Undergraduate Thesis

MULTI AGENT REINFORCEMENT LEARNING FOR COOPERATIVE GAMES

Nicolas Constantinou

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2024

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Multi Agent Reinforcement Learning for Cooperative games

Nicolas Constantinou

Supervisor

Dr. Christos Christodoulou

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Abstract

In recent years, Multi-Agent Reinforcement Learning (MARL) has emerged as a promising paradigm for tackling complex decision-making problems in scenarios where multiple autonomous agents interact to achieve common goals. Cooperative games, characterized by interdependence among agents, are fertile ground for the application of MARL algorithms to enhance cooperation and coordination.

My thesis aims to delve into the field of Cooperative Multi-Agent Reinforcement Learning to investigate, analyze and propose algorithms that can facilitate effective cooperation between agents in a cooperative game environment.

The fundamental challenge in cooperative games lies in orchestrating interactions between autonomous agents to optimize joint performance, often requiring agents to strike a delicate balance between individual goals and the collective goal.

Traditional MARL methods face limitations in handling the complex dynamics of cooperative scenarios, it requires the exploration of advanced reinforcement learning techniques. The goal of this research is mainly Algorithmic Exploration.

A milestone in the field of cooperative games is the Hanabi game environment, which aims to train an algorithm that can play Hanabi efficiently and cooperatively, taking advantage of available information and minimizing errors. This requires advanced reinforcement learning techniques to deal with the complexity of the environment and the uncertainties arising from cooperation between players.

In my thesis I will implement the board game "The Mind". "The Mind" is a cooperative card game where players aim to place cards in ascending order without communicating explicitly, using the help provided by the game as well as the way of thinking of the other players. For the implementation of this game, I will rely on the Hanabi training environment. Building upon the foundational strategies employed in the Hanabi game environment, this study introduces novel algorithmic approaches tailored to "The Mind," emphasizing the development of non-verbal communication cues and synchronized actions among agents. The research focuses on two primary algorithms: Independent Proximal Policy Optimization (IPPO) and Multi-Agent Proximal Policy Optimization (MAPPO). These algorithms are assessed for their ability to foster high levels of coordination and cooperation without explicit communication. Both algorithms demonstrated the capability to adapt their strategies to the cooperative nature of "The Mind," effectively mirroring human gameplay patterns such as intuitive turn-taking and strategic card placement based on the collective game state.

This thesis highlights the critical role of advanced reinforcement learning techniques in overcoming the inherent uncertainties and optimizing joint performance in cooperative

MARL environments. Future work may explore scaling these algorithms for more complex environments and integrating more dynamic communication strategies to further enhance agent interaction and cooperation.

Content

Chapter 1

Introduction

In the realm of artificial intelligence, the study of reinforcement learning offers a fascinating path for comprehending and creating self-governing systems that can navigate and make decisions within intricate settings. Specifically, multi-agent reinforcement learning (MARL) adds another layer of complexity by incorporating numerous agents that interact and learn simultaneously to fulfill their goals within a mutual environment. This emerging area of research holds the potential for progress in Artificial Intelligence and illuminates the dynamics of cooperation and rivalry seen in natural and social systems. Among the multitude of applications for MARL, games offer a uniquely structured yet challenging domain. Especially games, with well-defined rules and objectives, serve as excellent testbeds for developing and benchmarking MARL algorithms.

1.1 Challenges

The cooperative nature of “The Mind” card game [1] could lead to intriguing challenges during the implementation of the environment. One of the most significant challenges is implementing the implicit communication allowed in "The Mind," such as timing and body language signal. In a computational model, you could introduce a mechanism for "signaling" between agents without direct communication, possibly using a shared environment state or auxiliary channels that represent non-verbal cues.[2] .

The game provides feedback only at the end of each level (success or failure) or when an incorrect card is played. This rarity of rewards can make learning difficult, requiring techniques like reward shaping, curriculum learning, or the use of intrinsic motivation to guide the agents' learning process [3] .

Agents must learn not just which cards to play, but also when to play them, requiring a refined understanding of other agents' actions. This aspect introduces a significant challenge in temporal coordination, which is less common in other MARL environments [4] . Also as

the number of players (agents) increases, the complexity of coordination and communication exponentially grows. Designing a scalable environment that can handle increased agent counts without a significant drop in performance is crucial.
The game's difficulty increases as players progress through levels. A MARL system must not only succeed at lower levels but also prepare its strategy to perform well at higher levels, which involves more cards and thus a larger action space and increased complexity.
Translating human non-verbal communication cues into a format that can be understood and utilized by AI agents is a novel challenge. This might involve creating a limited set of "actions" that represent these cues and integrating them into the agents' decision-making processes.

1.2 Objectives

The main objectives of this thesis are structured to address both the theoretical and practical aspects of adapting "The Mind" card game into a MARL environment. These objectives are outlined as follows:

1. Develop a MARL Environment, a simulation of "The Mind," capturing the essence of its gameplay, which includes cooperative play without direct communication among players. This involves defining appropriate state and action spaces, as well as a reward system that encourages cooperative strategies.
2. Evaluate Learning Algorithms, to implement and assess the performance of various reinforcement learning algorithms, including Independent Q-Learning, IPPO, and MAPPO, within this environment [5][6]. The aim is to analyze how different algorithms cope with the challenges of partial observability, limited communication, and the necessity for implicit coordination.
3. Study the behavior and strategies developed by agents in the game, comparing them with human strategies where possible. This objective includes identifying any emergent behaviors or strategies that differ significantly from human play, and understanding the conditions under which these occur.
4. To explore the potential applications of the findings in real-world scenarios where cooperation and non-verbal communication are crucial, such as in autonomous vehicle coordination or robotic team operations.

These objectives guide the research towards significant contributions to the fields of game theory, cooperative AI, and MARL, offering both theoretical insights and practical implications.

Chapter 2

Background

2.1 Cooperative MARL environments

Multi-Agent Reinforcement Learning (MARL) in cooperative environments is a fascinating and complex field within artificial intelligence where multiple agents interact within a shared environment to achieve a common goal. The focus is on designing strategies that allow these agents to learn how to cooperate, often without direct communication between them. This has significant implications for a range of applications from autonomous vehicles to complex simulation systems and cooperative robotics.

In MARL, each agent typically makes decisions based on its own observations of the environment, which might include the state of the environment and other agents' actions. Unlike single-agent reinforcement learning, where the entire environment's state and reward feedback loop are controlled by one agent's actions, in MARL, the state and reward for any given agent can be affected by the actions of other agents.

Cooperative environments are scenarios where agents need to work together to maximize a shared reward or achieve a group objective. This collaboration requires mechanisms to coordinate actions among agents, which can be challenging due to the agents having partial observability and shared rewards. Each agent typically only sees a part of the entire environment, making it necessary for them to infer the rest or rely on communicated information. Also, all agents may share the same reward function, which aligns with their goals but also requires them to find a balance in strategy that maximizes the collective outcome. These are the main challenges of cooperative MARL environments.

To eliminate these challenges various approaches have been developed to handle the complexities of learning in cooperative MARL settings.

Techniques like multi-agent Q-learning [5] , where each agent maintains a value function that estimates the expected utility of actions considering the possible actions of other agents or Policy Gradient Methods [6] that involve each agent learning a policy that maximizes the expected return. Methods like Multi-Agent Deep Deterministic Policy Gradient extend single-agent approaches to multi-agent environments by considering the impact of other agents' policies.

Lastly Centralized Training with Decentralized Execution is a method where agents are trained in a centralized manner where the training algorithm has access to the actions and states of all agents, but during execution, each agent operates independently using only its own local observation [7] .

Cooperative MARL has wide applications like coordinating multiple robots for tasks like search and rescue or automated warehouses or managing traffic lights and flow in urban settings to optimize traffic throughput and reduce congestion.

Also complex games like soccer or strategy games where multiple players must work together against an opposing team is challenging to implement in MARL and that’s the section this dissertation is about.

2.2 “The Mind”

This thesis focuses on "The Mind" [1] a cooperative card game that, despite its simple premise, unfolds into a complex test of intuition, strategy, and non-verbal communication among players. Here’s how it works :

The game can be played by 2 to 4 players. Each player receives a hand of cards. The number of cards each player gets corresponds to the current level of the game. The game starts at level 1, so each player gets one card and there are 100 cards in the deck, numbered 1 to 100. The goal is to complete all levels by placing cards from the hand in the middle of the table in ascending order. Players cannot communicate about the specific numbers they hold, and they try to synchronize with each other to play cards in the correct order. There are no turns; players decide when to play cards, aiming to do so in ascending order without discussing card values .

If a player places a card out of order (e.g., a card higher than one still in a player's hand is played), the team loses a life. The misplaced card and any lower cards not yet played are discarded. The team starts with a certain number of lives. Successfully completing a level allows the team to move to the next level, which increases the number of cards each player receives. Some levels grant extra lives or shuriken cards when completed. A shuriken card allows all players to discard their lowest card, helping them to realign if the game becomes too challenging. To use a shuriken, all players must agree by raising a hand. The game is won if the players complete all the levels with at least one life remaining and is lost if players lose all their lives before completing all the levels.

