DISCRETE GCBF PROXIMAL POLICY OPTIMIZATION FOR MULTI-AGENT SAFE OPTIMAL CONTROL

Songyuan Zhang[†] **Oswin So**[†] **Mitchell Black**[*] **Chuchu Fan**[†]
[†]Department of Aeronautics and Astronautics, MIT [*]MIT Lincoln Laboratory
[†]{szhang21,oswinso,chuchu}@mit.edu [*]mitchell.black@ll.mit.edu

ABSTRACT

Control policies that can achieve high task performance and satisfy safety constraints are desirable for any system, including multi-agent systems (MAS). One promising technique for ensuring the safety of MAS is distributed control barrier functions (CBF). However, it is difficult to design distributed CBF-based policies for MAS that can tackle unknown discrete-time dynamics, partial observability, changing neighborhoods, and input constraints, especially when a distributed high-performance nominal policy that can achieve the task is unavailable. To tackle these challenges, we propose **DGPPO**, a new framework that *simultaneously* learns both a *discrete* graph CBF which handles neighborhood changes and input constraints, and a distributed high-performance safe policy for MAS with unknown discrete-time dynamics. We empirically validate our claims on a suite of multi-agent tasks spanning three different simulation engines. The results suggest that, compared with existing methods, our DGPPO framework obtains policies that achieve high task performance (matching baselines that ignore the safety constraints), and high safety rates (matching the most conservative baselines), with a *constant* set of hyperparameters across all environments.[1] [2]

1 INTRODUCTION

Multi-agent systems (MAS) have gained significant attention in recent years due to their potential applications in various domains, such as warehouse robotics (Kattepur et al., 2018), autonomous vehicles (Shalev-Shwartz et al., 2016), traffic routing (Wu et al., 2020), and power systems (Biagioni et al., 2022). However, a big challenge for MAS is designing distributed control policies that can achieve high task performance while ensuring safety, especially when the two are conflicting. In the single-agent continuous-time case, control barrier functions (CBF) are an effective tool to resolve the conflict via the solution of a safety filter quadratic program (QP) (Xu et al., 2015; Ames et al., 2017), minimally modifying a given performance-oriented nominal policy to be safe. While distributed CBFs have been proposed for multi-agent (Wang et al., 2017) and partially observable cases (Zhang et al., 2025), they have a limitation of requiring *known* continuous-time dynamics and a nominal policy that can achieve high task performance (albeit not necessarily safely).

While the assumptions above are reasonable for many applications, they do not apply when the dynamics are *unknown* and a performance-oriented nominal policy is unavailable. The challenge of requiring a nominal policy has been addressed by approaches that combine CBFs with reinforcement learning (RL) (Cheng et al., 2019; Emam et al., 2022), where the nominal policy is learned via an unconstrained RL algorithm to maximize task performance while the CBF is used as a safety filter to ensure safety. However, these works have only been applied to the single-agent case, and require known control-affine dynamics to ensure the resulting safety filter QP is computationally tractable, which is too strict for most systems, e.g., with contact dynamics, especially in discrete time.

A third challenge is that CBF-based methods require a CBF to be known. This can be challenging in the case of input constraints since not every function satisfies the CBF conditions (Chen et al., 2021). Constructing a CBF is even more challenging in the case of MAS with changing neighborhoods and limited sensing (Zhang et al., 2025). Zhang et al. (2025) proposed a learning framework for

[1]DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.
[2]Project website: https://mit-realm.github.io/dgppo/

constructing a graph CBF (GCBF) for MAS that guarantees safety while satisfying input constraints. However, they assume known continuous-time dynamics and require an existing nominal policy.

In this work, we address these challenges by proposing a novel framework that simultaneously learns a discrete graph CBF and a high-performance safe policy for a MAS under unknown discrete-time dynamics, changing neighborhoods, and input constraints. We summarize our contributions below.

- We propose a method of learning discrete CBFs (DCBF) for unknown discrete-time dynamics and with input constraints.
- We propose the discrete GCBF (DGCBF), a discrete-time extension of the GCBF, for ensuring safety under varying neighborhoods in the limited sensing setting and extend the DCBF learning above to the case of DGCBF.
- We propose **D**iscrete **G**raph CBF **P**roximal **P**olicy **O**ptimization (**DGPPO**), a framework combining RL and DGCBF for solving discrete-time multi-agent safe optimal control problems for MAS with unknown dynamics and limiting sensing without a known performant nominal policy.
- Through extensive simulations, we demonstrate that DGPPO outperforms existing methods and is not sensitive to hyperparameters. Specifically, compared to existing methods that require different choices of hyperparameters per environment, DGPPO achieves the lowest cost compared to baselines with near 100% safety rate using a single set of hyperparameters.

2 Related work

Constructing Decentralized CBFs. The challenge of applying CBFs for MAS has been explored via the construction of distributed CBFs (Borrmann et al., 2015; Glotfelter et al., 2017; Wang et al., 2017; Lindemann & Dimarogonas, 2019; Black & Panagou, 2023), which only take local observations as input. This simplifies the big centralized QP problem into small QP problems to be solved per agent. However, the construction is either limited to the case of unbounded control (Lindemann & Dimarogonas, 2019) or only for a specific dynamics model (e.g., double integrator) (Borrmann et al., 2015; Glotfelter et al., 2017; Wang et al., 2017). Recent advances in *learning* CBFs using neural networks (Saveriano & Lee, 2019; Srinivasan et al., 2020; Lindemann et al., 2021; Peruffo et al., 2021; Dawson et al., 2022; So et al., 2024; Knoedler et al., 2024) has resulted in works that investigate learning distributed CBFs (Qin et al., 2021; Zhang et al., 2023; 2025; Zinage et al., 2024). Nevertheless, these approaches assume *known* dynamics and are only applicable to *continuous-time* dynamics and hence cannot be applied to our problem setting. Moreover, it is assumed that a performant nominal policy is available, which we do not consider in this work.

CBF in RL. Originally inspired by the prospect of safety during training, recent works have integrated CBFs into the RL training process via the safety filter (Tearle et al., 2021; Hsu et al., 2023; Garg et al., 2024) for both single-agent (Cheng et al., 2019; Emam et al., 2022; Hailemichael et al., 2023) and multi-agent (Pereira et al., 2021; 2022) cases. Although both continuous-time (Emam et al., 2022; Hailemichael et al., 2023) and discrete-time (Cheng et al., 2019) dynamics have been considered, a major limitation is the requirement of affine (D)CBFs and control-affine dynamics up to a constant disturbance term to be learned. In contrast, the problem we tackle in this work does not make any such assumptions about the safety specifications or the structure of the dynamics.

Safe Multi-agent RL. The problem of constructing safe policies for MAS has also been studied in the RL community (Garg et al., 2024). Early works achieved safety via reward function design (Chen et al., 2017b;a; Long et al., 2018; Everett et al., 2018; Semnani et al., 2020). However, these approaches do not guarantee the satisfaction of the safety constraints even for the optimal policy (Massiani et al., 2023; Everett et al., 2018; Long et al., 2018). More recently, in the single-agent case, methods work with constraints in the form of the constrained Markov decision process (CMDP) problem and apply techniques from constrained optimization, including primal methods (Xu et al., 2021), primal-dual methods using Lagrange multipliers (Borkar, 2005; Tessler et al., 2019; He et al., 2023; Huang et al., 2024), and via trust-region-based approaches (Achiam et al., 2017; He et al., 2023). Of these, Lagrange-multiplier-based approaches are the most popular due to their simplicity, leading to multi-agent extensions (Gu et al., 2023; Liu et al., 2021b; Ding et al., 2023; Lu et al., 2021; Geng et al., 2023; Zhao et al., 2024). However, Lagrangian methods for CMDPs have been observed to have unstable training and convergence to poor policies when the constraint threshold is *zero* (Zanon & Gros, 2020; He et al., 2023; So & Fan, 2023; Ganai et al., 2024), which is the setting we target in this work.

3 PROBLEM SETTING AND PRELIMINARIES

3.1 MULTI-AGENT CONSTRAINED OPTIMAL CONTROL PROBLEM

Consider an N agent MAS. We aim to solve distributed control policies that minimize a joint cost describing a desired task while staying safe. Let the state and control of agent i at timestep k be $x_i^k \in \mathcal{X}$ and $u_i^k \in \mathcal{U}$, where x_i^k contains agent i's position $p_i^k \in \mathcal{P}$. The joint state is defined as $\mathbf{x}^k := [x_1^k; \dots; x_N^k; y^k] \in \boldsymbol{\mathcal{X}}$, where $y^k \in \mathcal{Y}$ is non-agent states (e.g., obstacles, goals). The joint action is defined with $\mathbf{u}^k := [u_1^k; \dots, u_N^k] \in \boldsymbol{\mathcal{U}}$. We assume $\mathbf{x}$ follows the general discrete-time dynamics

$$\mathbf{x}^{k+1} = f(\mathbf{x}^k, \mathbf{u}^k), \tag{1}$$

where $f : \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{U}} \to \boldsymbol{\mathcal{X}}$ describes the joint dynamics and is *unknown*. We consider the setting where agents only observe objects within their sensing radius $R > 0$. Let $\mathcal{N}_i(\mathbf{x}) = \{j \mid \|p_j - p_i\| \le R\}$ denote the *neighborhood* of agent i. At timestep k, agent i only has access to local observation $o_i^k := O_i(\mathbf{x}^k) \in \mathcal{O}$ for observation function O_i:

$$O_i(\mathbf{x}^k) = \left(\{o_{ij}\}_{j \in \mathcal{N}_i},\ o_i^y\right), \qquad o_{ij} := O^a(x_i, x_j), \quad o_i^y := O^y(x_i, y) \tag{2}$$

for inter-agent $O^a : \mathcal{X} \times \mathcal{X} \to \mathcal{O}_a$ and non-agent $O^y : \mathcal{X} \times \mathcal{Y} \to \mathcal{O}_y$ observation functions. Let the joint cost function describing the desired task be denoted as $l : \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{U}} \to \mathbb{R}$ [3]. Each agent has an avoid set defined as $\mathcal{A}_i := \{o_i \in \mathcal{O} \mid h_i^{(m)}(o_i) > 0, \forall m\}$ for the avoid functions $h_i^{(m)} : \mathcal{O} \to \mathbb{R}$, $m \in \{1, \dots, M\}$, such that the agent is *unsafe* it it enters $\mathcal{A}_i$ any time in the trajectory. We want to learn distributed policies $\mu_i : \mathcal{O} \to \mathcal{U}$ that minimize the joint cost l while avoiding the avoid set $\mathcal{A}_i$ at all times. Formally, denoting the joint policy by $\boldsymbol{\mu}(\mathbf{x}) = [\mu_1(o_1); \dots; \mu_N(o_N)]$, we want to solve the following discrete-time distributed multi-agent safe optimal control problem (MASOCP):

$$\min_{\mu_1, \dots, \mu_N} \quad \sum_{k=0}^{\infty} l(\mathbf{x}^k, \boldsymbol{\mu}(\mathbf{x}^k)) \tag{3a}$$

$$\text{s.t.} \quad \mathbf{x}^{k+1} = f(\mathbf{x}^k, \boldsymbol{\mu}(\mathbf{x}^k)), \qquad \forall i \in \{1, \dots, N\},\ k \ge 0, \tag{3b}$$

$$h_i^{(m)}(o_i^k) \le 0, \quad o_i^k = O_i(\mathbf{x}^k), \quad \forall i \in \{1, \dots, N\},\ \forall m \in \{1, \dots, M\},\ k \ge 0. \tag{3c}$$

The main challenge in solving (3) is satisfying the safety constraints (3c), especially in the multi-agent case with changing neighborhood, under unknown discrete-time dynamics with input constraints. We propose to tackle this challenge using the framework of DCBFs, which we review next.

