RL Project Report

Aaryan Ajith Dev IMT2022038
Shreyas S IMT2022078
Shashank Siddappa Devarmani IMT2022107

June 27, 2025

Abstract

We present a multi-agent reinforcement learning framework based on a competitive environment where agents interact through interdependent actions and rewards. In doing so, we have developed a hierarchy of environments, each one moving one step towards this goal. This model is inspired by how companies strive to achieve AGI (Artificial General Intelligence) currently. This environment is also related to Parrondo's paradox, where two losing strategies, when alternated, result in a winning outcome. By encoding joint actions and observing emergent dynamics, we study how cooperation and interference can lead to non-trivial outcomes. We implement a Deep Q-Network (DQN) agent to learn optimal joint policies and analyze the policies obtained by the agents.

1 Introduction

Connection to Parrondo's Paradox

Before delving into the hierarchical design of our environment, we first highlight the connection between our project and *Parrondo's paradox*. In our setup, the agent can either explore AGI independently or collaborate with other agents. Each strategy, when used in isolation, tends to yield suboptimal rewards, mirroring the essence of Parrondo games, where two losing strategies can be combined to produce a winning outcome.

According to Parrondo's paradox, optimal performance can emerge from alternating between suboptimal strategies. Translating this idea to our context, the agent should not rely solely on collaboration or independence, but rather switch between them judiciously. Our goal, therefore, is to use reinforcement learning to discover the optimal switching policy—learning when to explore alone and when to collaborate—to maximize long-term success.

To build intuition and validate our approach, we begin with simplified versions of the environment, gradually increasing complexity. These initial settings serve as stepping stones, helping us understand how RL can be applied effectively before addressing the full problem of strategic collaboration.

Work Hierarchy

This work has been divided into a hierarchy of environments, also called versions:

1. Single Agent (Simple)
2. Single Agent (Hiring and Firing)
3. Single Agent with "Dummy Agent"
4. Single Agent with "Dummy Agent" (Hiring and Firing)
5. Two Agent (Simple)
6. Two Agent (Hiring and Firing)

2 Single Agent (Simple)

2.1 Description

Here we model the simplest scenario: A single agent with the choice of giving up or continuing the quest, namely, explore and retreat. We model the chance of the agent completing the quest by taking the action explore simply as a random outcome with a low probability p_{AGI}.

2.2 MDP Formulation

This environment consists of three states: Exploring, Retreated, and Completed, with the actions being explore and retreat. The game ends when the agent enters the Retreated and Completed phase. The transitions from the Exploring phase are as follows:

1. **explore:** If explore is taken, then it moves to the completed phase with probability p_{AGI} and reward r_{AGI} (set to 100) and otherwise continues in the explore state with reward 0.
2. **retreat:** Here, the agent deterministically moves to the retreated state with a reward of 10.

2.3 Algorithm Choice and Modifications

The agent chosen for this environment was a tabular Q-Learning agent, well-suited for small, discrete state-action spaces.

The following hyperparameters were used for training:

- Number of episodes: 100,000
- Learning rate: $\alpha = 0.005$
- Initial exploration rate: $\epsilon_{\text{start}} = 1.0$
- Final exploration rate: $\epsilon_{\text{end}} = 0.1$
- Linear decay rate: $\epsilon_{\text{decay}} = \dfrac{\epsilon_{\text{start}}}{n_{\text{episodes}}/2} = 2 \times 10^{-5}$

Exploration Schedule

The agent uses an ϵ-greedy exploration strategy, where ϵ decays linearly over time as follows:

$$\epsilon \leftarrow \max(\epsilon_{\text{end}}, \epsilon - \epsilon_{\text{decay}})$$

This schedule ensures high exploration in the early stages of learning and gradually encourages exploitation of the learned policy.

Environment Statistics

To monitor performance, episode-level statistics such as rewards and lengths were recorded using Gym's `RecordEpisodeStatistics` wrapper, enabling detailed analysis of learning progression (as seen in Fig. 1).

2.4 Insights and Results

The behavior of the tabular Q-Learning agent used in the model-free setting was similar to the analysis one would make by writing down the Q-Bellman equations, assuming the environment was known. It can be shown that the policy obtained is a threshold policy that depends on p_{AGI} (keeping the rewards constant). This means that if $p_{AGI} \geq p_{thresh}$ for some threshold p_{thresh}, then it would try to achieve AGI repeatedly, and otherwise, it would instantly give up. This behavior was also observed in the model-free setting as shown in Fig. 2, but with the threshold being slightly higher than the analytically computed one.

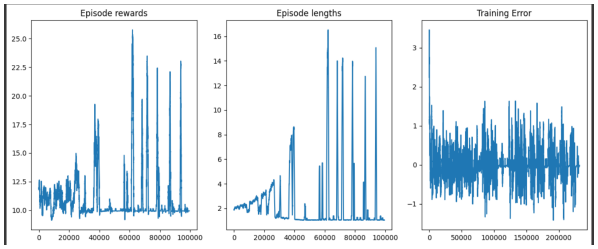

Figure 1: Tabular Q-Learning training results for a sample run with environment parameters, $p_{AGI} = 0.02$ and $r_{AGI} = 100$. The reward for retreating was set at 10, and the rest was set at 0.

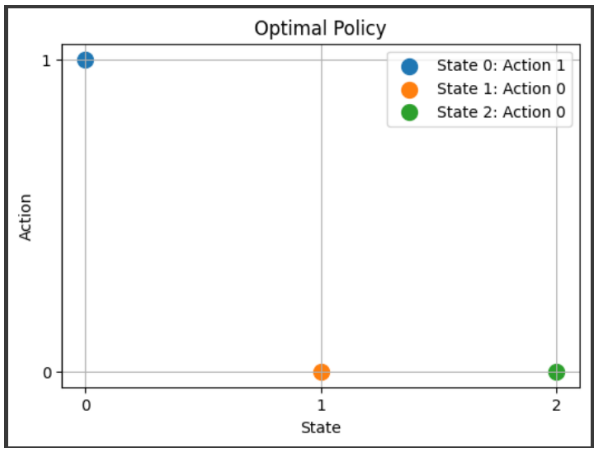

Figure 2: Results for a sample run with environment parameters, $p_{AGI} = 0.02$ and $r_{AGI} = 100$. The reward for retreating was set at 10 and the rest was set at 0.

3 Single Agent (Hiring and Firing)

3.1 Description

We look into the dynamics of an agent (company) which starts out with an initial team of some fixed size and has the choice of hiring an incoming employee (decided by the environment) and simultaneously firing an old employee, apart from the usual trying to find/giving up on AGI.

