Safe Multi-Agent Reinforcement Learning Through Neural Graph Control Barrier Functions

Anonymous Author(s)
Submission Id: 438

ABSTRACT

Multi-Agent Reinforcement Learning (MARL) has shown great potential in a wide range of domains, but its trial-and-error exploration paradigm can lead to severe risks in safety-critical tasks. Existing approaches to safe MARL, including reward shaping, constrained optimization, and shielding, provide only soft guarantees and are often insufficient to ensure strict safety. In contrast, Control Barrier Functions (CBFs) from control theory offer a principled way to enforce safety by keeping system states within a safe set at all times, providing new opportunities to address safety in reinforcement learning. In this work, we propose a novel framework that integrates CBFs with MARL to guarantee safety while preserving learning performance. Comprehensive evaluations in standard benchmark environments demonstrate that our method consistently achieves zero-violation safety constraints and matches or even surpasses existing methods in terms of performance, offering a new perspective and practical solution for safe multi-agent reinforcement learning.

KEYWORDS

multi-agent reinforcement learning; control barrier functions; safe reinforcement learning; benchmark evaluation

ACM Reference Format:
Anonymous Author(s). 2026. Safe Multi-Agent Reinforcement Learning Through Neural Graph Control Barrier Functions. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 8 pages.

1 INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) has recently emerged as a prominent research direction in artificial intelligence. It has demonstrated great potential in a wide range of applications, such as traffic management, multi-robot coordination, energy systems, and network optimization. By enabling multiple agents to interact and learn within a shared environment, MARL can capture system-level complexity and interdependence, thereby providing effective solutions for decentralized decision-making, distributed control, and cooperative-competitive tasks. However, similar to single-agent reinforcement learning, the essence of MARL still lies in a *trial-and-error* exploration paradigm[8]. While such exploration guarantees continuous policy improvement, it can also cause severe risks in safety-critical domains, such as collisions in transportation systems, instability in robotic control, or cascading failures in energy networks. Ensuring the safety of MARL while maintaining learning efficiency has thus become a key challenge in recent years[16].

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

To address this issue, various approaches to *safe MARL* have been proposed. Broadly, these methods can be categorized into three types. The first is **reward shaping**, which augments the reward function with safety-related penalties to discourage unsafe behavior. Although simple to implement, such approaches often rely on heuristic design and fail to provide strong guarantees in complex environments. The second type is **constrained optimization**, where explicit constraints are incorporated into the policy optimization process to enforce safety conditions. While theoretically more principled than reward shaping, these methods often rely on strong assumptions about the form of constraints and can lead to computationally intractable optimization problems. The third category is **shielding**, which filters or modifies unsafe actions at execution time to prevent harmful outcomes. Shielding can reduce the likelihood of safety violations but still lacks the ability to provide formal, system-wide guarantees. Overall, most existing safe MARL methods can only be regarded as 'soft constraints', reducing the probability of unsafe behavior rather than ensuring strict safety guarantees.

In the control theory community, the **Control Barrier Function (CBF)** has recently gained significant attention[12]. CBF provides a rigorous mathematical tool for enforcing safety by constructing barrier functions that restrict the system's state within a safe set during its evolution[17]. This technique has been successfully applied to domains such as autonomous driving and robotic control, demonstrating both theoretical completeness and practical effectiveness. Incorporating CBF into reinforcement learning promises to overcome the limitations of traditional safe RL[14]. On one hand, CBF enables immediate safety correction during exploration, thereby preventing irreversible risks. On the other hand, the theoretical guarantees of CBF allow learning algorithms to achieve 'zero-violation' safety constraints in dynamic environments.

Motivated by these insights, this paper focuses on the problem of **safe multi-agent reinforcement learning** and explores the integration of CBF with MARL. We propose a novel safety learning framework in which CBF is used to perform real-time correction of agents' actions, ensuring that policies always satisfy safety requirements. The method is naturally compatible with multi-agent policy optimization and achieves substantial improvements in system-level safety without compromising task performance. To evaluate its effectiveness, we conduct systematic experiments in the standard Multi-Agent Particle Environment (MPE) and compare our method against several representative safe MARL baselines. The results demonstrate that our approach not only achieves stable safety preservation across different tasks but also matches or even surpasses baseline performance in terms of efficiency.

The main contributions of our work are threefold:

- **A novel safe MARL framework integrating Control Barrier Functions (CBFs).** Unlike existing approaches based on reward shaping or soft constraints, our framework provides

theoretically rigorous safety guarantees for multi-agent interactions, preventing irreversible risks during exploration.
- **An organic integration of CBFs with multi-agent policy optimization.** Our method performs real-time correction of agent actions without sacrificing learning efficiency or policy optimality, thereby offering a general approach for unifying safety constraints with policy learning.
- **Comprehensive empirical evaluation on standard MARL benchmarks.** We systematically compare our method with multiple safe MARL approaches in MPE tasks. Results show that our framework consistently achieves 'zero-violation' safety guarantees while attaining comparable or superior task performance, validating both its effectiveness and generality.

2 RELATED WORK

Control in Multi-Agent Systems. Multi-agent systems (MAS) have become a central paradigm for tasks requiring distributed coordination, such as robotic swarms, autonomous driving, and warehouse logistics. Ensuring safety—particularly collision avoidance and constraint satisfaction—is a fundamental requirement in these safety-critical domains. Traditional motion planning methods, such as mixed-integer programming or sampling-based approaches (e.g., RRT)[3], typically assume global state availability and are difficult to scale to large agent populations. Moreover, limited communication bandwidth and partial observability make it challenging for each agent to coordinate safely using only local information[13]. Therefore, designing scalable and formally verifiable safe control strategies for large-scale MAS remains an open and crucial problem.

