Multiagent Reinforcement Learning in Competitive Environments

Athith Gobinathan
Department of Computer Science
Columbia University
New York, NY 10027
asg2278@columbia.edu

Mike Qu
Department of Computer Science
Columbia University
New York, NY 10027
zq2234@columbia.edu

Acknowledgments

The code for all experiments and environments described in this paper is available at `https://github.com/mikequ1/MultiagentRL`.

1 Introduction

In many real-world scenarios, decision-making involves interactions among various self-interested agents, each of whom may have competing or cooperating interests. Although game theory has long offered a robust framework to analyze these interactions through equilibrium strategies, a comprehensive and accurate understanding of the underlying environment model is required to generate these insights, which could be infeasible to obtain in complex or partially observable domains. In contrast, Multi-agent Reinforcement Learning (MARL) provides a powerful alternative by allowing agents to learn the environment through exploration and exploitation. Through this process, agents may be incentivized to cooperate, compete, or pursue a mixture of both, allowing MARL to model a wide-range of complex behaviors and phenomena.

1.1 Problem Formulation

In this project, we investigate the performance of four widely used multi-agent reinforcement learning (MARL) algorithms—CDQN, IDQN, MAPPO, and MADDPG—across two contrasting environments: a competitive two-agent economic game (the sequential bargaining environment) and a complex multi-agent card game (Monopoly Deal), both of which we build from scratch. We then evaluate these methods through both theoretical reasoning and empirical experiments, focusing on their relative strengths and limitations in terms of accuracy, stability, and convergence. Our findings suggest that centralized and deterministic approaches tend to outperform others in environments where equilibrium strategies are sharp and highly sensitive to deviations.

2 Related Work and Methods

2.1 Independent Agents

One of the simplest extensions of reinforcement learning to multi-agent settings is to treat each agent as an independent learner. Throughout the training process, each of these agents will optimize its own policy by interacting with the environment and ignoring the presence of other competing or cooperating agents. The behavior of these other agents will instead be treated as part of the environment dynamics. In this project, we incorporate the Independent DQN method (IDQN), in which each agent learns through a Deep Q-Network (DQN) (i.e. each agent i maintains a Q network

Preprint. Under review.

$\hat{Q}_i(o_i, a_i, \theta_i)$, where o_i is that agent's local observation, a_i is an action taken by that agent, and θ_i represents the parameters of the network).

While we focus on Independent Deep Q-Networks (IDQN) in our experiments, we begin by analyzing the convergence properties of classical tabular Q-learning in order to highlight the core instability introduced by multi-agent interactions. In single-agent settings, Q-learning is known to converge under mild assumptions, owing to the contraction property of the Bellman operator. However, this key property breaks down in multi-agent reinforcement learning. Here, each agent i no longer acts in a stationary environment, as the transition dynamics depend not only on its own actions but also on the actions of other agents. Formally, the marginal transition function becomes $P(s'|s, a_i) = \sum_{a_{\forall j \neq i}} P(a_j|s) \cdot P(s'|s, a_i, a_j)$, where a_j denotes the joint actions of all other agents. Since those agents are also learning and updating their policies over time, the distribution $P(a_{\forall j \neq i}|s)$ is non-stationary, resulting in time-varying transitions from the perspective of agent i. Moreover, each agent receives only a partial observation of the environment at each timestep, i.e., the observation o_i reveals only a subset of the full global state s. This partial observability, combined with the unobservable and evolving policies of other agents, invalidates the Markov assumption required for Q-learning convergence.