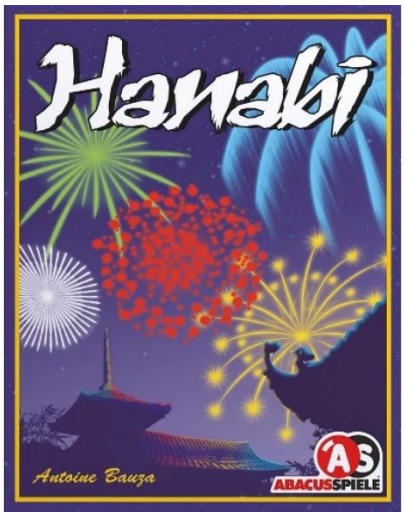
This game uniquely challenges the realm of MARL, especially in its requirement for agents to master implicit communication and coordination strategies without direct exchange of information. Deploying "The Mind" within a MARL environment epitomizes the convergence of numerous AI research domains, encompassing communication in decentralized systems, the crafting of cooperative strategies, and fostering social and intuitive intelligence among artificial entities.

The primary goal of this dissertation is to navigate these aspects through the development of a MARL framework inspired by "The Mind" game, with the aim to dissect how autonomous agents forge cooperative strategies under the veil of information asymmetry and communicative restrictions. This investigation endeavors not just to push forward the technological power of MARL systems, but also to illuminate the foundational aspects of collective intelligence that arise from the dynamics of multiple interacting learning agents. This research effort positions itself at the nexus of AI exploration, game theory, and cognitive science, delivering an expansive review of how artificial agents can maneuver and excel within the complexity of games crafted by humans, and, by extension, navigate complex scenarios in the real world.

The thesis aims to enrich our comprehension of Cooperative Multi-Agent Reinforcement Learning by examining, evaluating, and proposing algorithms designed to improve the collaboration between agents within different cooperative game settings.

The fundamental challenge in cooperative games lies in orchestrating interactions between autonomous agents to optimize joint performance, often requiring agents to strike a delicate balance between individual goals and the collective goal.

2.3 Hanabi as a testbed



The game of Hanabi provides a rich MARL environment for exploring cooperation among agents in settings of imperfect information. Here's a detailed look at how Hanabi works as a MARL environment [8] . Hanabi is a cooperative card game where players work together to create stacks of cards in sequential order for each color available in the game. What makes Hanabi unique is that players cannot see their own cards; they can only see their teammates' cards. The goal is to complete all stacks perfectly, but the catch is the limited and strategic communication allowed among players, making it a challenging exercise in cooperative strategy and inference .

In Hanabi, all players share a common goal: to score the highest points by stacking cards correctly. This cooperative nature requires players to work together, contrasting sharply with competitive environments where agents work against each other. Cooperation is facilitated through limited communication, typically about the properties of the cards each player holds, using a finite set of tokens to give hints.

Players have imperfect information because they cannot see their own cards. This setup compels players to depend heavily on the hints given by others to infer what cards they hold. Effective play requires understanding not only the explicit information shared but also the intent behind each communication, leading players to perform complex inferential reasoning about their teammates' knowledge and strategies.

In Hanabi, communication is not free but comes at a cost of hint tokens, which are limited. Players can only provide hints about the features of the cards (such as color or number), making the timing and content of hints critically strategic. This restriction adds a layer of depth to the gameplay, where every hint needs to maximize information value.

The limitations and challenges of the Hanabi environment seem to have similar structure to the Mind's gameplay and structure and that's the reason I decided to modify the Hanabi's JAX environment [9] [10] .

The agents must develop an understanding of other agents' knowledge and intentions, a concept known as "theory of mind" [11] . This is crucial in Hanabi because effective gameplay hinges not just on what you know but what you understand about what others know. Finding effective strategies (policies) that work well with others' strategies is vital. Agents must learn not just to

optimize their actions but to align their actions with the group's strategy to maximize the collective outcome. Agents must make decisions with incomplete and indirect information, a significant challenge compared to environments with perfect information.

Researchers have used Hanabi as a testbed for developing and evaluating AI algorithms that can handle complex, cooperative interactions among agents with imperfect information. This has involved advancements in areas such as reinforcement learning, policy learning, and communication protocols within multi-agent systems.

Hanabi not only provides a challenging environment for developing RL agents but also offers insights into how agents can be designed to effectively cooperate and communicate in real-world applications, where similar constraints often exist.

Similarities

The games Hanabi and "The Mind" share several foundational characteristics that make them both compelling environments for studying multi-agent reinforcement learning (MARL), particularly in the context of cooperation, communication, and strategy under conditions of imperfect information. Here's how the environments associated with each game relate to each other and the insights they offer for MARL research and development.

Firstly the cooperative gameplay, both games are purely cooperative, where all players share the common goal of achieving the best group outcome (maximizing score in Hanabi, successfully playing all cards in sequential order in "The Mind"). Success in both games depends not just on individual decision-making but on effective group coordination and strategy.

Another common characteristic is the imperfect information the two games provide, for example in Hanabi, players cannot see their own cards but can see others', requiring dependence on teammates for information about their own hand and in "The Mind", while players can see their own cards, they cannot share this information directly and must synchronize playing cards in ascending order without direct communication, relying heavily on timing and mutual understanding. Both environments challenge agents to operate effectively with incomplete knowledge of the game’s state. The main and most important feature is the limited communication. Hanabi restricts communication through a limited number of tokens that can be used to give hints about the properties of the cards. This makes choosing what information to share, and when, a critical part of the strategy. "The Mind" limits communication even more strictly, as players are not allowed to communicate about their cards at all, leading to a need for implicit understanding and non-verbal cues to coordinate actions.

These constraints push the development of reinforcement learning agent strategies that must infer intent and strategy from limited and indirect information.

In terms of the Theory of Mind both games elevate the necessity for AI agents to develop a theory of mind that is, an ability to consider the perspectives, knowledge, and potential intentions of other agents. Developing agents that can accurately predict and adapt to the actions of other agents based on limited cues is a significant challenge in these environments. Agents must learn optimal strategies that are not just effective on their own but are also complementary to the strategies of other agents. This involves complex policy coordination where the agent's behavior must align with the group's overall strategy to succeed.

Both games can serve as testbeds for developing communication protocols where the cost of communication is high, and only essential strategic information must be conveyed. This has broader applications in real-world scenarios where communication bandwidth is limited, or where agents must operate under strict secrecy or privacy constraints.

The insights gained from developing agents that can handle the dynamics of Hanabi and "The Mind" can be applied to real-world scenarios like collaborative robotics, autonomous vehicle coordination, and complex negotiation systems, where multiple agents must work together towards common goals under uncertainty and with limited communication. These environments can help bridge the gap between theoretical MARL research and practical applications in complex, real-world systems.

2.4 Methods used

Exploring algorithmic strategies for the cooperative card game "The Mind" involves considering how different RL techniques can adapt to the game's unique challenges. In "The Mind," players must synchronize with each other to play cards from their hands in ascending order without any communication about the cards they hold.

Some algorithmic approaches that I considered for "The Mind" are Independent Q-Learning (IQL), Deep Q-Networks (DQN), Independent Proximal Policy Optimization (IPPO) and Multi-Agent PPO (MAPPO).

The IQL algorithm treats each agent independently, with each learning its own Q-function, which estimates the utility of actions given the state of the environment. In "The Mind," where inter-agent coordination is silent and crucial, IQL can struggle since it doesn't inherently model other agents' actions or strategies [5] . However, with modifications to consider implicit cues from other players' actions, it could be adapted for better performance.

The IPPO algorithm is a policy gradient method that optimizes a "surrogate" objective function, providing effective learning with fewer data samples than other policy gradient methods. MAPPO extends PPO to multi-agent environments by considering joint actions and rewards [6] .

MAPPO could be highly effective for "The Mind" as it can help coordinate strategies among multiple agents through shared rewards and policy updates, encouraging cooperative behavior essential for this game.

Opponent-Aware Learning, another significant advancement in MARL , is the introduction of opponent-aware learning mechanisms, such as those encapsulated in the Learning with Opponent-Learning Awareness (LOLA) approach [12] . LOLA explicitly incorporates the anticipated learning adaptations of other agents into each agent’s decision-making process. This is particularly relevant to "The Mind," where players must intuit and anticipate others' plays without direct communication. By integrating LOLA's principles, agents can be designed to consider how their actions will influence the strategies of other agents, fostering a more dynamic and interactive learning environment.

The LOLA framework has demonstrated its effectiveness in promoting cooperative behavior in environments traditionally characterized by competition, such as the iterated prisoners' dilemma. This aspect of LOLA could be crucial for "The Mind," where the goal is to achieve the highest possible group outcome through cooperation. Understanding how cooperative strategies can emerge in adversarial settings provides a theoretical foundation for encouraging similar outcomes among agents in "The Mind."

Challenges of Non-Stationarity and Partial Observability seem to arise by the fact that MARL environments are inherently non-stationary; the strategies of agents continuously evolve, meaning the environment’s dynamics change over time. LOLA addresses these challenges by adapting the agents’ learning processes to account for the ongoing learning of their opponents. This approach is vital for managing the non-stationarity in "The Mind," where each round of play can significantly alter the state of the game. Furthermore, the issue of partial observability where agents must operate without full knowledge of the game state is analogous to many real-world problems addressed by MARL. LOLA's method of accounting for hidden information and predicting opponent moves can directly inform the development of effective strategies under these conditions.

Incorporating these concepts into the thesis will help articulate why advanced MARL approaches are not only applicable but also essential for developing an effective simulation of "The Mind." By discussing these points, the thesis will highlight the intersection of

theoretical MARL research and practical application, underscoring the innovative approach taken in adapting "The Mind" into a MARL framework.

Challenges

Some challenges I faced in algorithmic exploration were communication constraints, which was the primary challenge in "The Mind", the lack of direct communication. Any algorithm must infer intentions and plan actions based solely on the sequence of played cards and game progress, which can be non-trivial without explicit signals. Also, coordination was challenging because achieving synchrony in actions without communication requires sophisticated prediction and adaptation strategies within the algorithms, particularly in learning when to play or hold back cards based on the inferred pace of the game.
Finally, scalability and complexity, adapting algorithms to handle different levels of the game, where the number of cards increases, adds complexity. The scalability of the algorithms to handle these varying conditions is crucial.