3.2 DISCRETE CBF

To tackle the safety constraint of Problem (3), we review the notion of DCBF. Considering the discrete-time dynamics, we take the following definition of a DCBF from Ahmadi et al. (2019):

Definition 1. *A function $B : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ is a discrete CBF (DCBF) for (1) if there exists an extended class-κ function α satisfying $\alpha(-r) > -r$ for all $r > 0$ such that B satisfies the following property:*

$$B(\mathbf{x}) \le 0 \implies \inf_{\mathbf{u} \in \boldsymbol{\mathcal{U}}} B(f(\mathbf{x}, \mathbf{u})) - B(\mathbf{x}) + \alpha(B(\mathbf{x})) \le 0. \tag{4}$$

As shown in Ahmadi et al. (2019), the following theorem holds.

Theorem 1. *The set $\mathcal{C} := \{\mathbf{x} \mid B(\mathbf{x}) \le 0\}$ is control invariant under any policy $\boldsymbol{\mu}$ that satisfies*

$$B(f(\mathbf{x}, \boldsymbol{\mu}(\mathbf{x}))) - B(\mathbf{x}) + \alpha(B(\mathbf{x})) \le 0, \quad \forall \mathbf{x} \in \mathcal{C} \tag{5}$$

Thus, if $\mathcal{C} \cap \mathcal{A} = \emptyset$ for the avoid set $\mathcal{A}$, then the $\boldsymbol{\mu}$ from Theorem 1 renders the system safe.

Safe and Performant Policies via Safety Filtering. Given DCBFs $B^{(m)}$ for $m = 1, \dots, M$, we can construct a safe and performant policy using the safety filter framework. Assuming a nominal policy $\boldsymbol{\mu}_{\text{nom}}$ that is performant, e.g., minimizes the cost l, but not necessarily safe, we can obtain a safe *and* performant policy by solving the following nonlinear optimization problem:

$$\min_{\mathbf{u} \in \boldsymbol{\mathcal{U}}} \quad \|\mathbf{u} - \boldsymbol{\mu}_{\text{nom}}(\mathbf{x})\|^2, \tag{6a}$$

$$\text{s.t.} \quad C^{(m)}(\mathbf{x}, \mathbf{u}) \le 0, \quad C^{(m)}(\mathbf{x}, \mathbf{u}) := B^{(m)}(f(\mathbf{x}, \mathbf{u})) - B^{(m)}(\mathbf{x}) + \alpha(B^{(m)}(\mathbf{x})), \quad \forall m. \tag{6b}$$

[3] The cost function l here is **not** the *cost* in CMDP. Rather, it corresponds to the *negative reward* in CMDP.

In the general case, even if the dynamics f are known, (6) is potentially a nonlinear program that could be difficult to solve. In our problem setting, we assume f to be *unknown*, which renders this approach infeasible. Moreover, applying the safety filter framework assumes access to a performant, *distributed* nominal policy $\boldsymbol{\mu}_{\text{nom}}$, which we do not assume is available. Even if it were available, the safety filtering only minimizes the instantaneous deviation in control (6a) and can be *myopic*, potentially leading to liveness problems (Reis et al., 2020) and deadlocks (Jankovic et al., 2023).

Note that not every function satisfies (5) and is a DCBF. In continuous time with control-affine dynamics f, unbounded inputs are sufficient for any continuously differentiable function to be a CBF (Xiao & Belta, 2019) since the CBF condition will be *linear* in the controls. However, in discrete time, the DCBF condition (5) is potentially *nonlinear* in $\mathbf{u}$, making it difficult to construct a DCBF even with unbounded controls. While solutions have been proposed for the continuous-time case to construct valid CBFs under bounded inputs (Chen et al., 2021; So et al., 2024), the same is not true for DCBFs, potentially due to the requirement of solving a nonlinear program (6) even after such a DCBF has been found.

4 Tackling Challenges of DCBFs for MASOCP with DGPPO

We now address four challenges of extending a DCBF-based approach to our problem setting and propose DGPPO, our framework for solving (3). **All proofs can be found in Appendix A.**

4.1 Constraint-value function is DCBF

To construct a DCBF, we show that the constraint-value function of a policy *is* a DCBF, extending the insights of So et al. (2024) to discrete-time. This allows learning a DCBF with policy evaluation.

For an arbitrary function $\zeta : \boldsymbol{\mathcal{X}} \to \mathbb{R}$, let the avoid set be $\mathcal{A} \coloneqq \{\mathbf{x} \in \boldsymbol{\mathcal{X}} \mid \zeta(\mathbf{x}) > 0\}$. For a fixed deterministic policy $\boldsymbol{\mu}$, consider the constraint-value functions $V^{\zeta,\boldsymbol{\mu}}$, defined as

$$V^{\zeta,\boldsymbol{\mu}}(\mathbf{x}) = \max_{k \geq 0} \zeta(\mathbf{x}^k), \qquad \text{s.t.} \quad \mathbf{x}^0 = \mathbf{x}, \quad \mathbf{x}^{k+1} = f(\mathbf{x}^k, \boldsymbol{\mu}(\mathbf{x}^k)). \tag{7}$$

Then, $V^{\zeta,\boldsymbol{\mu}}$ is a DCBF, which we show in the following theorem.

Theorem 2 (Discrete Policy CBF)**.** *For a given $\boldsymbol{\mu}$, the constraint-value function $V^{h,\boldsymbol{\mu}}$ is a DCBF for any extended class-κ function α satisfying $\alpha(-r) > -r$ for all $r > 0$. Moreover, given $h^{(m)}$, $\boldsymbol{\mu}$ satisfies $V^{h^{(m)},\boldsymbol{\mu}}(f(\mathbf{x}, \mathbf{u})) - V^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x}) + \alpha(V^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x})) \leq 0$ for all $m \in \{1, \dots, M\}$.*

Theorem 2 enables the construction of a DCBF by choosing any policy $\boldsymbol{\mu}$ and evaluating its constraint-value function. Consequently, we can construct a DCBF by learning the value function.

4.2 Removing the Nominal Policy with Explicit Cost Optimization

We next address the challenge of requiring a performant nominal policy in the safety filter (6) by instead directly learning a policy using RL that minimizes the joint cost function (3a). This has been done previously in the single-agent setting via the framework of shielding for RL, where an unconstrained policy $\boldsymbol{\mu}_\theta$ with parameters θ is learned using existing unconstrained RL techniques, and the (D)CBF safety filter is incorporated into the environment dynamics (Cheng et al., 2019; Emam et al., 2022; Hailemichael et al., 2023). Formally, the following problem is considered:

$$\min_{\boldsymbol{\mu}_\theta} \quad \sum_{k=0}^{\infty} l(\mathbf{x}^k, \mathbf{u}^k), \qquad \text{s.t.} \quad \mathbf{u}^k = \text{SafetyFilter}(\boldsymbol{\mu}_\theta(\mathbf{x}^k)), \tag{8}$$

where SafetyFilter computes the minimizer of (6). However, solving (6) in the discrete case is a nonlinear program that is difficult unless $B^{(m)}$ is linear and the dynamics are control-affine, an assumption that we, unlike previous works (Cheng et al., 2019; Emam et al., 2022; Hailemichael et al., 2023), do not impose. To work around this, we constrain the learned policy to satisfy the DCBF conditions instead of using the safety filter framework.

$$\min_{\theta} \sum_{k=0}^{\infty} l(\mathbf{x}^k, \boldsymbol{\mu}_\theta(\mathbf{x}^k)), \quad \text{(9a)} \quad \text{s.t. } C^{(m)}(\mathbf{x}^k, \boldsymbol{\mu}_\theta(\mathbf{x}^k)) \leq 0, \ \ \forall m = \{1, \dots, M\}, \ \ k \geq 0, \quad \text{(9b)}$$

where $C^{(m)}$ is defined in (6b). This removes the need for a nominal policy and for solving the nonlinear safety filter program (6).

4.3 CONSTRAINED POLICY OPTIMIZATION USING DCBF UNDER UNKNOWN DYNAMICS

We now tackle the challenge of performing constrained policy optimization using DCBFs in (9). We choose to use a purely primal method inspired by constraint-rectified policy optimization (CRPO) (Xu et al., 2021) due to its simplicity (it does not have parameters related to the dual problem but simply chooses to take different gradient steps based on the current constraint satisfaction). Specifically, if the constraint (9b) is satisfied, our algorithm takes one gradient step to minimize the objective (9a). Otherwise, if the constraint is violated, our algorithm takes one gradient step to minimize the DCBF constraint violation C.

This procedure still cannot be implemented as-is since the gradient of C cannot be computed directly without knowledge of the dynamics f. To this end, we propose to use score function gradients (Williams, 1992) to compute gradients of (9b) without knowing the dynamics f. Since this requires a *stochastic* policy, we modify (9) accordingly. For clarity, let $\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})$ denote the probability density function of the stochastic policy with parameters θ conditioned on state $\mathbf{x}$. It can be tempting to consider the following stochastic version of the problem, which resembles the CMDP setting.

$$\min_\theta \quad \mathbb{E}_{\mathbf{x}\sim\rho_0,\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], \qquad \text{s.t.} \quad \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta},\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[C^{(m)}(\mathbf{x},\mathbf{u})\right] \leq 0, \tag{10}$$

where $Q^{\boldsymbol{\pi}_\theta}$ is the Q-function and ρ_0, $\rho^{\boldsymbol{\pi}_\theta}$ are the initial and stationary state distributions. Here, we constrain the *expectation* of the DCBF constraint. However, this formulation is *not* sufficient for safety, as the expectation does not guarantee satisfaction for all states as in (6), leading to an unsafe policy. To tackle this, we modify the constraint, leading to the following problem.

$$\min_\theta \quad \mathbb{E}_{\mathbf{x}\sim\rho_0,\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], \tag{11a}$$

$$\text{s.t.} \quad \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}} \underbrace{\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\max\left\{0, C^{(m)}(\mathbf{x},\mathbf{u})\right\}\right]}_{:=\tilde{C}_\theta^{(m)}(\mathbf{x})} \leq 0, \quad \forall m. \tag{11b}$$

Here, the satisfaction of (11b) guarantees that the DCBF constraint is satisfied almost surely (Appendix A.2). Applying gradient-manipulation style primal optimization as in CRPO (Xu et al., 2021) gives us the following expression for the gradient $\nabla_\theta L$ of the policy loss L.

$$\nabla_\theta L(\theta) = \begin{cases} \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho_0,\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], & \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\left[\tilde{C}_\theta^{(m)}(\mathbf{x})\right] \leq 0,\ \forall m, \\ \nu\, \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\left[\nabla_\theta \tilde{C}_\theta^{(m)}(\mathbf{x})\right] \text{ for any } m \text{ that violates}, & \text{otherwise}, \end{cases} \tag{12}$$

where $\nu > 0$ is a hyperparameter scaling the size of constraint minimization steps. The gradient of $\tilde{C}_\theta^{(m)}$ can be computed using score function gradients (see Appendix A.4, discussion on $\rho^{\boldsymbol{\pi}_\theta}$ in Appendix B.2).

