

# AppleGameEnv:

Mastering Massive Action Space in a Non-Tabular RL Environment

- Introduction
- AppleGame
- Applying RL to AppleGameEnv
- Employing Natural Language
- Limitations & Future Works

## ■ Introduction

## ■ AppleGame

## ■ Applying RL to AppleGameEnv

## ■ Employing Natural Language

## ■ Limitations & Future Works

## Motivation

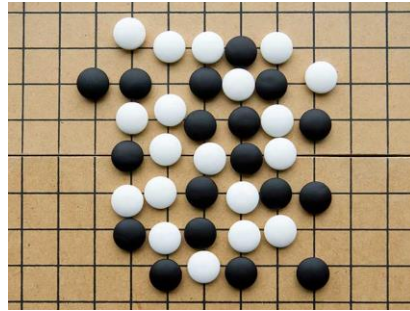
RL has been successful in solving **large state-space environments**,

But in reality...



## Motivation

Previous RL methods have conquered grid games.



Go



Atari

But, in common, they have **low action dimensionality**.

We wondered:

**“Would these algorithms work for grid games with a non-tabular action space?”**

## Motivation

---

And we didn't have to go far to find the perfect environment..!

A game which is

1. consistently loved by students (1M+ downloads)
2. has a high (4D) action space and state space
3. initialized randomly
4. not solved (no optimal solution)
5. yet has simple, straightforward dynamics

—————→ **AppleGame**

- Introduction
- AppleGame
- Applying RL to AppleGameEnv
- Employing Natural Language
- Limitations & Future Works



# AppleGame



- Top Puzzle Game on App Store & Google Play
- Non-tabular State/action spaces :  $10^{170}$  //  $170 \times 170$
- Randomly initialized
- Unknown optimal solution

A novel benchmark is born:  
AppleGameEnv



## Rules

1. Select a **rectangular area** on the grid.
2. If the sum of all apples within the selected area **equals 10**, all the apples in the rectangle disappear.

Your score is then increased by the **number of apples cleared**.

3. Repeat steps 1 and 2 for 120 seconds.

```
AppleGameRL2 — python3.9 AppleGame.py play — python3.9 — Python A...
python3.9 AppleGame.py play
Starting Apple Game - Human Play Mode
Instructions:
- Click and drag to select rectangles
- Rectangle sum must equal 10 to be valid
- Clear all apples to win!
- Press ESC to quit
2025-06-06 00:38:35.408 Python[18343:95755078] +[IMKClient subclass]: chose IMKClient_Modern
2025-06-06 00:38:35.409 Python[18343:95755078] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Invalid selection. Rectangle sum: 8 (need 10)
Valid selection! Cleared 2 apples. Score: 2
Valid selection! Cleared 2 apples. Score: 4
Valid selection! Cleared 2 apples. Score: 6
Valid selection! Cleared 2 apples. Score: 8
2025-06-06 00:38:43.930 Python[18343:95755078] TSM AdjustCapsLockLEDForKeyTransitionHandling - _ISetPhysicalKeyboardCapsLockLED Inhibit
2025-06-06 00:38:43.937 Python[18343:95755078] error messaging the mach port for IMKCFRunLoopWakeUpReliable
Game ended. Thanks for playing!
thomaskim@Book-Air-2: ~/AppleGame
python3.9 AppleGame.py play
```

## AppleGame: Difficulty

“The sequence of actions impacts the future”



Initial state



Bad move: +2



Good move: +4

## Notations for AppleGame

---

### State Space

$$S = \{(x, y, n) : 0 \leq x \leq 9, 0 \leq y \leq 16, 0 \leq n \leq 9\}$$



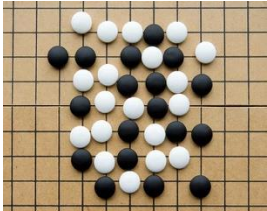
### Action Space

$$A = \{(x, y, w, h) : x \geq 0, y \geq 0, 0 < x + w \leq 9, 0 < y + h \leq 16\}$$

### Grid Size

$$(\text{height}) * (\text{width}) = 10 * 17$$

## AppleGame: Comparison

Games	State Space	Action Space
	$10^{170}$	$170 \times 170$
	$2.8 \times 10^4$	18
	$4.7 \times 10^{108}$	128

- Introduction
- AppleGame
- Applying RL to AppleGameEnv
- Employing Natural Language
- Limitations & Future Works

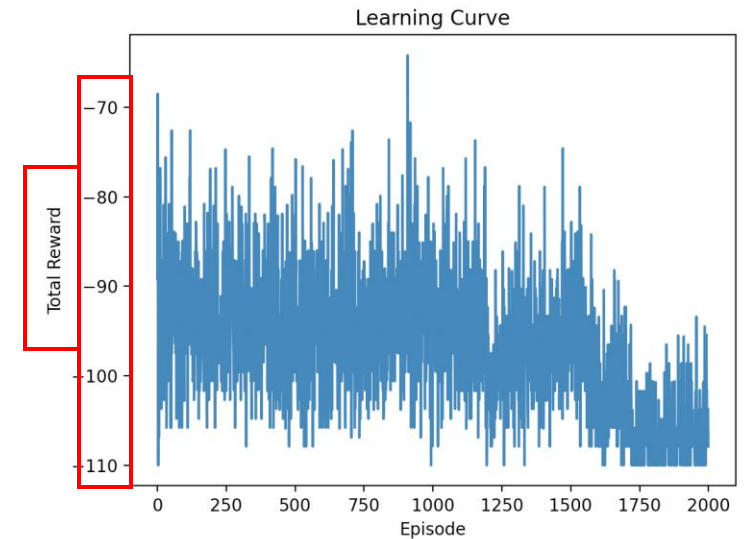
## Problem 1: Invalid Actions

The screenshot shows a code editor with a file named `QACAgent.py`. The code includes imports for `pygame`, `random`, and `sys`, and defines a `QACAgent` class. The `train` method is implemented, and the script is run using `python3.9 QACAgent.py`. A terminal window in the foreground shows the following output:

```

thomaskim@Book-Air-2: ~$ python3.9 QACAgent.py
pygame 2.1.2 (SDL 2.0.18, Python 3.9.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
Starting training...
Starting episode 1
[LW NNPack.cpp:64] Could not initialize NNPack! Reason: Unsupported hardware.

```

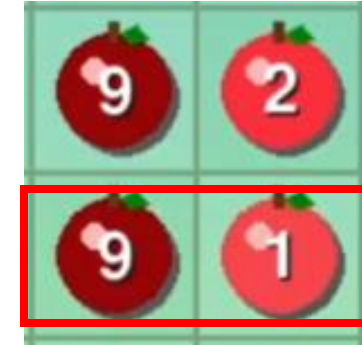


Leads to bad results  
(on average, 20/170)  
and longer runs

Most actions taken are invalid (not sum to 10)