Each employee is modeled as a tuple of skill (skill of the employee) and salary (demanded salary). Another modification done here is that the probability of getting AGI isn't constant anymore but rather varies w.r.t skill as a sigmoid:

$$\frac{1}{1 + \exp(-\alpha(\text{current\_skill} - s_0))}$$

where s_0 is like the threshold and α is the steepness parameter of the sigmoid function.

The skill of an incoming employee also depends on the current skill of the team and is modeled as follows:

$$\text{skill} = \text{clip}\left(\text{Normal}\left(\mu = \text{skill\_mean} + \text{skill\_bias} \times \frac{\text{current\_skill}}{\text{max\_skill}}, \sigma = \text{skill\_uncertainty}\right), 0, 1\right)$$

$$\text{salary} = \begin{cases} \text{clip}\left(\text{base\_skill} + \text{Normal}\left(\mu = 0.0, \sigma = \text{salary\_noise}\right), 0.0, 1.0\right) & \text{if salary is None} \\ \text{salary} & \text{otherwise} \end{cases}$$

3.2 MDP Formulation

3.2.1 States

The state / observable taken here was a combination of the team's skills and salary (both ensured to be between 0 and 1), a candidate (pair of skill and salary), and the total salary of the team.

3.2.2 Actions

Actions include the following:

- 0: Give up on the quest.
- 1: Try to achieve AGI.
- i ($i \geq 2$): Otherwise, hire the incoming member and fire the $(i-2)^{th}$ member of the current team (0 indexing).

3.2.3 Rewards

In the termination of the game, the agent receives a reward of 10 if it quits the quest, otherwise 100 if it completes the quest.

During the game, if the agent tries to achieve AGI but fails, then no penalty is explicitly given. However, during every iteration, an added penalty of some multiple of the total salary of the team is given to the agent.

When the agent tries to hire/fire a worker, the explicit reward for that action is proportional to (by explicit, we mean ignoring the salary penalty):

$$\text{skill of incoming worker} - \text{skill of outgoing worker}$$

3.3 Algorithm Choice and Modifications

The agent chosen for the above environment was a Deep Q-Learning Agent with the samples from an epsilon-greedy behavior policy. This choice was motivated by the sparse reward structure, along with the environment consisting of infinite states.

The deep Q Network used is a standard feed-forward neural network with two hidden layers (size 128).

The parameters used for training are as follows:

- Number of episodes: 20,000
- Batch size: 128
- Memory Buffer Size: 10000
- Discount factor: $\gamma = 0.99$
- Starting ϵ for ϵ-greedy policy: $\epsilon_{\text{start}} = 0.9$
- Final ϵ: $\epsilon_{\text{end}} = 0.05$
- Decay rate for ϵ: $\epsilon_{\text{decay}} = 1000$
- Soft update coefficient for target network: $\tau = 0.005$
- Learning rate: $\alpha = 10^{-4}$

Target Network Update Rule

The target network is updated using a soft update rule:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

where θ and θ' are the parameters of the policy network and the target network, respectively.

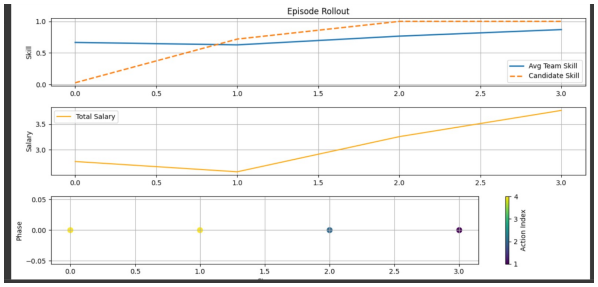

Figure 3: Results for a sample run with parameters, $s_0 = 4, \text{team\_size} = 5, \alpha = 2$. The above result was obtained on one particular run and is not guaranteed to occur.

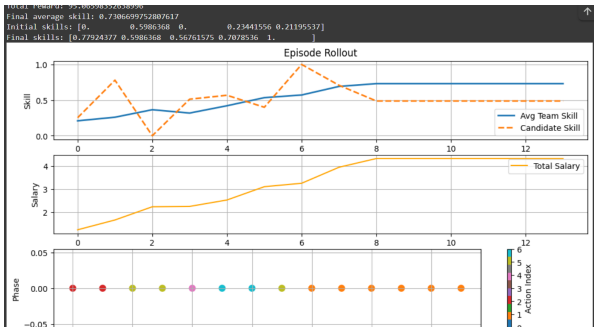

Figure 4: Results for a sample run with parameters, $s_0 = 4, \text{team\_size} = 5, \alpha = 2$, skill_bias=0.2, skill_uncertainty=0.8, skill_mean=0.5, salary_noise=0.2. The scaling factor for the salary penalty was chosen as 0.1, and the scaling for the hire/fire reward was kept at 1. The varying quantity initially kept was the team size. The x-axis shows the iterations for all sub-plots

Epsilon Decay Schedule

The ϵ-greedy threshold decays exponentially over time:

$$\epsilon_{\text{threshold}} = \epsilon_{\text{end}} + (\epsilon_{\text{start}} - \epsilon_{\text{end}}) \cdot \exp\left(-\frac{\text{steps\_done}}{\epsilon_{\text{decay}}}\right)$$

3.4 Insights and Results

We have run a Deep Q-Network (DQN) Agent on the above environment for a variety of parameters, and we observed that the agent was able to understand to fire lazy workers that demand more money, as shown in Fig. 3. Despite the initial penalty it gets by hiring an employee with a lesser skill, it realizes that in the long run, the salary difference would matter too, and hence makes tradeoffs.

Interestingly, it was seen that for a chosen set of parameters, the agent seems to hire / fire until its team skill is greater than some threshold, which was empirically estimated to be around 0.73 irrespective of the team's initial skills (as shown in Fig. 4).

4 Single Agent with "Dummy Agent"

4.1 Description

Now we look at the dynamics of an agent collaborating with a dummy agent, it is mimicking a situation of two agents, but here there is only one agent and the "dummy agent" just does whatever the agent wants so if the agent wants to collaborate it does collaborate, if it wants to break collaboration it breaks it. The "dummy agent" is just a agent which just obeys the actual agent. Later we will look at the situation two agents.

