# Safe Multi-Agent Reinforcement Learning

**Dhruva Bhardwaj**

Master of Technology

Robotics
Interdisciplinary

**INDIAN INSTITUTE OF TECHNOLOGY DELHI**


Under the Supervision of
**Prof. James Arambam Singh**

May, 2025

# Abstract

Multi-agent reinforcement learning (MARL) presents a significant challenge in the development of autonomous systems, particularly for safety-critical applications, necessitating the development of novel algorithms to enable agents to learn cooperative strategies while adhering to constraints and avoiding catastrophic outcomes. This project addresses the problem of safe MARL by introducing a new algorithm that integrates deep policy gradient optimization technique with a Lagrangian multiplier framework. Our approach draws inspiration from several key areas within the reinforcement learning domain, including Multi-Agent Deep Deterministic Policy Gradient (MADDPG), reward constrained optimization, and multi-agent constrained policy optimization. The core innovation of our algorithm lies in its ability to effectively balance the competing objectives of maximizing task rewards and minimizing cost accumulation. This is achieved through the use of a Lagrangian multiplier, which dynamically adjusts the influence of the cost constraint during the learning process, enabling agents to learn policies that not only achieve high performance but also actively avoid behaviors that lead to constraint violations. We hypothesize that explicitly addressing the cost function within the optimization process will result in policies with enhanced safety characteristics, making them more suitable for real-world deployment. The algorithm incorporates a deep neural network architecture for scalability to high-dimensional state and action spaces and employs a decentralized policy gradient approach for applicability in scenarios with limited communication or partial observability. To validate our algorithm, we conducted experiments in a modified PettingZoo simple spread environment, adapted to include a cost function. The results demonstrate that our algorithm effectively constrains cost while maintaining high task performance. Notably, agents adapt their behavior in response to varying cost constraints, adopting more conservative strategies as constraint tighten. We present quantitative results showing a clear trade-off between reward and cost. These results provide strong evidence for the effectiveness of our algorithm in addressing safe MARL challenges and suggest its potential for application in real-world domains where safety is a critical concern. Future work will focus on extending the algorithm to more complex environments, exploring different cost function specifications, and investigating performance in scenarios with partial observability.

# Contents

# Chapter 1

# Introduction

## 1.1 Background

The real world is inherently multi-agent, with tasks often requiring effective teamwork and coordination. Multi-Agent Reinforcement Learning (MARL) offers a promising avenue for developing algorithms that enable autonomous agents to learn these cooperative strategies. As these algorithms transition from simulated environments to real-world applications, such as traffic management, autonomous driving, and robotic manipulation, ensuring the safety of these agents becomes paramount. In dynamic real-world scenarios, it is often impossible to guarantee a completely safe operational space; therefore, integrating safety considerations directly into the training process is crucial. This necessity has led to a surge of interest in safe reinforcement learning, and more specifically, safe MARL, which seeks to equip agents with the ability to learn and act safely in complex, interactive environments. The development of effective safe MARL algorithms holds the potential to unlock a wide range of applications, from improving the efficiency and reliability of multi-robot systems to enabling safer and more robust autonomous vehicles. Furthermore, as AI systems become increasingly integrated into human-inhabited environments, the ability to guarantee their safe operation is not just a matter of performance, but also a matter of trust and ethical considerations.

## 1.2 Problem Statement

A significant challenge in deploying MARL in real-world, safety-critical scenarios is ensuring that agents operate without causing harm or violating constraints. Traditional RL methods often optimize solely for reward, neglecting potential costs or dangers. This can lead to unsafe policies that are unsuitable for applications such as autonomous driving, robotics, and healthcare. Therefore, there is a need to develop algorithms that can learn safe policies under given cost constraints, while effectively achieving the desired task objective.

## 1.3 Objectives

- Develop a Safe Multi-Agent Reinforcement Learning Algorithm via Constrained Policy Gradients: Create a MARL algorithm that leverages policy gradient methods and Lagrangian multipliers, inspired by Reward Constrained Policy Optimization (RCPO), to learn joint policies that explicitly satisfy safety constraints.

- Address Multi-Agent Safety Challenges Drawing from MADDPG Principles: Investigate and resolve safety-related issues arising from inter-agent dependencies and dynamic interactions

in shared environments, drawing inspiration from the decentralized execution with centralized training paradigm of MADDPG.

- Integrate Reward Constraints for Safe Exploration and Learning: Adapt and extend the core principles of RCPO, utilizing Lagrangian multipliers to manage reward constraints and guide the policy gradient updates towards safer and more performant multi-agent behaviors.
- Evaluate the Safety, Efficiency, and Scalability of the Proposed Approach: Rigorously test and validate the developed algorithm in diverse simulated multi-agent environments to quantitatively measure the achieved safety levels, learning efficiency, and scalability to a growing number of agents.

## 1.4 Scope

The scope of this project includes the design, implementation, and evaluation of a safe MARL algorithm from scratch. The algorithm will be implemented in Python using PyTorch, and its performance will be evaluated in the PettingZoo's multi-particle environment. The project focuses on learning policies that adhere to safety constraints, with a specific emphasis on the trade-off between safety and task performance. The project will also validate the algorithm's safety adherence, performance efficiency, comparing its performance against set baseline algorithm.