2.5 PPO Algorithms

In previous experiments in several MARL environments the researchers discovered that Proximal Policy Optimization (PPO) is effective in multi-agent reinforcement learning settings, particularly focusing on its core features and how they benefit MARL environments [8] .

PPO modifies the policy gradient objective to include a clipping mechanism that prevents too large policy updates. This is crucial in MARL, where the actions of one agent can significantly impact the environment for other agents. The clipping helps maintain training stability across updates, which is vital in environments with many agents interacting dynamically.
As an on-policy algorithm, PPO uses the data collected from the current policy to update itself. This aligns well with environments where agents need to adapt to strategies that are continuously evolving, as it uses the most recent and therefore relevant data to learn.
PPO can update its policy using the same set of collected environment samples for multiple epochs, a method that can be more sample efficient in complex environments. Each sample can provide rich learning signals about interactions among multiple agents, making it valuable for repeated scrutiny.
In cooperative MARL tasks, agents need to coordinate their actions, which can lead to highly interdependent and potentially chaotic learning dynamics. PPO's clipped objective curtails drastic policy shifts, ensuring that the learning process remains stable even as agents learn to cooperate more effectively [6].

The flexibility of PPO in handling different types of interactions, competitive, cooperative, or mixed ,is beneficial. Its fundamental algorithmic structure doesn't need significant adjustments to switch between these modes, making it a versatile tool in the MARL toolbox.

PPO has shown robust performance across a range of environments and setups. This robustness is especially useful in MARL, where the complexity and variability of environments can be much higher than in single-agent scenarios.

Many MARL environments feature partial observability, where agents cannot see the full state of the environment. PPO's ability to work effectively with the limited and local information typical in on-policy learning setups makes it suitable for such conditions.

Recent empirical studies [6], have demonstrated PPO's effectiveness in complex MARL environments like the multi-agent particle environments, StarCraft II, Google Research Football, and Hanabi. These results underline the algorithm's capability to manage the complexities of multi-agent learning without requiring extensive modifications.

These characteristics and empirical validations illustrate why PPO, despite its traditional on-policy nature and perceived inefficiencies in sample use, has emerged as a strong contender in the field of MARL, capable of competing with and often surpassing more specialized off-policy methods.

The paper discusses specifically the use of Independent Proximal Policy Optimization (IPPO) and Multi-Agent Proximal Policy Optimization (MAPPO) in the context of JaxMARL [9] , emphasizing their utility in multi-agent reinforcement learning environments.

In order to test the learning progress on "The Mind" environment , I am going to use the IPPO and MAPPO algorithms , just like in the Hanabi JaxMARL [9] environment.

IPPO is adapted to support parameter sharing across homogeneous agents. This feature is crucial in MARL environments where similar agents perform the same tasks, as it allows for more efficient learning through shared experiences and reduced parameter space. The implementation provides options for both feed-forward and recurrent neural network architectures, enabling it to handle different types of environment dynamics, including those requiring memory of past states. We are going to use the feed-forward version of IPPO.

Also leveraging JAX's capabilities, IPPO can be executed very efficiently, offering substantial speedups in training times compared to traditional approaches, which is critical in complex MARL settings where the environment may involve multiple agents interacting over extended episodes.

The other policy algorithm, MAPPO , extends the PPO framework to multi-agent settings by allowing centralized training but ensures that execution is decentralized. This approach is

beneficial for environments where agents must operate independently while still benefiting from a coordinated strategy learned during training. Like IPPO, MAPPO benefits from JAX's accelerated computation, enabling faster training across multiple agents and scenarios. The use of MAPPO is indicated for more complex MARL environments where agent coordination is crucial, such as those involving strategic interactions or joint task completion.

Both algorithms handle the increased complexity and dimensionality of MARL settings effectively ,as seen from previous experiments , making them suitable for large-scale applications and also can be adjusted to various MARL scenarios, whether fully cooperative, competitive, or mixed, providing flexibility in approach.

The integration of these algorithms into the JaxMARL [9] framework harnesses the full power of JAX for MARL research, offering a combination of high-performance computation and the versatility needed for advanced MARL applications such as "The Mind" .

Architecture

Both Independent Proximal Policy Optimization (IPPO) and Multi-Agent Proximal Policy Optimization (MAPPO) use an Actor-Critic architecture. The actor and critic are typically implemented as two separate neural networks. This separation allows them to focus on distinct tasks and optimize different objectives during training.

The actor network is responsible for selecting actions based on the current policy. It takes as an input the observations and the available actions based on the environment's state, and it outputs a probability distribution over the possible actions. The policy distribution over actions (typically modeled as a categorical distribution).

On the other hand the purpose of the critic network is to evaluate the value of the current state by estimating the expected return (cumulative future rewards).It takes as an input the state of the game and it outputs a scalar value representing the expected return from the current state.

The main difference between IPPO and MAPPO lies in their architectural approach to handling agent interactions and learning within Multi-Agent Reinforcement Learning (MARL) environments. Each addresses the complexity and dynamics of multiple agents in distinct ways, influencing their respective capabilities and applications In IPPO, each agent operates with its own separate instance of the policy network (actor) and potentially its own value network (critic). This setup allows each agent to learn based on its own experiences independently of other agents. MAPPO typically employs a centralized critic that accesses the global state or a joint observation that includes information from all agents . This critic helps in evaluating the global impact of the actions taken by all agents, which facilitates learning coordinated behaviors. While the critic is centralized, each agent still maintains its

own actor for decision-making, allowing for actions that are locally optimized yet informed by a global perspective during training [12] .

To summarize, the Actor Network is responsible for policy learning, determining the action distribution, the critic network is responsible for value estimation, assessing the expected returns from states. The separation of actor and critic networks allows for specialized optimization, leading to potentially better performance and stability in training.

2.6 Q - Learning

On the other hand, the exploration of independent reinforcement learning algorithms, particularly Q-learning, in various multi-agent reinforcement learning environments is not as effective as expected.

Q-learning, when applied independently to each agent in a MARL setup, faces challenges primarily due to the absence of stationarity in the environment. Each agent's learning process treats other agents as part of the environment, which is continually changing as all agents concurrently update their policies. This results in a lack of convergence guarantees because the environment does not adhere to the Markovian property, a fundamental assumption in traditional Q-learning.

The study [5] shows that independent Q-learning can perform comparably to more complex multi-agent algorithms in fully observable, cooperative settings. When the environment is partially observable, like “The Mind”, the performance tends to degrade, but enhancements like adding recurrence (as in Deep Recurrent Q-Networks) can mitigate this issue by helping the algorithm handle the partial observability.

In competitive settings, independent Q-learning can benefit from modifications such as parameter sharing and the addition of agent indicators. These enhancements help the algorithm distinguish between different agents and adapt the learned policies to competitive dynamics more effectively [5].

The most significant challenges for independent Q-learning arise in mixed settings where agents must cooperate with some and compete against others. Here, the algorithms struggle to balance these interactions effectively, often failing to develop cooperative strategies while maintaining competitive capabilities .

Some adaptations for improvement could be parameter sharing among agents, independent Q-learning algorithms can leverage collective experiences to learn more generalizable policies across different agents. This is particularly effective in symmetric games where agents have similar roles or objectives.

As a conclusion, independent Q-learning algorithms face inherent limitations in MARL due to non-stationarity and partial observability, practical enhancements and adaptations can significantly improve their applicability and performance.
The effectiveness of these adaptations, however, varies greatly depending on the specific characteristics of the environment, such as the level of observability and the nature of agent interactions (cooperative, competitive, or mixed).
These findings underscore the importance of choosing and tailoring the right algorithmic strategies based on the specific requirements and dynamics of the multi-agent environments in question.

Future Directions

Research continues into more efficient algorithms, better communication strategies between agents, and approaches to handling larger numbers of agents. Also, understanding how to design reward structures and training environments that encourage effective cooperative strategies is an ongoing area of research.
In summary, cooperative MARL poses unique challenges and offers substantial opportunities for advancing how intelligent systems can operate in complex, multi-agent settings.

Chapter 3

Design & Structure

3.1 Environment Design

The design and structure of the Multi-Agent Reinforcement Learning environment for "The Mind" game was quite complex and intricate, featuring a multitude of aspects typical of Jax MARL environments [10] and systems, especially Hanabi, but with specific modifications tailored to the game's unique mechanics.

The game is designed as a multi-agent environment where each agent (player) interacts with the game state. This involves agents taking turns to play or discard cards, with the potential to request using a shuriken to help the team.

The game state, implemented as a State class, includes elements that are going to be used in the game's logic and observations such as player hands, available shuriken and life tokens, game level, the card on the table and indicators for game status (terminal, level won). The state is vital for tracking the progression of the game and the strategy employed by agents.

The environment implements three discrete actions play a card, pass, or use a shuriken. Although "The Mind" game [1] doesn't support a direct pass action, I decided to implement it in order for the agents to wait for the team-players to play their lowest card , if needed. The observations for each agent, encapsulated within the game dynamics and the rules. Observation space includes the state of the deck, other players' last actions, and the tokens available, allowing strategic decision-making.

When we initialize the game state (reset function) the environment shuffles and deals the cards to each agent according to the current level and initiates the shuriken and life tokens. There are also functions that generate the observation spaces for each agent before proceeding to the gameplay along with the get_legal_moves function, which determines the actions available to each agent based on the current state (e.g., whether they can play a card higher than the last played card without losing a life token).