4.2 MDP formulation

States

The environment consists of 9 discrete states, formed by combinations of agent phase and collaboration status:

- **Phase**:
  - `Explore`
  - `Retreat` (terminal)
  - `AGI Discovered` (terminal)
- **Collaboration Status**:
  - `With Collaboration`
  - `Without Collaboration (never collaborated)`
  - `Broken Collaboration (collaborated, but now broken)`

This leads to the following 9 states:

1. State 0: Explore (with collaboration)
2. State 1: Explore (no collaboration, never collaborated)
3. State 2: Explore (collaboration broken)
4. State 3: Retreated (with collaboration) [terminal]
5. State 4: Retreated (never collaborated) [terminal]
6. State 5: Retreated (after breaking collaboration) [terminal]
7. State 6: AGI discovered (with collaboration) [terminal]
8. State 7: AGI discovered (never collaborated) [terminal]
9. State 8: AGI discovered (after breaking collaboration) [terminal]

Actions

The agent can perform the following actions when in a non-terminal state:

- **Action 0**: Explore
- **Action 1**: Retreat (ends episode)
- **Action 2**:
  - If in state 0 (with collaboration): break collaboration (transition to state 2)
  - If in state 1 (no collaboration): initiate collaboration (transition to state 0)
  - If in state 2 (collab broken): no effect (invalid)

Transitions

The transitions are probabilistic for exploration actions and deterministic for retreat and collaboration switching. Probabilities of discovering AGI vary by state:

- **State 0 (collab)**: AGI found with 9% probability
- **State 1 (no collab)**: AGI found with 1% probability
- **State 2 (broken collab)**: AGI found with 5% probability

Rewards

The reward system in the environment incentivizes AGI discovery and penalizes unnecessary exploration while providing moderate reward for safe retreat. The structure is as follows:

- **AGI Discovery Reward (+100):**
  - When the agent discovers AGI while in collaboration (from state 0 to state 6): reward = 100.
  - When the agent discovers AGI without collaboration (from state 1 to state 7): reward = 100.
  - When the agent discovers AGI after breaking collaboration (from state 2 to state 8): reward = 100.
- **Retreat Reward (+10):**
  - Retreating from any exploring state leads to a terminal retreat state with a reward of 10.
    - From state 0 to 3 (with collaboration)
    - From state 1 to 4 (never collaborated)
    - From state 2 to 5 (previously collaborated)
- **Exploration Cost:**
  - **With collaboration:** Exploring in state 0 leads back to itself with high probability (0.91) and incurs a cost of -2.
  - **Without collaboration:** Exploring in state 1 or 2 leads back with high probability and costs only -1.
- **Collaboration Action Cost:**
  - Initiating or breaking collaboration (action 2) costs -1 and transitions:
    - From state 1 to 0 (initiate collaboration)
    - From state 0 to 2 (break collaboration)
  - Attempting to re-initiate collaboration from state 2 is invalid and incurs a penalty of -10 (invalid action).

4.3 Algorithm Choice and Modifications

The algorithm chosen for solving the environment was **Q-learning**, a model-free reinforcement learning technique suitable for discrete state and action spaces. Q-learning learns the optimal action-selection policy by iteratively updating the Q-values based on observed rewards and transitions.

The following hyperparameters were used for training:

- **Learning Rate** (α): 0.1
  Determines the extent to which newly acquired information overrides the old information.
- **Discount Factor** (γ): 0.95
  Represents the importance of future rewards. A factor close to 1 values long-term rewards more.
- **Exploration Rate** (ϵ): 1.0
  Controls the trade-off between exploration (choosing a random action) and exploitation (choosing the best known action).
- **Exploration Decay**: 0.995
  Gradually reduces the exploration rate to favor exploitation as learning progresses.
- **Number of Episodes**: 5000
  Specifies how many episodes the agent trains over to converge on an optimal policy.

These hyperparameters were chosen to balance exploration and exploitation in the early stages and gradually focus on optimizing the policy through learned experience.