Improving sample efficiency with Gradient Projection. One drawback of (12) is that this scheme is sample-inefficient in the sense that if only a single state $\bar{\mathbf{x}}$ violates the DCBF constraint, then the gradient information of the total cost from all other safe states are thrown away. We propose to use this thrown-away gradient information by *projecting* the cost gradient to be *orthogonal* to the gradient direction of the violating constraints, similar to techniques from multi-objective optimization (Yu et al., 2020; Liu et al., 2021a). However, this requires computing the gradient $M+1$ times, which is expensive when M is large. Instead, we propose to use the following informal theorem:

Informal Theorem 3 (Approximate Gradient Projection for Decoupled Policy Parameters)**.** *Let $\sigma^{(m)} := \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho}[\tilde{C}_\theta^{(m)}(\mathbf{x})]$ denote the gradient of the m-th DCBF constraint violation for any state distribution ρ. Under suitable assumptions on the policy parametrization $\boldsymbol{\pi}_\theta$, modifying the gradient of the objective (11a) from $g_{orig} := \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}(\mathbf{x},\mathbf{u})\, Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right]$ to*

$$g := \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}(\mathbf{x},\mathbf{u})\, \mathbb{1}_{\{\max_m \tilde{C}_\theta^{(m)}(\mathbf{x})\leq 0\}}\, Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], \tag{13}$$

by multiplying the $Q^{\boldsymbol{\pi}_\theta}$ with an indicator function gives an approximate projection g of g_{orig} such that $g \cdot \sigma^{(m)} = 0\ \forall m$, i.e., it lies in the orthogonal complement of the constraint gradients $\sigma^{(m)}$.

We state this more formally in Theorem A2. By Informal Theorem 3, we can treat g as an approximate projection of the gradient of the objective (11a) so that it does not interfere with

the gradients from the safety constraints. Combining $\sigma^{(m)}$ (with $\rho = \rho^{\pi_\theta}$) and g from Informal Theorem 3 gives the following gradient $\nabla_\theta L$ of the policy loss L, where ψ denotes the *stop gradient* operation (see Appendix B for a detailed derivation and discussion about important details).

$$L(\theta) = \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})}\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))}\big[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\psi\big(\tilde{Q}(\mathbf{x},\mathbf{u},\theta)\big)\big], \tag{14}$$

$$\tilde{Q}(\mathbf{x},\mathbf{u},\theta) \coloneqq \mathbb{1}_{\{\max_m \tilde{C}_\theta^{(m)}(\mathbf{x})\le 0\}}\psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})) + \nu \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u}). \tag{15}$$

Although we cannot guarantee that the conditions of Informal Theorem 3 hold, we find that adding this gradient projection improves performance. We provide empirical comparisons between this gradient projection method (14) and methods without projection (10), (11) in Appendix C.6.1.

Scheduling the weight ν. Empirically, different values of ν result in a tradeoff between safety and cost minimization (potentially due to not satisfying Informal Theorem 3). To remedy this, we schedule ν by taking $\nu = 1$ initially to encourage exploration at the beginning, then doubling it after 50% and 75% of the total update steps to emphasize safety. We investigate this further in Section 5.3.

4.4 Extending to the Multi-Agent Case with DGCBF

Having proposed three solutions to tackle the difficulty of using CBFs in RL for unknown discrete-time dynamics, we now tackle the final challenge of changing neighborhoods due to the limited sensing radius of each agent. For this, we draw inspiration from GCBF (Zhang et al., 2025), which provides a theoretical framework for constructing distributed CBFs that can handle varying neighborhood sizes, albeit for the case of continuous-time dynamics.

To construct a distributed DCBF B, we need B to be a function of each agent's local observation o_i as opposed to the joint state $\mathbf{x}$. We make this concrete in the following definition.

Definition 2 (Discrete GCBF)**.** *A function $\tilde{B} : \mathcal{O} \to \mathbb{R}$ is a Discrete Graph CBF (DGCBF) if there exists a class-κ function α with $\alpha(-r) > -r$ for all $r > 0$ and a control policy $\mu : \mathcal{O} \to \mathcal{U}$ satisfying*

$$\tilde{B}(o_j(\mathbf{x})) \le 0,\ \forall j \implies \tilde{B}(o_i^+(\mathbf{x})) - \tilde{B}(o_i(\mathbf{x})) + \alpha(\tilde{B}(o_i(\mathbf{x}))) \le 0, \quad \forall \mathbf{x} \in \boldsymbol{\mathcal{X}},\ \forall i, \tag{16}$$

where $o_i^+(\mathbf{x}) = O_i(f(\mathbf{x}, \boldsymbol{\mu}(\mathbf{x})))$ and $\boldsymbol{\mu}$ denotes the resulting joint policy from each agent using μ.

Since a DCBF B is defined for a MAS with a *fixed* size N, B can not be used when N changes. Given a DGCBF $\tilde{B}$, we can construct a DCBF B as $B(\mathbf{x}) \coloneqq \max_i \tilde{B}(o_i(\mathbf{x}))$ (Appendix A.6), thus $\tilde{B}$ guarantees safety. However, the *same* DGCBF $\tilde{B}$ can *also* be used to guarantee safety for *any* N, which we show in Appendix A.7.

Remark 1 (Discontinuity due to neighborhood changes)**.** *Unlike the continuous-time case (Zhang et al., 2025), which assumes that the GCBF is unaffected by agents at the sensing-radius boundary, DGCBF does not have this requirement. This is because the proof of safety in GCBF relies on continuity (w.r.t. time) during neighborhood changes. However, in the discrete-time case, the proof of safety only looks at finite differences and hence does not require the continuity of the DGCBF.*

Finding a function $\tilde{B}$ that satisfies (16) is nontrivial, especially in the case of neighborhood changes due to the limited sensing radius. Though we can learn a DGCBF using Section 4.1, it is unclear how the learned function satisfies (16) when the neighborhood of agents changes. One sufficient way for this to hold is to take advantage of the attention mechanism to place zero weights on the features corresponding to agents far enough away such that the value of $\tilde{B}$ is not affected too much by such agents.

We state this informally in the following theorem (see Appendix A.5 for the formal version).

Informal Theorem 4 (Satisfying (16) during neighborhood changes)**.** *Let ξ_1, ξ_2, and ξ_3 be functions that encode the input observations into some feature space. Suppose $\tilde{B}$ is of the form*

$$\tilde{B}(O_i(\mathbf{x})) = \xi_1\left(\sum_{j\in\mathcal{N}_i} w(o_{ij})\xi_2(o_{ij}),\ \xi_3(o_i^y)\right), \tag{17}$$

where $w : \mathcal{O}_a \to \mathbb{R}$ is a weighting function that goes to 0 for observations o_{ij} of agents that are far enough away. Then, under technical conditions on the dynamics, $\tilde{B}$ satisfies (16) and is a DGCBF.

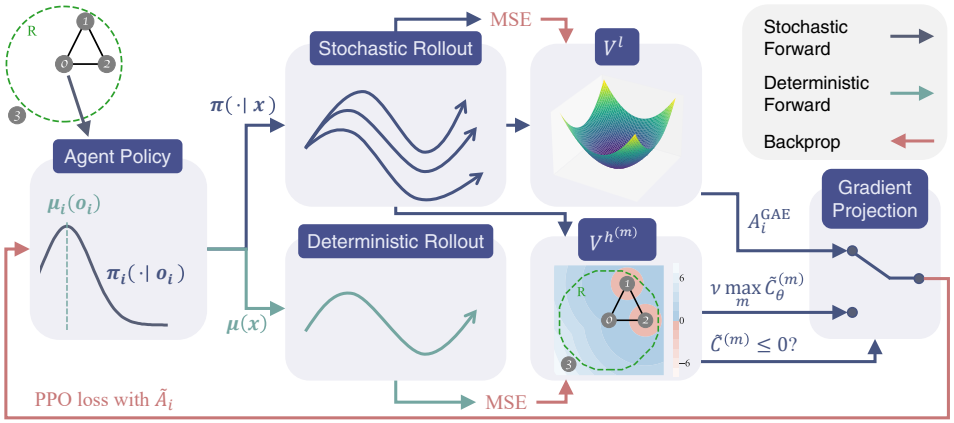


Figure 1: **DGPPO algorithm.** In addition to the normal MAPPO path (top) using **stochastic rollouts**, we introduce a second path (bottom) that uses **deterministic rollouts** to learn a DGCBF.

We encourage $\tilde{B}$ to satisfy (16) during neighborhood changes by parameterizing the value functions using graph neural networks (GNN) with graph attention (Veličković et al., 2017), which takes the form of (17) and hence is amenable to Informal Theorem 4. Similar to Zhang et al. (2025), the attention mechanism naturally learns to place zero weights on the features corresponding to neighboring agents that are far enough away, enabling the learned $\tilde{B}$ to satisfy (16) during neighborhood changes.

Finally, Theorem 2 can be extended to DGCBF, as shown in the following Corollary.

Corollary 1 (Discrete Policy GCBF). *Suppose $V_i^{h^{(m)},\boldsymbol{\mu}} : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ for agent i can be expressed using only local observations o_i, i.e., there exists some $\tilde{V}^{h^{(m)},\boldsymbol{\mu}} : \mathcal{O} \to \mathbb{R}$ such that*

$$\tilde{V}^{h^{(m)},\boldsymbol{\mu}}(o_i^0) = V_i^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x}^0) := \max_{k \geq 0} h^{(m)}(o_i^k). \tag{18}$$

Then, $\tilde{V}^{h^{(m)}}$ is a DGCBF.

4.5 DGPPO: Putting Everything Together

Combining the proposed solutions from the previous subsections, we present **DGPPO**, a framework for solving the discrete-time multi-agent safe optimal control problem (3). DGPPO follows the basic structure of on-policy MARL algorithms such as MAPPO (Yu et al., 2022).

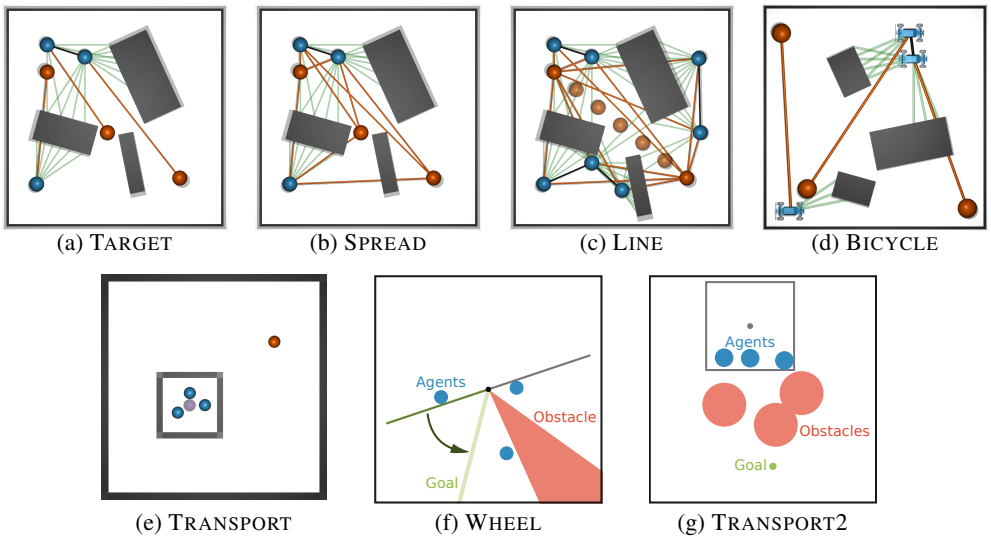
1. We perform a T-step stochastic rollout with the policy $\boldsymbol{\pi}_\theta$. However, unlike MAPPO, we additionally perform a T-step *deterministic* rollout using a deterministic version of $\boldsymbol{\pi}_\theta$ (by taking the mode), which we denote $\boldsymbol{\mu}$, to learn the DGCBF (per Theorem 2).
2. We update the value functions via regression on the corresponding targets computed using GAE (Schulman et al., 2015), where the targets for the cost-value function V^l uses the stochastic rollout, and the targets for the constraint-value functions $V^{h^{(m)},\boldsymbol{\mu}}$ use the deterministic rollout.
3. We update the policy $\boldsymbol{\pi}_\theta$ by replacing the Q-function with its GAE (Schulman et al., 2015), then combining the CRPO-style *decoupled* policy loss (14) with the PPO clipped loss (Schulman et al., 2017) using the learned constraint-value functions $V^{h^{(m)},\boldsymbol{\mu}}$ as the DGCBFs $\tilde{B}^{(m)}$. Specifically, we treat the expression within the expectation in (14) as a pseudo-advantage $\tilde{A}_i$ for agent i and use a single-sample estimator $\hat{C}_{\theta,i}^{(m)}$ of $\tilde{C}_{\theta,i}^{(m)}$ in (14), giving us

$$\hat{C}_{\theta,i}^{(m)} := \max\left\{0,\, V^{h^{(m)},\boldsymbol{\mu}}(o_i^+) - V^{h^{(m)},\boldsymbol{\mu}}(o_i) + \alpha(V^{h^{(m)},\boldsymbol{\mu}}(o_i))\right\}, \tag{19}$$

$$\tilde{A}_i := A^{\text{GAE}} \mathbb{1}_{\{\max_m \hat{C}_{\theta,i}^{(m)} \leq 0\}} + \nu \max_m \hat{C}_{\theta,i}^{(m)} \mathbb{1}_{\{\max_m \hat{C}_{\theta,i}^{(m)} > 0\}} \tag{20}$$

where A^{GAE} denotes the GAE (Schulman et al., 2015) for agent i. We then use $\tilde{A}_i$ in the PPO policy loss (Schulman et al., 2017) as done in MAPPO (Yu et al., 2022).