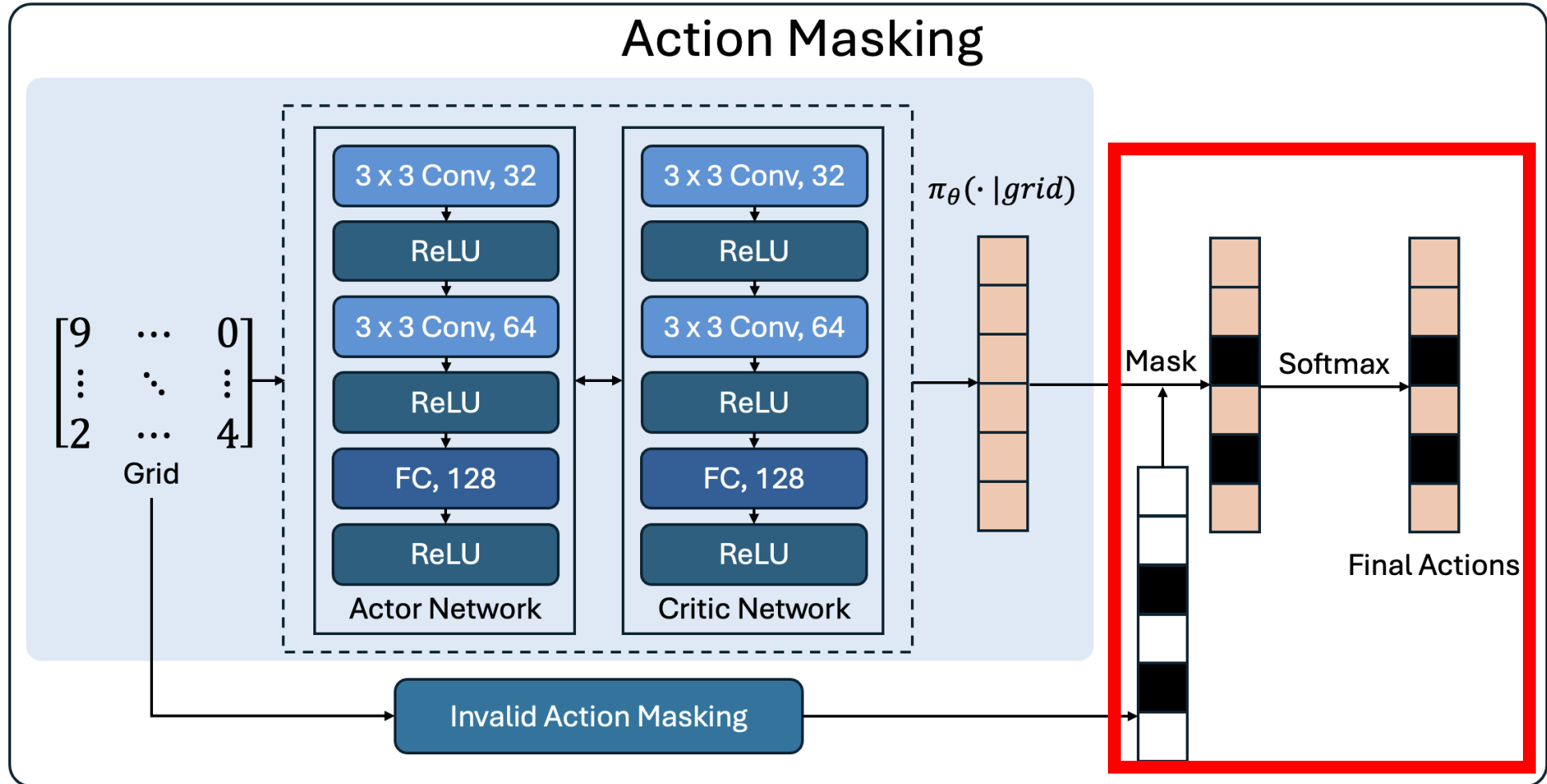
## Solution 1: Action Masking



(0,0,1,0)  
**Invalid** (0,0,0,1)  
(0,1,1,0)  
**Valid** (1,0,1,1)  
All **possible** actions

(1,0,1,1)  
only **valid** actions

## Solution 1: Action Masking



## Solution 1: Action Masking

### A Closer Look at Invalid Action Masking in Policy Gradient Algorithms

Shengyi Huang and Santiago Ontañón \*

College of Computing & Informatics, Drexel University

Philadelphia, PA 19104

{sh3397, so367}@drexel.edu

#### Abstract

In recent years, Deep Reinforcement Learning (DRL) algorithms have achieved state-of-the-art performance in many challenging strategy games. Because these games have complicated rules, an action sampled from the full discrete action distribution predicted by the learned policy is likely to be invalid according to the game rules (e.g., walking into a wall). The usual approach to deal with this problem in policy gradient algorithms is to “mask out” invalid actions and just sample from the set of valid actions. The implications of this process, however, remain under-investigated. In this paper, we 1) show theoretical justification for such a practice, 2) empirically demonstrate its importance as the space of invalid actions grows, and 3) provide further insights by evaluating different action masking regimes, such as removing masking after an agent has been trained using masking.

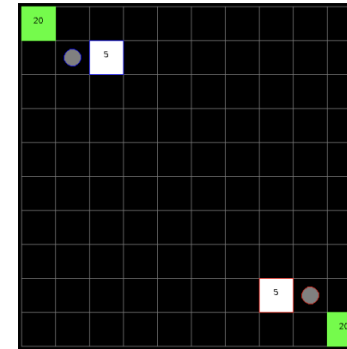


Figure 1: A screenshot of  $\mu$ RTS. Square units are “bases” (light grey, that can produce workers), “barracks” (dark grey, that can produce military units), and “resources mines” (green, from where workers can extract resources to produce).

The gradient produced by invalid action masking is a **valid policy gradient**, and it **works** by applying a state-dependent differentiable function

The screenshot displays a workspace for a CS377 Team Project. At the top, a navigation bar includes links to Naver Mail, Instagram, Colab, Dooray!, KAIST 포탈, 학성 메인, 학사시스템, KLMs, CSTL - Google Drive, and 산시공 학생회. Below this, a toolbar features buttons for 'upgrade', 'Code Editor', 'Visual Editor', 'Normal text', 'B', 'Recompile', and 'History'. The main area is divided into three panels:

- Code Editor:** Displays a Python script for 'RandomAgent.py'. The script includes comments and code for a game environment, such as `pygame 2.1.2 (SDL 2.0.18, Python 3.9.5)` and `Hello from the pygame community. https://www.pygame.org/contribute.html`. It also shows a `pygame` error message: `pygame 2.1.2 (SDL 2.0.18, Python 3.9.5) Hello from the pygame community. https://www.pygame.org/contribute.html`.
- Terminal:** Shows the output of the script, including the error message: `pygame 2.1.2 (SDL 2.0.18, Python 3.9.5) Hello from the pygame community. https://www.pygame.org/contribute.html`.
- Slide:** A presentation slide titled 'AppleGameEnv: Mastering Non-Tabular' by Sanggim Kim, Department of Computer Science, KAIST. The slide includes a bio for Sanggim Kim and a section titled '1 Introduction'.

18

## Experiments

We randomly generate 100 'golden grids' for evaluation



**Novice**  
**Sangoh Kim**

(<1 Month)



**Casual**  
**Yuseop Lee**

(~1 year)



**Expert**  
**Sejong Kim**

**Over 3 years**  
**Plays ~50 times a week**

## Learning Curves (Loss | Total Score)



The screenshot displays the CS377 Team Project environment. The top navigation bar includes icons for navigation, a search bar, and buttons for 'Review', 'Share', 'Submit', and 'History'. Below this is a toolbar with 'Code Editor', 'Visual Editor', 'Normal text', and 'Recompile' options. The main area shows a code editor with a Python script for 'AppleGameEnv'. The code defines a class with methods for setting up the environment, including loading models and initializing NNPACK. The code is highlighted in a dark theme. The sidebar on the left shows a file explorer with a tree view of the project files. The bottom of the screen shows a terminal window with the command 'python3 ./apple\_game\_env.py' and its output.