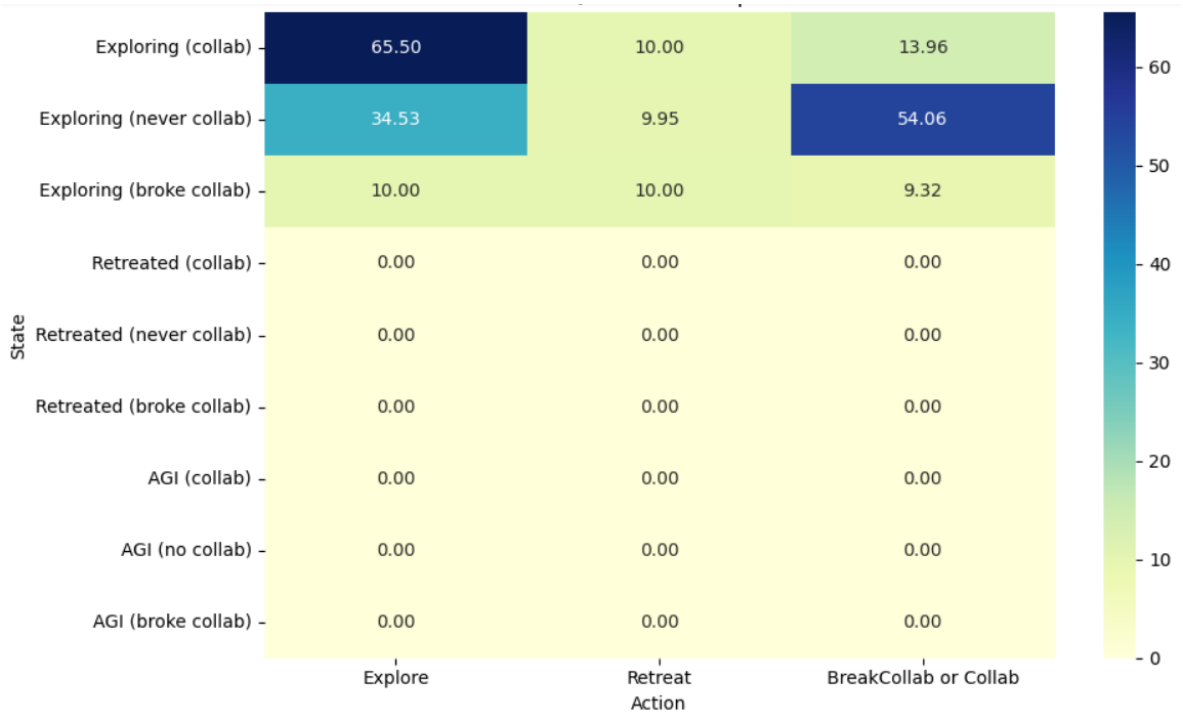

Figure 5: This plot shows the Q table for an example run

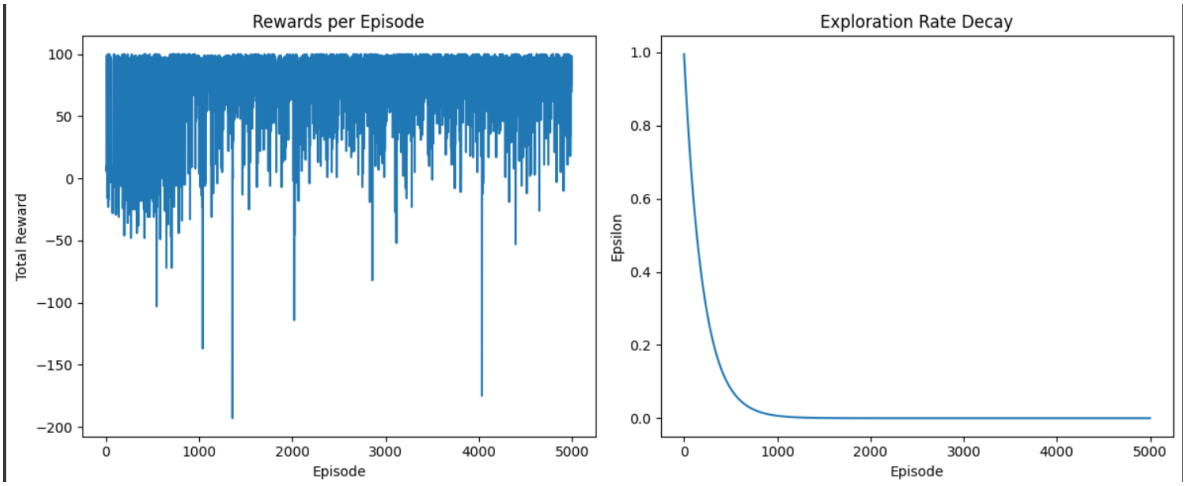

Figure 6: These graphs show how the rewards changes and the exploration rate decays per episode

5 Single Agent with "Dummy Agent" (Hiring and Firing)

5.1 Description

Before we looked at the dynamics of an agent just collaborating with a dummy agent, now we will also allow the agent the ability to hire a fixed team and also firing an old employee. The dynamics of hiring and firing was already discussed above, so we will just see how it is incorporated along with collaborating with a dummy agent

5.2 CombinedAGIEnv: State, Action, Transitions and Rewards

The `CombinedAGIEnv` extends the earlier dummy agent collaboration environment by introducing a hiring and salary management component. This environment integrates both strategic exploration for AGI and workforce optimization. Below is an overview of the key elements:

States

The environment maintains 9 discrete exploration states:

- **State 0:** Exploring with active collaboration
- **State 1:** Exploring, never collaborated
- **State 2:** Exploring, previously collaborated but broke it
- **States 3–5:** Retreated from states 0–2 respectively (terminal)
- **States 6–8:** Successfully discovered AGI from states 0–2 respectively (terminal)

Actions

The agent can perform the following actions:

- **0: Explore** — attempt to discover AGI or continue exploration
- **1: Retreat** — abandon the search and enter a terminal state
- **2: Collaborate Toggle** — initiate or break collaboration
- **3: Search** — draw a new candidate to consider hiring
- **4..(4+team_size): Hire/Fire** — replace team member at index ($i = action - 4$) with the candidate

Transitions

Exploration transitions are probabilistic and depend on the current collaboration state. For instance:

- In **State 0**, AGI is discovered with probability 0.07, else the agent remains and pays a collaboration cost.
- In **State 1**, the agent has a lower AGI discovery chance (0.01) but lower exploration cost.
- In **State 2**, the chance improves slightly (0.015), but the agent cannot re-initiate collaboration.
- All transitions to AGI or retreat states are terminal.

Rewards

- **AGI Discovery:** +100 reward
- **Retreat:** +10 reward
- **Exploration Costs:**
  - Collaborating: −2
  - Non-collaborating: −1
- **Collaboration Toggle:**
  - Initiating collaboration: −1
  - Breaking collaboration: −5
- **Search:** −3 cost to query a new candidate
- **Hiring:** Reward is the net skill improvement. Replacing a low-skilled worker with a higher-skilled candidate results in a positive reward.
- **Salary Maintenance:** A fixed penalty proportional to the average team salary: −5×`(total_salary / team_size)`

Candidate Generation: Skills and Salaries

In the `CombinedAGIEnv`, new candidate workers are generated dynamically using a stochastic process that depends on the current team's average skill level. Specifically, the skill of a candidate is sampled from a Gaussian distribution whose mean increases proportionally with the normalized team skill, encouraging stronger teams to attract better candidates. Formally, the skill is given by:

$$\texttt{skill} = \texttt{clip}\left(\mathcal{N}\left(\mu + \beta \cdot \frac{\texttt{current\_skill}}{\texttt{max\_skill}}, \sigma\right), 0, 1\right)$$

where $\mu = 0.5$, $\beta = 0.5$, and $\sigma = 0.5$ by default. The salary is then sampled as:

$$\texttt{salary} = \texttt{clip}\left(\texttt{skill} + \mathcal{N}(0, \sigma_{\text{salary}}), 0, 1\right)$$

with $\sigma_{\text{salary}} = 0.2$. This mechanism loosely couples higher skill with higher salary expectations.

This design introduces a realistic tradeoff between team performance and operating cost. A high-performing team will tend to attract skilled but expensive candidates, while weaker teams may only access low-skill, low-cost candidates. The `Search` action allows the agent to draw a new candidate (at a fixed cost of -3), enabling exploration of potentially better hiring opportunities. Overall, this encourages strategic team building and budget management as part of the AGI discovery process.

5.3 Algorithm Choice and Modifications

The chosen algorithm for training in the `CombinedAGIEnv` is Deep Q-Learning (DQN), a value-based reinforcement learning approach that approximates the action-value function using a deep neural network. Below are the implementation details and modifications specific to this environment:

Neural Network Architecture

The Q-network is a fully connected feedforward neural network with the following layers:

- Input layer: Flattened state vector (dimension varies based on team size, candidate, and state info)
- Two hidden layers: Each with 256 units and ReLU activation
- Output layer: Produces $Q(s, a)$ values for all discrete actions available in the environment

Key Hyperparameters

- Learning rate: 1×10^{-3}
- Discount factor (γ): 0.99
- Replay buffer size: 100,000
- Mini-batch size: 64
- Initial ϵ (for ϵ-greedy exploration): 1.0
- Minimum ϵ: 0.01
- ϵ decay rate per episode: 0.995
- Target network update frequency: every 10 episodes

Exploration Strategy

An ϵ-greedy policy is used where the agent selects a random action with probability ϵ, and the action with the highest predicted Q-value otherwise. The ϵ value decays exponentially over episodes to encourage more exploitation in later training stages.