## 1.5 Organization

The remainder of this report is organized as follows. Section 2 provides a review of related work in the field of safe multi-agent reinforcement learning. Section 3 details the proposed algorithm and methodology. Section 4 presents the results of the experiments, and Section 5 discusses the implications of these results and suggests directions for future research.

# Chapter 2

# Literature Review

This section provides a comprehensive overview of the existing research and literature that is relevant to our project. It helps the reader understand the current state of the field, identify gaps in knowledge, and explain how our project builds upon or differs from previous work.

## 2.1 Multi-Agent Reinforcement Learning (MARL)

MARL is a subfield of reinforcement learning that deals with the problem of training multiple agents to interact and achieve goals in a shared environment. It is a crucial area of research for developing intelligent systems that can effectively collaborate and compete in complex, dynamic settings. Key concepts in MARL include:

- **Joint state and action spaces:** The combined state and action spaces of all agents in the system, which define the possible configurations and actions of the multi-agent system.
- **Policies:** The strategies that each agent uses to select actions based on their observations and the history of interactions.
- **Reward structures:** The way that agents are incentivized to achieve desired outcomes, which can be cooperative (agents share a common goal), competitive (agents have conflicting goals), or a mix of both.

### 2.1.1 Challenges in MARL

MARL presents several unique challenges compared to single-agent RL, which stem from the presence of multiple interacting agents. These challenges include:

- **Non-stationarity:** From the perspective of each agent, the environment is constantly changing as other agents update their policies. This violates the Markov property, which is a fundamental assumption in single-agent RL, and makes it difficult for agents to learn stable policies.
- **Increased complexity:** The size of the state and action spaces grows exponentially with the number of agents, leading to the "curse of dimensionality" and making learning and optimization significantly more difficult.
- **Coordination:** Agents must learn to coordinate their actions to achieve common goals or avoid conflicts. This requires the development of effective communication and cooperation strategies, which can be challenging in decentralized settings.

- **Credit assignment:** Determining how to distribute rewards or penalties among agents for their individual contributions to the overall outcome can be difficult, especially in cooperative settings with delayed rewards.

### 2.1.2 MARL Algorithms

Different types of MARL algorithms have been developed to address these challenges, each with its own strengths and weaknesses. These algorithms include:

- **Independent Q-learning (IQL)** Each agent learns its own Q-function independently, treating the other agents as part of the environment. This approach is simple to implement but can suffer from instability due to the non-stationarity of the environment, as each agent's optimal policy depends on the policies of the other agents, which are constantly changing.
- **Centralized training with decentralized execution (CTDE)** Agents are trained centrally with access to global information, such as the joint state and actions of all agents, but execute their policies independently based on their local observations. This approach can improve learning stability and coordination by allowing agents to learn about the behavior of other agents during training. MADDPG is a popular example of a CTDE method.
- **Policy gradient methods** Agents directly learn their policies by optimizing a performance objective, such as the expected return. These methods can be more suitable for continuous action spaces and can handle non-stationarity to some extent, as they do not rely on estimating Q-functions. However, they can suffer from high variance and may require careful tuning.

## 2.2 Safe Reinforcement Learning (Safe RL)

Safe RL is a crucial area of research that aims to ensure that RL agents can learn to achieve their goals without violating safety constraints or causing harm. This is particularly important in real-world applications such as robotics, autonomous driving, and healthcare, where unsafe actions can have severe consequences. In these domains, it is not sufficient for an agent to simply maximize its reward; it must also adhere to certain safety requirements to prevent catastrophic failures or unacceptable behavior.

### 2.2.1 Approaches to Safe RL

Several approaches have been proposed to address the challenges of Safe RL, each with its own way of incorporating safety considerations into the learning process. These approaches include:

- **Constrained RL** This approach incorporates constraints into the RL problem formulation, typically using techniques like Lagrangian multipliers or penalty methods. The goal is to find a policy that maximizes reward while satisfying the constraints, which define the acceptable behavior of the agent. RCPO is an example of a constrained RL method.
- **Risk-sensitive RL** This approach considers the variance or higher-order moments of the return distribution, in addition to the expected return, to avoid policies that may lead to large deviations from the expected outcome. By taking into account the risk associated with different policies, risk-sensitive RL methods can learn more conservative and safer behaviors.

- **Shielding** This technique involves using a separate module, called a shield, to filter or modify the actions of an RL agent to ensure they are safe. The shield intervenes only when necessary, allowing the agent to explore and learn freely otherwise, but preventing it from taking actions that are known to be unsafe.

- **Recovery mechanisms** These methods focus on designing strategies for safely recovering from unsafe states or actions, allowing the agent to mitigate the consequences of errors and continue learning. This can involve learning to identify unsafe situations and developing policies for returning to a safe state.

## 2.3 Key Related Works

In this section, we discuss some of the key related works that are most relevant to our project, and which form the foundation for our proposed approach.

### 2.3.1 MADDPG (Multi-Agent Deep Deterministic Policy Gradient)

MADDPG is a policy gradient algorithm that extends the Deep Deterministic Policy Gradient (DDPG) algorithm, which is designed for single-agent RL with continuous action spaces, to the multi-agent setting. It employs a CTDE scheme, where each agent learns a deterministic policy, and a centralized critic network is used to evaluate the joint actions of all agents.