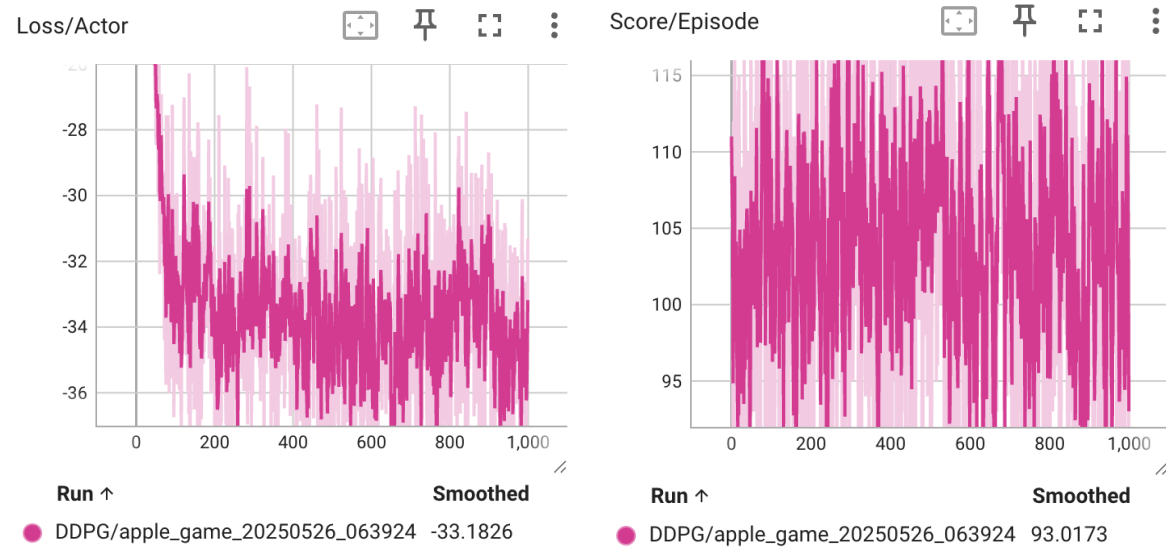
## DuelingDQN



# Experiments (2): Policy-Based Algorithm

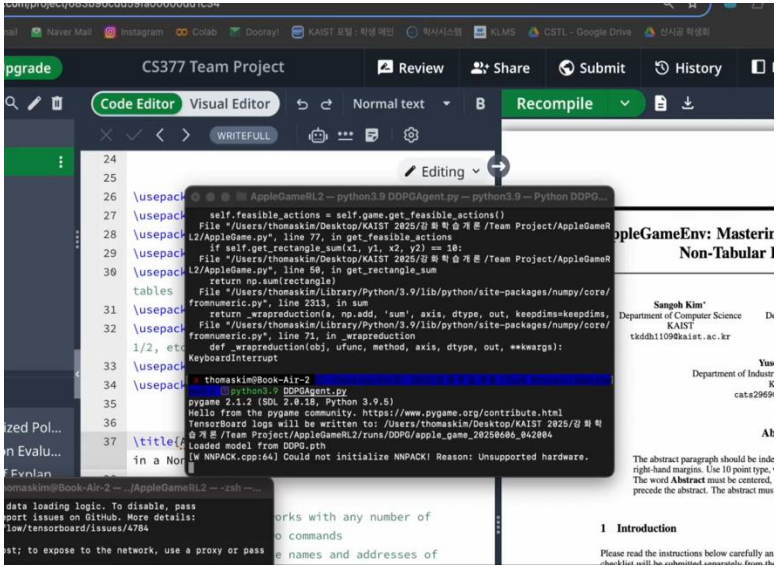
Algorithm	Average Total Score
REINFORCE	106.2
DDPG	105.6

## Learning Curves (Loss | Total Score)



DDPG

## Simulation Video

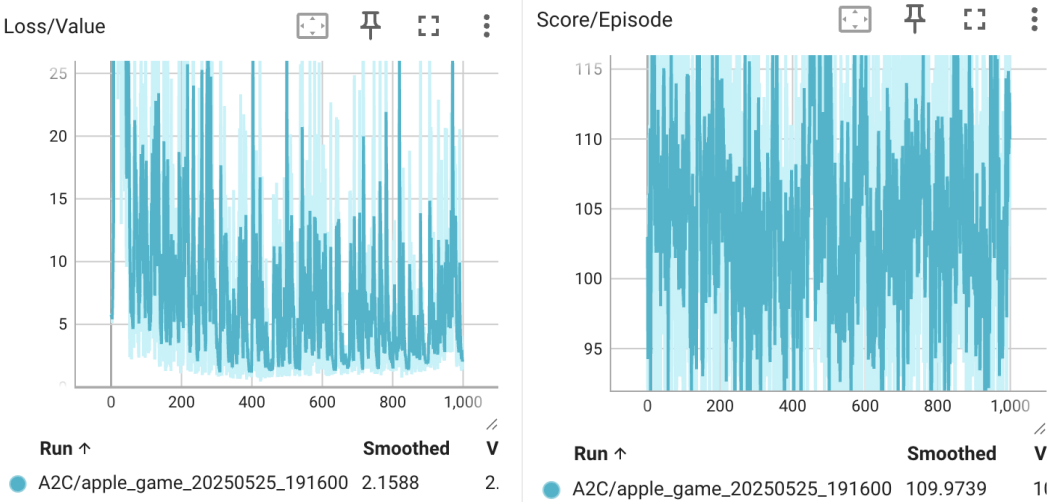


DDPG

# Experiments (3): Actor-Critic Algorithm

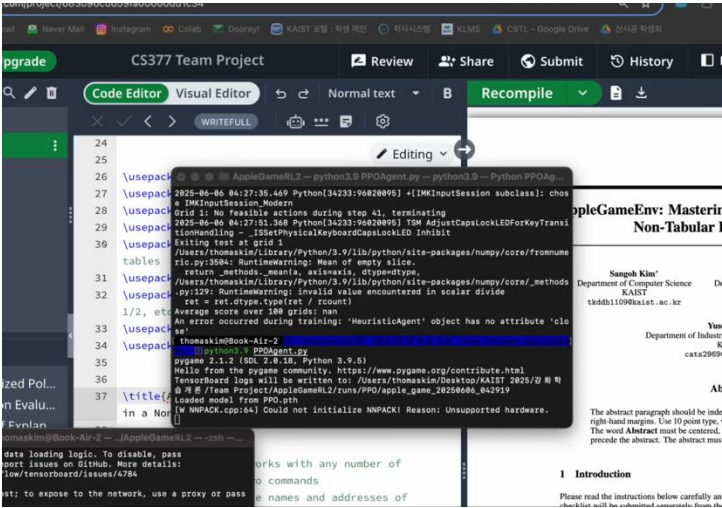
Algorithm	Average Total Score
A2C	105.3
QAC	103.2
PPO	106.2