Training Loop

The agent collects experiences in the form of $(s, a, r, s', \texttt{done})$ tuples using interactions with the environment and stores them in a replay buffer. At each step, a mini-batch of experiences is sampled to update the Q-network. The temporal-difference (TD) target is computed using a separate target Q-network that is periodically synchronized with the main Q-network.

Observation Preprocessing

Since the environment returns observations in dictionary format, a custom `flatten_observation()` function is used to convert them into a 1D float vector. This flattened vector includes:

- Scalar state identifier (explore state)
- Skill and salary of each team member
- Candidate skill and salary
- Total salary of the team

Policy Extraction

After training, the Q-network is used to define a greedy policy $\pi(s) = \arg\max_a Q(s, a)$ for evaluation or deployment. The policy function takes a dictionary-style observation and returns the best action based on current Q-values.

Remarks

This DQN implementation can handle the complex multi-faceted decisions in the `CombinedAGIEnv`, including exploration, collaboration management, and dynamic team optimization through hiring and firing. Using a target network, experience replay, and exploration decay helps stabilize training in this high-dimensional, partially stochastic environment.

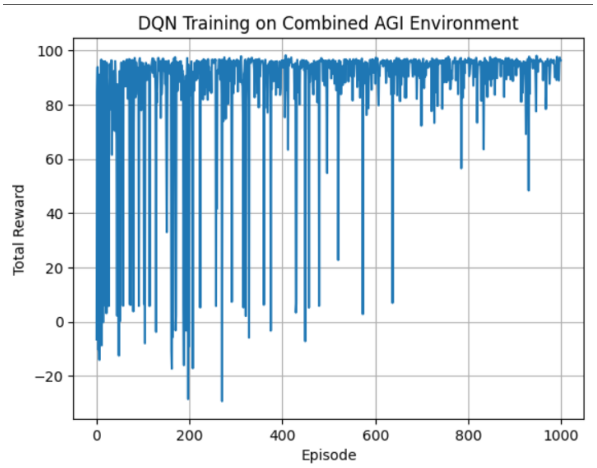

Figure 7: Rewards per episode for DQN

5.4 Policy Visualization

Figure 8 shows a heatmap of the learned DQN policy as a function of team skill and collaboration status. The X-axis represents the average skill level of the team, while the Y-axis distinguishes whether the agent is currently collaborating or not. Each color denotes the index of the action selected by the policy network for a given state.

The actions are indexed as follows:

- **0**: Explore with current team.

- **1**: Hire the presented candidate.
- **2**: Break collaboration.
- **3**: Retreat from exploration.

The policy generally favors **Action 0 (Explore)** when the average team skill is high, both with and without collaboration. When the skill is lower, **Action 1 (Hire)** becomes more prominent, especially when not collaborating. The agent rarely chooses to break collaboration or retreat, indicating that these are only used in specific edge cases. Overall, the policy shows strategic adaptation to skill level while using collaboration status as a secondary cue.

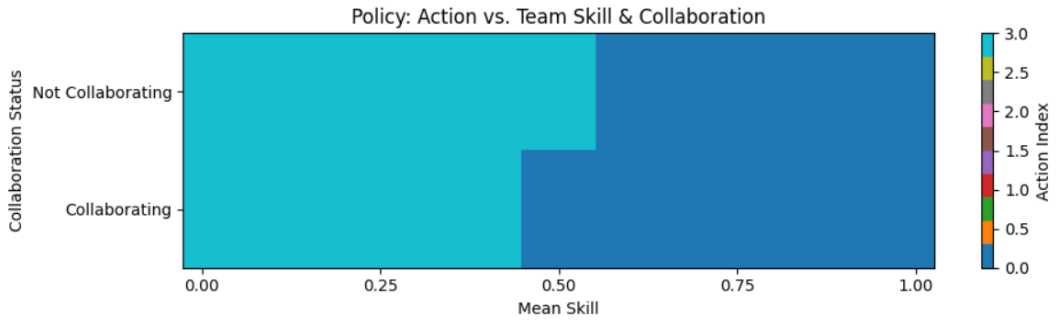

Figure 8: DQN policy heatmap showing selected actions vs. mean team skill and collaboration status.

5.5 Episode Rollout Visualization

Figure 9 presents a single episode rollout of the agent's behavior. The X-axis represents environment steps, while the Y-axis shows the mean skill level of the current team. The blue line traces how the average skill evolves throughout the episode.

Action decisions are marked by scatter points, color-coded using the colormap shown on the right. Each color corresponds to a different action taken by the agent:

- **0 - Explore**: Attempt AGI discovery with current team.
- **1 - Retreat**: Exit the AGI exploration phase.
- **2 - ToggleCollab**: Initiate or break collaboration with dummy agent.
- **3 - Search**: Evaluate a new candidate.
- **4+ - Hire/Fire** i: Replace team member i with the current candidate.

Light blue shaded regions indicate steps during which the agent is actively collaborating. A dashed green vertical line marks successful AGI discovery (explore states 6–8), while a red dashed line signals failure (explore states 3–5).

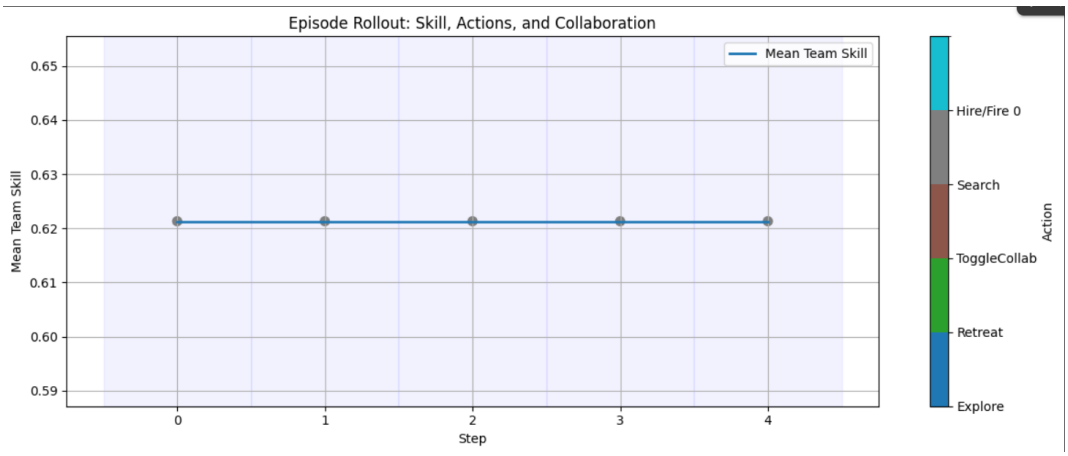

Figure 9: Episode rollout showing team skill, action decisions, and collaboration intervals.