As a result, the Q-learning update $Q_i(o_i, a_i) \leftarrow (1 - \alpha)Q_i(o_i, a_i) + \alpha \left(r_i + \gamma \max_{a'_i} Q_i(o'_i, a'_i)\right)$ is applied using targets derived from a non-stationary and partially observed environment. The underlying Bellman operator becomes time-varying and loses its contraction property, which is essential for convergence in standard Q-learning. This issue is further exacerbated in our Independent DQN (IDQN) architecture, where off-policy updates, bootstrapping, and function approximation (the deadly triad) further amplify the instability and make divergence even more likely. [Lee et al., 2021] [Sutton and Barto, 2018]

2.2 Centralized Agent

In order to address the non-stationarity of using independent agents, a simple solution is to directly model the joint state and joint action of all agents in one single network. In this project, we incorporate a centralized DQN (CDQN) that keeps track of a global Q function $Q(s, \bar{a})$ where s represents the full environmenment state and $\bar{a}$ is the joint action over all n agents, i.e. $\bar{a} = (a_1, a_2, ..., a_n)$.

From a convergence standpoint, centralized Q-learning reformulates the multi-agent environment as a single-agent, fully observable MDP, and therefore inherits the same convergence and stability guarantees as standard Q-learning. However, a major limitation of fully centralized approaches is that each agent requires access to the joint action and global state at execution time, which is unrealistic in most applications. In the case of competitive multi-agent games, agents often operate under partial observability and cannot access the exact states or actions of others, as agents are unwilling to reveal their intentions, valuations, and preferences. This makes fully centralized execution impractical outside of controlled or simulated environments.

2.3 Centralized Training, Decentralized Execution (CTDE)

To address the limitations of independent agents, who suffer from non-stationarity, and having one centralized agent, which would be unrealistic to deploy, the Centralized Training with Decentralized Execution is introduced [Amato, 2024]. As the name suggests, agents are trained using centralized information, i.e. the states, actions, and rewards seen by all other agents, but are restricted to using only their own local observations at execution time. This setup balances the sample efficiency and stability benefits of using centralized learning with the practical need for decentralized policies in real-world execution scenarios. Two widely-used multiagent RL frameworks, MADDPG [Lowe et al., 2020] and MAPPO [Yu et al., 2022], which we incorporate in our experiments, are built upon this framework.

2.3.1 Multiagent Deep Deterministic Policy Gradient (MADDPG)

Deep Deterministic Policy Gradient is an extension of Vanilla Policy Gradient (VPG), which, instead of optimizing stochastic policy $\pi_\theta(a|s)$ and corresponding gradient $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$, optimizes a deterministic policy $\mu_\theta(s)$ and its policy gradient $\nabla_\theta \mu(s) \nabla_a Q^\mu(s, a)|_{a=\mu(s)}$. Intuitively, we can see from the formula that DDPG attempts to maximize two objectives: $\nabla_a Q^\mu(s, a)\big|_{a=\mu(s)}$ identifies

how the deterministic action should change to increase the expected return (functionally similar to the log probabilities term in VPG), and the term $\nabla_\theta \mu(s)$ propagates this gradient through the actor network, adjusting its parameters to produce actions that yield higher Q-values (functionally similar to the advantage term in VPG).

MADDPG further extends DDPG to multiagent environments under the CTDE framework. During training, a centralized critic $Q_i^\mu(s, \mathbf{a})$ is used for each agent i, which conditions on the global state s and the joint action $\mathbf{a} = (a_1, \ldots, a_N)$. Meanwhile, each agent's actor network is updated using the deterministic policy gradient, optimizing how its own action affects its expected return. At execution time, the critic is discarded, and each agent selects actions using only its local observation: $a_i = \mu_i(o_i)$.

2.3.2 Multiagent Proximal Policy Optimization (MAPPO)

Proximal Policy Optimization is another widely used policy gradient method that greatly improves the stability over VPG by constraining policy updates to avoid large deviations from the previous policy. While Trust Region Policy Optimization (TRPO) enforces this constraint by explicitly limiting the KL divergence between the old and new policies to be within some threshold δ, PPO introduces a simpler and less computationally expensive approach by applying a clipping mechanism to the policy ratio, i.e. $[\min(r_\theta(s,a)A_{\theta_{old}}(s,a), clip(r_\theta(s,a), 1-\epsilon, 1+\epsilon)A_{\theta_{old}}(s,a)]$, where $r_\theta(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ represents this ratio. The clipping prevents excessively large updates by limiting how much the policy ratio can amplify the advantage, thus improving overall stability.