## Learning Curves (Loss | Total Score)



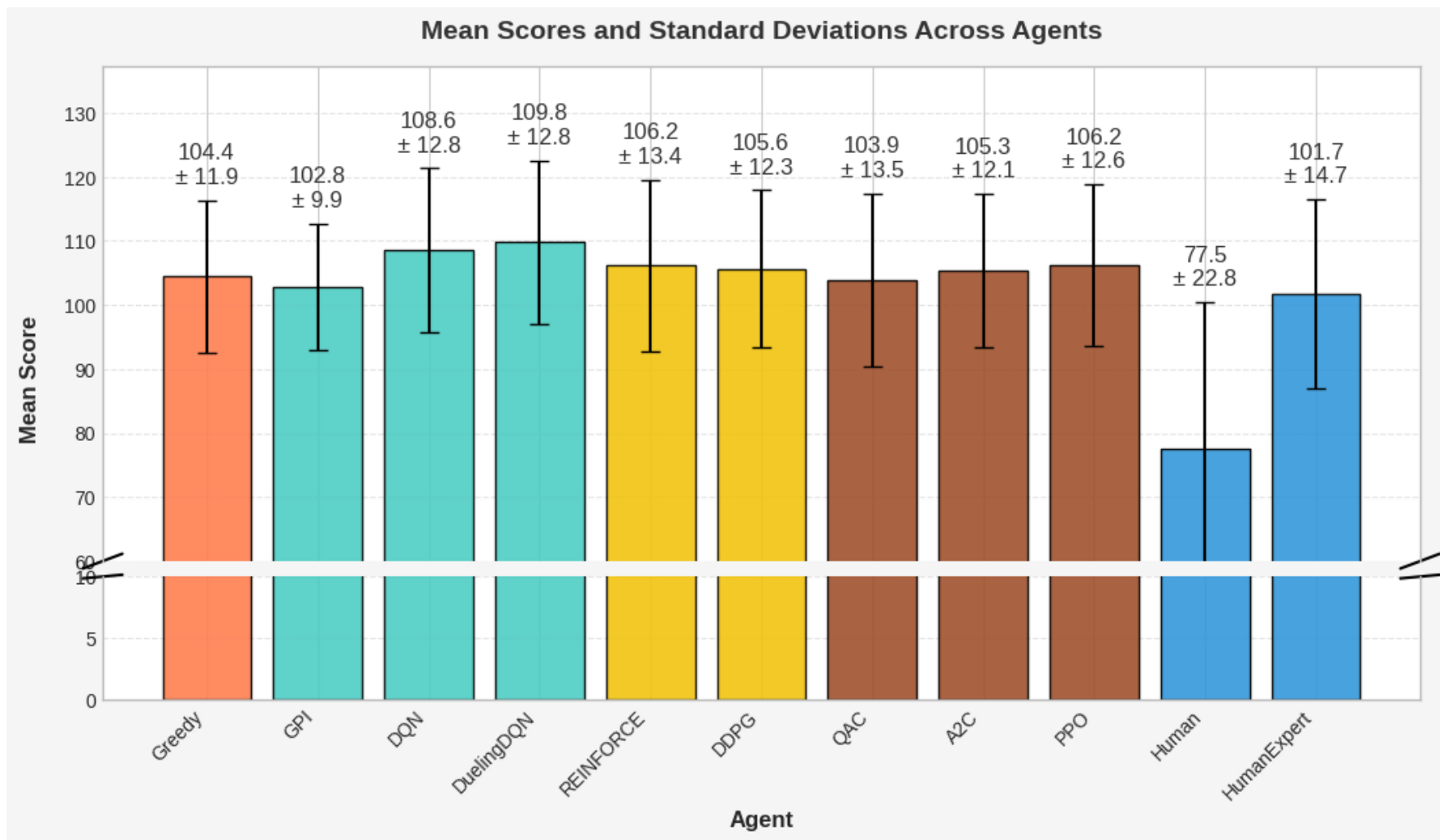
A2C

## Simulation Video

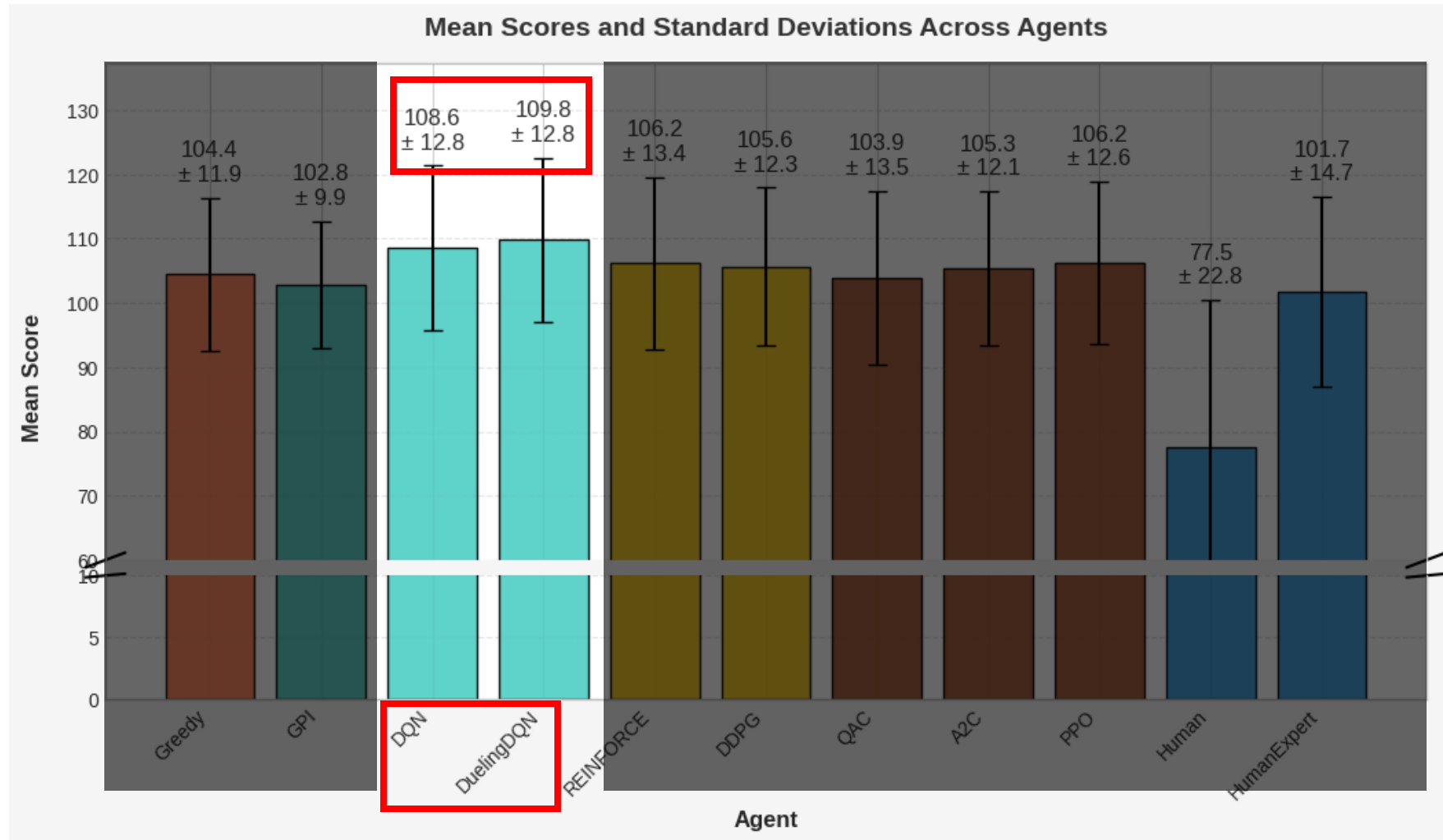


PPO

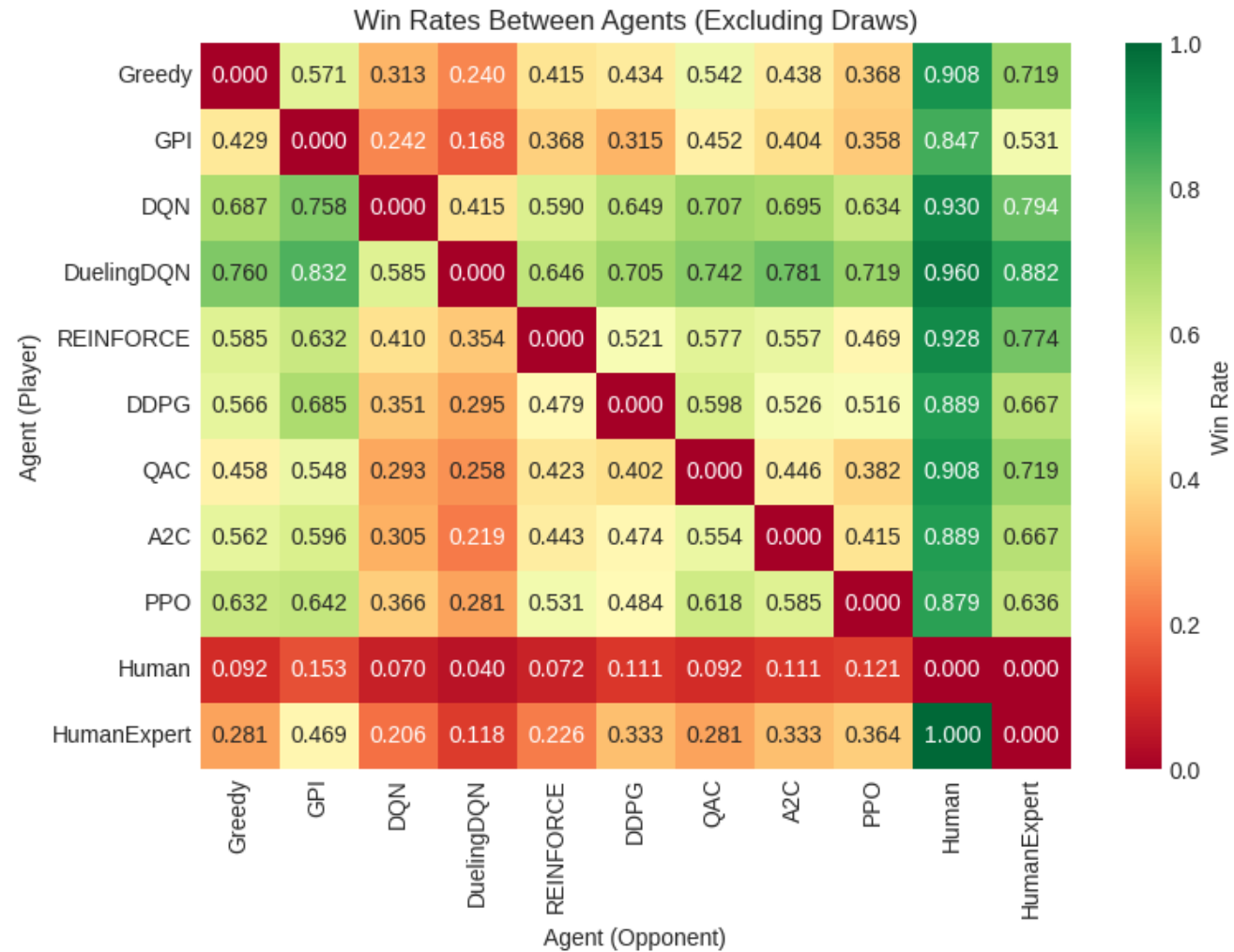
## Quantitative Results: Mean Score



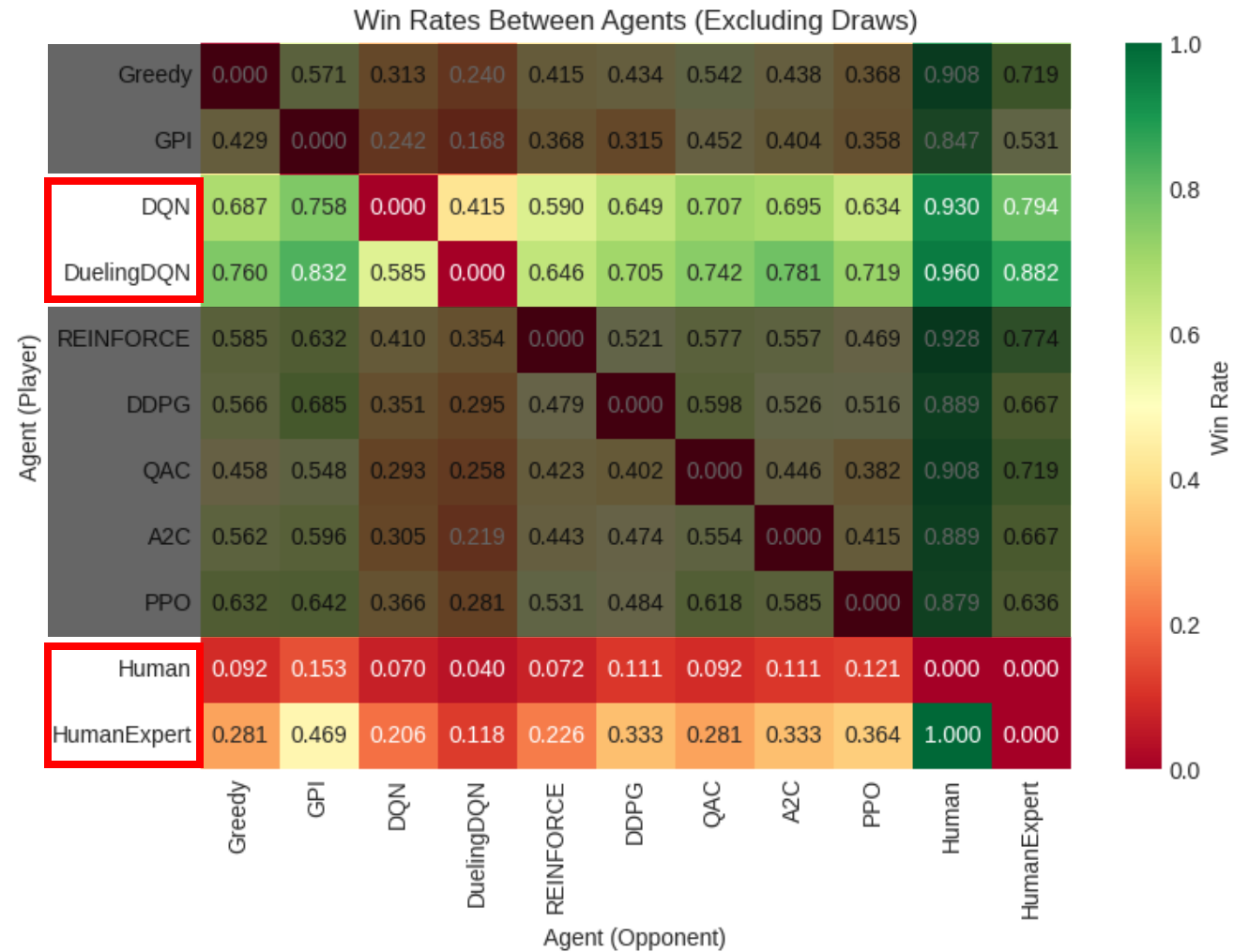
## Quantitative Results: Mean Score



## Quantitative Results: Win Rates



## Quantitative Results: Win Rates





## Problem 2: Interpreting Agents

```
113  
114 Please pay special attention to the ins.  
Section  
115 regarding  
reference  
116  
117  
118 \section  
119 \label{  
120  
121  
122 All head  
word and  
123 flush le  
124  
125  
126 First-le  
  
thomaskim@Book-Air-2 ~/AppleGameRL2 -- zsh --  
data loading logic. To disable, pass  
report issues on GitHub. More details:  
flow/tensorboard/issues/4784  
st; to expose to the network, use a proxy or pass  
  
Environment: AppleGameEnv}
```

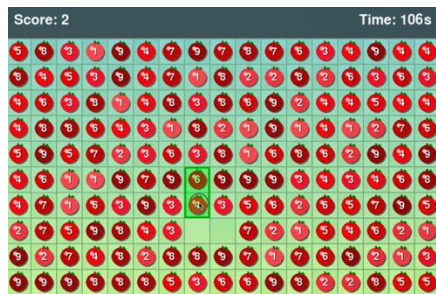
We can state that:

**“Tendency to remove neighboring apples from left to right.”**

**But, do we truly understand their black box models?**

- Introduction
- AppleGame
- Applying RL to AppleGameEnv
- Employing Natural Language
- Limitations & Future Works

## Solution 2: Language GPI(LGPI)



Environment



```
grid = np.array([
    [4, 5, 2, 6, 6, 1, 8, 7, 7, 3, 2, 9, 4, 7, 3, 1, 8],
    [4, 1, 8, 7, 1, 5, 8, 1, 4, 3, 4, 4, 4, 8, 8, 1, 1],
    [0, 7, 5, 6, 5, 1, 8, 8, 3, 3, 7, 4, 3, 1, 3, 4, 9],
    [0, 4, 8, 3, 7, 9, 7, 3, 8, 9, 4, 4, 3, 9, 2, 9, 2],