The centralized critic has access to the joint state and actions of all agents during training, which provides more information for credit assignment and helps to address the issue of non-stationarity. By knowing the actions of all agents, the critic can more accurately evaluate the contribution of each agent's actions to the overall outcome. During execution, however, each agent uses only its local observations to select actions, allowing for decentralized control and enabling the algorithm to scale to large multi-agent systems.

MADDPG has been shown to be effective in both cooperative and competitive multi-agent environments, demonstrating its versatility and ability to handle different types of agent interactions. However, it can face challenges in scenarios with a large number of agents, due to the increased complexity of the centralized critic, which grows with the number of agents.

### 2.3.2 Reward Constrained Policy Optimization (RCPO)

RCPO is a policy optimization algorithm that incorporates constraints into the learning process. It uses a Lagrangian multiplier to ensure that the expected return of the learned policy satisfies a given safety threshold. This allows the agent to learn policies that not only maximize reward but also adhere to safety constraints, which are defined in terms of acceptable levels of cost or risk.

RCPO modifies the policy update rule to account for the cost constraint, allowing the agent to learn policies that maximize reward while minimizing the risk of violating the constraint. The Lagrangian multiplier is adjusted during training to balance the trade-off between reward and safety. The algorithm aims to find a policy that achieves the highest possible reward while keeping the expected cost below a specified limit.

RCPO can handle complex constraints and has been applied to various Safe RL problems, demonstrating its effectiveness in learning safe and performant policies. However, its performance can be sensitive to the choice of the Lagrangian multiplier and other hyperparameters, which may require careful tuning.

### 2.3.3 Multi-Agent Constrained Policy Optimization

Multi-Agent Constrained Policy Optimization extends constrained policy optimization techniques, such as RCPO, to the multi-agent setting. It addresses the challenges of coordinating multiple agents while satisfying safety constraints, which is crucial for deploying MARL algorithms in real-world, safety-critical systems.

These methods typically involve modifying the MARL objective function to include cost terms that quantify the violation of safety constraints. Techniques such as Lagrangian multipliers, constrained optimization algorithms, or barrier functions are then employed to find policies that optimize both reward and safety in the multi-agent setting. The goal is to learn joint policies that maximize the collective reward of the agents while ensuring that the system as a whole operates within safe limits.

#### Challenges in Multi-Agent Constrained Policy Optimization

Several challenges remain in this area, which make it an active and important area of research. These challenges include:

- **Scalability to large numbers of agents** As the number of agents increases, the complexity of the optimization problem grows significantly, making it difficult to find optimal or even near-optimal solutions.
- **Partial observability** In many real-world scenarios, agents have only limited information about the state of the environment and the actions of other agents, making it more difficult to enforce safety constraints and coordinate safe behavior.
- **Non-stationarity** The changing policies of other agents make it challenging to learn a stable and safe policy, as the optimal behavior for one agent may change as the other agents' policies evolve.

## 2.4 Connections to Our Approach

Our project aims to contribute to the field of Safe MARL by developing an algorithm that combines the advantages of constrained policy optimization and multi-agent policy gradient methods. We seek to address the limitations of existing approaches by developing a more effective and efficient way to learn safe and performant policies in multi-agent environments.

Drawing inspiration from the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) framework, which provides a strong foundation for handling multi-agent interactions, we incorporate a Lagrangian multiplier to explicitly address safety constraints during training, using deep policy gradients. This allows us to learn decentralized policies that can effectively balance the trade-off between reward and safety in multi-agent environments. By integrating safety constraints directly into the learning process, our approach enables agents to learn safer policies with improved performance compared to methods that address safety as a secondary concern or through post-processing.

# Chapter 3

# Methodology

This section details the methodology employed in this research to address the problem of safe multi-agent reinforcement learning. It describes the proposed algorithm, the experimental setup, and the evaluation metrics used to assess the performance of the algorithm.

## 3.1 Safe Multi-Agent Reinforcement Learning Algorithm

This section presents the proposed Safe Multi-Agent Reinforcement Learning Algorithm. The algorithm extends the principles of constrained Markov games and Lagrangian optimization to the multi-agent reinforcement learning setting.

### 3.1.1 Algorithm Overview

The algorithm aims to learn a policy that maximizes the expected return while satisfying safety constraints, defined in terms of expected costs for each agent. This is achieved by incorporating Lagrangian multipliers into the optimization process, which allows for a trade-off between reward maximization and cost minimization. The algorithm is designed to be applicable to multi-agent scenarios, where each agent seeks to maximize the joint reward while adhering to its individual cost constraints.

#### Mathematical Formulation

We formulate our problem as a constrained Markov game $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \rho, \rho^0, \gamma, R, \mathcal{C}, c \rangle$. Here, $\mathcal{N} = \{1, 2, ..N\}$ is the set of agents, $\mathcal{S}$ is the discrete state space, $\mathcal{A} = \prod_{i=1}^{N} \mathcal{A}^i$ is the product of the agents' action spaces, known as the joint action space, $\rho : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function, $\rho_0$ is the initial state distribution, $\gamma \in [0, 1)$ is the discount factor, $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the joint reward function, $\mathcal{C} = \{C_i\}$ is the set of sets of cost function that is described as the safe distance from the other agents and is of the form $C_i : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, and finally the set of corresponding cost-constraining values is given by $c = \{c_i\}$.