We summarize our DGPPO algorithm in Figure 1.



(a) TARGET (b) SPREAD (c) LINE (d) BICYCLE

(e) TRANSPORT (f) WHEEL (g) TRANSPORT2

Figure 2: **Environments.** We test on (top) LiDAR, (bottom) MuJoCo, and VMAS environments.

5 EXPERIMENTS

In this section, we design experiments to answer the following research questions: **(Q1)** Does DGPPO learn a safe policy that also achieves low costs without hyperparameter tuning in different environments? **(Q2)** How stable is the training of DGPPO? **(Q3)** Can DGPPO maintain its performance with an increasing number of agents? **(Q4)** Is DGPPO sensitive to the hyperparameters?

To compare the methods, we look at the cost and safety rate. The **cost** is the trajectory cumulative cost $\sum_{k=0}^{T} l(\mathbf{x}^k, \mathbf{u}^k)$. The **safety rate** is the ratio of agents that are safe over the *entire* trajectory. Details on implementation, tasks, hyperparameters, and additional experiments are in Appendix C.

5.1 SETUP

Environments. We evaluate DGPPO in a wide range of environments including four LiDAR environments (TARGET, SPREAD, LINE, BICYCLE) where the agents use LiDAR to detect obstacles (Keyumarsi et al., 2023), one MuJoCo environment TRANSPORT (Todorov et al., 2012), and two VMAS environments (TRANSPORT2, WHEEL) (Bettini et al., 2022; 2024).

Baselines. We compare DGPPO against baseline methods that can solve MASOCP under unknown discrete-time dynamics, including the state-of-the-art MARL algorithm InforMARL (Nayak et al., 2023) and the safe MARL algorithm MAPPO-Lagrangian (Gu et al., 2021; 2023). For InforMARL, we add the constraint violations $\max\{0, \max_m h^{(m)}\}$ weighted by β to the cost function for different βs (Penalty(β)). We also try a weight-scheduling scheme where β starts at 0.01 and increases at 50% and 75% of the total steps (Schedule). For MAPPO-Lagrangian, we use a GNN backbone for fair comparison. We notice the official implementation (Gu et al., 2023) uses a tiny learning rate on the Lagrange multipliers (10^{-7}), so we consider two different initialization λ_0 (Lagr(λ_0)). We also increase the learning rate of λ to 0.1[4] (Lagr(lr)). We run each method for the same number of update steps, chosen to be large enough such that all methods converge.[5]

5.2 MAIN RESULTS

For each environment, we run each algorithm with 3 different seeds and evaluate each run on 32 different initial conditions. We draw the following conclusions.

(Q1): DGPPO has the best performance and is hyperparameter insensitive. We first compare the converged policies of all algorithms (Figure 3). DGPPO is closest to the top left corner in all environments, indicating that it performs the best. For Penalty and Lagr, different choices of

[4]This is the smallest learning rate of λ that does not make the algorithm ignore the safety constraints.

[5]We also test baselines with double environment steps similar to DGPPO for fairness (Appendix C.6.2).

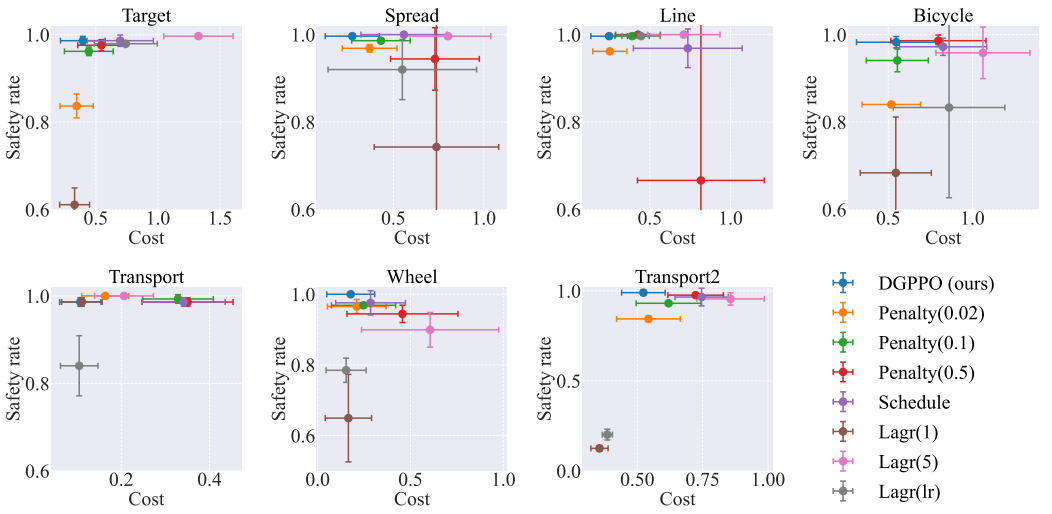


Figure 3: **Comparison on $N = 3$ agents.** ⊹ denotes the mean $\pm$ standard deviation. Methods closer to the top left yield lower costs and higher safety rates.

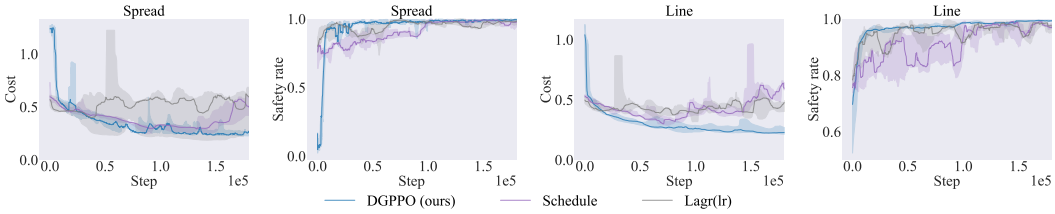


Figure 4: **Training stability.** DGPPO yields smoother training curves compared to the baselines.

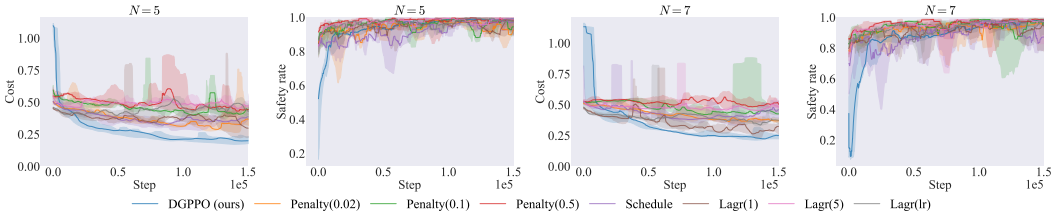
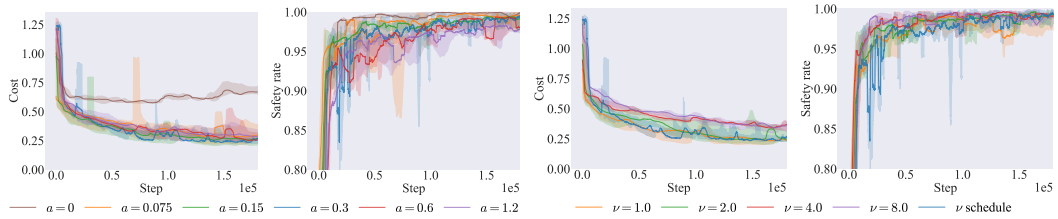


Figure 5: **Scaling to $N = 5, 7$.** Unlike other methods, DGPPO performs similarly with more agents.

hyperparameters result in either focusing only on safety or focusing only on performance. Even with a fixed hyperparameter, the performance of these two baselines also varies between environments. On the other hand, using the same set of hyperparameters in all environments, DGPPO consistently achieves the lowest cost among methods with a safety rate close to 100%.

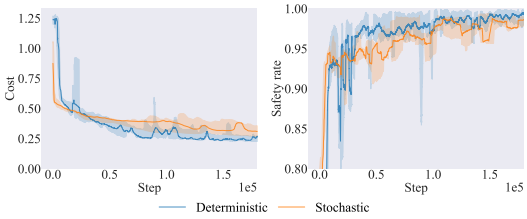
(Q2): Training of DGPPO is more stable. Next, we compare the training stability of DGPPO with Schedule, due to having a weight scheduled, and Lagr(lr), due to having a non-negligible learning rate (Figure 4). Schedule experiences an increase in cost as β increases throughout training. Lagr(lr) experiences high variance and many spikes in both cost and safety rate throughout training. This is similar to previous results obtained in the single-agent when the cost threshold is zero (So & Fan, 2023; He et al., 2023). DGPPO has a much smoother training curve than both. We provide training curves on other algorithms and environments in Appendix C.5.

(Q3): DGPPO scales well with more agents. Finally, we test the scalability of the methods on LINE by increasing the N from 3 to 5 and 7 (Figure 5). The same trends from before still hold, with DGPPO achieving the best performance and high safety rates. We also see that DGPPO performs well even with more agents, but the baseline methods are more inconsistent (e.g., Schedule is mostly safe with $N = 5$ but not so for $N = 7$), possibly due to their hyperparameter sensitivity.

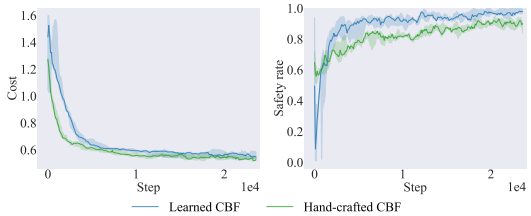


(a) Varying α in DGPPO.

(b) Varying ν in DGPPO.



(c) The necessity of deterministic rollouts to learn V^h.



(d) The necessity of learning a CBF.

Figure 6: **Ablations.** We vary hyperparameters (top) and verify our design decisions (bottom).

5.3 Ablation Studies

We now study hyperparameter sensitivity **(Q4)** by varying different hyperparameters in DGPPO.

Class-κ function α. For the class-κ function in (16), we use a linear $\alpha(r) = ar$ with $a = 0.3$. We test the sensitivity of DGPPO to this by varying a (Figure 6a) on Spread. We observe that $a = 0$ leads to conservative behavior with a high cost. $a = 1.2$ leads to an unsafe policy, which is to be expected since the $\alpha(-r) > -r$ condition is violated. For the other values that satisfy this condition, there is no significant difference in either cost or safety. We can thus choose any $a \in (0, 1)$.