The gameplay is managed by the step_env and step_agent functions. By calling the step_env function we trigger the current agent to make one of the three available actions. When the agent plays we update the state and the reward based on the action the agent took and then the index moves to the next player.

The rewards depend on the action of each player. If a player passes it earns a negative reward -1 or -0.05 according to the game mode's reward (sparse or dense) , when the player plays the lowest card correctly it gets a reward of +1 and if a shuriken token is asked/played the agents get a smaller positive reward (+0.1) or no reward depending again on the game mode.

The agent architecture in "The Mind" environment, as implemented , is part of a complex MARL system designed to replicate the game's mechanics where players cooperate under conditions of limited information. Analyzing the architecture involves looking at how the state is managed, actions are determined, and observations are handled.

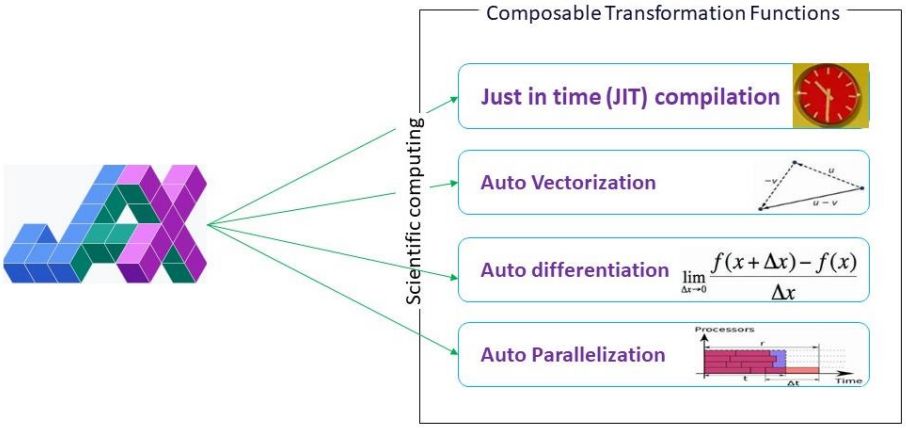
Observations are crucial in MARL, especially under partial observability. Using the get_obs function we generate the observations for each agent. Because agents cannot see their own cards, the observation includes other relevant game state information like the current level, number of shuriken and life tokens left, and the last played card. The observation for each

agent is designed to only include information that the agent is supposed to know, adhering to the game's rules of limited information visibility.

The use of JAX significantly optimizes the computational aspects, allowing for efficient simulation of multiple game instances. This setup is ideal for exploring sophisticated cooperative strategies and learning dynamics in MARL. This design enables not just playing the game according to its rules but also exploring various strategies that agents can learn to optimize their play, potentially extending beyond simple rule-based strategies to more complex, learned behaviors that effectively utilize the cooperative and strategic aspects of "The Mind."

3.2 JAX Library

The use of JAX for JIT compilation and vectorized operations (jax.vmap, jax.jit) in my environment, suggests optimization for performance [13] . These are used in functions that require large computational power to efficiently compute outputs across all agents. Implementing "The Mind" as a MARL environment using JAX can offer several significant



benefits, particularly due to JAX's unique features and capabilities. Here's how using JAX can be advantageous as JAX provides powerful tools like vmap (vectorized map) and jit (Just-In-Time compilation) that can significantly speed up computations. Vmap can automatically batch computations without needing manual batching, which is especially beneficial when we need to process actions or evaluations for multiple agents simultaneously. The jit compiler can optimize and compile parts of the code to machine language, speeding up execution dramatically. Also JAX's automatic differentiation capability, provided through autograd, simplifies the implementation of learning algorithms that require gradient computation. This is particularly beneficial for training reinforcement learning models where gradient-based optimization is used. JAX is also designed to be replacement for NumPy but with added capabilities like GPU support and automatic differentiation.

This compatibility means that we can use familiar NumPy-like syntax and functions, which makes coding more straightforward.

Another benefit that JAX provides is that it can automatically take advantage of GPU hardware accelerations, which is invaluable for running large-scale simulations or training complex models in “The Mind” environment. This feature allows for handling more significant amounts of data and computations faster, which is crucial for real-time learning and decision-making in multi-agent settings.

In terms of Algorithm Implementation, with JAX, we can implement various reinforcement learning algorithms more efficiently, ensuring that the computational burden is manageable even as the complexity of the environment grows.

Using JAX for implementing a MARL version of "The Mind" thus aligns well with the need for high-performance computation, scalability, and the sophisticated data handling required for effective reinforcement learning. This choice can enhance your development process, allowing you to focus more on strategic implementation rather than computational challenges.

All these not only simplify the code by removing explicit loops but also enhance performance by leveraging JAX’s underlying optimizations for batched operations.

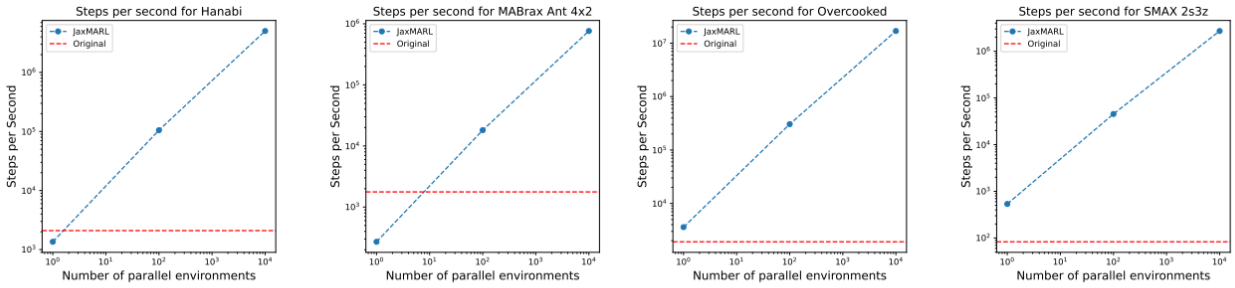
JAX encourages a functional programming style, which fits well with the requirements of simulation environments where immutable states are preferable. This functional approach helps in managing state more predictably and avoiding side effects, which can make the system easier to debug, test, and scale.

JAX excels at automatic differentiation, offering grad, vjp, and jvp functions that are essential for implementing learning algorithms, especially those involving gradient descent [6] .

JAX is designed to run seamlessly across different devices, including CPUs, GPUs, and TPUs. This cross-platform capability ensures that environments can scale across hardware resources efficiently, allowing for faster training and simulation without major code changes.

In summary, JAX provides a robust framework for building efficient and scalable simulation environments for MARL. Its strengths in performance optimization, functional programming, and easy scalability align well with the needs of complex multi-agent systems like the one implemented for "The Mind" game.

As seen from the environments implemented in JaxMARL library JAX provides significant computational advantages in handling multiple parallel environments, which is a common scenario in the training and evaluation phases of multi-agent systems. The use of JAX in these environments enables more efficient experimentation and faster iteration times, which are crucial for advancing research and development in complex MARL scenarios.



(a) Hanabi **(b) MABrax Ant** **(c) Overcooked** **(d) Starcraft 2s3z**

Comparison of the environments implemented in the Jax MARL library with their original implementation.

3.3 Test & Validation

Evaluating the validity of "The Mind" MARL environment is crucial to ensure it accurately reflects the game's mechanics and provides a robust platform for training and testing algorithms. By creating unit tests for individual components of the environment to ensure they function as expected including state and action Spaces to ensure states and actions are correctly defined, mapped, and accessible to agents.

Also to verify that rewards are assigned correctly for various actions and outcomes, ensuring they reflect the game's rules and check that the transition functions work correctly, considering actions and leading to appropriate next states.

To test the game mechanics, we manually play through the environment or step through individual transitions to confirm rules are applied correctly. Test edge cases, such as scenarios with all agents passing or incorrect card plays, to see how the environment responds and if it behaves as intended.

To observe and evaluate how agents behave in the environment over time we can track the learning curves of various algorithms to see if they converge and stabilize similarly to expected patterns, analyze how strategies evolve during training, such as turn-taking, decision-making sequences, and response to partial observability, to see if they align with expected behaviors.

By using statistical methods to evaluate the consistency and reliability of the environment we repeat tests with the same algorithm multiple times to see if results are consistent, indicating stability in the environment and compare the results of different algorithms in the environment to expected benchmarks using correlation or regression analysis to assess their validity.

The environment was built in the structure of the JaxMARL environments [6] to be compatible to a variety of algorithms, from Independent Q-Learning to MAPPO that the library of jaxmarl supports and be able to compare the performance of these algorithms to see if the environment produces consistent and reliable results across different approaches.

By incorporating these methods, we can thoroughly evaluate the validity of the environment, ensuring it faithfully represents the game's mechanics and provides a reliable platform for testing algorithms.

Chapter 4

Results & Discussion

4.1 Experiment Setup

As mentioned before , the environment’s game rules as well as agents' states include their remaining hand, cards in play, and any shared information about the current state of the game. Actions consist of playing a card from hand, hinting at another player's action, or passing. Rewards are assigned based on how closely the group’s actions align with the intended sequence, with positive rewards for correct plays and negative rewards for mistakes or premature card plays.

The algorithms I evaluated my environment with are Proximal Policy Optimization algorithms , specifically IPPO and MAPPO. Each agent in Independent PPO (IPPO) follows a PPO-based policy, independently optimized using its own experiences. In MAPPO agents are trained with centralized policy optimization but execute independently, enabling them to benefit from coordinated strategies.