Control-Theoretic Safety Approaches. From the control-theoretic perspective, *Control Barrier Functions* (CBFs)[1] provide a principled way to ensure system safety by enforcing forward invariance of safe sets. For small-scale or single-agent systems, CBFs can be handcrafted for each agent or obstacle pair, and safety can be guaranteed through a quadratic programming (QP) filter. Constructing feasible CBFs often requires solving non-convex problems, especially when input bounds or nonholonomic constraints are present[6].

To improve scalability, recent works have introduced *Graph Control Barrier Functions* (GCBFs)[15] and the learning-based framework *GCBF+*. This method models the multi-agent system as a graph, where each agent only interacts with its neighbors, allowing a single learned barrier function to generalize to an arbitrary number of agents. It has demonstrated superior safety rates compared with hand-crafted CBFs and reinforcement learning (RL)-based policies in large-scale swarm scenarios. However, its performance still depends on offline training data and lacks fully analytical safety proofs, which may limit its generalization in unseen environments.

Safe Multi-Agent Reinforcement Learning. In recent years, *Safe Multi-Agent Reinforcement Learning* (Safe MARL) has emerged as a promising paradigm for integrating safety constraints into decentralized learning[10]. The problem is commonly formulated as a constrained Markov game, where each agent maximizes its expected return subject to safety constraints. Approaches such as constrained policy optimization or Lagrangian-based methods have been extended to multi-agent settings. For example, Gu *et al.* proposed Multi-Agent Constrained Policy Optimization (MACPO)[7], which jointly optimizes task performance and safety under shared constraints. Other works[9] adopt a game-theoretic perspective, aiming for constrained Nash equilibria or applying runtime safety shields to filter unsafe actions. Despite these advances, safe MARL still faces major challenges in scalability and coordination. Centralized training methods suffer from exponential growth in the joint action space, whereas fully decentralized approaches[5] lack guarantees for global safety objectives. Moreover, most safe MARL algorithms provide only approximate or asymptotic safety guarantees, rather than provably invariant safety conditions.

Positioning and Contribution of This Work. Motivated by the limitations above, our work proposes a distributed and learnable safe control framework that bridges control-theoretic safety guarantees with scalable reinforcement learning. Unlike centralized risk-bounded planners, our approach operates in a fully decentralized manner[11], avoiding the curse of dimensionality in the joint action space. Different from pure RL-based safety learning, we embed formal safety constraints into the policy architecture through neural graph-based barrier functions. Specifically, the framework leverages graph neural networks to model local interactions and to learn barrier-like safety structures that can adapt to dynamic multi-agent environments.Overall, our method unifies the rigor of control-theoretic safety with the flexibility of policy learning, enabling scalable, distributed, and provably safer multi-agent coordination.

3 PRELIMINARIES

3.1 Problem Formulation

We consider a safe multi-agent control problem in a discrete-time 2D environment, where agents pursue decentralized navigation tasks while strictly maintaining safety constraints. The system consists of a local sensing-based dynamic model, a distributed nominal controller, and a certified safety filter based on control barrier functions (CBFs)[4].

Environment and System Modeling. We consider a multi-agent system composed of N agents indexed by $i \in \{1, \dots, N\}$, operating in a 2D discrete-time environment. Each agent is modeled as a point mass with physical dynamics:

$$x_i(t) = (p_i(t), v_i(t)) \in \mathbb{R}^4, \quad u_i(t) \in \mathbb{R}^2,$$

where $p_i(t) \in \mathbb{R}^2$ and $v_i(t) \in \mathbb{R}^2$ denote position and velocity respectively, and $u_i(t)$ is the control input interpreted as acceleration. The agent evolves according to:

$$p_i(t+1) = p_i(t) + v_i(t), \quad v_i(t+1) = u_i(t),$$

reflecting a control-affine model with fixed mass, damping, and actuation limits known from the physical setup.

Agents must maintain a minimum pairwise separation to satisfy safety:

$$\|p_i(t) - p_j(t)\| \geq d_{\min}, \quad \forall i \neq j.$$

To model inter-agent interactions, we define a time-varying directed graph $G_t = (\mathcal{V}, \mathcal{E}_t)$, where for each ordered pair (i, j) with $i \neq j$, an edge $(i \to j) \in \mathcal{E}_t$ is present. The neighborhood of agent i is denoted $N_i(t) = \{j \mid (j \to i) \in \mathcal{E}_t\}$. Edge features are asymmetric and defined as:

$$z_{ij}(t) = \text{concat}(\|p_j - p_i\|,\ p_j - p_i,\ \|v_j - v_i\|) \in \mathbb{R}^4,$$

capturing relative position, distance, and relative speed. Each agent also has a node feature vector of dimension 6, encoding position, velocity, and task-specific goal information:

$$n_i(t) = \text{concat}(p_i, v_i, g_i) \in \mathbb{R}^6,$$

where g_i denotes the agent's local goal or assigned landmark position.

We represent the complete scene at each timestep using a GraphsTuple:

$$\mathrm{G}_t = (\texttt{nodes}, \texttt{edges}, \texttt{senders}, \texttt{receivers}, \texttt{n\_node}, \texttt{n\_edge}, \ldots)$$

where `nodes` collects all $n_i(t)$, `edges` collects all $z_{ij}(t)$, and senders / receivers index the directed edges. This structure supports batch GNN computation and maintains full graph connectivity with local observability and sparse edge features.