    [3, 5, 1, 7, 2, 3, 8, 6, 5, 3, 1, 9, 1, 1, 9, 2, 2],
    [3, 6, 4, 2, 4, 4, 5, 2, 2, 6, 2, 4, 7, 1, 4, 7, 7],
    [6, 2, 8, 4, 8, 9, 7, 2, 9, 5, 8, 4, 9, 8, 9, 8, 6],
    [1, 2, 6, 5, 3, 8, 3, 4, 9, 9, 4, 5, 8, 7, 5, 6, 3],
    [1, 5, 3, 1, 4, 4, 6, 9, 8, 7, 8, 9, 8, 8, 6, 3, 8],
    [6, 5, 7, 2, 2, 1, 9, 9, 7, 6, 7, 8, 9, 5, 1, 6, 9]
])
```

Grid



LLM



Thought1: Compared to other actions  
Thought2: it is less .....  
Thought3: It's more impactful than....  
Thought4: It offers focused vertical clearing...  
...

Create thoughts for each action



Thought: The evaluation suggests that broader horizontal actions tend..

Deciding action

Action N



## Creating Thoughts for Each Action

---

**Thought1:** Compared to other actions, it offers a moderate vertical removal that could impact multiple elements but is less extensive than horizontal actions. It might be beneficial if vertical removal of this column is strategic, but **less impactful than broader horizontal options.**

**Thought2:** Compared to other actions, it is less extensive than the others with larger dimensions, making it more targeted but less impactful overall.

**Thought3:** It's more impactful than small vertical segments, **potentially clearing a full row segment**, which can significantly alter the state in that area.

**Thought4:** It offers focused **vertical clearing in a different column**, useful if targeting that specific column is strategic, but less extensive than larger horizontal actions.

...

