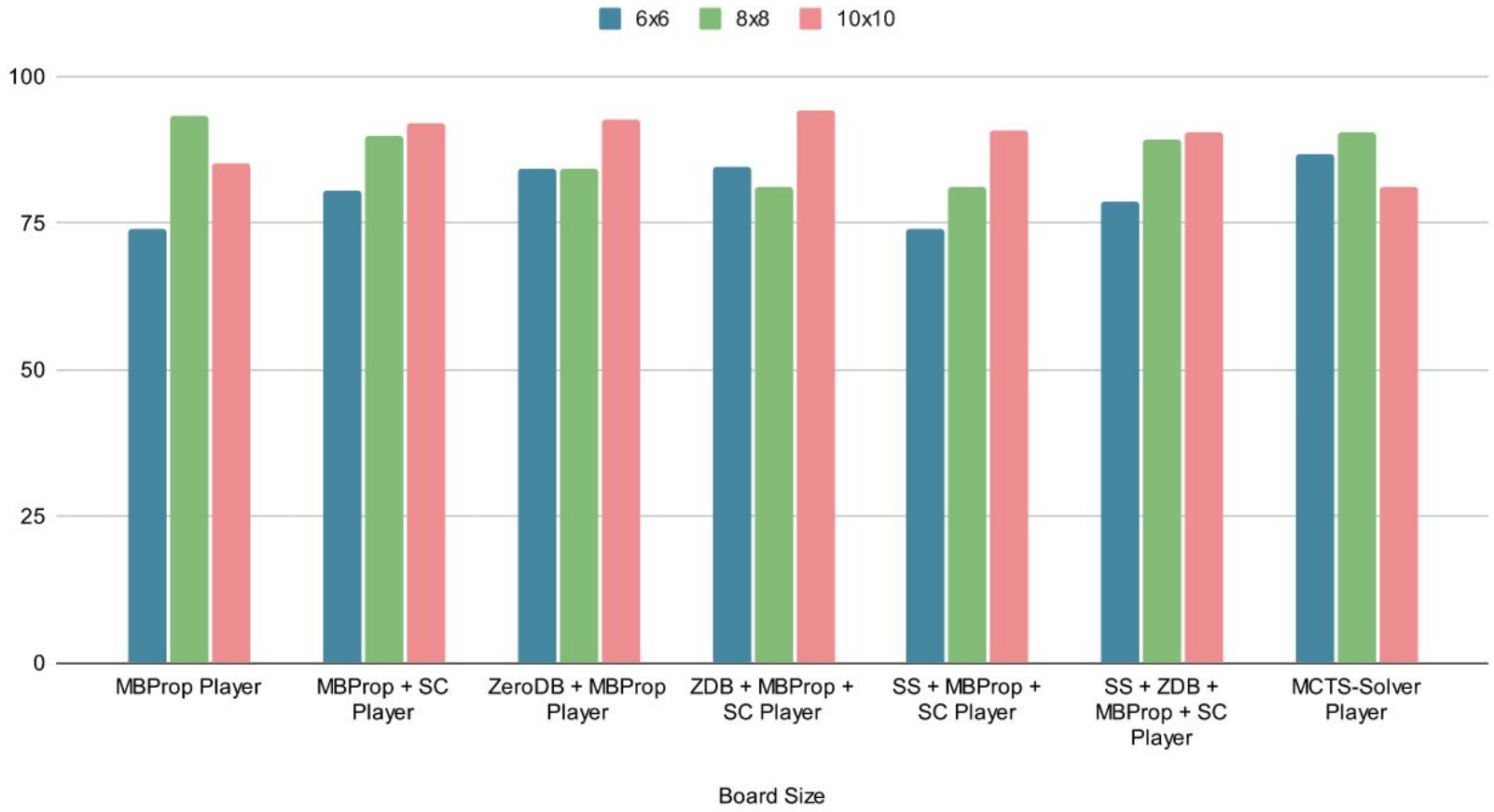
Results (7): MCTS-Solver Player

Board Size	MCTS-Solver Player (%)	De Asis' CGT Player (%)
6x6	86.66	13.33
8x8	90.47	9.52
10x10	81.08	18.91