MAPPO is an extension of the PPO algorithm to multiagent settings under the CTDE framework. Here, each agent maintains an independent stochastic policy $\pi_i(a_i|o_i)$ that is updated through PPO's clipped objective, in contrast to MADDPG's deterministic policy $\mu_i(o_i)$. During training, a centralized value function $V_i(s)$ is used to compute advantage estimates for each agent, where s is the global state, allowing each agent to leverage global information for better advantage estimates. At execution time, each agent acts independently by sampling from their local policy distributions and using their own local observations.

3 Experiments and Discussion

3.1 Environment 1: 2-Player Sequential Bargaining Environment

We begin by evaluating the learning and convergence propoerties of multi-agent reinforcement learning algorithms within a simple, 2-agent environment: the Sequential Bargaining Environment [Rubinstein, 1982]. Rooted in economic theory, the Sequential Bargaining problem involves two competing agents alternating offers to optimally split some finite, divisible resource (e.g. a pie). During each round, one agent acts as the proposer and the other as the responder. The proposer suggests a division of the resource, and the responder must either accept the offer—terminating the game with the agreed-upon allocation—or reject it, in which case roles reverse in the next round. Each round is discounted by some factor $\delta \in (0, 1)$, incentivizing early agreement to maximize the reward achieved by both parties. If no agreement is reached within the finite horizon of some T rounds, both agents receive a default reward of zero. In our implementation, the number of bargaining rounds is limited to 5. Agents are trained for 5000 episodes and evaluated on 500 episodes.

We analyze our MARL algorithms' performance by comparing our experimental results to this problem's Subgame-Perfect Nash Equilibrium (SPNE). An SPNE is a refinement on top of the Nash Equilibrium that ensures each player's strategies are optimal at every single step of a sequential game. To derive the SPNE here, we first note that when Agent A proposes some value V_A, Agent B will receive value $1 - \delta V_B$ should they accept the offer, assuming that the resource is normalized to 1. Vice versa, when Agent B proposes some value V_B, Agent A will receive value $1 - \delta V_A$ should they accept the offer. It is intuitive to see that the optimal equilibrium strategy is to make the other agent indifferent between accepting and rejecting the offer. Solving this system of equations yields us $V_1 = V_2 = \frac{1-\delta}{1-\delta^2} = \frac{1}{1+\delta}$. Because the proposer is not affected by the discount factor and that the responder will bear the discount factor δ should they re-propose in the next iteration, the SPNE is for the proposer to propose a share of $\frac{1}{1+\delta}$ for themselves and a share of $\frac{\delta}{1+\delta}$ for the responder [Rubinstein and Wolinsky, 1985].

Note that in these experiments, Gumbel-Softmax discretization is applied in the MADDPG implementation, as DDPG requires a continuous action space to have a differentiable loss landscape.

3.1.1 Performance vs. Action-space size

We first evaluate the performance of the four multi-agent RL algorithms across varying grid sizes in the action space, where each grid size controls the resolution of possible offers in the sequential bargaining environment. For example, a resolution of 10 allows the agents to make offers only at intervals of 0.1. The results are shown in Figure 1. Note that the standard deviations are obtained across 5 training-execution runs for each method and grid size combination.

First, we notice the Centralized DQN (CDQN) achieves very stable performance for all action space sizes. This is intuitive as CDQN operates on the global state and joint action space, providing each agent with complete information on the other agent's actions and intents. The complete observability significantly reduces learning variance and allows for coordination across all steps of the execution process.