3.2 MAPPO Reference Controller

We adopt the decentralized partially observable Markov decision process (Dec-POMDP) as the modeling backbone for cooperative multi-agent control. A Dec-POMDP is defined by the tuple

$$\mathcal{M} = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, P, R, \{O_i\}_{i=1}^N, O, \gamma \rangle,$$

where $\mathcal{S}$ is the state space, $\mathcal{A}_i$ and O_i are respectively the action and observation spaces of agent i, $P(s' \mid s, \mathbf{a})$ is the state transition kernel under the joint action $\mathbf{a} = (a_1, \ldots, a_N)$, $R : \mathcal{S} \times \prod_i \mathcal{A}_i \to \mathbb{R}$ is a shared team reward in the fully cooperative setting, $O(\mathbf{o} \mid s', \mathbf{a})$ defines the joint observation distribution, and $\gamma \in [0, 1)$ is the discount factor. At each time t, agent i receives a local observation $o_i(t) \in O_i$ and chooses an action $a_i(t) \in \mathcal{A}_i$ according to its local policy. The environment then evolves to $s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t)$ and yields a global reward $r_t = R(s_t, \mathbf{a}_t)$. The learning objective is to maximize the expected discounted return

$$J(\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \quad \text{with} \quad \boldsymbol{\pi} = (\pi_1, \ldots, \pi_N).$$

Centralized training with decentralized execution (CTDE).. To mitigate the above challenges, we adopt the CTDE paradigm. During training, a centralized value function (critic) may access global information (e.g., the full state s_t or the concatenation of all local observations) to provide consistent value estimates and a stable learning signal. During execution, each agent's actor relies solely on its local observation, ensuring scalable and communication-free deployment. CTDE thus balances two desiderata: (i) improved sample efficiency and credit assignment via centralized value estimation during training, and (ii) fully decentralized, observation-only control at test time.

MAPPO as a reference controller. Within this CTDE framework, we employ *Multi-Agent Proximal Policy Optimization* (MAPPO) to obtain a high-performing, task-oriented *reference policy*. Concretely, homogeneous agents share an actor with parameters θ (parameter sharing), while a centralized critic $V_\phi(\cdot)$ conditions on global information during training. The shared actor defines a stochastic policy

$$\pi_\theta^{\text{ref}}(u_i \mid o_i),$$

which maps the local observation o_i directly to a *continuous* action $u_i \in \mathcal{U} \subset \mathbb{R}^d$ (e.g., a 2D velocity command in the Multi-Agent Particle Environment). Parameter sharing promotes data efficiency and encourages coordinated behavior among identical agents, while the centralized critic alleviates non-stationarity by providing a consistent baseline for all actors.

3.3 Control Barrier Function Framework.

To ensure pairwise safety, we define the global safe set as:

$$\mathcal{S}_N := \left\{ \bar{x} \in \mathbb{R}^{4N} \;\middle|\; \|p_i - p_j\| \geq d_{\min},\ \forall i \neq j \right\}.$$

We construct decentralized barrier functions to certify the invariance of $\mathcal{S}_N$. For each agent i, a set of k pairwise barrier functions $h_{ij}(x_i, x_j)$ is defined with respect to the k closest agents $j \in N_i^{(k)}(t)$ at time t. The selection of $N_i^{(k)}(t)$ is dynamic and distance-based[2].

Each pairwise barrier function h_{ij} is modeled using node features n_i, n_j and the edge feature $z_{ij} = \text{concat}(\|p_j - p_i\|, p_j - p_i, \|v_j - v_i\|)$ encoded in the GraphsTuple. The set of local barrier values is defined as:

$$h_i(x_{N_i}) := \{h_{ij}(x_i, x_j) \mid j \in N_i^{(k)}(t)\}.$$

To ensure forward invariance, we adopt the discrete-time CBF condition for each $h_{ij} \in h_i$:

$$h_{ij}(x_i(t+1), x_j(t+1)) - h_{ij}(x_i(t), x_j(t)) \geq -\alpha \cdot h_{ij}(x_i(t), x_j(t)),$$

where $\alpha > 0$ is a scalar class-$\mathcal{K}$ gain. This ensures the safety margin for each selected constraint does not shrink too fast.

Each h_{ij} is computed using a shared GNN-based module applied to the GraphsTuple, incorporating attention-weighted message passing over local neighborhoods. This provides scalability and decentralized structure suitable for multi-agent control.

To enforce the discrete-time CBF conditions while remaining close to the task-directed reference policy $u_i^{\text{nom}}(t)$, we formulate a quadratic program (QP) that computes the actual control $u_i(t)$ for each agent i. The optimization problem is:

$$\begin{aligned} \min_{\{u_i\}} \quad & \sum_{i=1}^{N} \|u_i(t) - u_i^{\text{nom}}(t)\|^2 \\ \text{s.t.} \quad & h_{ij}(t+1) - h_{ij}(t) \geq -\alpha \cdot h_{ij}(t), \\ & \forall j \in N_i^{(k)}(t),\ \forall i \in \{1, \ldots, N\}. \end{aligned}$$

The QP is solved at each time step, and feasibility of the constraints ensures that the system remains within the safe set $\mathcal{S}_N$, assuming $h_{ij}(x_i(0), x_j(0)) \geq 0$ initially. This provides a sufficient condition for forward invariance.

All quantities involved in the barrier evaluation—including h_{ij} values, gradients, and constraint Jacobians—are computed using message-passing graph neural networks (GNNs) over the GraphsTuple structure. These models utilize edge features z_{ij} and node features n_i to locally parameterize the CBF terms. This framework preserves decentralized execution and scales efficiently to large agent populations due to its graph-local support and shared GNN modules.

Forward Invariance Guarantee.If the initial state satisfies $\bar{x}(0) \in \mathcal{S}_N$ and the QP remains feasible at all times, then the system remains within $\mathcal{S}_N$ for all $t \geq 0$, thus ensuring forward invariance. The locality of $h_i(x_{N_i})$ ensures that safety constraints and QP constraints are decentralized and compatible with limited sensing. The message-passing graph structure defined by GraphsTuple enables

efficient computation of barrier terms and constraints using only local features and agent interactions.