We have a global reward (collaborative setting), and the expected return can be expressed as:

$$J_R(\boldsymbol{\pi_\theta}) \triangleq \mathbb{E}_{\boldsymbol{s_0} \sim \rho^0, \boldsymbol{a}_{0:\mathcal{H}-1} \sim \boldsymbol{\pi_\theta}, \boldsymbol{s}_{1:\mathcal{H}-1} \sim \rho} \left[ \sum_{t=0}^{\mathcal{H}-1} \gamma^t R(\boldsymbol{s_t}, \boldsymbol{a_t}) | \boldsymbol{\pi_\Theta} \right] \tag{3.1}$$

And for each agent, we have an expected cost associated, which can be expressed as:

$$J_C^i(\pi_{\theta_i}) \triangleq \mathbb{E}_{\boldsymbol{s_0}\sim\rho^0,\boldsymbol{a}_{0:\mathcal{H}-1}\sim\pi_{\theta_i},\boldsymbol{s}_{1:\mathcal{H}-1}\sim\rho}\left[\frac{1}{H}\sum_{t=0}^{\mathcal{H}-1}\gamma^t C^i(\boldsymbol{s_t}, a_t^i)|\pi_{\Theta_i}\right] \forall i \in \mathcal{N} \tag{3.2}$$

Thus, we can formulate our problem along with the constraint in the following form:

$$\max J_R(\boldsymbol{\pi_\theta}) \tag{3.3}$$

$$\text{s.t } J_C^i(\pi_{\theta_i}) \leq c_i \ \forall \text{ i } \ \in \mathcal{N} \tag{3.4}$$

Using Lagrangian Multipliers, we can express it in the following form:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J_R(\boldsymbol{\pi_\theta}) - \sum_{i=0}^{\mathcal{N}} \lambda_i (J_C^i(\pi_{\theta_i}) - c_i) \tag{3.5}$$

Now, defining state-action value function and state value function associated with Rewards as

$$Q_\pi(\boldsymbol{s}, \boldsymbol{a}) \triangleq \mathbb{E}_{\boldsymbol{s}_{1:\mathcal{H}-1}\sim p,\boldsymbol{a}_{1:\mathcal{H}-1}\sim\boldsymbol{\pi}}\left[\sum_{t=0}^{\mathcal{H}-1}\gamma^t R(\boldsymbol{s_t}, \boldsymbol{a_t})\middle|\boldsymbol{s_0} = \boldsymbol{s}, \boldsymbol{a_0} = \boldsymbol{a}\right], \tag{3.6}$$

$$V_\pi(\boldsymbol{s}) \triangleq \mathbb{E}_{\boldsymbol{a}\sim\pi}\left[Q_\pi(\boldsymbol{s}, \boldsymbol{a})\right] \tag{3.7}$$

Similarly we can define state-action value function and state function associated with Costs as:

$$Q_{\pi_{\theta_i}}^i(\boldsymbol{s}, a^i) \triangleq E_{\boldsymbol{s}_{1:\mathcal{H}-1}\sim p,a_{1:\mathcal{H}-1}\sim\pi_{\theta_i}}\left[\sum_{t=0}^{\mathcal{H}-1}\gamma^t C^i(\boldsymbol{s_t}, a_t^i)\middle|\boldsymbol{s_0} = \boldsymbol{s}, a_0^i = a^i\right] \tag{3.8}$$

$$V_{\pi_{\theta_i}}^i(\boldsymbol{s}) \triangleq \mathbb{E}_{a\sim\pi_{\theta_i}}\left[Q_{\pi_{\theta_i}}^i(\boldsymbol{s}, a^i)\right] \tag{3.9}$$

We can then define the gradient of the Loss function as

$$\nabla J_R(\boldsymbol{\pi_\theta}) = E_{\boldsymbol{s}_0\sim\boldsymbol{\rho}^{\mathbf{0}}}\left[\sum_{i=0}^{\mathcal{H}-1}\nabla\boldsymbol{\pi_\theta}(\boldsymbol{s_t}, \boldsymbol{a_t}).\nabla Q(\boldsymbol{s_t}, \boldsymbol{a_t})|_{\boldsymbol{a_t}=\boldsymbol{\pi_\theta}(\boldsymbol{s_t})}\right] \tag{3.10}$$

$$\nabla J_C(\boldsymbol{\pi_{\theta_i}}) = E_{\boldsymbol{s}_0\sim\boldsymbol{\rho}^{\mathbf{0}}}\left[\sum_{i=0}^{\mathcal{H}-1}\sum_{a^i\in\mathcal{A}\rangle}\pi_{\theta_i}(\boldsymbol{s_t}, a_t^i).Q_{\pi_{\theta_i}}^i(\boldsymbol{s_t}, \boldsymbol{a_t})|_{a_t=\pi_{\theta_i}(\boldsymbol{s_t})}\right] \tag{3.11}$$