In contrast, the Independent DQN (IDQN) performs poorly and exhibits very high variance across all grid sizes. This supports our earlier analysis that IDQN suffers from non-stationarity due to independent learning in a multi-agent setting. As the grid size increases, the resulting larger action space further amplifies the instability, since agents lack the coordination needed to navigate a finer-grained negotiation space.

MAPPO demonstrates more stable learning than IDQN, but still experiences substantial variations across all grid sizes. The performance hovers around, but does not converge to the theoretical optimum, suggesting a tendency toward some mixture across both agents' valuations. In the sequential bargaining environment, a sharp, low-variance policy is highly desirable, as the proposer should make threshold-level offers that are just barely acceptable to the responder. When stochasticity is involved in the policies, the resulting small deviations can often trigger new rounds of negotiations and therefore a discounted pie, and thereby lead to inefficient proposals and unstable convergence.

Last but not least, MADDPG exhibits the strongest performance in tracking the theoretical optimum. This is likely because the deterministic actor enables low-variance decisions during execution, allowing for both agents to learn policies that optimally converge to the SPNE optimum, in contrast to MAPPO's stochastic policies. Another observation is that variance decreases as the grid size increases. This is further indication of MADDPG's ability to consistently "lock on" to the optimal offer, as the increasing grid size allows for a more precise deterministic policy to optimally partition the pie.

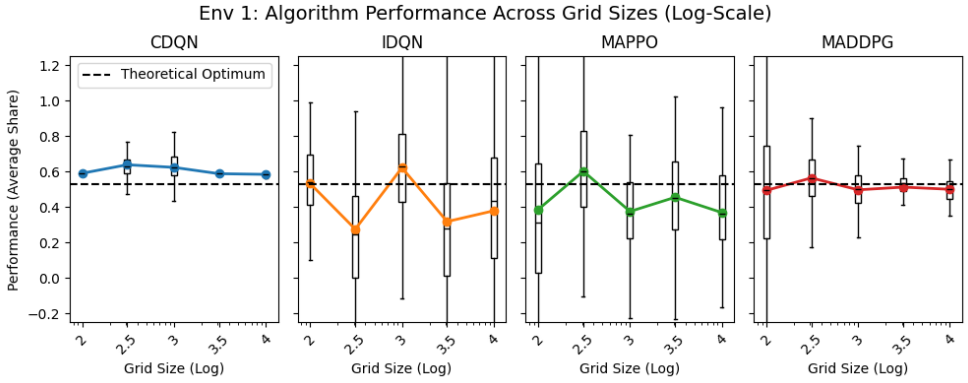

Figure 1: Average Share vs. Grid Sizes Comparison by method (CDQN, IDQN, MAPPO, MADDPG)

3.1.2 Convergence Dynamics

We next analyze the convergence dynamics of each algorithms over training in our sequential bargaining environment. The results are shown in Figure 5. Note that because SPNE favors the proposer, an effective algorithm should converge to a policy where agent A (proposer of the first iteration) obtains a larger share of the pie, and agent B obtains a smaller share.

CDQN demonstrates clear convergence after approximately 2000 training episodes, stabilizing at a point where Agent A obtains a consistently high reward while Agent B receives significantly less. This indicates that Agent A reliably learns to exploit its advantage as the proposer, which is enabled by CDQN's access to the full global state and joint action space during training.

In contrast, IDQN exhibits highly unstable learning dynamics. Agent A's running average reward fluctuates significantly throughout training before ultimately collapsing to below 0.1. This suggests that IDQN fails to learn an equilibrium-consistent strategy, which aligns with expectations given the method's inability to model other agents in a non-stationary multi-agent environment. The resulting lack of coordination leads to high variance and consistently suboptimal outcomes.