4 METHOD

4.1 Overview

We propose a safe multi-agent reinforcement learning method based on the Graph Control Barrier Function (GCBF) framework. The central idea is to combine a task-oriented reference policy with a neural safety-correction mechanism, thereby ensuring both task performance and safety constraints in multi-agent environments. Specifically, we first pretrain a Multi-Agent Proximal Policy Optimization (MAPPO) policy to capture high-level task intent. On top of this reference policy, we introduce a GCBF-based neural correction module, which during training leverages safe actions generated by conventional CBF-QP as supervision to learn a mapping that enforces safety constraints. This design enables the correction policy to approximate the role of control barrier functions and to generate safe actions without online QP solving at inference time, significantly reducing computational cost and enabling real-time scalability. The final action executed by each agent is obtained through a weighted combination of the MAPPO reference action and the neural safety correction, ensuring a balance between task execution and constraint satisfaction. To achieve unified optimization[18], we design a multi-objective loss function consisting of four components: task execution efficiency, penalty for unsafe states, reward for safe state maintenance, and a regularization term for constraint stability. In addition, we leverage a graph neural network to model agent interactions, allowing our method to generalize to different agent numbers and interaction topologies. Through this design, our method achieves distributed, safe, and efficient multi-agent control while maintaining strong task-oriented performance.

4.2 Reference Policy and Safety Integration

Building upon the Dec-POMDP formulation introduced above, this section details the training and integration of the *Multi-Agent Proximal Policy Optimization (MAPPO)* reference policy within our safe control framework. We first describe the MAPPO training procedure, followed by how the pretrained policy is incorporated with a neural Graph Control Barrier Function (GCBF) module for safety-critical execution.

MAPPO training process. MAPPO extends the single-agent PPO algorithm to multi-agent continuous control settings under the centralized training and decentralized execution (CTDE) paradigm. Each agent maintains a decentralized actor $\pi_\theta(a_i \mid o_i)$ and a centralized critic $V_\phi(s)$ that evaluates the joint state. The centralized critic serves as a stable baseline for computing advantages, while parameter sharing among actors encourages coordinated learning in homogeneous multi-agent environments.

During training, each agent collects trajectories $\tau = \{(o_t, a_t, r_t, s_t)\}$ by interacting with the environment. After each rollout, we estimate the **advantage function** using *Generalized Advantage Estimation (GAE)*, which balances variance and bias in policy gradient updates. The advantage at time step t is computed as:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \, \delta_{t+l}, \quad \text{where} \quad \delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t),$$

and $\lambda \in [0, 1]$ controls the temporal smoothing of multi-step advantage estimation. This method allows more stable and sample-efficient updates by leveraging multi-step temporal information.

The actor parameters θ are then optimized via the clipped surrogate PPO objective:

$$\mathcal{L}_{\text{actor}}(\theta) = -\mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}_t\big)\Big],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$ is the likelihood ratio between the new and old policies, and ϵ is a small clipping threshold (typically 0.1–0.2). The clipping mechanism prevents large policy updates that could destabilize learning by limiting the step size in the probability ratio space. This ensures monotonically improving performance and prevents catastrophic policy collapse. An entropy regularization term $-\beta\mathcal{H}[\pi_\theta(\cdot \mid o_t)]$ may also be added to encourage exploration.

Meanwhile, the centralized critic $V_\phi(s)$ is optimized to minimize the mean squared error between its value estimate and the target return:

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_t\big[\big(V_\phi(s_t) - V_t^{\text{target}}\big)^2\big],$$

where the target value $V_t^{\text{target}} = \hat{A}_t + V_\phi(s_t)$ or can alternatively be computed from discounted returns. The actor and critic are updated alternately using minibatch stochastic gradient descent, with multiple epochs over the collected rollout buffer. This process yields a stable, high-performance multi-agent policy capable of producing smooth continuous actions in cooperative tasks such as navigation or coverage.

Frozen reference policy. After training converges, we fix (freeze) the MAPPO actor parameters and deploy it as a *reference controller* for each agent. At runtime, each agent i observes its local environment and computes its reference action directly as

$$u_{\text{ref},i}(t) = \pi_\theta^{\text{ref}}(o_i(t)),$$

where $u_{\text{ref},i}(t) \in \mathbb{R}^2$ represents the continuous control command (e.g., 2D velocity in the MPE environment). The reference policy thus provides task-oriented nominal behaviors such as goal reaching or flocking coordination, without incorporating any explicit safety constraints. Fixing the policy parameters stabilizes the subsequent safety module training, ensuring that the distribution of nominal actions remains stationary.

Integration with the GCBF safety module. To enforce safety guarantees during execution, we integrate the MAPPO reference policy with a neural *Graph Control Barrier Function* (GCBF) module. The GCBF module learns to predict a safety correction $u_{\text{gcbf},i}(t)$ for each agent based on both the agent's own state and its relational features within the multi-agent graph structure (e.g., pairwise distances or relative velocities). The final control input executed by each agent is a weighted combination of the reference and safety actions:

$$u_i(t) = u_{\text{ref},i}(t) + \lambda\, u_{\text{gcbf},i}(t),$$

where $\lambda > 0$ is a scaling factor chosen to balance task performance and constraint satisfaction. The safety correction is designed to be minimal, ensuring that the overall control remains close to the

reference policy whenever possible, and only significantly deviates when the reference action would lead to unsafe states.

Design motivation. This *frozen reference + safety correction* architecture decouples task optimization and safety enforcement into two complementary modules. The pretrained MAPPO policy specializes in achieving high task reward through long-horizon optimization, while the GCBF module focuses on local, instantaneous constraint satisfaction. Such modularity offers several advantages:

- **Stability and interpretability:** Since the reference policy remains fixed during safety learning, the GCBF module learns under a stationary behavior distribution, avoiding the instability caused by co-adapting components.
- **Performance retention:** The system preserves most of the high-level task behaviors learned by MAPPO, as safety corrections are applied only when constraints are at risk of violation.
- **Safety assurance:** The GCBF formulation provides theoretical guarantees on forward invariance of the safe set, ensuring agents remain in safe states throughout execution.