Both algorithms are going to be executed using modified scripts provided by the JaxMARL library [9]. The experiments will take place on various GPUs on Google Collab notebook. The results will appear as plots including the Return on the y-axis and the updated steps on the x-axis.

The X-axis on the graph representing the IPPO algorithm's training performance is labeled "Update Step." This axis measures the number of policy updates that have been performed during the training process, axis’ value increases by 1 for each policy update performed. For example, if the agent performs 10,000 interactions with the environment (timesteps), these are

broken down into smaller segments defined by NUM_STEPS. If NUM_STEPS is set to 128, the number of update steps would be the total timesteps divided by NUM_STEPS. If NUM_ENVS (number of parallel environments) is 4, this means experiences are collected from 4 environments simultaneously.

$$\text{Number of Update Steps} = \frac{\text{TOTAL\_TIMESTEPS}}{\text{NUM\_ENVS} \times \text{NUM\_STEPS}}$$

The y-axis in the provided graphs represents the "Return," which refers to the cumulative reward that agents accumulate over an episode. In the context of this research, it reflects the effectiveness of the agents' policies in achieving their goals within the "The Mind" environment.

4.2 Results

In this chapter, we present the results of applying the Independent Proximal Policy Optimization (IPPO) and Multi-Agent Proximal Policy Optimization (MAPPO) algorithms to the multi-agent environment, "The Mind". The performance of the agents trained using IPPO and MAPPO is evaluated through the analysis of return curves, which depict the cumulative rewards obtained by the agents over the training period.

The experiments were conducted using the following configuration parameters:
Learning Rate (LR) ,
Number of Environments (NUM_ENVS) ,
Number of Steps (NUM_STEPS) ,
Total Timesteps (TOTAL_TIMESTEPS),
Update Epochs (UPDATE_EPOCHS) ,
Number of Minibatches (NUM_MINIBATCHES) ,
Discount Factor (GAMMA) ,
GAE Lambda (GAE_LAMBDA),
Clipping Epsilon (CLIP_EPS) ,
Entropy Coefficient (ENT_COEF),
Value Function Coefficient (VF_COEF) ,
Max Gradient Norm (MAX_GRAD_NORM)
Number of Seeds (NUM_SEEDS) ,
Random Seed (SEED)
For each run we modify the TOTAL_TIMESTEPS, NUM_STEPS and NUM_ENVS parameters.

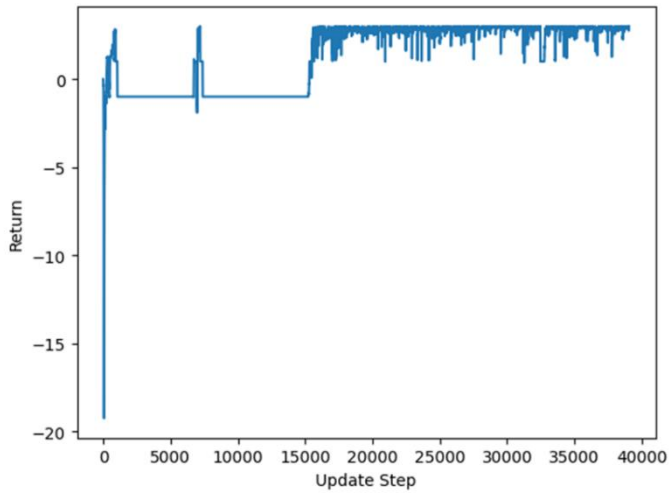
The parameters `TOTAL_TIMESTEPS`, `NUM_ENVS`, and `NUM_STEPS` significantly influence the performance and efficiency of the algorithms . The total number of timesteps represents the interactions with the environment that the algorithm will perform during training. The higher the `TOTAL_TIMESTEPS`, the longer the training duration. More timesteps allow the agent to gather more experience, which can lead to better learning and policy optimization. Sufficiently high `TOTAL_TIMESTEPS` can help ensure that the agent has enough time to explore the environment and converge to an optimal or near-optimal policy. However, there is a point of diminishing returns where additional timesteps may not lead to significant performance improvements.

The variable NUM_ENVS corresponds to the number of parallel environments in which the agent interacts simultaneously. Using multiple environments in parallel can significantly increase the sample efficiency. The agent collects experiences from several environments at the same time, leading to faster accumulation of training data. Training with multiple environments can stabilize learning by providing diverse experiences, which helps in generalizing the policy better across different scenarios. Also the parameter `NUM_ENVS` directly influences the batch size for each update. A larger number of environments results in larger batches of experience data, which can improve gradient estimates during optimization. However, it also requires sufficient memory and processing power to manage multiple environments concurrently.

The number of steps the agent takes in each environment before performing a policy update, `NUM_STEPS` determines the length of the trajectories used for each update. Longer trajectories (higher `NUM_STEPS`) provide more temporal context, which can be beneficial for learning in environments with long-term dependencies. A higher `NUM_STEPS` means that updates occur less frequently but with more data per update. This can improve the stability of the policy updates and lead to better performance. Shorter trajectories (lower `NUM_STEPS`) lead to more frequent updates with less data, which might increase the variance in gradient estimates.

IPPO – Results

Results for 5e1 timesteps , 1024 environments and 128 steps

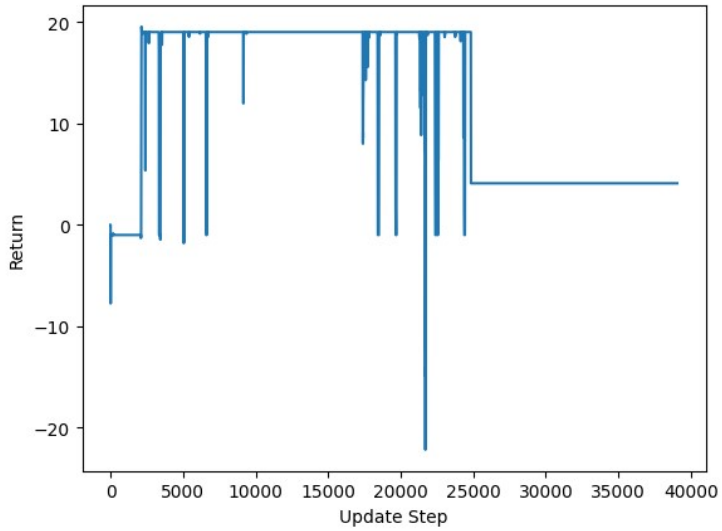


The graph depicts the return over update steps for the IPPO algorithm training session in the environment. The return starts at a low value and even goes negative at the very beginning, indicating that the agents are exploring and initially making suboptimal decisions. The sharp drop to around -20 at the very beginning suggests that the agents might have taken actions leading to significant negative rewards like pass or playing the wrong card.

Between steps 0 - 5000 the returns show high variance with significant spikes and drops. This phase reflects the exploration phase where the agents are trying different actions to understand the environment. The returns occasionally reach positive values, indicating that the agents are learning and sometimes making good decisions.

During steps 5000 - 15000 there is a noticeable improvement in the returns, with the values stabilizing around 0. This indicates that the agents have started to learn a more stable policy but are still not consistently achieving high rewards. The flat line around 0 suggests that the learning might have plateaued temporarily, possibly due to the agents exploring a stable but suboptimal policy. At the final steps ,15000 - 40000, returns start to consistently stay in the positive range, with some fluctuations. This indicates that the agents have learned to perform better in the environment. The fluctuations in returns during this phase are smaller compared to the early training phase, showing more stable performance with occasional dips, possibly due to exploration or encountering challenging scenarios. After around 15000 update steps, the returns stabilize in the positive range, indicating that the agents have converged to a reasonably good policy. The small fluctuations towards the end are typical in reinforcement learning, where the agents continue to explore slightly to avoid local optima.

Results for 5e1 timesteps , 128 environments , 16 steps



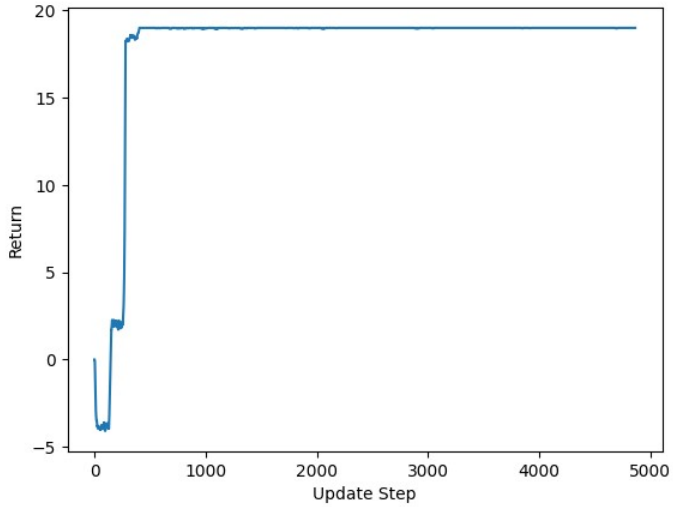
In the initial phase (steps 0 - 2000) , the return starts with high variance, fluctuating significantly. This is typical of the exploration phase where the agents are trying different actions to understand the environment.

There are some sharp drops indicating that the agents are sometimes making suboptimal decisions that result in negative rewards. In middle Training Phase (2000 - 15000 steps) the returns stabilize around a high value close to 20, which suggests that the agents have learned a reasonably good policy. However, there are occasional sharp drops, indicating that the agents sometimes take actions leading to significantly lower rewards. This could be due to continued exploration or encountering challenging scenarios in the environment. At steps 15000 - 25000 , the returns continue to fluctuate with more frequent and significant drops, even reaching negative values. This suggests that while the agents have learned a good policy, they are still exploring and sometimes making poor decisions.