MAPPO exhibits more stable and gradual convergence compared to IDQN, but still falls short of learning the equilibrium strategy. Agent A's average reward remains low, indicating that it fails to fully leverage its strategic advantage as the proposer. This supports our earlier analysis that the stochastic structure of MAPPO's policy produces blurred strategic behavior, generating suboptimal or inconsistent offers, and leading to compromised results in an environment that rewards precise and deterministic proposals.

Lastly, MADDPG displays the strongest performance in terms of both convergence speed and equilibrium fidelity. It stabilizes in fewer than 200 episodes and consistently achieves a high reward for Agent A, on par with CDQN. This suggests that MADDPG's centralized critic and Gumbel-Softmax-based discrete policy allow Agent A to effectively learn sharp offer thresholds, enabling it to exploit its proposer role in a manner consistent with SPNE behavior.

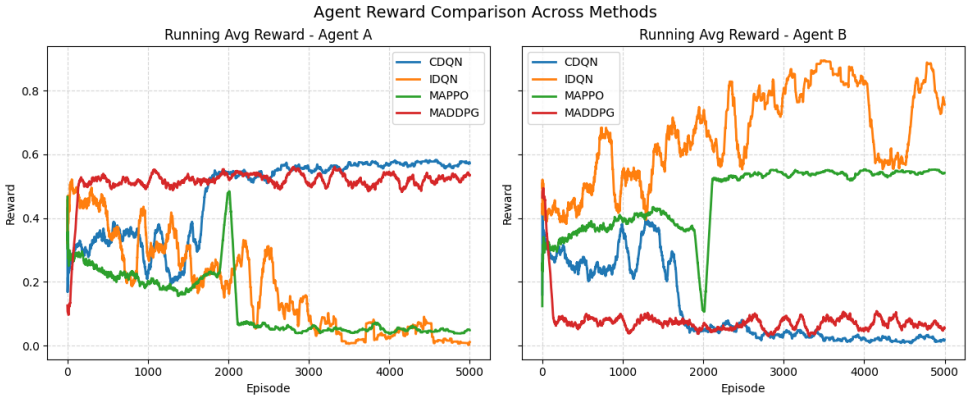

Figure 2: Convergence Dynamics by method.

3.2 Environment 2: Creating and Experimenting with a Monopoly Deal Environment

3.2.1 Objectives

Our primary objective was to design and implement a fully functional Agent-Environment Cycle (AEC) environment tailored to the mechanics of Monopoly Deal. Constructing an AEC environment is a foundational step in multi-agent reinforcement learning (MARL), as it provides the necessary framework for sequential, interleaved decision-making among multiple agents. By building our own environment, we gain complete control over the dynamics, game rules, and observation/action interfaces, allowing for custom experimentation and benchmarking. Our two main goals were to:

1. Create a working Monopoly Deal environment for future MARL work.
2. Produce an agent that performs better than random action selection.

3.2.2 Game Summary

Monopoly Deal is a strategic, turn-based card game derived from the traditional Monopoly board game, reformulated to operate exclusively through a deck of specialized cards. The primary objective is for a player to assemble three complete sets of property cards, each set corresponding to a unique

color group. Gameplay consists of drawing cards and executing up to three actions per turn, with possible actions including the placement of property or money cards, as well as the use of action cards to influence the game state. Action cards enable players to perform a range of interactions such as charging rent, stealing property, demanding payments, or countering opponents' moves. All the rules can be found here.

3.2.3 Building the Environment and Game Setup

The Monopoly Deal environment was built using the Petting Zoo API. This was done in order to leverage the support of the Petting Zoo community as well as to open the door for this environment to be used by others in the future using a standard API.

The observation space in the Monopoly Deal environment is a flattened, fixed-length vector that compactly encodes the agent's current game state. It includes the agent's hand (with up to 12 cards), the total money in their bank, and the occupancy count of each of their 10 property slots. Additionally, the observation captures how many cards are in each property slot of the two other players (for a 3-player game), along with their respective bank totals. The agent's own completed property sets are also counted as a single scalar. This representation allows a learning agent to infer relative game progress, opponent threat levels, and strategic opportunities from a compact feature vector. A previous iteration of the observation space included the individual cards in each player's bank and property slots. Experimentation yielded similar results between the simplified observation space and the previous iteration, so we went with the simpler one. The observation space is show in Table 2.