Weight ν on the gradient of $\tilde{C}$. We introduced a schedule for ν, which weights the constraint minimization step (Section 4.3). We test the sensitivity to ν on Spread with different static schedules (Figure 6b). The safety rate is lower with $\nu = 1$, while the convergence in cost for $\nu = 4, 8$ is slower. The proposed schedule leads to faster cost convergence and a high safety rate.

Learning V^h with a stochastic policy. Theorem 2, used to learn the DGCBF $V^{h^{(m)},\boldsymbol{\mu}}$, requires *deterministic* rollouts. Consequently, DGPPO uses double the environment samples by performing both a stochastic and deterministic rollout. We verify whether this is necessary by seeing how the type of rollout (deterministic vs stochastic) used to learn the DGCBF affects performance (Figure 6c), which shows that using a stochastic rollout degrades both the cost and safety rate. Thus, the use of a deterministic rollout to learn $V^{h^{(m)},\boldsymbol{\mu}}$ is necessary despite the increased data use.

Using a hand-crafted DGCBF. One motivation for DGPPO is that it is difficult to construct a DGCBF with changing neighborhoods and input constraints. We test this by using $h^{(m)}$ *directly* as the DGCBF (instead of the learned $V^{h^{(m)},\boldsymbol{\mu}}$,), as is commonly done for CBFs, on Transport2 (Figure 6d). Using this hand-crafted "DGCBF" results in a decreased safety rate ($\sim 15\%$), validating the need to learn a DGCBF. If no DGCBF is used, it performs even worse (Appendix C.6.3).

6 Conclusion

We propose DGPPO to learn distributed safe policies for discrete-time MAS with unknown dynamics under a limited sensing range. We extend CBFs to this problem setting with DGCBFs, propose a construction using constraint-value functions, and apply CBFs to the case of unknown dynamics using score function gradients. Experimental results across three simulation engines suggest that DGPPO is robust to hyperparameters and performs well, achieving a safety rate matching conservative baselines while matching the performance of the performant but unsafe baselines.

Limitations. DGPPO uses both stochastic and deterministic rollouts, decreasing the sample efficiency. Moreover, safety under stochastic dynamics has not been considered. Finally, although safety is guaranteed when the DGCBF constraints are satisfied at *all* states, achieving this in practice using learning is hard. We leave these limitations to future work.

ACKNOWLEDGEMENT

This work was partly supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. In addition, Zhang, So, and Fan are supported by the MIT-DSTA program. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and don't necessarily reflect the views of the sponsors.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Mohamadreza Ahmadi, Andrew Singletary, Joel W Burdick, and Aaron D Ames. Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4797–4803. IEEE, 2019.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2017.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

Matteo Bettini, Amanda Prorok, and Vincent Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024.

David Biagioni, Xiangyu Zhang, Dylan Wald, Deepthi Vaidhynathan, Rohit Chintala, Jennifer King, and Ahmed S Zamzam. Powergridworld: A framework for multi-agent reinforcement learning in power systems. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pp. 565–570, 2022.

Mitchell Black and Dimitra Panagou. Adaptation for validation of consolidated control barrier functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 751–757, 2023. doi: 10.1109/CDC49753.2023.10383597.

Vivek S Borkar. An actor-critic algorithm for constrained markov decision processes. *Systems & Control Letters*, 54(3):207–213, 2005.

Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350. IEEE, 2017a.

Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285–292. IEEE, 2017b.

Yuxiao Chen, Mrdjan Jankovic, Mario Santillo, and Aaron D Ames. Backup control barrier functions: Formulation and comparative study. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6835–6841. IEEE, 2021.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3387–3395, 2019.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pp. 1724–1735. PMLR, 2022.

Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient generalized lagrangian policy optimization for safe multi-agent reinforcement learning. In *Learning for Dynamics and Control Conference*, pp. 315–332. PMLR, 2023.

Yousef Emam, Gennaro Notomista, Paul Glotfelter, Zsolt Kira, and Magnus Egerstedt. Safe reinforcement learning using robust control barrier functions. *IEEE Robotics and Automation Letters*, 2022.

Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059. IEEE, 2018.

Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Kunal Garg, Songyuan Zhang, Oswin So, Charles Dawson, and Chuchu Fan. Learning safe control for multi-robot systems: Methods, verification, and open challenges. *Annual Reviews in Control*, 57:100948, 2024.

Nan Geng, Qinbo Bai, Chenyi Liu, Tian Lan, Vaneet Aggarwal, Yuan Yang, and Mingwei Xu. A reinforcement learning framework for vehicular network routing under peak and average constraints. *IEEE Transactions on Vehicular Technology*, 2023.

Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters*, 1(2):310–315, 2017.

Jaskaran Singh Grover, Changliu Liu, and Katia Sycara. Deadlock analysis and resolution for multi-robot systems. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pp. 294–312. Springer, 2021.

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. *arXiv preprint arXiv:2110.02793*, 2021.

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

Habtamu Hailemichael, Beshah Ayalew, and Andrej Ivanco. Optimal control barrier functions for rl based safe powertrain control. *IFAC-PapersOnLine*, 56(3):385–390, 2023.

Tairan He, Weiye Zhao, and Changliu Liu. Autocost: Evolving intrinsic cost for zero-violation reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 14847–14855, 2023.

Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tsE5HLYtYg.

Mrdjan Jankovic, Mario Santillo, and Yan Wang. Multiagent systems with cbf-based controllers: Collision avoidance and liveness from instability. *IEEE Transactions on Control Systems Technology*, 2023.

Ajay Kattepur, Hemant Kumar Rath, Anantha Simha, and Arijit Mukherjee. Distributed optimization in multi-agent robotics for industry 4.0 warehouses. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pp. 808–815, 2018.

Shaghayegh Keyumarsi, Made Widhi Surya Atman, and Azwirman Gusrialdi. Lidar-based online control barrier function synthesis for safe navigation in unknown environments. *IEEE Robotics and Automation Letters*, 2023.

Luzia Knoedler, Oswin So, Ji Yin, Mitchell Black, Zachary Serlin, Panagiotis Tsiotras, Javier Alonso-Mora, and Chuchu Fan. Rpcbf: Constructing safety filters robust to model error and disturbances via policy control barrier functions. *arXiv preprint*, 2024.

Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks. *IEEE control systems letters*, 3(3):757–762, 2019.

Lars Lindemann, Haimin Hu, Alexander Robey, Hanwen Zhang, Dimos Dimarogonas, Stephen Tu, and Nikolai Matni. Learning hybrid control barrier functions from data. In *Conference on robot learning*, pp. 1351–1370. PMLR, 2021.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021a.

Chenyi Liu, Nan Geng, Vaneet Aggarwal, Tian Lan, Yuan Yang, and Mingwei Xu. Cmix: Deep multi-agent reinforcement learning with peak and average constraints. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pp. 157–173. Springer, 2021b.

Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6252–6259. IEEE, 2018.

Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Başar, and Lior Horesh. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8767–8775, 2021.

Pierre-François Massiani, Steve Heim, Friedrich Solowjow, and Sebastian Trimpe. Safe value functions. *IEEE Transactions on Automatic Control*, 68(5):2743–2757, 2023.

Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International Conference on Machine Learning*, pp. 25817–25833. PMLR, 2023.

Marcus Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos Theodorou. Safe optimal control using stochastic barrier functions and deep forward-backward sdes. In *Conference on Robot Learning*, pp. 1783–1801. PMLR, 2021.

Marcus A Pereira, Augustinos D Saravanos, Oswin So, and Evangelos A Theodorou. Decentralized safe multi-agent stochastic optimal control using deep fbsdes and admm. *arXiv preprint arXiv:2202.10658*, 2022.

Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*, pp. 370–388. Springer, 2021.

Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=P6_q1BRxY8Q.

Matheus F Reis, A Pedro Aguiar, and Paulo Tabuada. Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria. *IEEE Control Systems Letters*, 5(2):731–736, 2020.

Matteo Saveriano and Dongheui Lee. Learning barrier functions for constrained motion planning with dynamical systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 112–119. IEEE, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2023.

Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11532–11539. IEEE, 2024.

Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio A Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7139–7145. IEEE, 2020.

Ben Tearle, Kim P Wabersich, Andrea Carron, and Melanie N Zeilinger. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters*, 6(4):7635–7642, 2021.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(8):8243–8256, 2020.

Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *2019 IEEE 58th conference on decision and control (CDC)*, pp. 474–479. IEEE, 2019.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pp. 11480–11491. PMLR, 2021.

Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

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. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2020.

Songyuan Zhang, Kunal Garg, and Chuchu Fan. Neural graph control barrier functions guided distributed collision-avoidance multi-agent control. In *Conference on Robot Learning*, pp. 2373–2392. PMLR, 2023.

Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. Gcbf+: A neural graph control barrier function framework for distributed safe multiagent control. *IEEE Transactions on Robotics*, 41: 1533–1552, 2025.

Youpeng Zhao, Yaodong Yang, Zhenbo Lu, Wengang Zhou, and Houqiang Li. Multi-agent first order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 36, 2024.

Vrushabh Zinage, Abhishek Jha, Rohan Chandra, and Efstathios Bakolas. Decentralized safe and scalable multi-agent control under limited actuation. *arXiv preprint arXiv:2409.09573*, 2024.

A PROOFS

A.1 PROOF OF THEOREM 2

Proof. Dynamic programming on $V^{h,\mu}$ gives

$$V^{h,\mu}(x) = \max\Big\{h(x),\ V^{h,\mu}(f(x,\mu(x)))\Big\}. \tag{21}$$

Let $V^{h,\mu}(x) \leq 0$. This gives us two cases depending on which argument in the $\max$ is larger.

Case 1: $h(x) \leq V^{h,\mu}(f(x,\mu(x)))$**:** Here, we have that $V^{h,\mu}(x) = V^{h,\mu}(f(x,\mu(x)))$, which implies

$$V^{h,\mu}(f(x,\mu(x))) - V^{h,\mu}(x) = 0 \leq 0 \leq -\alpha(V^{h,\mu}(x)). \tag{22}$$

Case 2: $h(x) > V^{h,\mu}(f(x,\mu(x)))$**:** Here, we have that $V^{h,\mu}(x) = h(x) > V^{h,\mu}(f(x,\mu(x)))$, which implies

$$V^{h,\mu}(f(x,\mu(x))) - V^{h,\mu}(x) < 0 \leq 0 \leq -\alpha(V^{h,\mu}(x)). \tag{23}$$

Thus, $V^{h,\mu}(f(x,\mu(x))) - V^{h,\mu}(x) + \alpha(V^{h,\mu}(x)) \leq 0$ if $V^{h,\mu}(x) \leq 0$, and $V^{h,\mu}$ is a DCBF. □

A.2 PROOF THAT EQUATION (11b) IMPLIES $C^{(m)}(\mathbf{x},\mathbf{u}) \leq 0$ ALMOST SURELY

Theorem A1. *Suppose*

$$\mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}} \underbrace{\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\max\left\{0, C^{(m)}(\mathbf{x},\mathbf{u})\right\}\right]}_{\coloneqq \tilde{C}_\theta^{(m)}(\mathbf{x})} \leq 0. \tag{24}$$

Then, $C^{(m)}(\mathbf{x},\mathbf{u}) \leq 0$ almost surely.

Proof. Note that (24) can only be satisfied when the expectation equals 0 since $\max\{0,\cdot\}$ is non-negative.