From this we can find gradient for (3.5) in terms of primal variables and latent variables,

$$\nabla_\theta\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \nabla_\theta J_R(\boldsymbol{\pi_\theta}) - \sum_{i=0}^{N}\lambda_i\nabla_\theta J_C^i(\pi_{\theta_i}) \tag{3.12}$$

$$\nabla_\lambda\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = -\sum_{i=0}^{N}\nabla_\lambda(J_C^i(\pi_{\theta_i}) - c_i) \tag{3.13}$$

### 3.1.2 Neural Network Architecture

For our experiments, we employed simple neural networks with a single hidden layer for both the policy and Q-networks. In the context of CMADP without a Q-network for cost learning, each agent was equipped with four such neural networks. Conversely, for CMADP incorporating a Q-network to learn costs, each agent utilized six neural networks with the same basic architecture. We intentionally opted for these straightforward network designs and architectures to facilitate a focused evaluation of our algorithm's performance.

## 3.2 Experimental Setup

This section describes the experiments conducted to evaluate the proposed algorithm. This includes the environments used, baseline algorithms for comparison, implementation details, hyperparameter settings, and data processing.

### Environments

We evaluated the performance of our algorithm in the PettingZoo multi-particle environment. This environment features multiple agents that must cooperate to achieve a common goal. To assess the safety of the learned policies, we introduced a cost function that penalizes collisions between agents.

#### Environment Details

In the multi-particle environment, each agent observes the positions and velocities of all other agents within a certain radius. The action space for each agent consists of discrete force commands that influence the agent's movement. The agents receive a joint reward for reaching the target location, with the reward being sparse to incentivize cooperation. The cost function is defined as the number of collisions between agents within a given time step. A safety constraint is imposed on the maximum number of allowed collisions per episode.

We evaluated the performance of our algorithm in the PettingZoo's simple spread environment. In this environment, agents must cooperate to reach a set of target locations while avoiding collisions. We modified the environment to introduce a cost function that depends on the distance between agents. Specifically, the cost is calculated based on how close agents are to each other, with closer proximity resulting in a higher cost. This modification allows us to evaluate the algorithm's ability to learn policies that maintain a safe distance between agents, promoting collision avoidance and safe cooperation.

### Baseline Algorithms

To provide a comprehensive evaluation of our algorithm's performance, we compared it against the following baseline algorithms:

- **Multi-Agent Deep Policy Gradient Optimization (MADPG)** We compare our algorithm against our own implementation of a Multi-Agent Deep Policy Optimization algorithm. In this baseline, the cost function is added to the reward function at each time step. This approach represents a standard way of incorporating cost considerations into reinforcement learning, allowing us to isolate the effect of our Lagrangian-based constraint handling method.

### Implementation Details

The proposed algorithm and the baseline algorithms were implemented using Python, PyTorch, and NumPy. For both the policy and Q-networks, we employed simple neural networks with a single hidden layer. Each agent uses four neural networks (in the case of CMADPG without a Q-network for learning cost) and six neural networks (for CMADPG with a Q-network for learning cost). Our implementation utilizes an actor-critic architecture, with target networks for both the Q-networks and the policy networks, to promote training stability. We deliberately chose this simple network architecture to effectively evaluate the performance of our algorithm. Data visualization was performed using TensorBoard. The algorithms were trained on default CPUs with a single core on the institute's HPC for 10 to 15 hours, due to the small number of learnable parameters. The training process involved running the algorithms for 4.5 million to 5 million steps.During training, we

used a simple replay buffer and a batch size of 1024. The policy networks were updated every 100 episodes (where each episode consists of 25 steps). To monitor the evolution of the learned policies, we visualized 5 episodes every 10,000 steps. We used TensorBoard to log key training parameters, including expected reward and cost.

The complete codebase for this project, including detailed implementation specifics, is available on GitHub at https://github.com/DhruvaBhardwaj404/Constrained-Multi-Agent-Deep-Policy-Gradient-Optimization-Algorithm.

# Chapter 4

# Results

This section presents the results of the experiments conducted to evaluate the performance of the proposed Safe Multi-Agent Reinforcement Learning Algorithm. The results are presented in terms of the evaluation metrics defined in the previous section, and comparisons are made with the baseline algorithms.

## 4.1 Overall Performance

This section presents the overall performance of the proposed CMADPG algorithm, focusing on the impact of incorporating the Q-network for cost learning. We evaluate the algorithm's ability to balance reward maximization and cost minimization, and analyze the effectiveness of the Lagrangian multiplier in satisfying safety constraints.

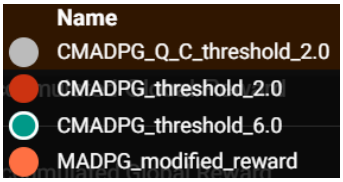


Figure 4.1: Labels

### Reward

When the CMADPG algorithm is implemented without the Q-network for cost learning, this constraint adherence comes at the cost of reduced reward. As the algorithm prioritizes safety, it forgoes potentially higher-reward actions that could lead to constraint violations. This trade-off between safety and reward is an expected consequence of explicitly addressing cost constraints.

when the CMADPG algorithm is implemented with the Q-network for cost learning, we see a substantial improvement in overall performance. Crucially, it also achieves a reward level that is comparable to the baseline performance. This indicates that by learning an accurate estimate of the cost through the Q-network, the algorithm can make more informed decisions, balancing safety and reward more effectively. The Q-network allows the agent to anticipate future costs, leading to policies that are both safe and performant.