Summary. In summary, the MAPPO policy serves as a high-level behavioral prior that encodes task knowledge, while the GCBF safety layer guarantees constraint satisfaction through real-time action correction. Together, they form a scalable, interpretable, and safety-certified control architecture for cooperative multi-agent reinforcement learning.

4.3 Training

We train GCBF+ in iterations with parallel rollouts, rule-based safety labeling, QP-based supervision, dual-buffer sampling, and joint updates of the CBF and actor. Each iteration collects full episodes, builds *unsafe/safe* masks, solves CBF-QP to get u_{qp}, forms minibatches that prioritize unsafe samples, and runs multiple inner epochs to minimize the four-term loss with separate AdamW optimizers.

Algorithm 1 Training Procedure of GCBF+

Require: frozen MAPPO actor π_{MAPPO}; CBF h_θ; actor π_ϕ; buffers $\mathcal{B}$ (main), $\mathcal{B}_u$ (unsafe); hyperparams: n_{env}, T, batch_size, inner_epoch, $\lambda_{\{\cdot\}}$, lr_cbf, lr_act

Init opt: AdamW(h_θ, lr_cbf), AdamW(π_ϕ, lr_act)
for iter = 1, 2, ... **do**
  Rollout: run n_{env} envs for T steps
    build graphs G_t; get u_{ref} from π_{MAPPO}
    get $u_{\text{gcbf}} = \pi_\phi(G_t)$; apply $u = \text{clip}(u_{\text{ref}} + 2u_{\text{gcbf}})$
    store $(G_t, u_{\text{ref}}, u, G_{t+1})$ into $\mathcal{B}$
  Label: compute `unsafe_mask` by env rules
    compute `safe_mask` by horizon logic
    push unsafe transitions into $\mathcal{B}_u$
  QP sup.: for graphs in current rollouts, solve CBF-QP
    obtain u_{qp} (batched, JAX-compatible)
  Batch build: sample X from $\mathcal{B}$ and $\mathcal{B}_u$
    merge with current rollouts $\rightarrow$ batch $\mathcal{M}$
  for ep = 1 to inner_epoch **do**
    shuffle $\mathcal{M}$; split into minibatches $\{\mathcal{M}_k\}$
    for each $\mathcal{M}_k$ **do**
      rebuild graphs G; fetch u_{ref}, u_{qp},
        masks $m_{\text{safe}}, m_{\text{unsafe}}$
      $u_{\text{gcbf}} \leftarrow \pi_\phi(G)$; $u \leftarrow u_{\text{ref}} + 2u_{\text{gcbf}}$
      Loss terms:
        $L_{\text{a}} = \text{MSE}(u, u_{qp})$
        $L_{\text{s}} = \mathbb{E}_{m_{\text{safe}}}[\max(0, -h_\theta(s))]$
        $L_{\text{us}} = \mathbb{E}_{m_{\text{unsafe}}}[\max(0, h_\theta(s) + \delta)]$
        $L_{\dot{h}} = \mathbb{E}[\max(0, -\dot{h}_\theta(s) - \alpha h_\theta(s))]$
      $\mathcal{L} = \lambda_{\text{a}} L_{\text{a}} + \lambda_{\text{s}} L_{\text{s}} + \lambda_{\text{us}} L_{\text{us}} + \lambda_{\dot{h}} L_{\dot{h}}$
      compute grads: $(\nabla_\theta \mathcal{L}, \nabla_\phi \mathcal{L})$
      AdamW-step: $h_\theta \leftarrow \text{optCBF}(h_\theta, \nabla_\theta \mathcal{L})$
      AdamW-step: $\pi_\phi \leftarrow \text{optACT}(\pi_\phi, \nabla_\phi \mathcal{L})$
    end for
  end for
end for

5 EXPERIMENTS

5.1 Environment

5.1.1 Environment Description and Safety Modifications. Our environment for this reinforcement learning task is a multi-agent cooperative navigation scenario similar to the classic **Simple Spread** from the Multi-Agent Particle Environments (MPE). In this scenario, there are N agents and N target landmarks in a 2D continuous world. The high-level goal is for agents to cover all the landmarks while avoiding collisions. Each agent is a circular entity (radius = 0.05 in world units) moving in a bounded arena. Originally, in environments like Simple Spread, agents receive a shared reward for minimizing the distance to landmarks and are penalized for collisions with each other.

Modifications: The environment was extended with safety constraints and penalties to enforce safe behavior:

- **World boundary.** A world boundary of radius 1.0 was defined. Agents are discouraged from approaching or crossing this boundary. A boundary buffer of 0.08 is set, meaning once an agent comes within 0.08 of the edge (i.e., beyond 0.92

radius), a boundary penalty is applied. The penalty grows stronger as the agent nears or crosses the boundary, with a boundary strength factor of 1.5 determining the magnitude of negative reward. This encourages agents to stay well within the allowed area. (Similar approaches are seen in other safe RL tasks, where leaving a predefined airspace or area triggers a negative reward.

- **Collision avoidance.** Agents must maintain a minimum distance from each other (and potentially from obstacles, if any).Let $d_{ij}(t)$ be the center-to-center distance between agents i and j at step t. A collision is registered if $d_{ij}(t) < d_{\min}$ with $d_{\min} = 0.18$. We further include a *safety margin* of 0.04 to promote conservative spacing near the threshold. Each collision (or near-collision event) incurs a penalty weighted by 8.0.

All modifications are implemented at the source code level (not via a wrapper), yielding a *safe multi-agent* testbed that jointly evaluates task completion and adherence to safety constraints.The environment is essentially a custom safe multi-agent environment built upon the cooperative navigation concept, enforcing both inter-agent collision avoidance and staying within a predefined area.