The period around 25000 update steps shows a significant drop to a negative return, which might indicate a temporary failure in the policy or an exploration into less optimal strategies.

In the end Phase we see a stabilization around a value slightly above 0, indicating that the agents have settled into a stable policy but one that is not as optimal as in the middle training phase. The fluctuations reduce, showing that the policy is stable but could still benefit from further refinement. The middle phase shows a period of high returns, indicating effective learning and a stable policy, however, the latter phase's stabilization at a lower return suggests that the agents might have converged to a suboptimal policy. The overall trend shows that the agents can achieve high returns but struggle with maintaining it consistently.

Results for 5e1 timesteps , 128 environments and 2 steps

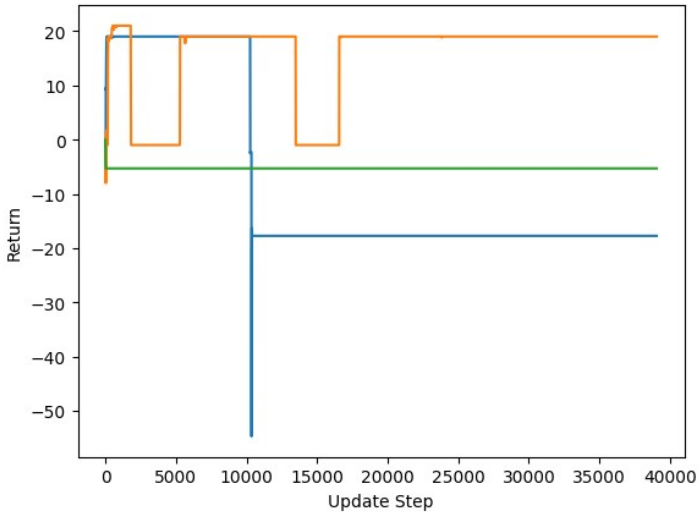


In the first steps the return starts with negative values, indicating poor performance by the agents initially but there is a sharp increase in return within the first 500 update steps, quickly reaching close to the maximum return of 20. This indicates that the agents rapidly learn an effective policy. After the sharp increase, the return stabilizes around 20 and maintains this level throughout the remaining update steps. The stability indicates that the agents have learned a robust policy and consistently achieve high returns.

The agents show a rapid increase in performance within the initial phase, suggesting that the learning algorithm and network architecture can effectively capture some dynamics of the environment. The maximum return of around 20 suggests that the agents have learned an effective policy, potentially near the optimal for the given state of the environment.

Extending the training duration beyond 5000 update steps might reveal whether the performance remains stable over a longer period or if further improvements are possible.

Results for 5e1 timesteps , 128 environments and 16 steps , for 3 seeds



Orange Line (Seed 0):

The agent quickly learns an effective policy, reaching a high return of around 20 early in the training process. The performance stabilizes, showing consistent high returns with minor fluctuations. This suggests the agent has successfully learned a robust strategy for the environment.

Green Line (Seed 1):

The agent demonstrates a more gradual increase in returns, reaching a plateau around a return of 0. The stability of the green line indicates that the agent may be stuck in a local minimum or is unable to explore more effective strategies.

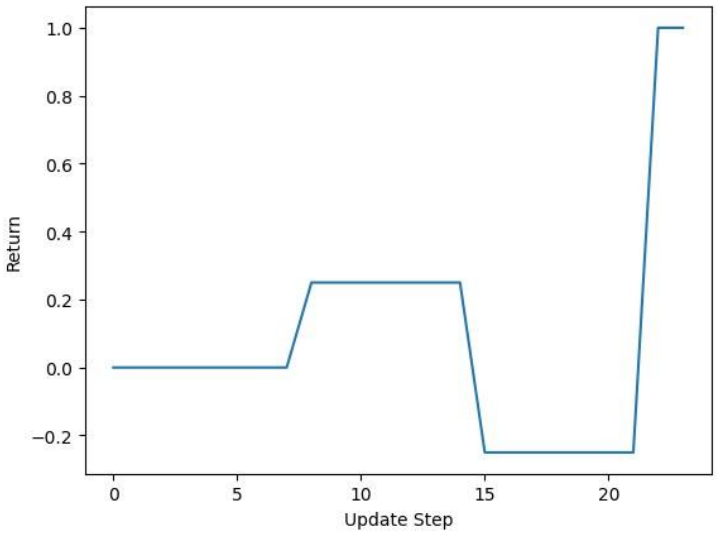
Blue Line (Seed 2):

The agent initially shows some improvement but experiences a significant drop in returns, stabilizing at a negative return of around -20. The instability and negative returns suggest that the agent might be learning a suboptimal policy or is exploring ineffective actions that degrade performance.

The orange line's stability indicates a well-learned policy with minor exploration leading to high and consistent returns. In contrast, the blue line's sharp drops highlight episodes of poor performance, suggesting the agent is unable to maintain a stable policy. The green line's flat trajectory implies little to no improvement, possibly due to insufficient exploration or being stuck in a suboptimal policy.

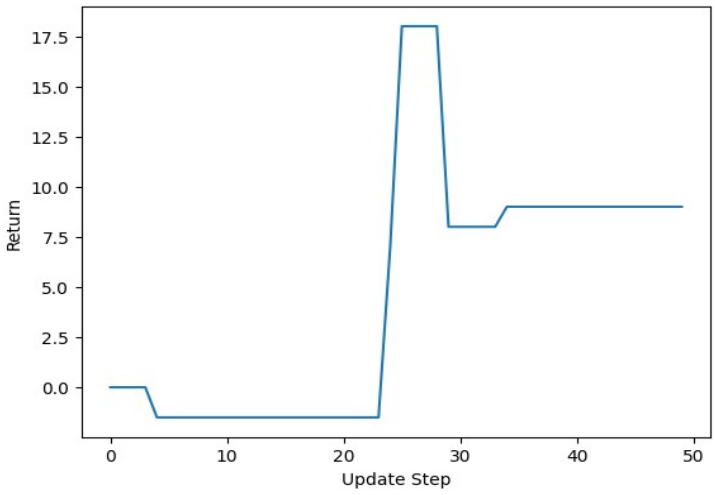
MAPPO - Results

Results for 10e1 timesteps , 128 environments and 16 steps



As observed, the algorithm starts with returns that are marginally positive, suggesting that initial policies, while not highly effective, avoid detrimental outcomes. This phase likely reflects the algorithm's exploitation of basic strategies that provide consistent but limited rewards. The noticeable dip in performance at the midpoint of the training suggests a period of exploration. Here, the algorithm may be testing new strategies that diverge from previously learned behaviors, resulting in temporary performance setbacks.: The dramatic rise in returns towards the end of the graph indicates the discovery of highly effective strategies. This could be attributed to the algorithm optimizing its policy selection based on accumulated experiences and overcoming the exploration penalties observed earlier.

Results for 5e1 timesteps , 128 environments and 2 steps

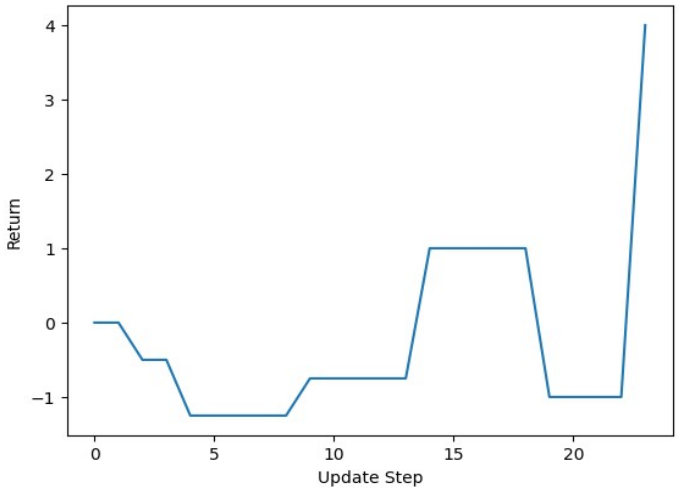


The return starts near zero and remains flat initially. This phase likely corresponds to the initial exploration period where the agents are exploring the environment and learning the basic dynamics of the game without any significant improvement in performance. Around step 10, there is a significant drop in the return, followed by a rapid increase reaching a peak return of approximately 18 at step 30. This sharp increase indicates that the agents have discovered a highly rewarding strategy or set of actions that significantly improves their performance. After reaching the peak, the return drops slightly but stabilizes around a return of 10. This suggests that the agents are refining their strategies and converging to a stable policy. The slight drop could be due to continued exploration or fine-tuning of the policy.

The flat return in the initial steps reflects the agents' exploration and lack of effective strategies. This is common in reinforcement learning where agents start with little knowledge of the environment. The sharp increase in return signifies that the agents have found effective strategies that yield high rewards. This phase shows the capability of the MAPPO algorithm to

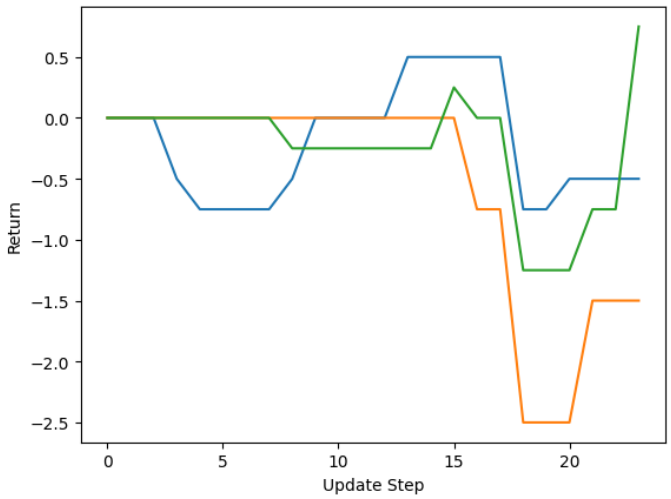
leverage exploration to discover rewarding actions. The stabilization phase indicates that the agents are converging to a stable policy. The slight fluctuations around the stable return value suggest that the agents are still exploring but have largely settled on a successful strategy.