5.6 Conclusion

The `CombinedAGIEnv` successfully integrates two complementary perspectives on AGI discovery: the collaborative dynamics between agents and the strategic process of building and managing a skilled team. By allowing the agent to recruit, fire, and collaborate with dummy partners under resource constraints, the environment captures both social and operational complexities that mirror real-world AGI exploration scenarios.

This hybrid design encourages policies that balance exploration with team optimization, and collaboration with autonomy. The resulting environment serves as a powerful testbed for evaluating multi-faceted decision-making algorithms under uncertainty, and for developing agents capable of long-term strategic planning in complex, dynamic environments.

Policy Behavior: Experiments reveal that trained agents tend to explore independently at low skill levels, initiate collaboration only when sufficient team strength is reached, and strategically toggle collaboration to maximize reward while minimizing redundancy. The policies exhibit sensitivity to both team composition and environmental phase transitions, demonstrating adaptive, context-aware decision-making.

6 MultiAgent system (simple)

6.1 MDP formulation

Our environment is formulated as a Markov Decision Process (MDP) with multi-agent dynamics. The key components are:

6.2 States

The state space captures the current status of both parties and their collaborative relationship:

- **Collaboration Mode**: One of three discrete states - Independent (never collaborated), Collaborative (actively collaborating), or Post-Collaborative (previously collaborated but now independent)
- **Party Status**: For each party, one of three discrete states - Exploring (actively pursuing AGI), Retreated (withdrawn from the race), or Found AGI (successfully discovered AGI)

Formally, the observation space is defined as:

$$S = (C, P_1, P_2) \tag{1}$$

Where:

- $C \in \{0, 1, 2\}$ (Independent, Collaborative, Post-Collaborative)
- $P_1, P_2 \in \{0, 1, 2\}$ (Exploring, Retreated, Found AGI)

6.3 Actions

Each party can take one of four discrete actions independently:

- **Explore (0)**: Attempt to discover AGI, with success probability dependent on collaboration mode
- **Retreat (1)**: Withdraw from the race and secure a modest reward
- **Collaborate (2)**: Initiate or maintain collaboration (effective only if both parties choose this action)
- **Break Collaboration (3)**: End an active collaboration

The joint action space is therefore $A = A_1 \times A_2 = \{0, 1, 2, 3\} \times \{0, 1, 2, 3\}$, where A_1 and A_2 are the action spaces for party 1 and party 2, respectively.

6.4 Transition Dynamics

The environment dynamics are governed by probabilistic transitions based on the current state and joint actions. Key transition dynamics include:

- **AGI Discovery Probabilities**:
  - Independent mode: 1% chance per explore action
  - Collaborative mode: 3% chance per explore action
  - Post-collaborative mode: 1.5% chance per explore action
- **Collaboration Dynamics**:
  - Collaboration is initiated only if both parties simultaneously choose the collaborative action
  - Collaboration is broken if either party chooses to break the collaboration
  - Collaboration status transitions to post-collaborative once broken
- **Terminal States**:
  - Any party discovers AGI
  - Any party retreats
  - Maximum number of steps is reached

6.5 Rewards

The reward structure captures the strategic trade-offs in the AGI race:

- **AGI Discovery**: +100 reward for the party that discovers AGI
- **Competitor Success Penalty**: -50 penalty if the opposing party discovers AGI
- **Retreat Reward**: +10 reward for choosing to retreat
- **Exploration Costs**:
  - Independent exploration: -1 per step
  - Collaborative exploration: -3 per step (higher cost but higher AGI discovery probability)
- **Collaboration Costs**:
  - Initiating collaboration: -5 for each party
  - Breaking collaboration: No direct penalty, but transition to lower AGI discovery probability

This reward structure creates tensions between individual and collective incentives, short-term costs, and long-term payoffs, mirroring real-world strategic dilemmas in technological competition.

6.6 Environment Design

The implementation follows object-oriented design principles using Gymnasium (formerly Gym) for reinforcement learning environments. The core environment class, `CompetitiveAGIEnv`, inherits from Gymnasium's `Env` class and implements the MDP formulation described above.

6.7 Key Environment Features

- **Transition Matrices**: The environment uses structured dictionaries to define state transition probabilities for each party based on current state and action. This approach enhances readability and allows for precise control over different probability distributions.
- **Side Effects**: The transition system includes a mechanism for actions to have side effects on the opposing party, such as when collaborative AGI discovery results in both parties receiving the discovery reward.

6.8 State Preprocessing

A dedicated `AGIRaceObsPreprocessor` class handles the conversion of raw environment observations into a format suitable for neural network processing. This includes:

- One-hot encoding of categorical state variables (collaboration mode, party statuses)
- Batching and tensor conversion for efficient processing

The preprocessor ensures that the state representation captures all relevant information while being in a format that facilitates learning by the DQN agents.

6.9 Algorithm Choice and Modifications

6.10 Deep Q-Network (DQN)

We chose Deep Q-Learning as our primary algorithm due to its demonstrated effectiveness in handling discrete action spaces and complex state dynamics. The implementation includes several key features:

- **Neural Network Architecture**: A three-layer fully connected architecture with 128 units in each hidden layer and ReLU activations.
- **Experience Replay**: A replay buffer stores transitions (s, a, r, s') to break correlations between consecutive samples and stabilize learning.
- **Target Network**: A separate target network is used to compute target Q-values, reducing overestimation bias and improving stability.
- **Soft Updates**: The target network is updated using soft updates with parameter $\tau = 0.005$, balancing stability and learning speed.
- **Exploration Strategy**: An ϵ-greedy policy with exponential decay from 0.9 to 0.05 encourages initial exploration and gradual transition to exploitation.

6.11 Multi-Agent Learning Approach

Our multi-agent implementation uses independent learning where each agent maintains its own DQN and learns from its individual experiences. This approach is known as Independent DQN (IDQN) and treats other agents as part of the environment from each agent's perspective.

Key parameters for both agents include:

- Learning rate: 1×10^{-4}
- Discount factor: 0.99
- Replay buffer size: 10,000

- Batch size: 128
- Target network update rate (τ): 0.005

This approach balances simplicity with effectiveness, though it does not explicitly account for the non-stationarity introduced by simultaneously learning agents.

6.12 Experiments and Results

6.13 Experimental Setup

We conducted training runs with the following configuration:

- Training episodes: 10,000 (for GPU/MPS systems) or 50 (for CPU-only systems)
- Maximum steps per episode: 1,000
- Initial resources per party: 100 units
- Environment parameters as defined in the MDP formulation

6.14 Key Metrics

We tracked several metrics to evaluate agent performance and interaction dynamics:

- Episode duration
- Terminal state distribution (AGI discovery, retreat, resource depletion)
- Collaboration statistics (frequency, duration, breaking patterns)
- Resource utilization patterns
- Action preferences over time

6.15 Results Visualization

The results were visualized through multi-panel plots showing:

- Resource trajectories for both parties
- Reward accumulation over steps
- Collaboration mode transitions
- Action selections by both agents

These visualizations reveal interesting patterns in agent behavior and strategic decision-making that emerge from the reinforcement learning process.

