Representation vs. Planning in Cops and Robbers

Dennis Kim | Advisors: Professor Eric Vigoda and Professor Ambuj Singh

Computer Science Department, University of California, Santa Barbara

COMPUTER SCIENCE
UC SANTA BARBARA

MOTIVATION

Effective decision-making in graph environments requires:

- a useful **state representation**
- a **planning procedure** for choosing actions

In Cops and Robbers, agents must reason about graph structure, legal moves, and long-term pursuit-evasion.

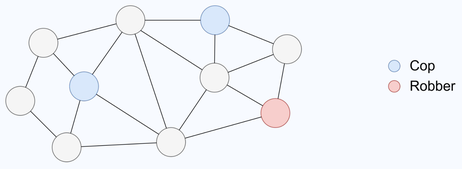
This project asks whether stronger **graph representations** reduce the need for **expensive planning**.

RESEARCH QUESTIONS

- How much of an agent's performance comes from its internal **representation of the game state**?
- How much comes from the **lookahead planning** used before choosing a move?
- How much **additional planning budget** is required to compensate for weaker representations?

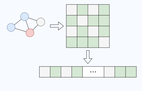
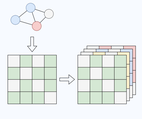
GAME SETUP

- Graph setup
  - Connected 3-regular graphs
  - 20 nodes
- Game setup
  - 2 cops and 1 robber
  - Randomized starting positions
  - Robber starts at least 3 edges away from each cop
- Win condition
  - Cops win by capturing the robber
  - Robber wins by evading until move 20



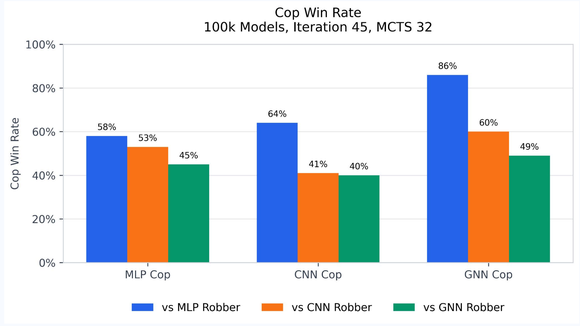
REPRESENTATION MODES

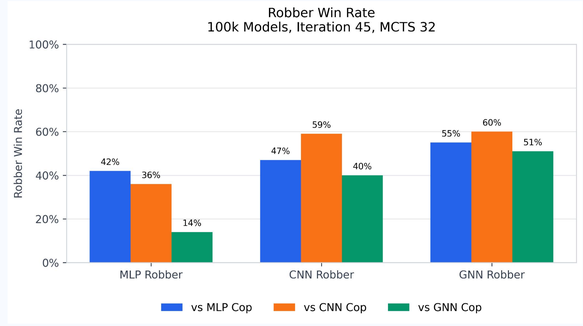
We compare three representation modes.
All models receive the same game state, but use different encodings.

MLP		• 1D representation of the graph • Flatten adjacency matrix into a vector
CNN		• 2D representation of the graph • Use adjacency matrix as image-like channels
GNN		• Direct representation of the graph • Keep the graph as nodes and edges

MLP: Multilayer Perceptron; CNN: Convolutional Neural Network; GNN: Graph Neural Network

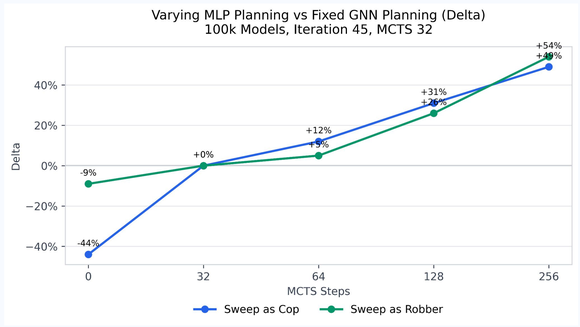
HEAD-TO-HEAD EVALUATION

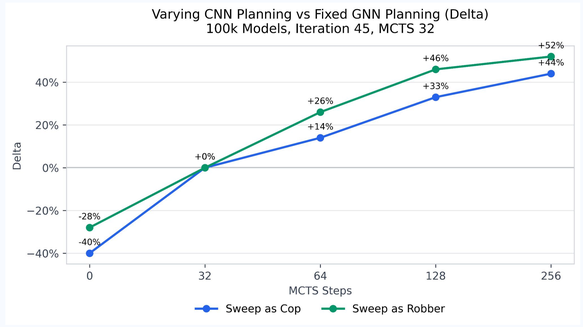




Cop-side results suggest a clear ordering trend: MLP robbers are easiest to capture, followed by CNN, then GNN.
This trend is weaker on the robber side, but robbers appear to perform better against MLP cops than against GNN cops.