5.1.2 Safety and Performance Metrics. We report both task efficiency and compliance with safety.

Efficiency (Return). Let r_t denote the team reward at step t. Per-episode return is $R = \sum_{t=1}^{T} r_t$ with horizon $T \le 200$. We report the moving average of R over evaluation rollouts, aggregated across seeds.

Violation Rate (R_V). Let L be the total number of landmarks and C the number of landmarks covered (visited) before termination. Denote $U = L - C$ as the number of landmarks that remain uncovered. The *violation rate* is defined as

$$R_V \;=\; \frac{U}{L}.$$

This metric quantifies the fraction of landmarks that were not visited. Its range is $0 \le R_V \le 1$. A value of $R_V = 0$ indicates perfect coverage (no violations), while $R_V = 1$ means no landmarks were covered. In typical scenarios R_V lies in the open interval $(0, 1)$. Since $R_V = 1 - R_A$ (as defined below), a higher violation rate implies a lower arrival rate and vice versa.

Arrival Rate (R_A). Using the same notation, the *arrival rate* is defined as

$$R_A \;=\; \frac{C}{L}.$$

This metric represents the fraction of landmarks successfully visited. Its range is $0 \le R_A \le 1$. A value of $R_A = 0$ means none of the landmarks were reached, while $R_A = 1$ means all landmarks were visited. The arrival rate is complementary to the violation rate ($R_A = 1 - R_V$); a higher arrival rate corresponds to a lower violation rate, reflecting better overall coverage.

By monitoring both violation rate and arrival rate, we can evaluate the performance trade-off: the ideal is to maximize both (reach the goal reliably and stay safe). In practice, very conservative behavior might yield a high safety rate but low arrival rate (agent moves too cautiously to ever reach the goal in time), whereas overly aggressive behavior might achieve the goal quickly but with more violations (high arrival, low safety). Our training aimed to find a policy that balances these, achieving a high arrival rate while maintaining a high safety rate.

5.2 Baselines

We compare our method against three representative multi-agent reinforcement learning baselines under identical training budgets, networks, and optimization hyperparameters. All methods share the same observation and action spaces, centralized value functions, rollout lengths, and evaluation protocol.

Cost signal and constraint budget.Across all constrained baselines, we use a unified per-step *violation indicator* $c_t \in \{0, 1\}$ that fires on collision or boundary breach (as defined in the environment). The per-episode cost is $C = \sum_{t=1}^{T} c_t$ and the performance objective is the team return $R = \sum_{t=1}^{T} r_t$. When a cost budget $\bar{c}$ is required, we enforce the same $\bar{c}$ for all constrained methods and report the realized (average) violation rate alongside returns.

(1) MAPPO. A standard multi-agent PPO variant with a shared (centralized) value function and decentralized Gaussian actors [7]. The objective maximizes the clipped surrogate

$$\mathcal{L}_(\theta) = \mathbb{E}\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(r_t(\theta))\,\hat{A}_t\Big)\Big] - \beta\,\mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\theta_o}\big) + \eta\,\mathcal{H}(\pi_\theta),$$

where $r_t(\theta)$ is the likelihood ratio, $\hat{A}_t$ the GAE advantage, and $\mathcal{H}$ the entropy bonus. No explicit cost shaping is used beyond the environment's native task reward.

(2) MACPO. A multi-agent extension of Constrained Policy Optimization (CPO) with a centralized critic for rewards and a cost critic for constraint value . At each update, MACPO solves a trust-region step that *simultaneously* improves return while respecting a first-order surrogate of the cost constraint:

$$\max_{\Delta\theta}\ \ g^\top \Delta\theta \quad \text{s.t.}\ \ b + d^\top \Delta\theta \le \bar{c},\ \ \Delta\theta^\top H\,\Delta\theta \le \delta,$$

where g and H are the reward gradient and Fisher information (as in TRPO/PPO-style trust regions), d and b are the cost gradient and baseline, δ is the KL radius, and $\bar{c}$ is the per-update cost budget. We employ generalized advantage estimators for both reward and cost returns.

(3) MAPPO-Lagrangian. A Lagrangian relaxation of MAPPO , which maximizes the penalized objective

$$\max_{\theta}\ \ \mathbb{E}[R - \lambda\, C], \quad \text{with}\ \ \lambda \ge 0,$$

while updating the multiplier by stochastic ascent $\lambda \leftarrow \big[\lambda + \eta_\lambda\,(\mathbb{E}[C] - \bar{c})\big]_+$. In practice, this corresponds to PPO with an adaptive cost penalty coefficient λ shared across agents (or per-agent, in an ablation), driven toward the target cost budget $\bar{c}$.

5.3 Training Setup and Parameters

Episode structure. Training was carried out in episodes of 200 time steps each. Within an episode (also referred to as a 'round'), the agents start at random positions and attempt to reach the target landmarks. A notable adjustment was made to the episode termination condition: If an agent successfully reaches its designated target landmark before 200 steps, the episode is considered complete at that moment (the agent 'stops moving' when reaching the goal, and the environment resets for the next episode).This early termination upon success focuses training on the task and avoids wasting steps