6.16 Conclusion

Our multi-agent reinforcement learning approach to modeling AGI race dynamics demonstrates how complex strategic behaviors can emerge from relatively simple reward structures and learning algorithms. The agents learn to balance the individual incentives of AGI discovery against the collaborative benefits of shared research, while managing limited resources and responding to the actions of their competitor.

7 Version 6: MultiAgent with Hiring Firing

7.1 MDP Formulation

Our environment is formulated as a Markov Decision Process (MDP) with multi-agent dynamics. The key components are:

7.2 States

The state space captures the current status of both parties, their collaborative relationship, and team compositions:

- **Collaboration Mode**: One of three discrete states - Independent (never collaborated), Collaborative (actively collaborating), or Post-Collaborative (previously collaborated but now independent)
- **Party Status**: For each party, one of three discrete states - Exploring (actively pursuing AGI), Retreated (withdrawn from the race), or Found AGI (successfully discovered AGI)
- **Team Composition**: For each party:
  - Team member skills (continuous values between 0 and 1 for each team member)
  - Team member salaries (continuous values between 0 and 1 for each team member)
  - Current candidate skill and salary (for potential hiring)

Formally, the observation space is defined as a dictionary:

$$S = \{\text{collab\_mode}, \text{party\_1\_status}, \text{party\_2\_status}, \tag{2}$$
$$\text{party\_1\_team\_skills}, \text{party\_1\_team\_salaries}, \text{party\_2\_team\_skills}, \text{party\_2\_team\_salaries}, \tag{3}$$
$$\text{party\_1\_candidate}, \text{party\_2\_candidate}\} \tag{4}$$

Where:

- $\text{collab\_mode} \in \{0, 1, 2\}$ (Independent, Collaborative, Post-Collaborative)
- $\text{party\_1\_status}, \text{party\_2\_status} \in \{0, 1, 2\}$ (Exploring, Retreated, Found AGI)
- $\text{party\_1\_team\_skills}, \text{party\_1\_team\_salaries}, \text{party\_2\_team\_skills}, \text{party\_2\_team\_salaries} \in [0, 1]^{\text{team\_size}}$ (Vectors of team member skills and salaries)
- $\text{party\_1\_candidate}, \text{party\_2\_candidate} \in [0, 1]^2$ (Skill and salary of the current candidate for each party)

7.3 Actions

Each party can take one of the following discrete actions independently:

- **Retreat (0)**: Withdraw from the race and secure a modest reward
- **Explore (1)**: Attempt to discover AGI, with success probability dependent on collaboration mode and team skill
- **Collaborate (2)**: Initiate or maintain collaboration (effective only if both parties choose this action)
- **Break Collaboration (3)**: End an active collaboration
- **Hire/Fire (4+i)**: Replace team member at index i with the current candidate, where $i \in \{0, 1, \ldots, \text{team\_size} - 1\}$

For a team size of n, each party has $4 + n$ possible actions. The joint action space is therefore $A = A_1 \times A_2 = \{0, 1, \ldots, n + 3\} \times \{0, 1, \ldots, n + 3\}$, where A_1 and A_2 are the action spaces for party 1 and party 2, respectively.

This extended action space allows agents to make strategic decisions not only about exploration and collaboration but also about team composition, balancing individual skills and salary costs to optimize their chances of discovering AGI.

7.4 Transition Dynamics

The environment dynamics are governed by probabilistic transitions based on the current state and joint actions. Key transition dynamics include:

- **AGI Discovery Probabilities**:
  - Base probability factors:
    - Independent mode: 1% base chance per explore action
    - Collaborative mode: 3% base chance per explore action
    - Post-collaborative mode: 1.5% base chance per explore action
  - These base probabilities are modified by team skill using a sigmoid function:

$$P(\text{AGI}) = \frac{\text{base\_factor}}{1 + e^{-\alpha(\text{team\_skill} - s_0)}} \tag{5}$$

    where α controls the steepness of the sigmoid curve and s_0 is the skill threshold at which the probability reaches half of the base factor

- **Team Dynamics**:
  - New candidates are generated with stochastic skill and salary values
  - Skill values are drawn from a normal distribution, with mean and variance as parameters
  - Salary values are loosely correlated with skill, with added noise
  - Replacing a team member updates the team's total skill and salary

- **Collaboration Dynamics**:
  - Collaboration is initiated only if both parties simultaneously choose the collaborative action
  - Collaboration is broken if either party chooses to break the collaboration
  - Collaboration status transitions to post-collaborative once broken

- **Terminal States**:
  - Any party discovers AGI
  - Any party retreats
  - Any party depletes their resources
  - Maximum number of steps is reached

7.5 Rewards

The reward structure captures the strategic trade-offs in the AGI race with team management:

- **AGI Discovery**: +100 reward for the party that discovers AGI
- **Competitor Success Penalty**: -50 penalty if the opposing party discovers AGI
- **Retreat Reward**: +10 reward for choosing to retreat
- **Exploration Costs**:
  - Independent exploration: -1 per step
  - Collaborative exploration: -3 per step (higher cost but higher AGI discovery probability)
- **Collaboration Costs**:
  - Initiating collaboration: -5 for each party
  - Breaking collaboration: No direct penalty, but transition to lower AGI discovery probability

- **Team Management**:
  - Salary costs: Proportional to the total team salary (team_salary × salary_cost_factor) deducted every step
  - Hiring reward: Difference between new team member's skill and the fired member's skill (can be positive or negative)

This expanded reward structure creates complex tensions between:

- Individual vs. collective incentives in collaboration
- Short-term costs vs. long-term payoffs
- Team skill maximization vs. salary cost minimization
- Exploration vs. team optimization

These trade-offs mirror real-world strategic dilemmas in technological competition, where organizations must balance immediate research progress with talent acquisition and retention.

7.6 Environment Design

The implementation follows object-oriented design principles using Gymnasium (formerly Gym) for reinforcement learning environments. The core environment class, `CompetitiveAGIEnv`, inherits from Gymnasium's `Env` class and implements the MDP formulation described above.