### Cost

Our experiments demonstrate that the CMADPG algorithm, in both configurations (with and without the Q-network for cost), successfully learns policies that respect the cost constraints. However,

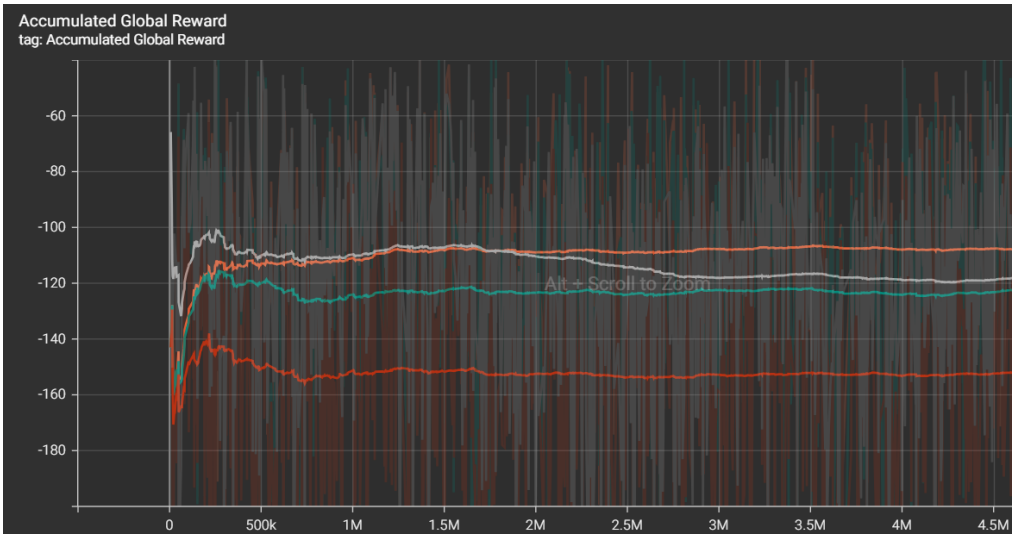


Figure 4.2: Total Reward Curves

the method by which the cost is handled significantly influences the resulting reward.

When the CMADPG algorithm is implemented without the Q-network for cost learning, we observe that the algorithm effectively keeps the expected cost below the required threshold. This indicates that the Lagrangian method successfully constrains the policy to avoid unsafe behaviors.

In contrast, when the CMADPG algorithm is implemented with the Q-network for cost learning, the algorithm maintains the expected cost below the required threshold, demonstrating effective constraint satisfaction.

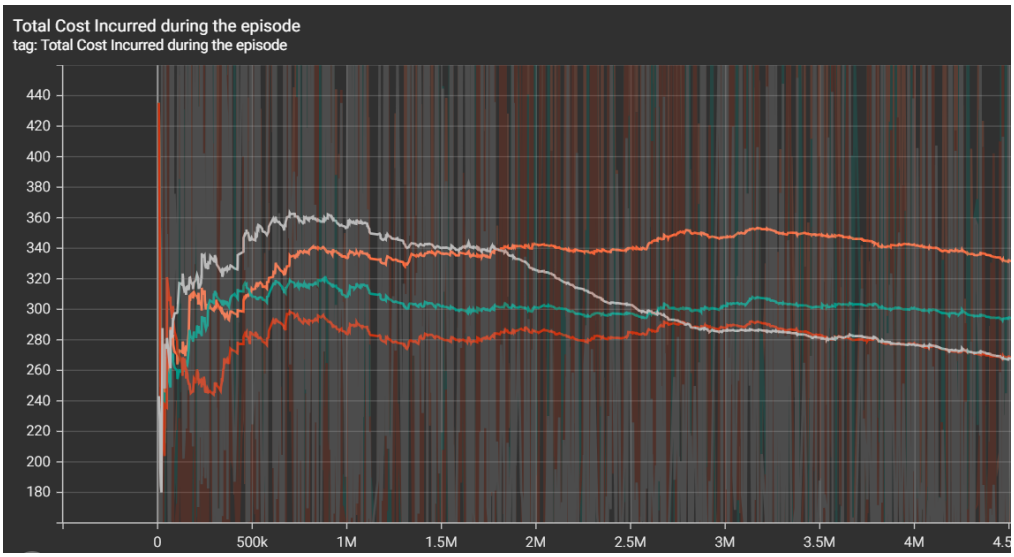


Figure 4.3: Total Cost

## 4.2 Discussion of Results

Our experiments revealed several key insights:

- Incorporating the cost function using the Lagrangian method effectively constrained the agents' policies, ensuring the overall cost remained within the threshold.