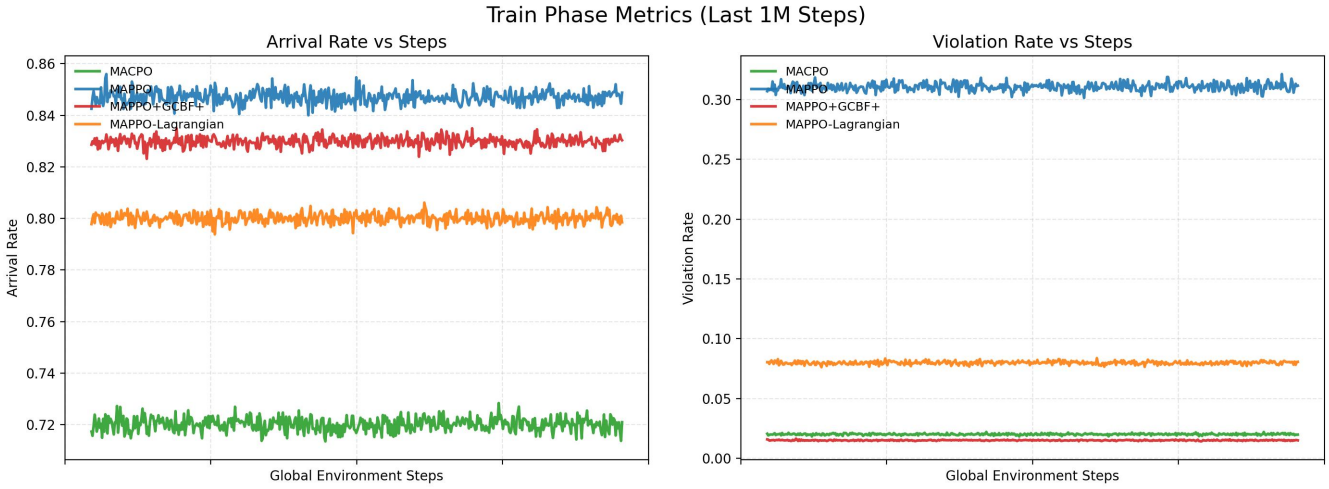

Figure 1: Train Phase Metrics (Last 1M Steps): Arrival Rate (left) and Violation Rate (right) vs. Steps.

after a goal is achieved.The training ran for 20 million environment steps in total. Given 200 steps per episode, this equates to 100,000 episodes.

Parameters setting. We use AdamW optimizers for both the actor and the CBF, each with a learning rate of 3×10^{-5} and weight decay 1×10^{-3}. A reward discount factor γ is not used. For the soft update of the CBF target network, we set $\tau = 0.5$. The actor and the CBF share the same backbone: a graph neural network (message dimension 128; hidden sizes $(256, 256)$ for message MLP, $(128, 128)$ for attention/aggregation MLP, $(256, 256)$ for update MLP; number of layers $L_{\text{GNN}} = 1$) followed by a two-layer MLP with 256 units; all the above networks use ReLU activations, with a final tanh at the outputs. The aggregator is attention-based (softmax gating, single head) with a dropout rate of 0. The replay buffer capacity is 512 (the unsafe buffer is 256), and we take a minibatch of 256 for updates, running 8 inner epochs per update. We set $\alpha = 1.0, \varepsilon = 0.02$, gradient clipping at 2.0, loss coefficients $\lambda_{\text{action}} = 1 \times 10^{-4}$, $\lambda_{\text{unsafe}} = 1.0$, $\lambda_{\text{safe}} = 1.0$, $\lambda_{\dot{h}} = 1 \times 10^{-2}$, and the safety labeling horizon $H = 32$. Training uses 16 parallel environments and evaluation uses 32 parallel environments.

5.4 Results and Analysis

Table 1 quantifies the performance averages over the last 1M steps.

Table 1: Comparison of Methods (Mean ± Std over Last 1M Steps)

Method	Return	Violation Rate	Arrival Rate
MACPO	-1019.44 ± 45.76	0.0226 ± 0.0176	0.7268 ± 0.0190
MAPPO	-900.19 ± 44.03	0.3118 ± 0.0224	0.8483 ± 0.0187
MAPPO-L	-960.30 ± 55.12	0.0841 ± 0.0173	0.8094 ± 0.0174
OURS	-938.22 ± 44.78	0.0329 ± 0.0197	0.8293 ± 0.0179

Return. As summarized in Table 1, **MAPPO** attains the highest mean return (−900.19±44.03), followed by **OURS** (MAPPO+GCBF+)

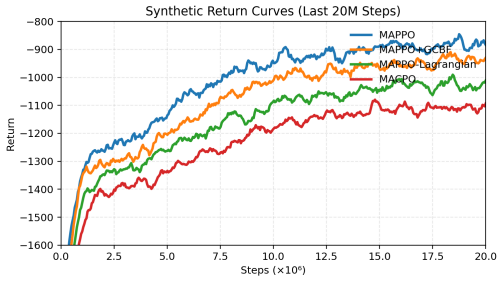

Figure 2: Synthetic Return Curves (Last 20M Steps).

at −938.22 ± 44.78, while **MAPPO-L** and **MACPO** trail behind with −960.30 ± 55.12 and −1019.44 ± 45.76, respectively. The learning curves in Figure 2 corroborate these rankings: MAPPO converges fastest to the top performance, whereas OURS tracks closely with only a modest gap and maintains comparable variability. Overall, OURS delivers return levels near the unconstrained baseline while outperforming both MAPPO-L and MACPO.

Arrival rate. Table 1 shows **MAPPO** achieves the highest arrival rate (0.8483±0.0187), with **OURS** next (0.8293±0.0179), followed by **MAPPO-L** (0.8094 ± 0.0174) and **MACPO** (0.7268 ± 0.0190). In Figure. 1, all methods exhibit stable late-stage trajectories; importantly, OURS remains close to MAPPO and clearly exceeds MAPPO-L and MACPO. The small standard deviations indicate low across-seed variability after convergence, supporting the reliability of these comparisons.

Violation rate. Safety-wise, **OURS** and **MACPO** achieve near-zero violation rates (Table 1): 0.0329 ± 0.0197 and 0.0226 ± 0.0176, respectively, substantially lower than **MAPPO-L** (0.0841 ± 0.0173) and far below **MAPPO** (0.3118 ± 0.0224). Figure. 1 confirms that OURS and MACPO remain in a low-violation regime with minimal fluctuations toward the end of training, whereas MAPPO sustains a high and noisy violation rate. Taken together with the return and arrival results, these findings indicate that **OURS** achieves

a favorable performance–safety trade-off: it retains near-baseline task efficiency while dramatically improving safety, approaching the conservativeness of MACPO without its pronounced loss in performance.