The action space, in contrast, is large and sparse, consisting of 611 discrete indices that encode every possible legal action the agent can take, including banking cards, playing rent or action cards, performing targeted moves like "Deal Breaker" or "Forced Deal", and meta-actions like discarding or passing. To manage this complexity and prevent the agent from selecting invalid or illegal actions, an **action masking** mechanism is employed.

At each decision step, a binary mask of length 611 is generated, where each element corresponds to an action index. The action represent by each index is shown in Table 1. A value of 1 indicates that the action is valid in the current state, while a 0 masks out illegal options. This mask is integrated into the learning process to ensure the agent only considers valid moves during action selection and gradient updates. Not only does this improve sample efficiency and convergence speed, but it also dramatically reduces the likelihood of wasted transitions due to invalid actions. Descriptions of both spaces are in the appendix.

The reward function did not remain static throughout experimentation. Depending on the results of experimentation, rewards for different actions were added or removed. We introduced a set of shaped rewards tied to specific in-game actions. These reward signals were designed to reinforce desirable behaviors—such as placing properties, completing sets, and ultimately winning, while also penalizing inefficient or stalled actions like discarding or hoarding cards. These rewards could be turned on/off using a shape option in the environment.

3.2.4 Training a DQN to Beat Random Agents

First, the DQN was trained with a reward of +10 for a win and -10 for a loss, with no intermediate actions receiving rewards. This sparse reward setting was intended to incentivize long-term planning rather than short-sighted strategies. While the agent's average reward per 5000 steps improved over time, the win rate consistently declined after an initial spike. This suggests that the agent learned to exploit aspects of the game that increased its expected reward without necessarily learning how to win.

To try and improve our model, the DQN was trained with shaped rewards. Shaped rewards are intermediate rewards provided throughout the episode to encourage useful behavior and accelerate learning, rather than only rewarding the agent at the end of the game for winning. These rewards were given for actions such as placing properties or completing a full set. Over the course of training, the agent's average reward steadily increased, indicating that it was learning to exploit the shaped reward structure effectively. However, despite this improvement in average reward, the agent's win rate did not show a corresponding increase. This suggests that the agent learned to maximize short-term gains encouraged by the shaped rewards, but failed to develop long-term strategies that would lead to actually winning the game.

3.2.5 Application of MAPPO on Monopoly Deal

Our next attempt at creating a good agent in Monopoly Deal was through Multi-Agent Proximal Policy Optimization (MAPPO). While the average loss per 100 updates remained stable, fluctuating between 3,250 and 4,750, the average reward per 5,000 steps across all three agents showed high variance, typically remaining between -4 and -1. This suggests that although agents were learning to optimize local reward signals, they were not converging to consistent high-reward strategies. Win rate evaluation against random opponents every 500 episodes further reinforced this, as agents achieved moderate success, with win rates fluctuating between 0.25 and 0.42, but without clear upward trends. These results highlight the difficulty of training stable and generalizable MAPPO agents under noisy reward signals and emphasize the need for more targeted reward shaping or opponent diversity to improve learning outcomes.

3.2.6 General Strategies for Improvement

The poor results of our models prompted us to try several techniques to push the performance of the models passed the random-selection agents. One technique was to train at en epsilon of 1 for a third of the episodes. The motivation behind this was to let the agent explore rare actions (such as using a Deal Breaker card, of which there are only 2 in the entire deck). Furthermore, the "bank" action removes a card from play for the entire game, so we ran the exploratory phase with the ability to bank cards turned off. Unfortunately, neither attempt yielded significant performance improvements.

