Heuristic-Based Multi-Agent Mind-Boggle UNO: A Strategic Card Game Simulation

Ali Jafar

Course: AI
Instructor: Sir Khalid
Submission Date: May 11, 2025

1. Executive Summary

Project Overview: This project presents a simulation of a multi-agent card game that merges traditional UNO mechanics with elements from the chaotic and unpredictable game *Mind-Boggle*. The goal was to explore AI decision-making in dynamic environments by introducing rule-modifying cards and memory-based logic. A heuristic-based AI approach enhanced with Bayesian inference allows agents to navigate and strategize in this game.

2. Introduction

Background: UNO is a deterministic card game with fixed rules and player actions. *Mind-Boggle*, on the other hand, introduces evolving rules and memory-based challenges. By combining both, this project aimed to create a novel game format where each card can affect not only gameplay but also an agent's decision-making process.

Objectives of the Project:

- Develop a simulation environment for the modified Mind-Boggle UNO game.
- Implement AI agents with limited and decaying memory.
- Incorporate heuristic and Bayesian decision-making strategies.
- Evaluate performance under chaotic and unpredictable game scenarios.

3. Game Description

Original Game Rules: UNO is a card game where players take turns matching a card in their hand with the current card shown on top of the deck by either color or number. Special action cards add game dynamics.

Innovations and Modifications:

- Wild Card resets and reviews the agent's hand.
- 9 Card allows viewing and shuffling opponent's cards, introducing memory decay.
- 7 Card swaps a card with any opponent.
- Memory and decision state of AI are affected dynamically by these cards.

4. AI Approach and Methodology

AI Techniques Used:

- Heuristic Evaluation
- Bayesian Inference using pgmpy library
- Memory confidence decay logic

Algorithm and Heuristic Design: Each AI agent calculates a weighted score for each card based on:

- Card Weight
- Memory reliability (based on `shuffled` tracker)

AI Performance Evaluation: AI agents were evaluated by:

- Their win/loss rates across simulations (they shared an equal chance to win and sometimes excel the human player)
- Ability to act rationally under decaying memory (the probability of remembering your hand correctly decreases overtime depicting real life scenario)
- Confidence scores and strategic card usage over time

5. Game Mechanics and Rules

Modified Game Rules:

- Each special card affects game state or memory.
- Agents track whether their hand is shuffled after last seen (`shuffled` metric).

- Bayesian models update beliefs about opponents' hands based on observed actions.

Turn-based Mechanics: Each turn consists of drawing or playing a card, followed by memory updates and inference.

Winning Conditions: A player wins when they have no cards left. The game ends when the draw deck is exhausted.

6. Implementation and Development

Development Process: The game and AI were implemented in Python, using core libraries and `pgmpy` for Bayesian logic. Game logic supports real-time AI decision making and memory management.

Programming Languages and Tools:

- **Programming Language:** Python 3.11
- **Libraries:** random, typing, pgmpy.models : DiscreteBayesianNetwork, pgmpy.inference : VariableElimination, pgmpy.factors.discrete : TabularCPD
- **Tools:** Internal debug logging, VSCode, GitHub for version control

Challenges Encountered:

- Balancing memory decay with meaningful decision-making
- Modeling uncertainty with Bayesian inference
- Designing strategic AI without full observability

7. Team Contributions

Ali Jafar: Sole developer responsible for:

- AI algorithm development (Heuristics and Bayesian logic)
- Rule modifications and game simulation
- Debug logging and performance evaluation

8. Results and Discussion

AI Performance: The AI agents demonstrated adaptive strategy by leveraging memory decay and probabilistic beliefs. The shuffled metric improved decision quality under uncertainty.

Figures:

```
Player - 1 Cards: ['B9', 'G1', 'R8']
Player - 2 Cards: ['GSkip', 'B1', 'B8']
Player - 3 Cards: ['G2', 'G9', 'G8']
Player - 4 Cards: ['Y+2', 'G+2', 'R+2']
```

Figure 1: AI and Player hands during simulation.

```
Player - 1 Cards: ['B1', 'B0', 'R1']
Player - 2 Cards: ['B7', 'Y0', 'Y5']
Player - 3 Cards: ['G9', 'Y3', 'Y2']
Player - 4 Cards: ['G0', 'R0', 'B6']
Card Drawn: Y6 & Turn Off - 3
Number of Turns played by Now: 0
Hueristic Index: 0
Memory: ['G9', 'B0', 'Y2']
Card: G9
Bot Called 9
Before: ['B7', 'Y0', 'Y5']
After: ['Y0', 'B7', 'Y5']
```

Figure 2: AI shuffling another player's cards with powerup-9.

```
Player - 1 Cards: ['B1', 'B0', 'R1']
Player - 2 Cards: ['Y0', 'Y6', 'Y5']
Player - 3 Cards: ['B7', 'Y3', 'Y2']
Player - 4 Cards: ['G0', 'R0', 'B6']
Card Drawn: B5 & Turn Off - 3
Number of Turns played by Now: 1
Hueristic Index: 0
Memory: ['Y3', 'B7', 'Y2']
Card: B7
me - ['B5', 'Y3', 'Y2']
Player - ['B1', 'B0', 'R1']
bot1 - ['Y0', 'Y6', 'Y5']
bot2 - ['G0', 'R0', 'B6']
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
Bot Called 7 with - 1
me - ['Y6', 'Y3', 'Y2']
Player - ['B1', 'B0', 'R1']
bot1 - ['Y0', 'B5', 'Y5']
bot2 - ['G0', 'R0', 'B6']
```

Figure 3: AI swapping cards using powerup-7.

```
Player - 1 Cards: ['Y0', 'B0', 'R1']
Player - 2 Cards: ['B1', 'B2', 'R2']
Player - 3 Cards: ['Y2', 'G1', 'Y1']
Player - 4 Cards: ['G0', 'R0', 'G2']
Card Drawn: R3 & Turn Off - 4
[('Player', 1), ('AI - 3', 2), ('AI - 2', 4), ('AI - 1', 5)]
Player Won the game with 1 Weight Values
```

Figure 4: Winning state when draw deck is exhausted.

9. References

- Python Documentation

Appendix

- **Code Base:** Python 3.11
- **Libraries:** random, typing, pgmpy

- **Agent Memory Logic:** Turn-based probabilistic recall decay with Bayesian inference for belief updating under uncertainty
- **Debug Features:** Logging memory trust scores and inference outcomes