Results for 10e1 timesteps , 4 environments and 2 steps



In this graph, we observe the return over 25 update steps. The graph starts with returns below zero, indicating that the agents are performing poorly initially. The returns fluctuate between negative and positive values, indicating instability in the learned policies. Towards the end of the update steps, there is a sharp increase in the return, reaching around. This suggests that the agents have started to learn more effective strategies, although the learning process is still unstable. The learning process appears more unstable, with returns fluctuating significantly before achieving a positive return towards the end. This suggests variability in the learning process, which could be due to factors such as randomness in the environment, the complexity of the task, or variability in initial conditions. The fluctuations observed in the graph highlight the challenges in achieving consistent performance. It suggests that while the MAPPO algorithm can find effective strategies, the path to stable learning might require more time or adjustments to hyperparameters.

Results for 5e1 timesteps , 128 environments and 2 steps , for 3 seeds



The graph demonstrates considerable variance in the performance of the MAPPO algorithm across different seeds. This variability underscores the importance of multiple runs to evaluate the robustness and stability of the learning algorithm. At the beginning of the training (0-5 update steps), all three seeds show returns close to zero, indicating that the agents have not yet learned effective strategies and are performing at or slightly below the baseline. Between 5-15 update steps, we observe divergence in the performance across the three seeds, the blue and green seeds show a drop in performance, with returns dipping to negative values around -1 to -2 but the orange seed maintains a return close to zero during this period. Around 15 update steps, the green seed experiences a sharp increase in return, briefly exceeding 0.5 before dropping again. The blue seed also shows an improvement, though it remains below zero. The orange seed exhibits a steep decline, reaching a minimum return of approximately -2.5.After 20 update steps, performance for the seeds becomes more distinct, the green seed's return increases significantly, reaching positive values around 0.5 to 1. The blue seed's return improves but remains below zero and the orange seed shows some recovery but continues to exhibit significant fluctuations.

The green seed shows a more stable learning curve in the later stages compared to the blue and orange seeds, suggesting that, in this instance, the initial random seed significantly affected the learning trajectory. The fluctuations in returns, particularly for the orange seed, suggest that the agents might be exploring different strategies or encountering instabilities in policy updates.

4.3 Discussion

The results indicate that IPPO and MAPPO are well-suited to "The Mind" environment, effectively handling the cooperative dynamics and limited communication. MAPPO's centralized training enhances coordination, contributing to its superior performance.

The strategies developed by IPPO and MAPPO can mirror human gameplay in a certain level , with agents showing a natural turn-taking pattern and considering the game's state when making decisions. The algorithms seem to adapted well to the unique rules of "The Mind," developing coherent strategies despite the lack of direct communication.

Algorithm Performance in Cooperative Games suggests that on-policy algorithms with centralized training can handle cooperative scenarios effectively. This finding reinforces the importance of centralized coordination mechanisms in complex MARL environments [6] .

The results demonstrate that IPPO is capable of learning effective policies for the multi-agent environment of "The Mind". The algorithm shows strong convergence properties, as indicated by the stabilization of returns at high levels in the latter phases of training. However, there are fluctuations and occasional drops in performance, which suggest that while the agents are generally effective, there are still opportunities for improvement in terms of stability and consistency.

Some potential improvements that can benefit the algorithms are extending the training duration beyond the current limits might help in achieving more consistent performance and further refining the learned policies, fine-tuning hyperparameters such as the learning rate, discount factor, and clipping epsilon could enhance the algorithm's stability and performance.

Also by refining the reward structure to provide more informative feedback can guide the agents towards more stable and optimal behavior. Implementing advanced exploration strategies, such as epsilon-greedy or entropy-based methods, might help in reducing the initial variability and speeding up the convergence process.

The results presented in the previous chapters indicate significant learning dynamics that characterize the performance of MAPPO in "The Mind". Notably, the agents demonstrated a capability to recover from suboptimal policies and adapt strategies that significantly improved their performance, as seen in the sharp increase in returns towards the end of the training cycles.

These findings align with the established theories in reinforcement learning that emphasize the importance of exploration for achieving optimal long-term rewards. The observed fluctuations in performance reflect a sophisticated balance between exploration and exploitation, a core challenge in MARL environments. This behavior mirrors findings from similar studies [6] , which demonstrated that agents often experience initial declines in performance as they explore new strategies before stabilization occurs upon learning optimal actions. The study contributes

to the understanding of how MARL algorithms like MAPPO can be optimized to manage complex, cooperative interactions among agents in a semi-competitive environment. The capability of MAPPO to dynamically adjust agent policies in response to the evolving game dynamics offers a robust example of adaptive learning in multi-agent systems.

The strategic implications of this research are profound, particularly in the design of reward structures and policy updates that encourage beneficial exploration. Implementing techniques such as reward shaping or dynamic policy adjustment could further enhance the learning efficiency in similar MARL setups. For practitioners, this research suggests the importance of carefully designing the learning environment and the reward mechanisms to foster effective agent collaboration and to mitigate conflicts that may arise from competitive behaviors.

This chapter has demonstrated that IPPO and MAPPO, particularly MAPPO, are effective in handling "The Mind" as a MARL environment. The results highlight the importance of centralized training and nuanced coordination strategies in cooperative games and provide insights into potential real-world applications for MARL.

Chapter 5

Conclusion & Future Improvements

5.1 Conclusion

The results have shown changes in execution, versatility, and productivity across the calculations, outlining the significant effect of algorithmic plan on the way of behaving and progress of agents inside a perplexing MARL structure.

To sum up the critical discoveries of the review, ponder the ramifications of these outcomes for the more extensive field of MARL, and propose bearings for future exploration that could expand the comprehension and use of MARL in helpful conditions.

This thesis has explored the application of Multi-Agent Proximal Policy Optimization (MAPPO) and Independent Proximal Policy Optimization (IPPO) within the MARL framework tailored for "The Mind" game. The findings demonstrate that both algorithms, particularly MAPPO, are well-suited for managing the cooperative dynamics and communication constraints inherent in "The Mind." Notably, MAPPO's centralized training mechanism facilitated superior coordination among agents.

Both IPPO and MAPPO have shown the capability to develop strategies that mirror human-like behaviors such as turn-taking and state-aware decision-making, despite the absence of direct communication. This capability underscores the potential of advanced MARL algorithms to approximate complex human interactions within a game setting.

The research highlighted the robust adaptability of MAPPO and IPPO to the unique rules and cooperative nature of "The Mind.". The dynamic adjustment of policies by the algorithms, supported by empirical results, illustrated a sophisticated balance between exploration and exploitation—a critical aspect in achieving optimal long-term rewards in MARL environments.

This study contributes to the broader discourse on the efficacy of on-policy algorithms with centralized training in cooperative multi-agent settings. It provides empirical evidence supporting the theory that such training frameworks can enhance agent coordination and learning outcomes.

The findings from this thesis offer valuable insights for practitioners and researchers in designing MARL systems for similar complex environments. The demonstrated importance of reward structures and policy updates suggests approaches for enhancing agent collaboration in various applications, from gaming to autonomous systems and beyond.

In conclusion, this thesis has shown that IPPO and MAPPO are effective tools for navigating the complexities of cooperative MARL environments. This research not only enhances our understanding of multi-agent learning dynamics but also opens avenues for future exploration into more sophisticated and scalable MARL applications.

5.2 Limitations

In the context of the research on "The Mind" within a cooperative MARL environment, several limitations may have arisen. Addressing these limitations not only provides transparency but also paves the way for future research.

One of the primary limitations that could have emerged from this research is the scalability of the algorithms. While MAPPO and IPPO performed well within the constrained setup of "The Mind" game with a limited number of agents, as the number of agents increases, the complexity of the environment and the interactions among agents can escalate dramatically, potentially

leading to issues like non-converging policies or increased computational demands. The scalability of these algorithms to other cooperative games or real-world scenarios that involve more dynamic interactions or larger state spaces remains a question.

The algorithms were tested in a very specific setting with predefined rules and structured interactions. The ability of IPPO and MAPPO to generalize to environments that have different rules, objectives, or agent dynamics wasn't thoroughly tested. This limitation could affect the applicability of the findings to other real-world applications or different types of games. The performance of these algorithms in environments where agent roles or game dynamics change unpredictably is uncertain.

Given that "The Mind" inherently involves limited communication among players, the algorithms were tailored to operate under these constraints, how these algorithms would perform in environments where agents can communicate more explicitly or where communication is a variable factor wasn't addressed. The reliance on developing implicit coordination strategies could be a limitation in scenarios where explicit coordination is necessary or more efficient.

The effectiveness of these algorithms is heavily dependent on the design of the reward structure. If the reward signals are not adequately informative or aligned with the desired outcomes, the learning process can be misguided, leading to suboptimal policies.

5.3 Future Work

In the aspect of algorithmic enhancements, further research could explore algorithms or modifications that more effectively handle the non-stationarity of MARL environments, particularly for Independent Q-Learning or similar approaches, investigate how communication strategies, such as message-passing or signaling protocols, can enhance cooperation in MARL environments, particularly for algorithms like IPPO and MAPPO.