Assume for contradiction that $P(C^{(m)}(\mathbf{x},\mathbf{u}) \leq 0) \leq 1-\epsilon$ for $\epsilon > 0$, i.e., $P(C^{(m)}(\mathbf{x},\mathbf{u}) > 0) \geq \epsilon$. Then,

$$\begin{aligned}
0 &= \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\max\left\{0, C^{(m)}(\mathbf{x},\mathbf{u})\right\}\right] && (25)\\
&\geq P\Big(C^{(m)}(\mathbf{x},\mathbf{u}) > 0\Big)\mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta},\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\max\left\{0, C^{(m)}(\mathbf{x},\mathbf{u})\right\} \mid C^{(m)}(\mathbf{x},\mathbf{u}) > 0\right] && (26)\\
&= \epsilon\,\mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta},\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\max\left\{0, C^{(m)}(\mathbf{x},\mathbf{u})\right\} \mid C^{(m)}(\mathbf{x},\mathbf{u}) > 0\right] && (27)\\
&> 0. && (28)
\end{aligned}$$

which is a contradiction. Thus, $P(C^{(m)}(\mathbf{x},\mathbf{u}) \leq 0) = 1$, and $C^{(m)}(\mathbf{x},\mathbf{u}) \leq 0$ almost surely. □

A.3 FORMAL STATEMENT AND PROOF OF INFORMAL THEOREM 3

We first formally state Informal Theorem 3 below.

Theorem A2 (Approximate Gradient Projection for Decoupled Policy Parameters)**.** *Suppose that for all $\mathbf{x}_1 \neq \mathbf{x}_2$, the parameters θ of the stochastic policy $\boldsymbol{\pi}_\theta$ are orthogonal, i.e., $(\nabla_\theta \boldsymbol{\pi}_\theta(\mathbf{u}_1 \mid \mathbf{x}_1)) \cdot (\nabla_\theta \boldsymbol{\pi}_\theta(\mathbf{u}_2 \mid \mathbf{x}_2)) = 0$ for all $\mathbf{u}_1, \mathbf{u}_2 \in \boldsymbol{\mathcal{U}}$ (e.g., a finite state-space $\boldsymbol{\mathcal{X}}$ with independent distribution at each state). Let $\sigma^{(m)} \coloneqq \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho}[\tilde{C}_\theta^{(m)}(\mathbf{x})]$ denote the gradient of the m-th DCBF constraint violation for any state distribution ρ. Then, the gradient of the objective (11a), modified with an extra indicator as follows:*

$$g \coloneqq E_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\left[\mathbb{1}_{\{\max_m \tilde{C}_\theta^{(m)}(\mathbf{x})\leq 0\}}\,\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}(\mathbf{x},\mathbf{u})\, Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right]\right], \tag{29}$$

satisfies $g \cdot \sigma^{(m)} = 0\ \forall m$, i.e., it lies in the orthogonal complement of the constraint gradients $\sigma^{(m)}$.

Proof. For $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, define the set $\Theta_{\mathbf{x}}$ as the column space of the gradient of the policy $\boldsymbol{\pi}$ at $\mathbf{x}$, i.e.,

$$\Theta_{\mathbf{x}} \coloneqq \operatorname{span}\{\nabla_\theta \boldsymbol{\pi}_\theta(\mathbf{u} \mid \mathbf{x}) : \mathbf{u} \in \boldsymbol{\mathcal{U}}\}. \tag{30}$$

By assumption, this implies that for $\mathbf{x}_1 \neq \mathbf{x}_2$,

$$\theta_1 \in \Theta_{\mathbf{x}_1},\, \theta_2 \in \Theta_{\mathbf{x}_2} \implies \theta_1 \cdot \theta_2 = 0. \tag{31}$$

Now, note that

$$\begin{aligned}
\sigma^{(m)} &\coloneqq \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho}[\tilde{C}_\theta^{(m)}(\mathbf{x})] && (32)\\
&= \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}[\max(0, C_\theta^{(m)}(\mathbf{x},\mathbf{u}))] && (33)\\
&= \mathbb{E}_{\mathbf{x}\sim\rho}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{u} \mid \mathbf{x}) \max(0, C_\theta^{(m)}(\mathbf{x},\mathbf{u}))\right] && (34)\\
&= \mathbb{E}_{\mathbf{x}\sim\rho}\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\mathbb{1}_{\{C_\theta^{(m)}(\mathbf{x},\mathbf{u})>0\}} \nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{u} \mid \mathbf{x}) C_\theta^{(m)}(\mathbf{x},\mathbf{u})\right] && (35)\\
&\subseteq \bigcup_{\mathbf{x}:\mathcal{E}^{(m)}} \Theta_{\mathbf{x}}, \qquad \mathcal{E}^{(m)} \coloneqq \left\{\mathbf{x} \in \boldsymbol{\mathcal{X}} : \operatorname*{ess\,sup}_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} C_\theta^{(m)}(\mathbf{x},\mathbf{u}) > 0\right\}, && (36)
\end{aligned}$$

where we have used the score function gradient estimator of $\tilde{C}_\theta$ (Appendix A.4). Similarly,

$$\begin{aligned}
g &\coloneqq \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta},\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{u} \mid \mathbf{x}) \mathbb{1}_{\{\max_m \tilde{C}_\theta^{(m)}(\mathbf{x})\leq 0\}} Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], && (37)\\
&= \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta},\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{u} \mid \mathbf{x}) \mathbb{1}_{\{\max_m \operatorname{ess\,sup}_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} C_\theta^{(m)}(\mathbf{x})\leq 0\}} Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], && (38)\\
&\subseteq \bigcup_{\mathbf{x}:\mathcal{F}} \Theta_{\mathbf{x}}, && (39)
\end{aligned}$$

where

$$\begin{aligned}
\mathcal{F} &\coloneqq \left\{\mathbf{x} \in \boldsymbol{\mathcal{X}} : \max_m \operatorname*{ess\,sup}_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} C_\theta^{(m)}(\mathbf{x},\mathbf{u}) \leq 0\right\}, && (40)\\
&= \left\{\mathbf{x} \in \boldsymbol{\mathcal{X}} : \operatorname*{ess\,sup}_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} C_\theta^{(m)}(\mathbf{x},\mathbf{u}) \leq 0,\ \forall m\right\}. && (41)
\end{aligned}$$

Since $\mathcal{E}^{(m)} \cap \mathcal{F} = \emptyset$ for all m, we have that $\sigma^{(m)} \cdot g = 0$ for all m. □

A.4 Score Function Gradient Estimator of $\tilde{C}_\theta$

Proof. Using the log trick,

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\mathbf{x})}\left[C(\mathbf{x}, f(\mathbf{x},\mathbf{u}))\right] &= \nabla_\theta \int C(\mathbf{x}, f(\mathbf{x},\mathbf{u}))\, \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\, d\mathbf{u},\\
&= \int C(\mathbf{x}, f(\mathbf{x},\mathbf{u}))\, (\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}))\, \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\, d\mathbf{u},\\
&= \mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\mathbf{x})}\left[\nabla \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\, C(\mathbf{x}, f(\mathbf{x},\mathbf{u}))\right].
\end{aligned} \tag{42}$$

□

A.5 Formal Statement and Proof of Informal Theorem 4

We first formally state Informal Theorem 4 below.

Theorem A3 (Satisfying (16) during neighborhood changes)**.** *Suppose that the maximum distance an agent can travel in a time step is $\bar{d}$. Let $\tilde{B}$ be of the following form.*

$$\tilde{B}(O_i(\mathbf{x})) = \xi_1\left(\sum_{j\in\mathcal{N}_i} w(o_{ij})\xi_2(o_{ij}),\ \xi_3(o_i^y)\right), \tag{43}$$

where $\xi_1 : \mathbb{R}^{\rho_1} \times \mathbb{R}^{\rho_2} \to \mathbb{R}$, $\xi_2 : \mathcal{O}_a \to \mathbb{R}^{\rho_1}$, and $\xi_3 : \mathcal{O}_y \to \mathbb{R}^{\rho_2}$ encode the observations into some feature space, and $w : \mathcal{O}_a \to \mathbb{R}$ is a weighting function such that

$$w(o_{ij}) = 0, \quad \textit{for all } x_i, x_j \textit{ such that } \|p_i - p_j\| \geq R - 2\bar{d}. \tag{44}$$

If 1) $\tilde{B}$ satisfies (16) for all transitions where the neighborhood does not change, and 2) for any transition $\mathbf{x} \to \mathbf{x}^+$ with a neighborhood change $\mathcal{N}_i(\mathbf{x}) \neq \mathcal{N}_i(\mathbf{x}^+)$, there exists a transition $\bar{\mathbf{x}} \to \bar{\mathbf{x}}^+$ where all agents that either enter or leave the neighborhood (i.e., the complement of $\mathcal{N}_i(\mathbf{x}) \cap \mathcal{N}_i(\mathbf{x}^+)$) are moved outside the sensing radius, and all the remaining agents move identically in $\mathbf{x}$ and $\bar{\mathbf{x}}$, then $\tilde{B}$ is a DGCBF.

Proof. Let $\mathbf{x}$ and $\mathbf{x}^+$ be consecutive states such that the neighborhood of agent i changes, i.e., $\mathcal{N}_i(\mathbf{x}) \neq \mathcal{N}_i(\mathbf{x}^+)$. Let $E \coloneqq \mathcal{N}_i(\mathbf{x}) \setminus \mathcal{N}_i(\mathbf{x}^+)$ and $F \coloneqq \mathcal{N}_i(\mathbf{x}^+) \setminus \mathcal{N}_i(\mathbf{x})$ denote the set of agents that leave and enter the neighborhood of agent i, respectively. Since the maximum distance agents can travel in one timestep is $\bar{d}$, the distance between agents can change by at most $2\bar{d}$ in one timestep. Hence, all agents exiting the neighborhood are at least $R - 2\bar{d}$ away agent i, i.e.,

$$j \in E \implies \|p_i - p_j\| \geq R - 2\bar{d}. \tag{45}$$

Similarly, all agents entering the neighborhood are at least $R - 2\bar{d}$ away from agent i at the $\mathbf{x}^+$, i.e.,

$$j \in F \implies \|p_i^+ - p_j^+\| \geq R - 2\bar{d}. \tag{46}$$

Hence, by (44), we have that $w(o_{ij}) = 0$ for all $j \in E$ and $w(o_{ij}^+) = 0$ for all $j \in F$.

Now, by asumption, there exists consecutive states $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}^+$ with the same neighborhood $\mathcal{N}_i(\bar{\mathbf{x}}) = \mathcal{N}_i(\bar{\mathbf{x}}^+) = \mathcal{N}_i(\mathbf{x}) \cap \mathcal{N}_i(\mathbf{x}^+)$, such that $\bar{x}_j = x_j$ and $\bar{x}_j^+ = x_j^+$ for $j \in \mathcal{N}_i(\bar{\mathbf{x}})$, and similarly for non-agent states $y = \bar{y}$, $y^+ = \bar{y}^+$. By definition of the observation function O_i, $\mathbf{x}$ and $\bar{\mathbf{x}}$ share the same observation except for the o_{ij} for $j \in E$, and similarly for $\mathbf{x}^+$ and $\bar{\mathbf{x}}^+$ for $j \in F$. Since $w_{ij} = 0$ for all $j \in E \cup F$, the form of $\tilde{B}$ (43) implies that

$$\sum_{j \in \mathcal{N}_i} w(o_{ij}) \xi_2(o_{ij}) = \sum_{j \in \mathcal{N}_i} w(\bar{o}_{ij}) \xi_2(\bar{o}_{ij}) \tag{47}$$

$$\sum_{j \in \mathcal{N}_i} w(o_{ij}^+) \xi_2(o_{ij}) = \sum_{j \in \mathcal{N}_i} w(\bar{o}_{ij}) \xi_2(\bar{o}_{ij}^+) \tag{48}$$

Hence, we must have that

$$\tilde{B}(O_i(\mathbf{x})) = \tilde{B}(O_i(\bar{\mathbf{x}})), \quad \tilde{B}(O_i(\mathbf{x}^+)) = \tilde{B}(O_i(\bar{\mathbf{x}}^+)). \tag{49}$$

By assumption, $\tilde{B}$ satisfies the DGCBF condition (16) for all transitions where the neighborhood does not change, which includes $\bar{\mathbf{x}} \to \bar{\mathbf{x}}^+$. Hence,

$$\tilde{B}(o_i^+) - \tilde{B}(o_i) + \alpha\left(\tilde{B}(o_i)\right) = \tilde{B}(\bar{o}_i^+) - \tilde{B}(\bar{o}_i) + \alpha\left(\tilde{B}(\bar{o}_i)\right) \tag{50}$$

$$\leq 0, \tag{51}$$

and $\tilde{B}$ also satisfies the DGCBF condition (16) for transitions where the neighborhood changes. Thus, $\tilde{B}$ is a DGCBF. □

A.6 Proof that a DGCBF can be used to construct a DCBF

Theorem A4. *For a N-agent MAS, define $B : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ as*

$$B(\mathbf{x}) \coloneqq \max_i \tilde{B}(O_i(\mathbf{x})). \tag{52}$$

Then, B is a DCBF.