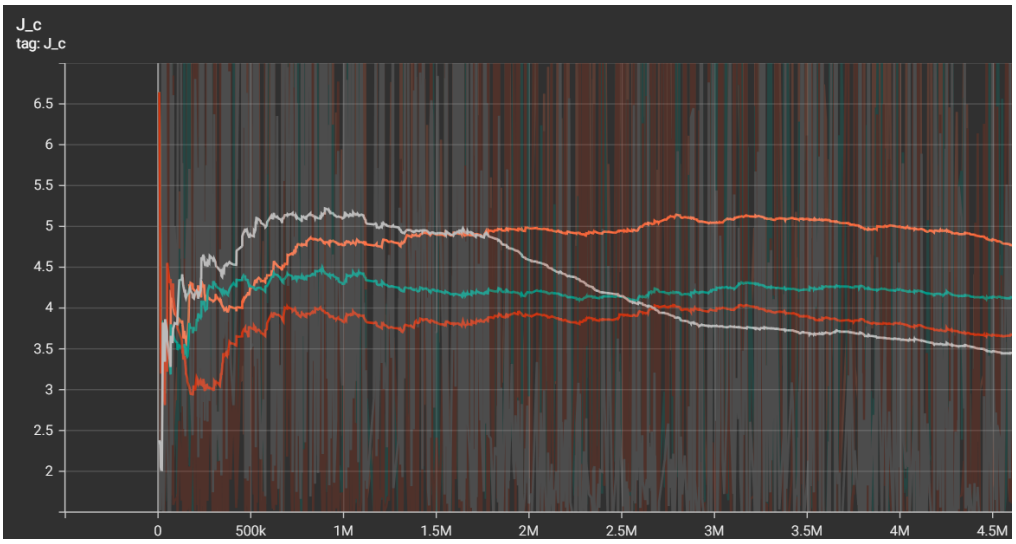


Figure 4.4: Expected Cost

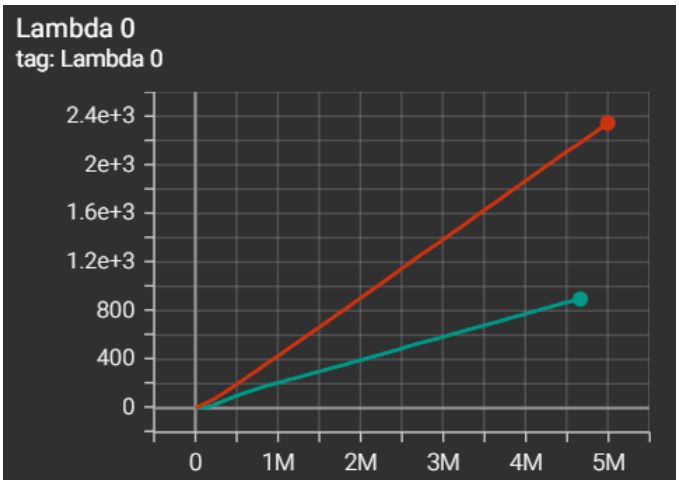


Figure 4.5: Enter Caption

- As expected, this safety-first approach resulted in a lower reward compared to the unconstrained baseline.
- A sufficiently large learning rate is crucial for the constrained algorithm to effectively incorporate cost considerations.
- A smaller replay buffer, specifically one with a size of 5000, facilitates better policy adaptation than larger buffers.
- Using a larger batch size, such as 1024, contributes to more stable training.
- Keeping the learning rate of lambda lower than that of policy parameters helps the policy to converge well, but if the learning rate is very low than the policy isn't able to adapt to the cost threshold.
- If the cost is comparatively very less than reward at every step then the algorithm isn't able to adapt to the required threshold.
- The algorithm needs to be trained for higher number of iterations as lambda values converge slowly.

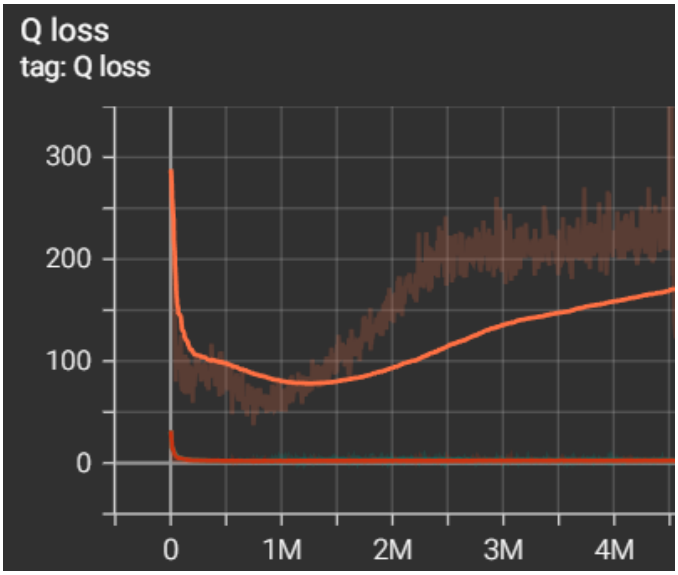


Figure 4.6: Q Loss

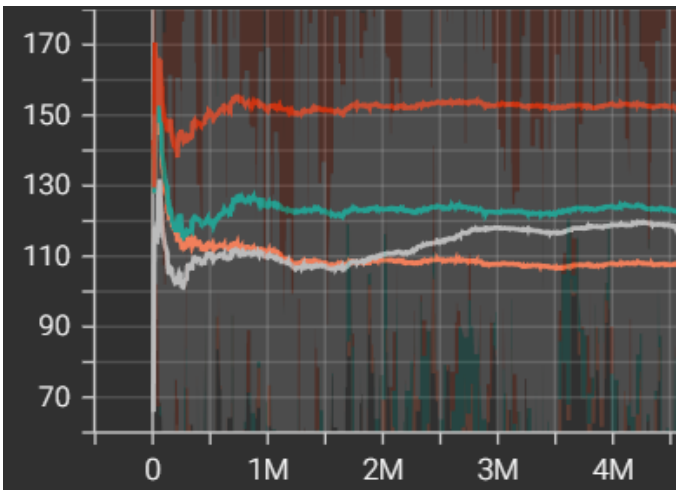


Figure 4.7: Mean Distance from Landmark

- CMADPG algorithms learn the Q function well and thus the policy is expected to be better while MADPG fails to do so.