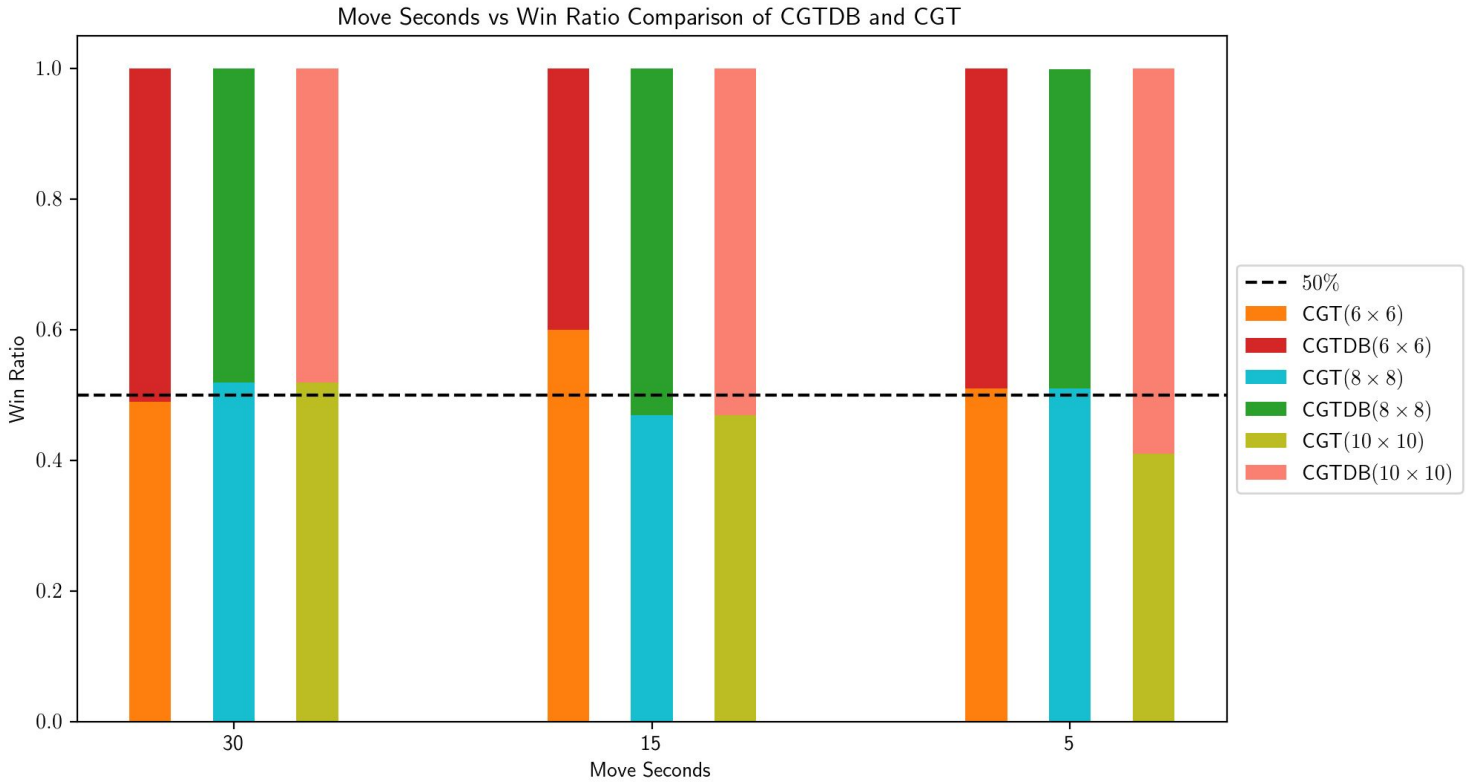
Results (8): MCTS-Solver Player

Board Size	MCTS-Solver Player (%)	MCTS Player (%)
6x6	88.00	12.00
8x8	93.00	7.00
10x10	91.00	9.00

Visual Representation of Proposed CGT Clobber Players



Case Study

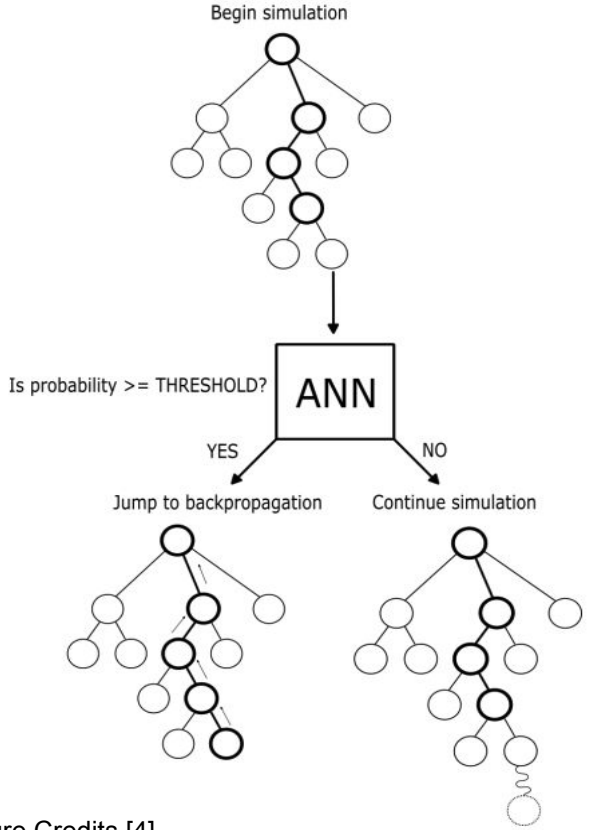

Move Seconds vs Win Ratio Comparison of CGTDB and CGT
Win Ratio
1.0
0.8
0.6
0.4
0.2
0.0
30
15
5
Move Seconds
50%
CGT(6 × 6)
CGTDB(6 × 6)
CGT(8 × 8)
CGTDB(8 × 8)
CGT(10 × 10)
CGTDB(10 × 10)