Proof. Let $\mu : \mathcal{O} \to \mathbb{R}$ denote the per-agent control policy corresponding to the DGCBF $\tilde{B}$ in Definition 2, and let $\boldsymbol{\mu}$ denote the resulting joint control policy. Then, under μ, (16) implies that for any i,

$$\tilde{B}(o_i^+(\mathbf{x})) - \tilde{B}(o_i(\mathbf{x})) + \alpha(\tilde{B}(o_i(\mathbf{x}))) \leq 0, \quad \forall \mathbf{x} \in \boldsymbol{\mathcal{X}}. \tag{53}$$

Now, for a given $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, let $\mathbf{x}^+ = f(\mathbf{x}, \boldsymbol{\mu}(\mathbf{x}))$ denote the next state following $\boldsymbol{\mu}$. Let i_1 and i_2 denote the index that maximizes (52) at $\mathbf{x}$ and $\mathbf{x}^+$ respectively, i.e.,

$$i_1 \coloneqq \arg\min_i \tilde{B}(O_i(\mathbf{x})) \quad \implies B(\mathbf{x}) = \tilde{B}(O_{i_1}(\mathbf{x})), \tag{54}$$

$$i_2 \coloneqq \arg\min_i \tilde{B}(O_i(\mathbf{x}^+)) \quad \implies B(\mathbf{x}^+) = \tilde{B}(O_{i_2}(\mathbf{x}^+)). \tag{55}$$

$$\tag{56}$$

Then, using (53) and the fact that $(1-\alpha)$ is also an extended class-κ function and thus is a monotonic function (Ahmadi et al., 2019):

$$0 \geq \tilde{B}(O_{i_2}(\mathbf{x}^+)) - \tilde{B}(O_{i_2}(\mathbf{x})) + \alpha(\tilde{B}(O_{i_2}(\mathbf{x}))), \tag{57}$$
$$= \tilde{B}(O_{i_2}(\mathbf{x}^+)) - (1-\alpha) \circ \tilde{B}(O_{i_2}(\mathbf{x})), \tag{58}$$
$$\geq \tilde{B}(O_{i_2}(\mathbf{x}^+)) - (1-\alpha) \circ \tilde{B}(O_{i_1}(\mathbf{x})), \tag{59}$$
$$= B(\mathbf{x}^+) - (1-\alpha) \circ B(\mathbf{x}), \tag{60}$$
$$= B(\mathbf{x}^+) - B(\mathbf{x}) + \alpha(B(\mathbf{x})). \tag{61}$$

and (4) holds. Thus, B is a DCBF. □

Since $\tilde{B}$ enables the construction of a DCBF, this implies that $\mathcal{C}$, the zero sub-level set of B, i.e.,

$$\mathcal{C} \coloneqq \{\mathbf{x} \mid B(\mathbf{x}) \leq 0\} = \bigcap_i \{\mathbf{x} \mid \tilde{B}(o_i(\mathbf{x})) \leq 0\}, \tag{62}$$

is forward-invariant under $\boldsymbol{\mu}$ and hence control invariant.

A.7 DGCBF has Generalizable Safety Guarantees

In this subsection, we prove that the same DGCBF $\tilde{B}$ from Definition 2 can guarantee the safety of a MAS with any number of agents N. We will do this by showing that there exists an $\bar{N}$ such that if $\tilde{B}$ satisfies the DGCBF conditions (16) for $\bar{N}$ agents, then the **same** DGCBF $\tilde{B}$ will also satisfy the DGCBF conditions (16).

Let the maximum distance an agent can travel in a time step is $\bar{d}$. Given a sensing radius R, define $\bar{N}$ as the maximum number of agents that can be located within a ball of radius $2R + 2\bar{d}$. For a N-agent MAS, let $\boldsymbol{\mathcal{X}}_N$ and $\boldsymbol{\mathcal{U}}_N$ denote the joint state and control space respectively, and let f_N denote the corresponding dynamics function. For generalizability to an arbitrary number of agents, we make the additional assumption that the dynamics f_N are decoupled for each agent, i.e.,

$$x_i^{k+1} = f_0(x^k, u^k), \tag{63}$$

for per-agent dynamics function f_0.

For a state $\mathbf{x} \in \boldsymbol{\mathcal{X}}_N$, let $\mathbf{x}_{i,\bar{N}} \in \boldsymbol{\mathcal{X}}_{\bar{N}}$ denote the restriction of $\mathbf{x}$ to that of agent i and its $\bar{N}-1$ closest neighbors. We then have the following theorem.

Theorem A5. *Let $\tilde{B}$ satisfy the DGCBF conditions (16) for $\bar{N}$ agents, i.e., there exists a class-κ function α with $\alpha(-r) > -r$ for all $r > 0$ and a control policy $\mu : \mathcal{O} \to \mathcal{U}$ satisfying*

$$\tilde{B}(O_j(\tilde{\mathbf{x}})) \leq 0,\ \forall j \implies \tilde{B}(O_i^+(\tilde{\mathbf{x}})) - \tilde{B}(O_i(\tilde{\mathbf{x}})) + \alpha(\tilde{B}(O_i(\tilde{\mathbf{x}}))) \leq 0, \quad \forall \tilde{\mathbf{x}} \in \boldsymbol{\mathcal{X}}_{\bar{N}},\ \forall i, \tag{64}$$

Then, $\tilde{B}$ also satisfies the DGCBF conditions (16) for any $N > \bar{N}$ agents, i.e.,

$$\tilde{B}(O_j(\mathbf{x})) \leq 0,\ \forall j \implies \tilde{B}(O_i^+(\mathbf{x})) - \tilde{B}(O_i(\mathbf{x})) + \alpha(\tilde{B}(O_i(\mathbf{x}))) \leq 0, \quad \forall \mathbf{x} \in \boldsymbol{\mathcal{X}}_N,\ \forall i, \tag{65}$$

We will prove Theorem A5 by showing that (65) holds because it can be reduced to the case of (64). For convenience, define B as in Theorem A4 so that

$$\tilde{B}(o_j(\mathbf{x})) \leq 0,\ \forall j \quad \iff \quad B(\mathbf{x}) \leq 0. \tag{66}$$

Before we prove Theorem A5, we first prove a few helpful lemmas.

Lemma 1 (Restriction leaves observations of all agents within $R + 2\bar{d}$ invariant)**.** *For any $\mathbf{x} \in \boldsymbol{\mathcal{X}}_N$ such that $B(\mathbf{x}) \leq 0$, the restriction of $\mathbf{x}$ to that of agent i and its $\bar{N}-1$ closest neighbors leaves the observation of agent i and all agent within $R + 2\bar{d}$ of i invariant, i.e., for any i,*

$$\|p_j - p_i\| \leq R + 2\bar{d} \quad \implies \quad O_j(\mathbf{x}) = O_j(\mathbf{x}_{i,\bar{N}}) \tag{67}$$

Proof. Since $B(\mathbf{x}) \leq 0$, by definition of $\bar{N}$ there can be no more than $\bar{N}-1$ agents within radius of $2R + \bar{d}$ of agent i. Hence, $\mathbf{x}_{i,\bar{N}}$ will include all agents within a radius of $2R + \bar{d}$ of agent i.

By definition of the observation function O_i (2),

$$O_i(\mathbf{x}) = \left(\{o_{ij}\}_{j\in\mathcal{N}_i(\mathbf{x})},\ o_i^y\right). \tag{68}$$

$O_j(\mathbf{x})$ depends only on all agents l that are at most R away from j. Since $\mathbf{x}$ and $\mathbf{x}_{i,\bar{N}}$ agree on all agents up to $2R + 2\bar{d}$ away from i, this implies that for all j that are $R + 2\bar{d}$ away from i,

$$\mathcal{N}_j(\mathbf{x}) = \mathcal{N}_j(\mathbf{x}_{i,\bar{N}}), \tag{69}$$

thus the observation is unchanged as well. □

Next, we show that the observation of the *next* state for agent i is also left unchanged after restriction.

Lemma 2 (Restriction leaves the next observation for agent i unchanged)**.** *For any $\mathbf{x} \in \boldsymbol{\mathcal{X}}_N$ such that $B(\mathbf{x}) \leq 0$, let $\mathbf{x}^+ = f_N(\mathbf{x}, \boldsymbol{\mu}(\mathbf{x}))$ denote the next state under $\boldsymbol{\mu}$ for $\mathbf{x}$, and $\tilde{\mathbf{x}}^+ = f_{\bar{N}}(\mathbf{x}_{i,\bar{N}}, \boldsymbol{\mu}(\mathbf{x}_{i,\bar{N}}))$ the next state under $\boldsymbol{\mu}$ starting from $\mathbf{x}_{i,\bar{N}}$. Then, for any i,*

$$O_i(\mathbf{x}^+) = O_i(\tilde{\mathbf{x}}^+). \tag{70}$$

Proof. We will prove this by showing that the states of all neighbors $\mathcal{N}_i(\mathbf{x}^+)$ agree. For each new neighbor $j \in \mathcal{N}_i(\mathbf{x}^+)$, since each agent can travel at most $\bar{d}$ in one step, this implies that

$$\|p_j - p_i\| \leq R + 2\bar{d}. \tag{71}$$

Applying Lemma 1 gives us that agent j's observation is equal in both cases, i.e.,

$$O_j(\mathbf{x}) = O_j(\mathbf{x}_{i,\bar{N}}). \tag{72}$$

Hence, the controls $\mu(O_j(\mathbf{x}))$, and thus the new states match, i.e.,

$$x_j^+ = f_0(x_j, \mu(O_j(\mathbf{x}))) = f_0(x_j, \mu(O_j(\mathbf{x}_{i,\bar{N}}))) = \tilde{x}_j^+. \tag{73}$$

Since the new states for all agents in $\mathcal{N}_i(\mathbf{x}^+)$ agree, this implies that the new observation for agent i must also agree. □

We are now ready to prove Theorem A5.