PLANNING BUDGET EVALUATION





This sweep directly tests whether extra planning budget can compensate for non-GNN representations.
MLP and CNN reach similar final gains, but CNN improves more at intermediate budgets, suggesting representation still affects how efficiently planning is used.

OBSERVED ISSUES

- Early training showed unstable and degenerate strategies
- Agents learned to stand still instead of pursuing or evading
- One side sometimes dominated, reaching very high win rates
- Robber policies often made clearly poor decisions
- The losing side failed to learn effective counter-strategies

IMPLEMENTED FIXES

- Cop win value shaped by capture speed
- Lower temperature during self-play
- 50/50 replay buffer by cop-win and robber-win outcomes
- Previous-checkpoint self-play to prevent strategy forgetting
- Disallow both cops from staying still

FINDINGS & CONCLUSION

WHAT WE LEARNED

- Different encodings learned different levels of strategy, showing that representation matters
- Current results follow the trend MLP < CNN < GNN
- Graph-aware representations appear to help models learn more competitive strategies
- MCTS planning improves performance, and the gain also depends on the representation

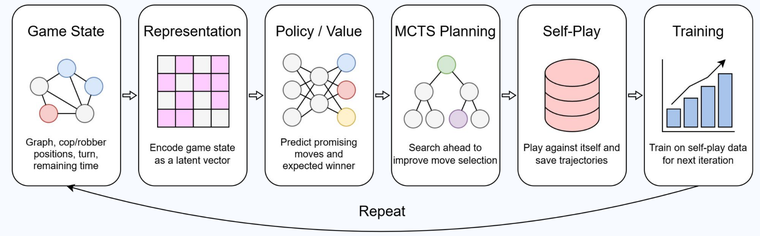
LIMITATIONS

- Self-play RL is highly unstable
- Preventing collapse into degenerate strategies remains difficult
- Current models are small
- Stronger results likely require larger models and more compute
- Results are based on a limited evaluation setting and need more comprehensive validation

KEY TAKEAWAY

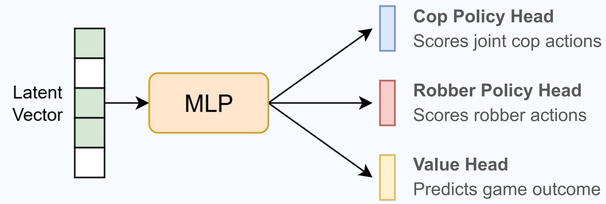
- **Graph-aware representations** learned stronger strategies, following **MLP < CNN < GNN**
- **MCTS planning** improves performance, and **stronger representations** tend to **benefit more** from additional planning
- Results come from a **limited evaluation setting**; longer and more robust experiments are needed

PIPELINE OVERVIEW



POLICY / VALUE NETWORK

Each representation outputs a latent vector, which is passed into the same policy/value MLP.



MONTE CARLO TREE SEARCH

Monte Carlo Tree Search (MCTS) gives the model extra planning before selecting a move.

MCTS balances:

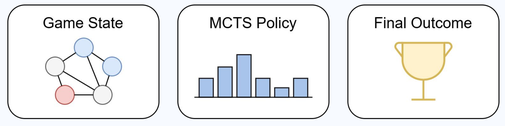
- **Exploitation** of actions with high estimated value
- **Exploration** of actions with high potential but fewer visits

The search is repeated for a chosen planning budget.

After search, the MCTS policy is given by the action **visit-count distribution**.

SELF-PLAY & TRAINING

At each iteration, the current model generates new games through **self-play**.
For **each move** in every game, we save:



These self-play trajectories are used to update the model for the next iteration.

Training includes three loss terms:

- **Value loss**: align predicted outcome with final game result
- **Policy loss**: align predicted policy with the MCTS policy
- **Weight decay**: regularize model parameters to prevent overly large weights

REFERENCES

- Silver et al. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815.
- Lees & Matiisen (2026). Reproducing AlphaZero on Tablut: Self-Play RL for an Asymmetric Board Game. arXiv:2604.05476.

ACKNOWLEDGMENTS

Thank you to Professor Eric Vigoda and Professor Ambuj Singh for their guidance and support throughout this project.