3.2.7 Further Work

There are directions we will take to improve agent performance and stability in the Monopoly Deal environment. One avenue is to reduce stochasticity during training by fixing the initial game state—specifically, by training agents from the same starting hand in each episode. This would allow for more consistent learning signals and help isolate the effects of strategic decision-making.

Furthermore, the action space is large, and some cards (like Deal Breakers) appear extremely infrequently, making it hard to explore the Q values of using those cards. Work has been done to address these sparse actions that can be applied to this environment [Pang et al., 2021]. Reward shaping also turned out to be extremely difficult, as evident by the fact that higher shaped rewards did not translate to higher win rates, but methods exist to avoid shaping rewards manually as well [Hu et al., 2020].

Finally, we hope to apply algorithms used in other card games to Monopoly Deal, as the game may not be a suitable environment for our chosen algorithms. For example, the Pluribus program has achieved state of the art performance in Poker, and may be able to be applied to Monopoly Deal [Brown and Sandholm, 2019].

4 Conclusion

In this project, we explored a variety of multi-agent reinforcement learning methods, evaluating their stability, convergence, and performance in both controlled (Sequential Bargaining) and complex (Monopoly Deal) environments. While centralized and deterministic approaches like MADDPG excelled in low-variance settings with clear objectives, more expressive policy gradient methods like MAPPO struggled in environments with noisy rewards and high action sparsity. Our work highlights the importance of environment design, reward structure, and agent coordination in MARL. Future work will explore better initialization, action space pruning, and reward shaping strategies to improve sample efficiency and policy generalization in large, stochastic games.

References

Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning, 2024. URL https://arxiv.org/abs/2409.03052.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019. doi: 10.1126/science.aay2400. URL https://www.science.org/doi/10.1126/science.aay2400.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping, 2020. URL https://arxiv.org/abs/2011.02669. Accepted by NeurIPS 2020.

Ken Ming Lee, Sriram Ganapathi Subramanian, and Mark Crowley. Investigation of independent reinforcement learning algorithms in multi-agent environments, 2021. URL https://arxiv.org/abs/2111.01100.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments, 2020. URL https://arxiv.org/abs/1706.02275.

Jing-Cheng Pang, Tian Xu, Shengyi Jiang, Yu-Ren Liu, and Yang Yu. Reinforcement learning with sparse–executing actions via sparsity regularization, 2021. URL https://arxiv.org/abs/2105.08666. Version 4, last revised 22 Jul 2024.

Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1912531.

Ariel Rubinstein and Asher Wolinsky. Equilibrium in a market with sequential bargaining. *Econometrica*, 53(5):1133–1150, 1985. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1911015.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022. URL https://arxiv.org/abs/2103.01955.

Appendix

Additional Figures

Action Type	Start Index	End Index	Size
bank_card	0	66	67
deal_breaker	67	86	20
just_say_no	87	87	1
pass_go	88	88	1
sly_deal	89	108	20
forced_deal	109	308	200
debt_collector	309	311	3
birthday	312	312	1
rent	313	332	20
place_property	333	342	10
place_wildcard	343	352	10
flip_wildcard	353	362	10
place_wildcard_all	363	372	10
move_wildcard_all	373	472	100
place_house	473	482	10
place_hotel	483	492	10
pay_sum_from_property	493	502	10
accept_action	503	503	1
discard_card	504	609	106
do_nothing	610	610	1

Table 1: Flattened action space layout for Monopoly Deal with 3 players

Component	Start Index	End Index	Size	Description
Hand	0	11	12	Card IDs in the agent's hand
Bank	12	12	1	Sum of money in the agent's bank
Properties	13	22	10	Number of cards in each of the agent's 10 property slots
Completed Sets	23	23	1	Number of complete property sets the agent owns
Other Properties	24	43	20	Number of cards in each of the 10 property slots of 2 opponents
Other Banks	44	45	2	Sum of money in each of the 2 opponent banks