**Create thoughts  
for each action**

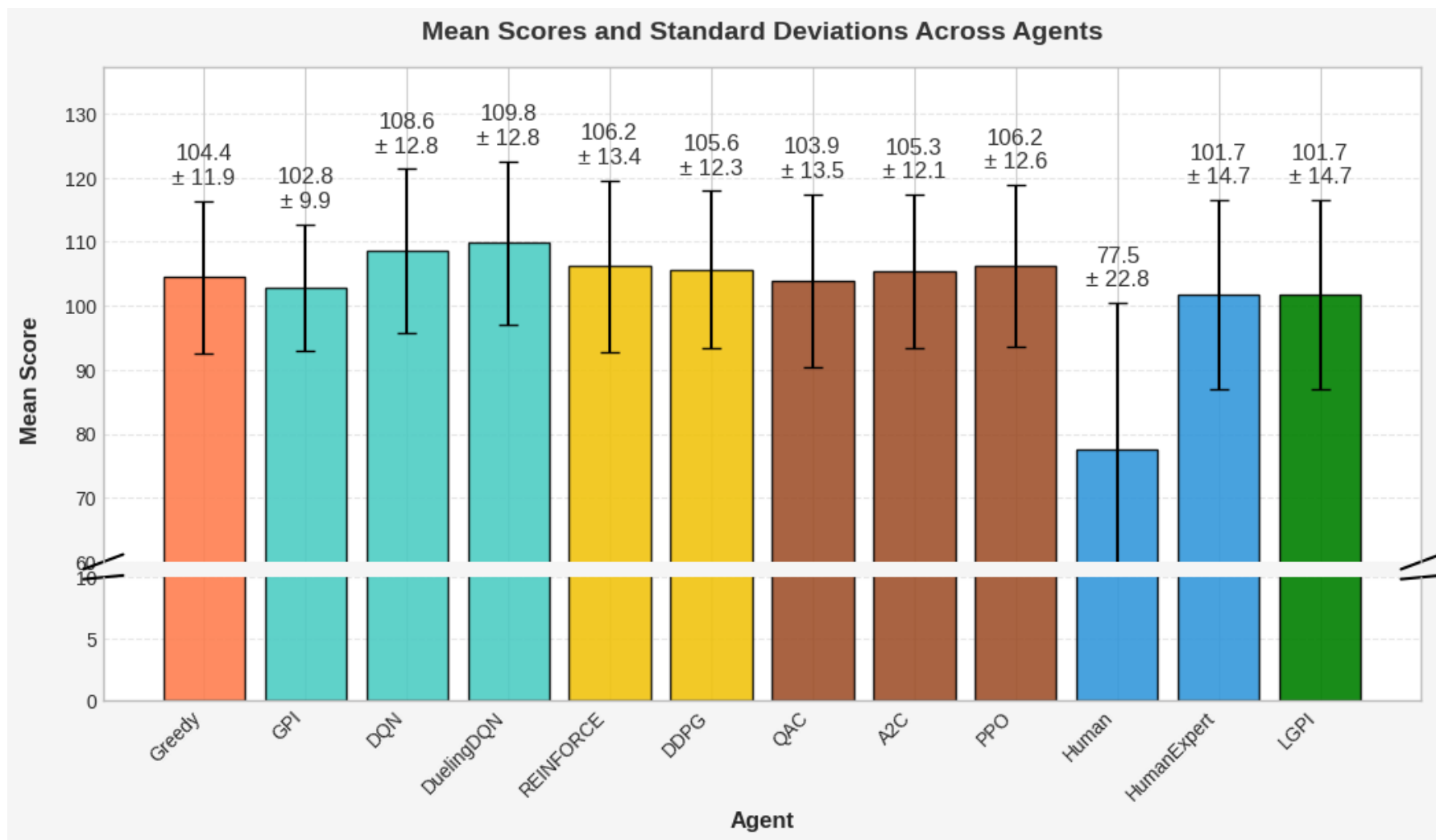
## Deciding action

---

**Thought:** The evaluation suggests that broader horizontal actions tend to be more impactful for clearing multiple cells and potentially completing lines, which is often more advantageous in such puzzle games. **For instance, action 3 (horizontal removal across 4 cells in row 8) and action 13 (horizontal removal of 3 cells in row 5) both offer significant impact.** Comparing these, action 3 covers more cells in a crucial row, possibly leading to larger clearing effects. Similarly, actions 8 and 9 target specific segments, but their impact is more localized. Vertical actions tend to be less impactful unless they target high-value or strategic cells, but broader horizontal moves (like action 3) typically maximize the clearing potential and score. **Therefore, action 3 appears to be the most impactful candidate with a high potential to clear a significant portion of the grid, making it the best choice among the options.**