7.7 Key Environment Features

- **Transition Matrices with Dynamic Probabilities**: The environment uses structured dictionaries to define state transition probabilities for each party. Unlike the previous version, these transition probabilities now use lambda functions that dynamically calculate AGI discovery probabilities based on current team skill, allowing for more complex and realistic team performance modeling.
- **Side Effects**: The transition system includes a mechanism for actions to have side effects on the opposing party, such as when collaborative AGI discovery results in both parties receiving the discovery reward.
- **Condition Functions**: Transition effects can be conditional, allowing for complex game dynamics where effects only apply in certain states.
- **Resource Dynamics**: Each party has resources that deplete based on their actions and team salary costs, introducing a multi-dimensional budget constraint that forces strategic decision-making.
- **Team Management**: The environment includes a worker generation system that creates candidates with stochastic skill and salary values. This system includes:
  - Skills drawn from a normal distribution with parameters that can be adjusted
  - Salary values loosely correlated with skill but with added noise
  - A mechanism for updating team statistics when team composition changes
  - A sigmoid function mapping team skill to AGI discovery probability
- **Dictionary Observation Space**: The observation space is implemented as a dictionary with typed spaces for each component, making it easier to handle the increased complexity of team-related observations.

7.8 State Preprocessing

A dedicated `MultiAgentDictObsPreprocessor` class handles the conversion of raw environment observations into a format suitable for neural network processing. This preprocessing is more complex than in previous versions due to the dictionary-based observation space and the additional team-related features. The preprocessing includes:

- One-hot encoding of categorical state variables (collaboration mode, party statuses)
- Processing of team skill and salary vectors
- Encoding of candidate worker information
- Concatenation of all features into a single flat tensor
- Batching and tensor conversion for efficient processing

The preprocessor ensures that the state representation captures all relevant information while being in a format that facilitates learning by the DQN agents. The increased dimensionality of the state space (due to team information) makes effective preprocessing particularly important in this version.

8 Algorithm Choice and Modifications

8.1 Deep Q-Network (DQN)

We chose Deep Q-Learning as our primary algorithm due to its demonstrated effectiveness in handling discrete action spaces and complex state dynamics. The implementation includes several key features:

- **Neural Network Architecture**: A three-layer fully connected architecture with 128 units in each hidden layer and ReLU activations.
- **Experience Replay**: A replay buffer stores transitions (s, a, r, s') to break correlations between consecutive samples and stabilize learning.
- **Target Network**: A separate target network is used to compute target Q-values, reducing overestimation bias and improving stability.
- **Soft Updates**: The target network is updated using soft updates with parameter $\tau = 0.005$, balancing stability and learning speed.
- **Exploration Strategy**: An ϵ-greedy policy with exponential decay from 0.9 to 0.05 encourages initial exploration and gradual transition to exploitation.

8.2 Multi-Agent Learning Approach

Our multi-agent implementation uses independent learning where each agent maintains its own DQN and learns from its individual experiences. This approach is known as Independent DQN (IDQN) and treats other agents as part of the environment from each agent's perspective.

To handle the increased complexity of the state space and action space in this version, we've made several enhancements to the DQN implementation:

- **Improved Replay Memory Handling**: The replay memory management now properly handles the higher-dimensional states and ensures proper batching during optimization.
- **Enhanced Action Selection**: The action selection function has been updated to handle the larger action space that includes team management actions (hiring/firing).
- **Robust Gradient Handling**: Additional gradient clipping and more careful tensor handling to ensure stable learning despite the increased complexity.
- **State Dimension Tracking**: The DQN agent now explicitly tracks the state dimension to ensure proper tensor operations during learning.

Key parameters for both agents include:

- Learning rate: 1×10^{-4}
- Discount factor: 0.99
- Replay buffer size: 10,000
- Batch size: 128
- Target network update rate (τ): 0.005

This enhanced IDQN approach balances computational efficiency with effectiveness.

9 Experiments and Results

9.1 Experimental Setup

We conducted training runs with the following configuration:

- Training episodes: 10,000 (for GPU/MPS systems) or 1,000 (for CPU-only systems)
- Maximum steps per episode: 1,000
- Initial resources per party: 100 units
- Team size: 5 members per team
- AGI threshold skill (s_0): 4.0 (requiring significant team skill to achieve high AGI discovery probability)
- Sigmoid steepness (α): 1.5 (controlling how sharply the AGI probability rises with team skill)
- Salary cost factor: 0.1 (proportional cost of maintaining team salaries)
- Environment parameters as defined in the MDP formulation

9.2 Some insights

- **Coordination Challenges**: Establishing collaboration required simultaneous action selection by both agents, presenting a coordination challenge that agents gradually learned to overcome.
- **Post-Collaboration Behavior**: After collaboration was broken, agents typically accelerated their exploration efforts, attempting to capitalize on the slightly improved AGI discovery probability compared to the never-collaborated state.

10 Conclusion

The addition of team management to the model significantly increases its realism and complexity, allowing for richer strategic behaviors to emerge. In particular, the trade-offs between team skill maximization and salary cost minimization create interesting dynamics that mirror real-world organizational challenges in competitive research environments. The agents' learned strategies for timing their hiring actions, balancing team composition, and adapting their collaboration decisions based on relative team strength demonstrate sophisticated emergent behaviors that arise from the multi-objective optimization problem they face.

This work advances our understanding of how artificial agents might approach complex strategic scenarios involving both competition and potential collaboration, while managing internal organizational resources and team composition. The insights gained may have applications not only to understanding technological races but also to developing AI systems that can assist in strategic decision-making in competitive environments with team management components.

Author Contributions

While all team members collaborated throughout the project, including literature review and idea generation, each member took primary responsibility for specific components. The main contributions are summarized below:

Aaryan Ajith Dev (IMT2022038)

Primary Focus: Versions 2 and 5 of the environment
Key Contributions:

- Designed and implemented a probabilistic environment simulating realistic Hiring-Firing for single-agent and mult-agent scenarios
- Developed and optimized the Deep Q-Network (DQN) implementation, which was reused in later versions.
- Created training and evaluation frameworks for the agents
- Developed visualization tools to analyze agents' behavior and visualized emergent behaviors in agents.

Shreyas S (IMT2022078)

Primary Focus: Versions 3 and 4 of the environment
Key Contributions:

- Enhanced the mathematical rigor of the environment formulations
- Developed probability threshold calculations for AGI discovery in Version 1
- Contributed to theoretical analysis and documentation

Shashank Siddappa Devarmani (IMT2022107)

Primary Focus: Versions 1 and 6 of the environment
Key Contributions:

- Created the initial environment design for the single-agent case
- Implemented the multi-agent environment with both dummy and true competitive agents
- Developed the transition matrices and side effect mechanisms
- Integrated various version components with the DQN implementation
- Ensured consistency and compatibility across environment versions