Table 2: Flattened observation space layout for Monopoly Deal (3-player game)

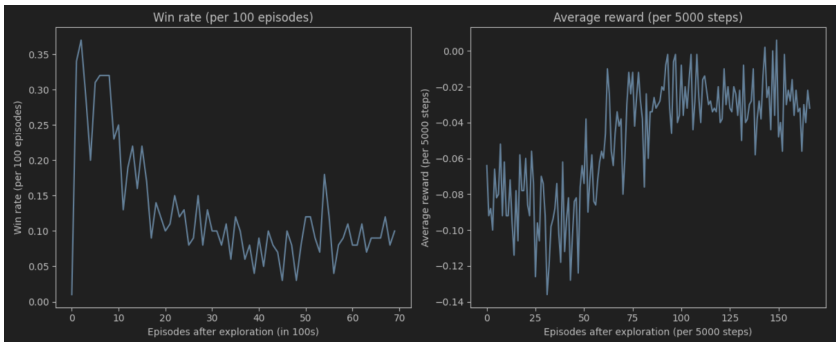

Figure 3: Win rate and rewards for initial environment. Note: Since there were 3 players, the benchmark was a 33% win rate.

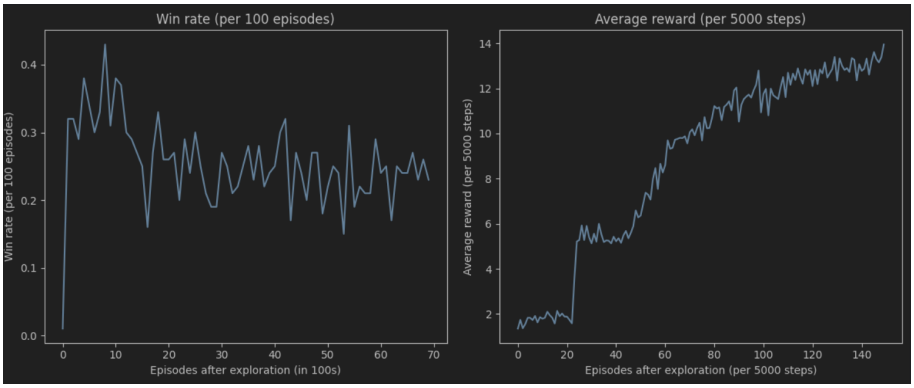

Figure 4: Win rate and rewards for DQN shaped rewards environment.

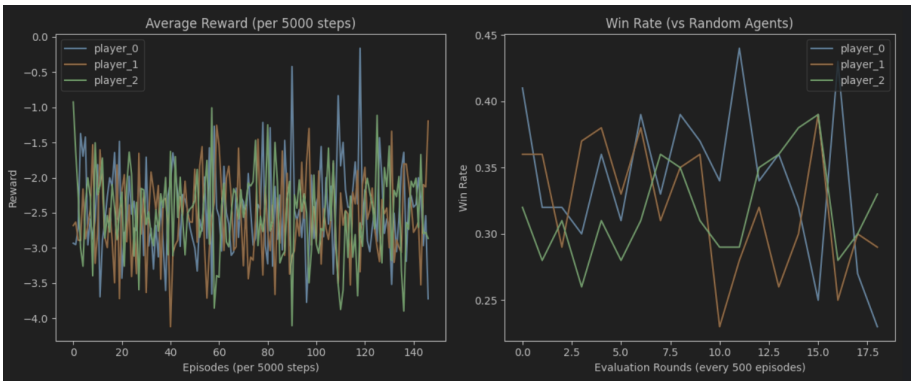

Figure 5: Win rate and rewards for MAPPO.

Work Distribution

Athith: Environment creation, all experiments and analysis for the Monopoly Deal Environment

Mike: Writing of the methods section. Environment creation, all experiments and analysis for the Sequantial Bargaining Environment