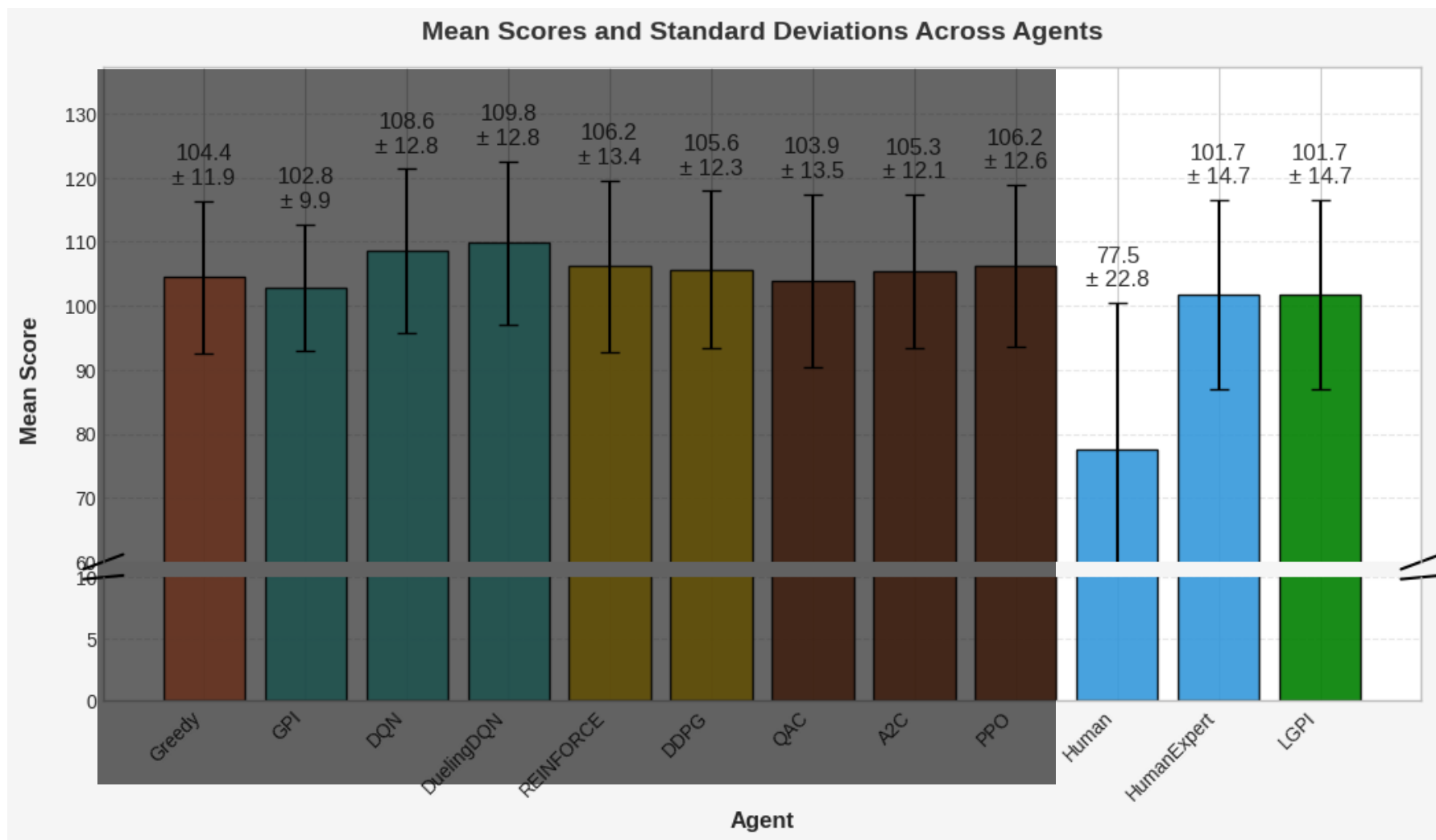
Deciding action

## Results: LGPI

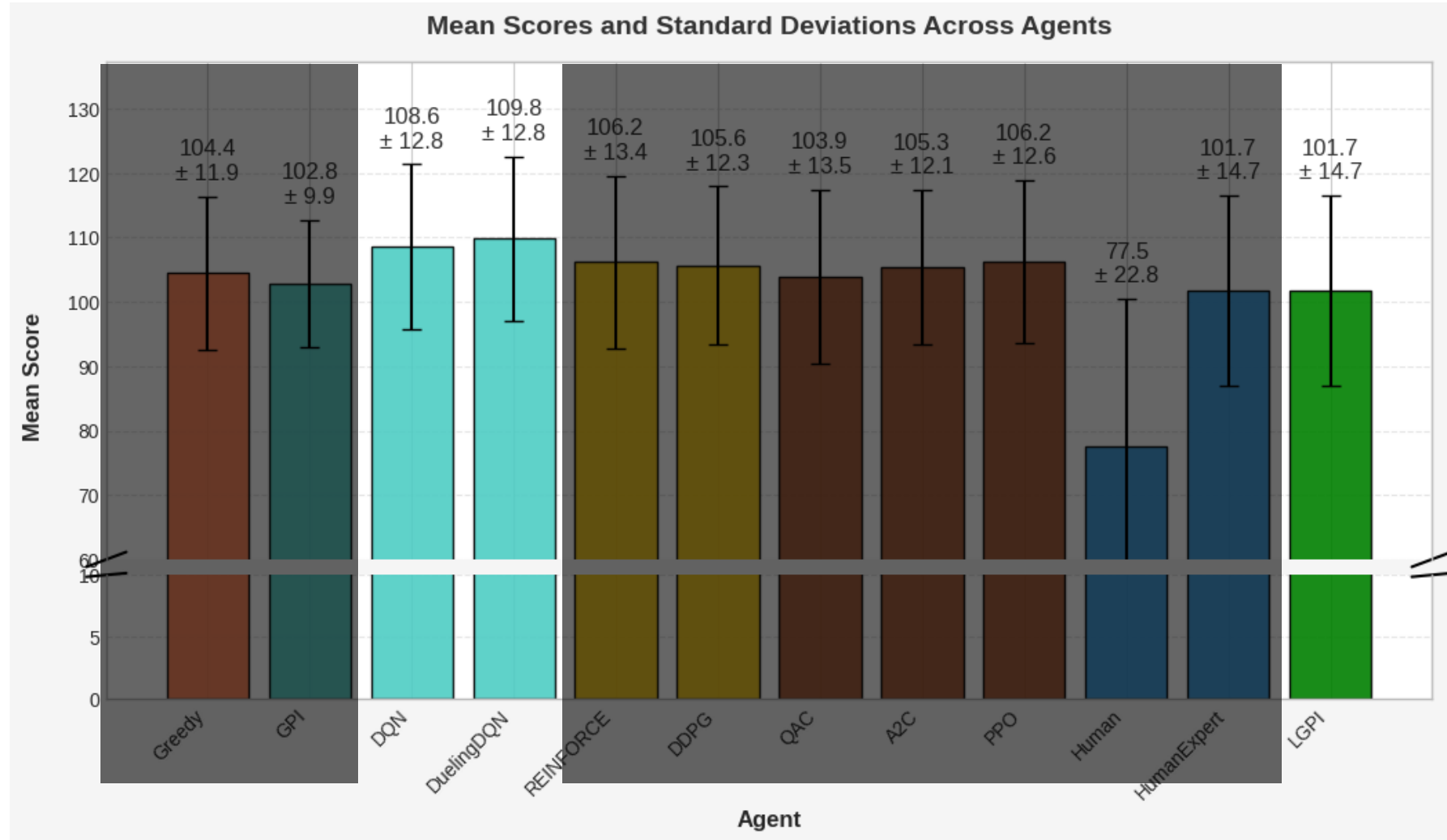




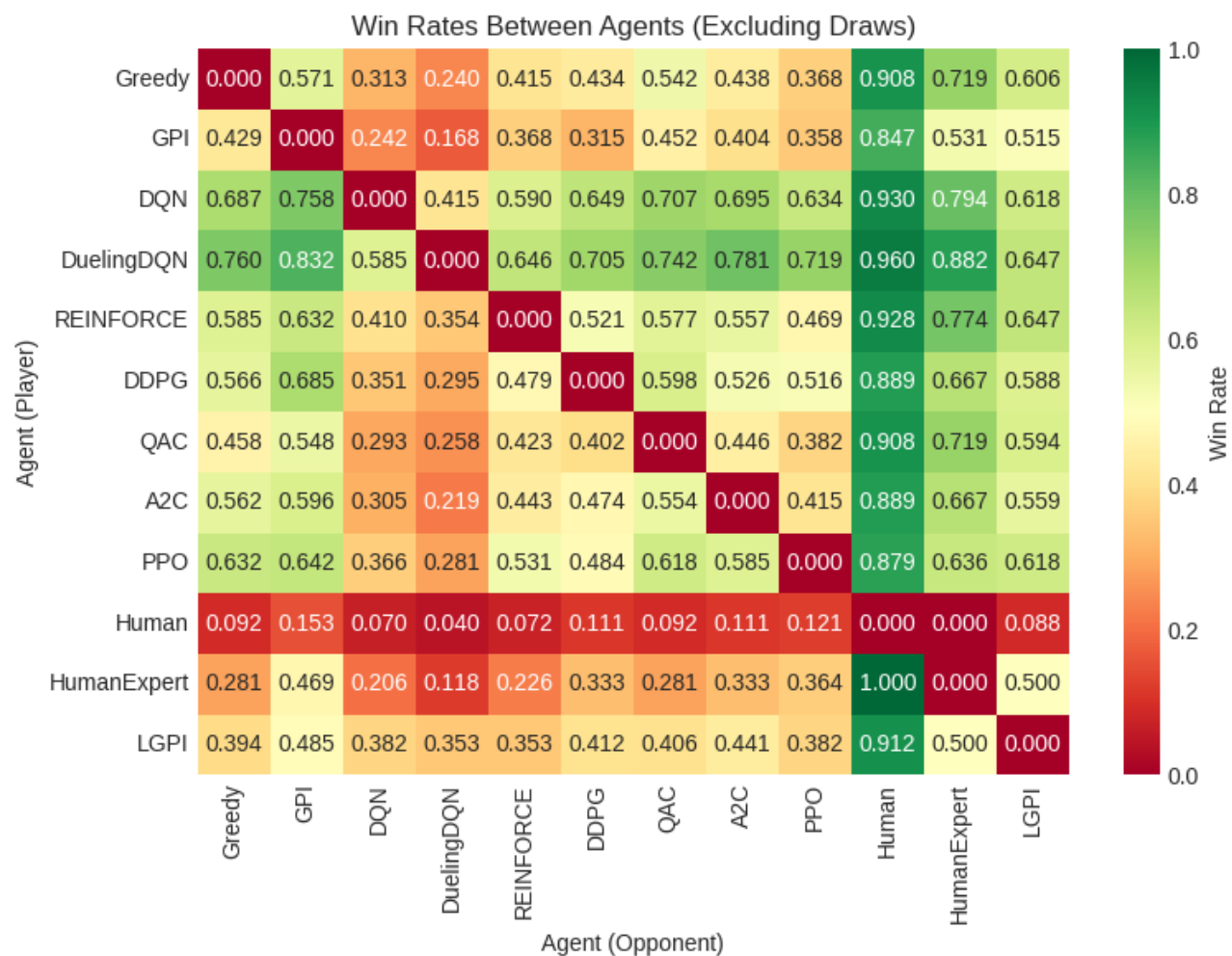
## Results: LGPI



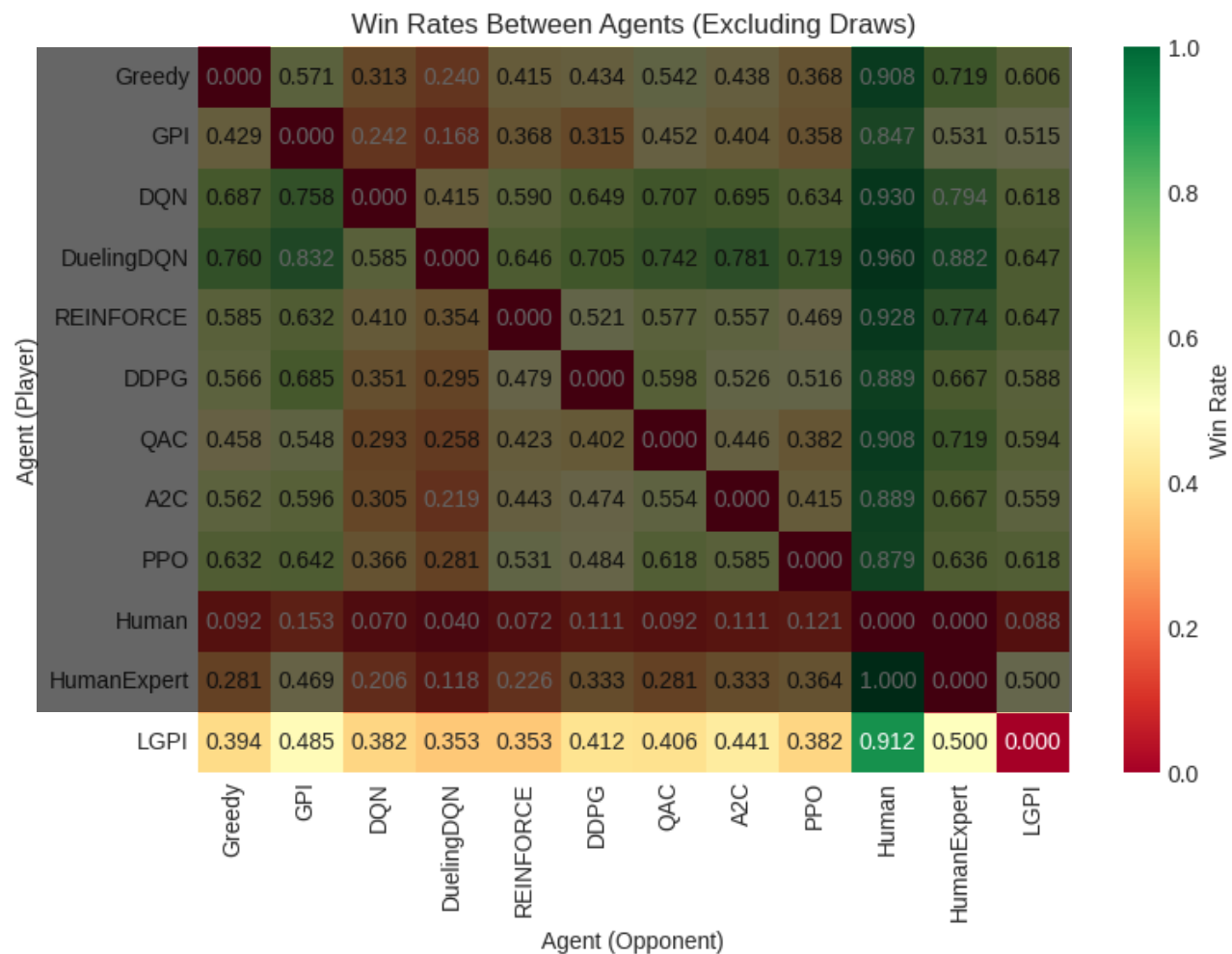
## Results: LGPI



# Results: LGPI



# Results: LGPI



- Introduction
- AppleGame
- Applying RL to AppleGameEnv
- Employing Natural Language
- Limitations & Future Works

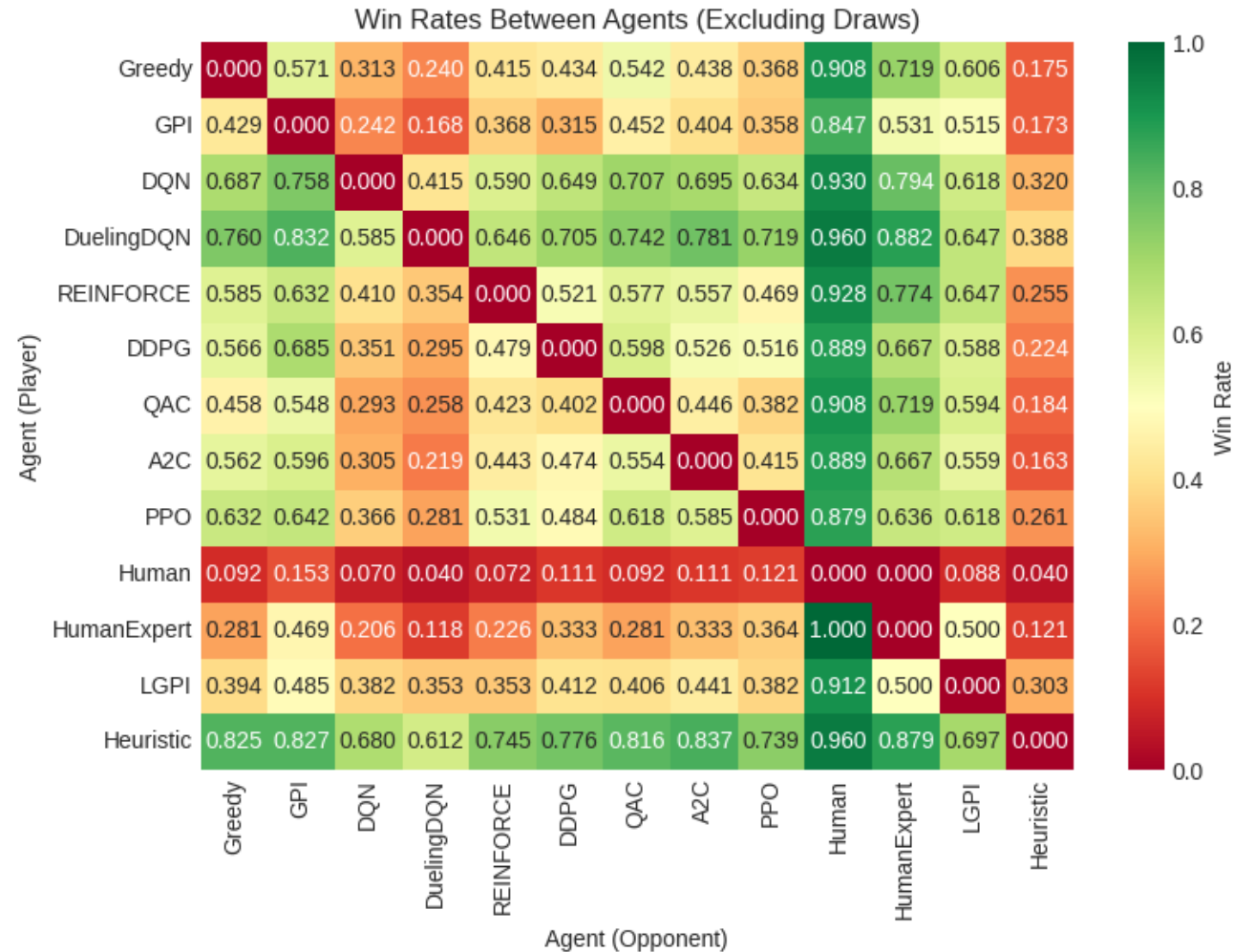
## Problem 3: HeuristicAgent

```
class HeuristicAgent:

    def test(self, golden_grids, render=False, start_idx=1, model_path=None,
             deterministic=None):
        scores = []
        clock = pygame.time.Clock()
        total_grids = len(golden_grids)
        for idx, grid in enumerate(golden_grids):
            global_idx = start_idx + idx
            state, info = self.env.reset(fixed_grid=grid)
            feasible_actions = info.get('feasible_actions', [])
            done = False
            episode_score = 0
            step = 0
            max_steps = 1000
            state_render = "playing"
            if render and not self.env.visualizer.initialized:
                self.env.visualizer.initialize()
```

- 1) Early game, prioritize **clearing smaller groups** of adjacent apples (**e.g., 9-1, 8-2**)
- 2) End game, maximize **apples cleared per move** (**e.g., 3-3-4, 1-2-3-4, 1-1-2-3-2-1**)

## Results: HeuristicAgent





# Results: HeuristicAgent



## Limitations & Future Works

---

Due to financial / time constraints, we were unable to conduct numerous epochs & post-training for NLRL. (as discussed in the original paper).

However, even with off-the-shelf models, natural language proved useful in providing **interpretability with spatial reasoning** for value-based algorithms(TD).

Also, existent methods demonstrated a **human-like strategy**, showing their effectiveness in AppleGameEnv.

Despite this, all 10 agents fell short of a simple heuristic agent in performance. This implies a **gap in algorithms for large, dynamic action-space MDPs** like AppleGame.

## What We learned

---

- How to model a non-tabular environment for RL
- Engineering techniques (action masking, hyperparameter tuning)
- How to use Natural Language for interpretability

**Thank you for listening!!**

For more information, visit <https://github.com/thomaskim1130/AppleGameRL>