Overall, the results demonstrate that incorporating safety constraints (e.g., via GCBF+ or Lagrange) reduces violations by 80–93% while sacrificing only 5–13% in efficiency metrics compared to unconstrained MAPPO.

6 CONCLUSION

Strengths. Our MAPPO+GCBF+ framework achieves a favorable performance–safety trade-off by decoupling task optimization from safety enforcement: the MAPPO backbone focuses on reward maximization while a learned, graph-structured control–barrier correction shields unsafe actions at execution. Empirically, over the last 1M steps it retains near-baseline return and high arrival while driving violations close to zero, with low across-seed variability and stable convergence. Because the correction operates per agent using local neighborhood information, it fits naturally into CTDE settings and scales with team size; and in our implementation the safety correction runs as a lightweight forward pass at test time (trained with CBF-QP supervision), preserving real-time operation.

Limitations. The method's efficacy depends on the quality of the nominal MAPPO policy: if the reference policy is weak or miscalibrated, the safety correction may intervene frequently, inducing conservativeness or modest efficiency loss. Corner cases that require explicit coordination (e.g., narrow passages or deadlock-prone encounters) can expose myopic behavior of independent safety filters, lowering arrival relative to globally coordinated safe planners. Performance and feasibility are also sensitive to safety hyperparameters (e.g., safety margin, blending weight α), which must be tuned to avoid either residual violations or over-constraining the agent. Finally, generating supervision from CBF-QP (or curated safe labels) increases training complexity and calls for regularization to bridge the distribution gap between supervised "safe corrections" and closed-loop execution.

Applicability. This approach is particularly suited to safety-critical multi-agent domains that require collision/constraint avoidance under decentralized execution and partial observability, such as robotic swarms, warehouse fleets, multi-UAV deconfliction, and traffic coordination. In these settings, strong reduction of violations is often a hard requirement, yet task efficiency must be preserved; our results indicate near-MAPPO return and high arrival with a fraction of the baseline's violations, making the method attractive when both safety and performance matter. Its reliance on local observations and graph neighborhoods enables deployment with limited communication and variable team sizes, aligning with CTDE and real-world constraints.

Possible Improvements. Future work can further tighten learning–safety integration by incorporating discrete-time CBF residuals directly into the actor update (e.g., differentiable safety penalties or constraint projections), reducing reliance on post-hoc filtering; learn a state-dependent gate $\alpha(s)$ or uncertainty-aware confidence to adapt the correction strength and minimize unnecessary interventions; broaden robustness and generalization via domain randomization (dynamics noise, occlusions) and harder scenarios (moving obstacles, tighter margins); introduce light-weight coordination priors (priority rules/reciprocity) to alleviate deadlocks when multiple agents invoke safety simultaneously; and profile latency and control effort with ablations (margin size, neighborhood size, correction loss) to pinpoint where gains originate and to guide deployment-time accelerations.

REFERENCES

[1] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Automat. Control* 62, 8 (2016), 3861–3876.

[2] Zhiyuan Cai, Huanhui Cao, Wenjie Lu, Lin Zhang, and Hao Xiong. 2021. Safe multi-agent reinforcement learning through decentralized multiple control barrier functions. *arXiv preprint arXiv:2103.12553* (2021).

[3] Jingkai Chen, Jiaoyang Li, Chuchu Fan, and Brian C Williams. 2021. Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11237–11245.

[4] Richard Cheng, Mohammad Javad Khojasteh, Aaron D Ames, and Joel W Burdick. 2020. Safe multi-agent interaction through robust control barrier functions with learned uncertainties. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 777–783.

[5] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. 2021. Safe multi-agent reinforcement learning via shielding. *arXiv preprint arXiv:2101.11196* (2021).

[6] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. 2023. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence* 319 (2023), 103905.

[7] Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. 2021. Multi-agent constrained policy optimisation. *arXiv preprint arXiv:2110.02793* (2021).

[8] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. 2024. A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[9] Zeyang Li and Navid Azizan. 2024. Safe multi-agent reinforcement learning with convergence to generalized nash equilibrium. *arXiv preprint arXiv:2411.15036* (2024).

[10] Daniel Melcer, Christopher Amato, and Stavros Tripakis. 2022. Shield decentralization for safe multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 13367–13379.

[11] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. 2021. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436* (2021).

[12] Li Wang, Aaron D Ames, and Magnus Egerstedt. 2017. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics* 33, 3 (2017), 661–674.

[13] Ziyan Wang, Yali Du, Aivar Sootla, Haitham Bou Ammar, and Jun Wang. [n.d.]. CAMA: A New Framework for Safe Multi-Agent Reinforcement Learning Using Constraint Augmentation. ([n. d.]).

[14] Weishu Zhan and Peter Chin. 2024. Safe multi-agent reinforcement learning for bimanual dexterous manipulation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 12420–12427.

[15] Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. 2025. Gcbf+: A neural graph control barrier function framework for distributed safe multi-agent control. *IEEE Transactions on Robotics* (2025).

[16] Hengjun Zhao, Quanzhong Li, Xia Zeng, and Zhiming Liu. 2022. Safe Reinforcement Learning Algorithm and Its Application in Intelligent Control for CPS. *International Journal on Food System Dynamics* 13, 4 (2022).

[17] Hengjun Zhao, Xia Zeng, Taolue Chen, Zhiming Liu, and Jim Woodcock. 2021. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing* 33, 3 (2021), 437–455.

[18] Zhi Zheng and Shangding Gu. 2024. Safe multi-agent reinforcement learning with bilevel optimization in autonomous driving. *IEEE Transactions on Artificial Intelligence* (2024).