Case Study

Move Time →	**30s**	**15s**
Avg. Hit	34	29
Avg. Miss	16	16
Avg. Moves	33	33
Avg. Subgame Start	14	13
Median Hit	30	25
Median Miss	15	15
Median Moves	33	33
Median Subgame Start	14	14
Median DB Access Time	0.000173	0.00018
Ratio of Subgame Start	0.40	0.39

Interesting Ideas: Neural Network with MCTS

- Train a neural network to predict the winner
  - Given the board configuration, the move, board after the move is player, the player
  - Learns to map to the winner
  - Dataset generated for boards using AlphaBeta Solver
- Before the simulation step, query ANN over node to be simulated
- If winner is our player with probability > threshold (0.5)
  - Jump to backpropagation phase ELSE simulate node



Picture Credits [4]

Interesting Ideas: Neural Network with MCTS (Cont.)

- Trained a simple MLP model
- Dataset generated of board sizes:
  - 1x1, ..., 1x10
  - 2x1, ..., 2x10
  - 3x1, ..., 3x7
  - 4x4, 4x5
- As the board sizes increases, the time spent in the NN part increases
  - Pre-process string representation overhead
  - Query model and type conversions overhead
  - More losses comparatively
- No significant gain compared to standard CGT+MCTS player (on given board sizes)

Next Steps

- Improve MCTS-Solver and try different combinations of enhancements with it
- Perform analysis on:
  - Different move times
  - Extended case studies to better understand behavior

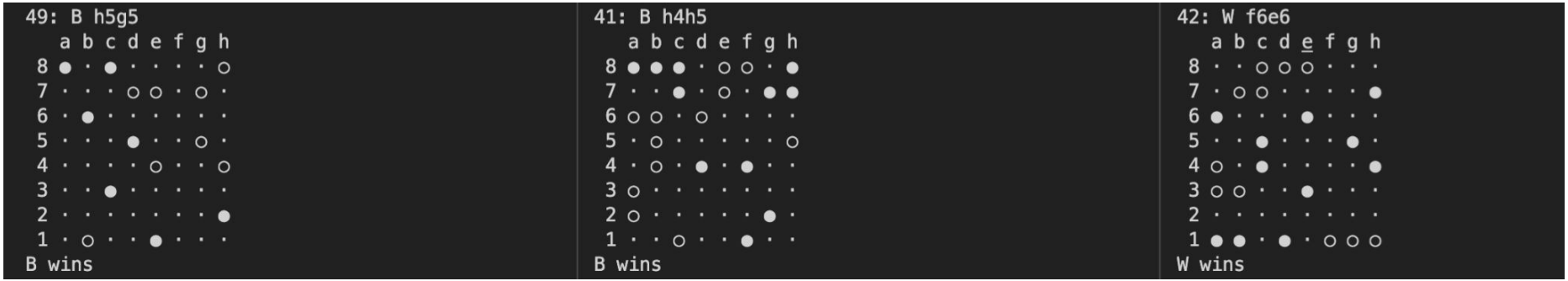
Conclusion

- Integrated CGT enhancements within MCTS
- Improved CGT-aware Clobber Player (Game Simplification)
- Implemented MCTS-Solver Player
- Conducted ablation studies of database
  - Query time vs. Hits/Misses
- Implemented ANNs + MCTS

Acknowledgement

- J. Fernando Hernandez and Kris De Asis for their previous work on building databases and CGT-MCTS player.
- Martin for helping us navigate through the project.

Demo!



Screenshot from class demo: CGT-MCTS-Solver (Ours) Black to play vs CGT (De Asis') White to play. 8x8 board with 2 seconds per move.

References

[1] Wieczorek, W., Skinderowicz, R., Kozak, J., & Juszczuk, P. (2011). New Trends in Clobber Programming. ICGA Journal, 34(3), 150-158.

[2] Griebel, J., & Uiterwijk, J. W. H. M. (2016). Combining Combinatorial Game Theory with an α-β Solver for Clobber. In BNAIC (pp. 48-55).

[3] Damhuis L., and Kosters W. A. (2017). Convolutional Neural Network for Clobber. Poster presented at: Leide Institute of Advanced Computer Science Posters Bachelorklas; 2017 Mar 1; Leiden, Netherlands.

[4] Cotarelo, A., García-Díaz, V., Núñez-Valdez, E. R., González García, C., Gómez, A., & Chun-Wei Lin, J. (2021). Improving monte carlo tree search with artificial neural networks without heuristics. Applied Sciences, 11(5), 2056.

[5] Chaslot, G. M. J., Winands, M. H., Herik, H. J. V. D., Uiterwijk, J. W., & Bouzy, B. (2008). Progressive strategies for Monte-Carlo tree search. New Mathematics and Natural Computation, 4(03), 343-357.

[6] Claessen, J. (2011). Combinatorial Game Theory In Clobber (dissertation).

[7] De Asis, K. (n.d.). (rep.). A CGT-Informed Clobber Player.

Thank You!

Questions?