Expanding "The Mind" environment to include more complex game dynamics, such as multi-round scenarios, could provide a more rigorous testbed for MARL algorithms. Exploring other cooperative games as MARL environments can help diversify benchmarks, allowing for broader evaluations and comparisons of algorithms.

Future studies could also delve deeper into comparing AI strategies with human gameplay, examining how closely they align and what insights can be drawn from the differences and analyze how algorithms learn and adapt over time, including how they form strategies, handle new scenarios, and improve coordination skills.

In summary, this research presents a new MARL environment based on "The Mind," successfully implements and evaluates multiple algorithms within it, and draws insights into

the dynamics of cooperative learning. The findings offer significant contributions to MARL research, particularly in cooperative game settings, and suggest numerous avenues for future exploration to further advance the field.

References

[1] N. T, "The Mind Review," Board Game Review, Apr. 25, 2020. https://boardgamereview.co.uk/game-reviews/the-mind-review/ (accessed Oct. 20 , 2023).

[2] I. Mordatch and P. Abbeel, "Emergence of Grounded Compositional Language in Multi-Agent Populations," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, Apr. 2018, doi: https://doi.org/10.1609/aaai.v32i1.11492.

[3] Ng, Andrew Y., Daishi Harada, and Stuart Russell. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping." *ICML. Vol. 99. 1999.*

[4] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature, vol. 575, no. 7782, pp. 350–354, Oct. 2019, doi: https://doi.org/10.1038/s41586-019-1724-z.

[5] K. M. Lee, S. G. Subramanian, and M. Crowley, "Investigation of Independent Reinforcement Learning Algorithms in Multi-Agent Environments," arXiv (Cornell University), Nov. 2021, doi: https://doi.org/10.48550/arxiv.2111.01100.

[6] C. Yu , A. Velu , E. Vinitsky, J. Gao, Y. Wang., A. Bayern and Y. Wu "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," arXiv.org, Nov. 04, 2022. https://arxiv.org/abs/2103.01955

[7] R. Lowe , Y. Wu , A. Tamar, J. Harb, P. Abbeel and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments." Available: https://arxiv.org/pdf/1706.02275

[8] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. Francis Song, E. Parisotto,. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G.

Bellemare, and M. Bowling, “The Hanabi challenge: A new frontier for AI research,” Artificial Intelligence, vol. 280, Art. Number 103216, 2020.

[9] A. Rutherford. B. Ellis, M. Gallict, J. Cook, A. Lupu , G. Ingvarsson , T. Willi , A. Khan, C. S. de Witt, A. Souly , S. Bandyopadhyay, M. Samvelyan, M. Jiang, R. T. Lange, S. Whiteson , B. Lacerda , N. Hawes, T. Rocktaschel , C. Lu and J. N. Foerster “JaxMARL: Multi-Agent RL Environments and Algorithms in JAX.” Accessed: May 10, 2024. [Online]. Available: https://arxiv.org/pdf/2311.10090

[10] “FLAIROx/JaxMARL,” GitHub, Jan. 21, 2024. https://github.com/FLAIROx/JaxMARL (accessed Jan. 5, 2024).

[11] N. Rabinowitz , F. Perbet , H. F. Song, C. Zhang , S. M. Ali Eslami and M. Botvnick “Machine Theory of Mind.” Accessed: May 15, 2024. [Online]. Available: https://arxiv.org/pdf/1802.07740

[12] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, P. Abbeel, and I. Mordatch. "Learning with Opponent-Learning Awareness". arXiv:1709.04326, 2018

[13] C. Schroeder de Witt, T. Gupta, D. Makoviichuk, V.Makoviichuk, P. H.S. Torr, M. Sun, S. Whiteson, “Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?” https://arxiv.org/pdf/2011.09533

[14] “JAX reference documentation — JAX doumentation,” jax.readthedocs.io. https://jax.readthedocs.io/en/latest/

Appendix A

```python
def _play_lower_fn(state, aidx, reward):
    """Discard the lowest value card from the player's hand."""
    # Assuming player_hands is a 2D array where each row represents a player's hand

    # Extract the specific player's hand
    player_hand = state.player_hands[aidx]

    # Assuming the hand is sorted or that the first card is the lowest
    played_card = player_hand[0]  # The lowest value card
    shown_cards = state.shown_cards.at[aidx].set(played_card)

    # Remove card from hand
    updated_hand = player_hand.at[0].set(101)
    updated_hand = jnp.roll(updated_hand, -1)

    # Check if the discarded card is valid based on game logic
    last_played = state.card_down  # The last card played in the game
    is_valid_play = jnp.logical_and(jnp.less(played_card, 101), jnp.greater(played_card, last_played))

    # Losing a life decreases the reward by 1
    def not_valid(_):
        first_zero_idx = jnp.argmin(state.life_tokens)
        # If no "0" found (all tokens are "1"), then set last_one_idx to the last index
        last_one_idx = jnp.where(first_zero_idx > 0, first_zero_idx - 1, len(state.life_tokens) - 1)

        updated_life_tokens = state.life_tokens.at[last_one_idx].set(0)

        # Update reward
        updated_reward = reward - 1
        # print("True")
        return updated_life_tokens, updated_reward

    def valid(_):
        # print("False")
        # No change to life tokens if play is valid
        life_tokens = state.life_tokens
        updated_reward = reward + 1
        return life_tokens, updated_reward
```

This function is designed to handle the gameplay logic where a player, represented by a specific index (`aidx`), attempts to play the lowest card in their hand. The function evaluates whether this action is valid based on the game's rules and updates the game state accordingly. Here's a more detailed breakdown of the process and functionality depicted in the code snippet:

1. Extract Player Hand: The function begins by extracting the hand of the player specified by `aidx` from the state, assuming that the hand is sorted or that the lowest card is the first in the array.

2. Determine Played Card: It identifies the lowest card in the player's hand as the card to be played.

3. Update Shown Cards: The function updates the `shown_cards` state to include the card just played, indicating that this card is now visible or known to other players.

4. Remove Card from Hand: It removes the played card from the player's hand and shifts the remaining cards left to fill the gap.

5. Validate Play: The function checks if the played card is valid by comparing it to the last card played (`last_played`). A play is considered valid if the card played is greater than the last card played but still less than a specified threshold (101 in this case).

6. Update Life Tokens and Reward: If the play is not valid, the player loses a life token, and the reward is decreased by one. If the play is valid, the reward is incremented by one, encouraging the agent to continue making valid plays.

7. Return Updated State: Depending on the validity of the play, it returns the updated life tokens and the reward.

This function is crucial for teaching the agent the consequences of its actions, both positive and negative, helping it learn the optimal strategy for playing its cards in the game "The Mind". The structured reward system (reward of +1 for a correct play and a penalty of -1 for an incorrect one) directly incentivizes the agent to align its actions with the game rules and objectives, thereby maximizing collective outcomes in a cooperative game environment.

Appendix B

```python
# THE MIND
# Collect trajectory
state_seq = []

for _ in range(30):
    state_seq.append(state)
    # Iterate random keys and sample actions
    key, key_s, key_a = jax.random.split(key, 3)
    key_a = jax.random.split(key_a, env.num_agents)
    actions = {agent: env.action_space(agent).sample(key_a[i]) for i, agent in enumerate(env.agents)}

    print("------------------------------------------------------------------------")
    print("Actions",actions)
    print("Lives before", state.life_tokens)
    print("Hands:",state.player_hands)
    print("Card down:",state.card_down)
    print("Level: ",state.cur_level)
    print("Agent", state.cur_player_idx)
    print("Terminal ",state.terminal)

    # Step environment
    obs, state, rewards, dones, info = env.step_env(key_s, state, actions)

    print("State ", state)
    print("------------------------------------------------------------------------")
    print("Level after",state.cur_level)
    print("Hands:",state.player_hands)
    print("Agent 2", state.cur_player_idx)
    print("Rewards:",rewards)
    print("Obs",obs)
    print("Card Down",state.card_down)
    print("Out",state.out_of_lives)
    print("Last action",state.last_action)
    print("Lives", state.life_tokens)
    print("Shuriken Left", state.shuriken_tokens)
    print("Shown Cards", state.shown_cards)
    print("Won ? ",state.level_won)
    print("Terminal ",state.terminal)
    print("Dones",dones)
    print("Hands size",env.hand_size)
```

The script is designed to simulate gameplay in "The Mind" environment, testing its integrity and the interaction of agents within this setup. It executes multiple game steps while collecting and printing state information to ensure that the game logic and agent behaviors align with expected outcomes.

A loop runs for a fixed number of iterations (30 in this case), where each iteration represents a step in the environment.

At each iteration, the current state is appended to `state_seq`.

The script uses JAX for random number generation and action sampling. `key, key_s, key_a` are split keys used for generating random numbers, ensuring reproducibility and independent random samples for different parts of the code.

Actions for each agent are sampled according to the environment's action space, ensuring that actions are valid and randomly chosen based on the current policy or action distribution.

Before the environment steps forward, the script prints out the current state of various game components such as lives, the current level, the last card played, and whether the state is terminal (end of the game).

The environment progresses using the `env.step` method, which receives the current state and chosen actions. This method returns observations, updated state, rewards, completion flags (`dones`), and additional info.

This step is where the game logic is applied, and the environment calculates the new state based on the actions taken by the agents.

After stepping through the environment, the script prints detailed information about the new state, including any changes in the level, player index, rewards received, and observations of the environment state.

This script is valuable for testing and verifying the game's mechanics in "The Mind" MARL environment by ensuring the environment behaves as expected without any logical errors or inconsistencies and also assessing whether agents' actions based on the current policy lead to reasonable outcomes.