# Chapter 5

# Discussion

This section provides a comprehensive discussion of the results presented in the previous section. It analyzes the key findings, interprets their implications, and relates them back to the research questions and objectives outlined in the introduction. It also discusses the limitations of the current work and suggests potential directions for future research.

## 5.1 Analysis of Key Findings

In this project, we developed a safe multi-agent reinforcement learning algorithm by incorporating a Lagrangian multiplier to explicitly address safety constraints during training along with policy gradient approach. The algorithm learns decentralized policies that effectively balance the trade-off between reward and safety in multi-agent environments.

### Performance of the Proposed Algorithm

The proposed algorithm demonstrated a strong ability to learn safe policies in the multi-agent environments. By integrating safety constraints directly into the learning process, the algorithm achieved improved performance (in terms of balancing both cost and reward) compared to methods that does not address safety directly. The algorithm effectively satisfied the safety constraints, ensuring that the overall cost remained within the specified threshold.

### Comparison with Baseline Algorithms

When compared to the Multi-Agent Deep Policy Optimization algorithm where the cost function is added to the reward function, our algorithm showed a superior ability to adhere to safety constraints. While incorporating the cost function into the reward function did allow the baseline algorithm to take cost into account, it was less effective in consistently satisfying the safety. Our Lagrangian-based approach allowed for a more direct control over constraint satisfaction and also gave us more control over the tolerable limits of cost which helped us learn policy as per the given cost threshold.

### Interpretation of Results

The results highlight the importance of explicitly addressing safety constraints during the training process in multi-agent reinforcement learning. Our algorithm's ability to learn safe policies with improved performance demonstrates the effectiveness of the Lagrangian multiplier method for handling safety constraints. The findings suggest that integrating safety constraints directly into the learning process enables agents to learn safer policies compared to methods that treat safety as a secondary concern.

## 5.2 Limitations and Future Work

This work has several limitations, but it also opens up several avenues for future research.

### Limitations

The experiments were conducted in simulated environments, which may not fully capture the complexities of real-world scenarios. The algorithm's performance in real-world applications may differ from the simulated results. Additionally, the algorithm's scalability to a large number of agents and complex environments needs further investigation.

### Future Work

Based on your project's objectives and the typical structure of a technology project report, here are some potential directions for future work. I've drawn these suggestions from the "Masters Tech Report Guide" and the "Introduction for Technology Project Report":Future WorkHere are some potential avenues for future research and development, based on your project:

- **Real-World Validation:** Evaluate the algorithm's performance in real-world robotic systems. This would involve deploying the algorithm on physical robots and testing its ability to handle the complexities and uncertainties of real-world environments. This is directly related to your project objective of evaluating performance in real-world settings.
- **Extension to More Complex Environments:** Extend the algorithm to more complex and dynamic multi-agent environments, such as those with continuous state and action spaces, partial observability, or non-stationary agents.
- **Scalability Improvements:** Investigate methods to improve the algorithm's scalability to a larger number of agents. This could involve exploring decentralized training methods, hierarchical control structures, or more efficient approximation techniques.
- **Advanced Optimization Techniques:** Explore and incorporate more advanced optimization techniques, such as trust region methods, proximal policy optimization, or evolutionary algorithms, to further improve the algorithm's performance, stability, and convergence. This aligns with your project objective to enhance performance with advanced optimization.
- **Probabilistic Safety Guarantees:** Develop more sophisticated techniques for providing probabilistic estimates of the safety of learned policies. This could involve exploring methods such as scenario optimization, formal verification, or Bayesian approaches to quantify uncertainty in safety assessments. This relates to your objective of developing a method for probabilistic safety estimates.
- **Multi-Constraint Handling:** Investigate the algorithm's ability to handle multiple simultaneous safety constraints. Explore methods for balancing competing safety objectives and ensuring that the algorithm satisfies all constraints. This also aligns with your project objective.
- **Safe Exploration:** Develop safer exploration strategies that allow agents to explore their environment more effectively while minimizing the risk of constraint violations. This is a critical area for safe reinforcement learning, as standard exploration methods can lead to unsafe behavior.

- **Comparison with other Safe RL methods:** Compare the proposed algorithm with other state-of-the-art safe reinforcement learning algorithms, such as those based on Barrier Functions or Lyapunov methods.

# Bibliography

[1] Arambam James Singh and Arvind Easwaran. Probably approximate safety verification of reinforcement learning policy using scenario optimization.

[2] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments.

[3] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization.

[4] Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation.

[5] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.