Proof of Theorem A5. Let $\mathbf{x} \in \boldsymbol{\mathcal{X}}_N$ such that $B(\mathbf{x}) \leq 0$. Then, applying Lemma 1 and Lemma 2 implies that the current and next observations for agent i remain unchanged even when considering only the $\bar{N}$ closest agents, i.e.,

$$O_i(\mathbf{x}_{i,\bar{N}}) = O_i(\mathbf{x}), \tag{74}$$

$$O_i^+(\mathbf{x}_{i,\bar{N}}) = O_i^+(\mathbf{x}). \tag{75}$$

Hence, taking $\tilde{x} = \mathbf{x}_{i,\bar{N}} \in \boldsymbol{\mathcal{X}}_{\bar{N}}$ and using (64),

$$\tilde{B}(O_i^+(\mathbf{x})) - \tilde{B}(O_i(\mathbf{x})) + \alpha(\tilde{B}(O_i(\mathbf{x}))) = \tilde{B}(O_i^+(\mathbf{x}_{i,\bar{N}})) - \tilde{B}(O_i(\mathbf{x}_{i,\bar{N}})) + \alpha(\tilde{B}(O_i(\mathbf{x}_{i,\bar{N}}))) \tag{76}$$

$$= \tilde{B}(O_i^+(\tilde{\mathbf{x}})) - \tilde{B}(O_i(\tilde{\mathbf{x}})) + \alpha(\tilde{B}(O_i(\tilde{\mathbf{x}}))) \tag{77}$$

$$\leq 0. \tag{78}$$

□

Theorem A5 implies that finding a **single** DGCBF $\tilde{B}$ that satisfies the DGCBF condition for $\bar{N}$ agents enables the **same** DGCBF $\tilde{B}$ to *also* be applied to larger numbers of agents $N > \bar{N}$.

A.8 Proof of Corollary 1

Proof. Since $\tilde{V}^{h^{(m)},\boldsymbol{\mu}}(o_i^0) = V_i^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x}^0) \coloneqq \max_{k\geq 0} h^{(m)}(o_i^k)$, applying Theorem 2 gives us that

$$\tilde{V}^{h^{(m)},\boldsymbol{\mu}}(o_i^+) - \tilde{V}^{h^{(m)},\boldsymbol{\mu}}(o_i) + \alpha\left(\tilde{V}^{h^{(m)},\boldsymbol{\mu}}(o_i)\right) \tag{79}$$

$$= V_i^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x}^+) - V_i^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x}) + \alpha\left(V_i^{h^{(m)},\boldsymbol{\mu}}(\mathbf{x})\right) \tag{80}$$

$$\leq 0. \tag{81}$$

Thus, $\tilde{V}^{h^{(m)},\boldsymbol{\mu}}$ is a DGCBF. □

B Policy Loss Details

B.1 Derivation of (14)

We first start from (11), which we repeat below for convenience.

$$\min_{\theta} \quad \mathbb{E}_{\mathbf{x}\sim\rho_0,\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\big[Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\big], \tag{82a}$$

$$\text{s.t.} \quad \underbrace{\mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\,\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\Big[\underbrace{\max\big\{0, C^{(m)}(\mathbf{x},\mathbf{u})\big\}}_{:=\tilde{C}^{(m)}(\mathbf{x},\mathbf{u})}\Big]}_{:=\tilde{C}^{(m)}_\theta(\mathbf{x})} \leq 0, \quad \forall m. \tag{82b}$$

In the above, we additionally define $\tilde{C}^{(m)}_\theta(\mathbf{x}) \coloneqq \max\{0, C^{(m)}(\mathbf{x},\mathbf{u})\}$ for convenience.

Borrowing the ideas of gradient projection from multi-objective optimization (Yu et al., 2020; Liu et al., 2021a), we combine the gradient from the objective minimization (82a), and the gradient from constraint satisfaction (82b) (equivalent to constraint minimization due to the constraints being non-negative) by projecting the gradient of (82a) such that it is orthogonal to the gradient of *all* m constraints (82b). We can do this in a **single** backward pass by using Informal Theorem 3.

Let ψ denote the *stop gradient* function, such that the gradient of ψ is equal to zero. Then, for a state distribution ρ,

$$\begin{aligned}
\sigma &\coloneqq \mathbb{E}_{\mathbf{x}\sim\rho}[\nabla_\theta \max_m \tilde{C}^{(m)}_\theta(\mathbf{x})] && (83)\\
&= \mathbb{E}_{\mathbf{x}\sim\rho}\,\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\mathbf{x})}\Big[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\ \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})\Big], && (84)\\
&= \mathbb{E}_{\mathbf{x}\sim\psi(\rho)}\,\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\mathbf{x}))}\Big[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\ \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})\Big], && (85)\\
&= \nabla_\theta\,\mathbb{E}_{\mathbf{x}\sim\psi(\rho)}\,\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\mathbf{x}))}\Big[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\ \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})\Big], && (86)
\end{aligned}$$

where the second line uses the score function gradient (Appendix A.4), and

$$\begin{aligned}
g &= \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}\Big[\mathbb{1}_{\{\max_m \tilde{C}^{(m)}_\theta(\mathbf{x})\leq 0\}}\,\mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right]\Big], && (87)\\
&= \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})}\Big[\mathbb{1}_{\{\max_m \tilde{C}^{(m)}_\theta(\mathbf{x})\leq 0\}}\,\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u}))\right]\Big], && (88)\\
&= \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})}\Big[\mathbb{1}_{\{\max_m \tilde{C}^{(m)}_\theta(\mathbf{x})\leq 0\}}\,\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))}\left[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u}))\right]\Big], && (89)
\end{aligned}$$

where the second line follows from the policy gradient theorem. Let $\tilde{g} \coloneqq \nu\sigma^{(m)} + g.$, where we take the ρ in the definition of $\sigma^{(m)}$ to be equal to $\rho^{\boldsymbol{\pi}_\theta}$ (we discuss the implications of this in the next subsection Appendix B.2):

Theorem A6. *The combined gradient $\tilde{g}$ can be computed as the gradient of the loss function L defined in (14), i.e.,*

$$\tilde{g} = \nabla_\theta L(\theta), \tag{90}$$

where

$$\begin{aligned}
L(\theta) &\coloneqq \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})}\mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))}\big[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\,\psi\big(\tilde{Q}(\mathbf{x},\mathbf{u},\theta)\big)\big], && (91)\\
\tilde{Q}(\mathbf{x},\mathbf{u},\theta) &\coloneqq \begin{cases} Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u}), & \tilde{C}^{(m)}_\theta(\mathbf{x}) \leq 0,\ \forall m,\\ \nu \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u}), & \textit{otherwise}. \end{cases} && (92)\\
&= \mathbb{1}_{\{\max_m \tilde{C}^{(m)}_\theta(\mathbf{x})\leq 0\}}\psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})) + \nu \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u}). && (93)
\end{aligned}$$

Proof. Taking $\rho = \rho^{\boldsymbol{\pi}_\theta}$ for σ and summing the expressions for σ and g in (86) and (89) respectively gives

$$\begin{aligned}
\nu\sigma + g &= \nu\nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})} \mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))} \left[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})\right] \\
&\quad + \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})} \left[\mathbb{1}_{\{\max_m \tilde{C}_\theta^{(m)}(\mathbf{x})\leq 0\}} \mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))} \left[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u}))\right]\right] && (94) \\
&= \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})} \mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))} \left[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \left(\nu \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u}) + \psi(Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u}))\right)\right] && (95) \\
&= \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})} \mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))} \Big[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \psi\big(\underbrace{\nu \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u}) + Q^{\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})}_{:=\tilde{Q}(\mathbf{x},\mathbf{u},\theta)}\big)\Big] && (96) \\
&= \nabla_\theta \mathbb{E}_{\mathbf{x}\sim\psi(\rho^{\boldsymbol{\pi}_\theta})} \mathbb{E}_{\mathbf{u}\sim\psi(\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}))} \left[\log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \psi\big(\tilde{Q}(\mathbf{x},\mathbf{u},\theta)\big)\right] && (97)
\end{aligned}$$

□

In other words, Theorem A6 implies that $\tilde{g}$ can be computed using a single backward pass.

B.2 Implications of minimizing the constraint violation over ρ^{π_θ}

Note that a key step that allows us to move the gradient operator between the inside (83) and outside (86) for the expression of σ is our use of ψ, because this allows this equivalent holding even when we take $\rho = \rho^{\pi_\theta}$.

Without the ψ and taking $\rho = \rho^{\boldsymbol{\pi}_\theta}$, we would obtain that (86) is *instead* equivalent to

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}} \mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})} [\max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})] &= \nabla_\theta \mathbb{E}_{\mathbf{x}_0\sim\rho_0, \mathbf{u}^k\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x}^k)} \left[\underbrace{\sum_{k=0}^{\infty} \max_m \tilde{C}^{(m)}(\mathbf{x}^k,\mathbf{u}^k)}_{:=Q^{C,\boldsymbol{\pi}_\theta}(\mathbf{x}_0,\mathbf{u})}\right] && (98) \\
&\propto \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}} \mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\cdot|\mathbf{x})} \left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u})\, Q^{C,\boldsymbol{\pi}_\theta}(\mathbf{x},\mathbf{u})\right], && (99) \\
&\neq \mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}} \mathbb{E}_{\mathbf{u}\sim\boldsymbol{\pi}_\theta(\mathbf{x})} \left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{x},\mathbf{u}) \max_m \tilde{C}^{(m)}(\mathbf{x},\mathbf{u})\right]. && (100)
\end{aligned}$$

Note that the second line comes from the use of the **policy gradient theorem**, while the expression in (100) comes from the application of the **score function gradient** Appendix A.4. Consequently, while (100) will minimize the maximum DCBF violation $\tilde{C}^{(m)}$, (99) minimizes the *sum* of **future** DCBF violations as well.

Another way to see this is to note that (98) can be viewed as an optimal control problem with *cost* equal to the DCBF violation. In other words, in (98) the policy will additionally try to move to states where the DCBF violation is small as opposed to changing only the control $\mathbf{u}$ such that it satisfies the DCBF conditions in (100).

Experimental Validation. To validate our intuition above, we conduct experiments in the TRANSPORT2 environment to compare using (98) with (100), and plot the training curves in Figure 7. The results show that using (98) converges much slower in cost, which matches the intuition above. Namely, (98) *additionally* tries to avoid *states* where the DCBF constraint violation is high, which leads to unnecessary conservatism.

C Experiments

C.1 Computation resources

The experiments are run on a 13th Gen Intel(R) Core(TM) i7-13700KF CPU with 64GB RAM and an NVIDIA GeForce RTX 4090 GPU. The training time is around 12 hours for 2×10^5 steps for DGPPO, 14 hours for Lagr and 10 hours for Penalty.

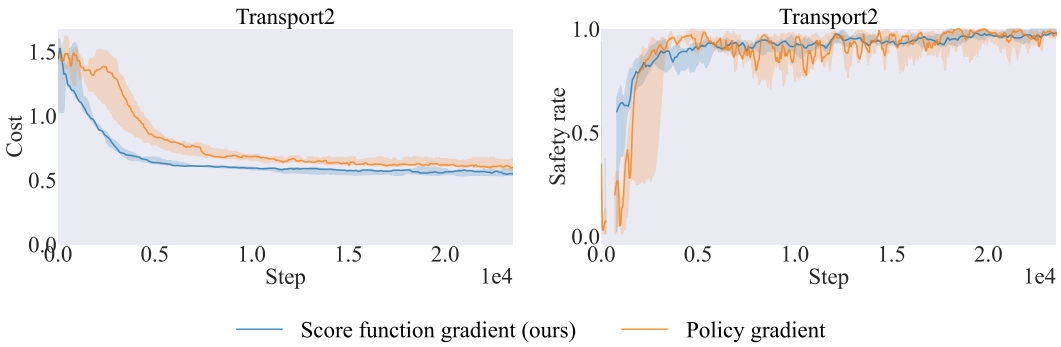


Figure 7: In comparison to DGPPO which uses (100), using (98) converge much slower in cost, which matches the intuition that (98) is optimizing for the wrong objective due to unnecessarily avoiding states where the DCBF constraint violation is high.

C.2 Environments

C.2.1 LiDAR environments

In the LiDAR environments, we assume that the agents have a radius of $r = 0.05$ and a local sensing radius $R = 0.5$ such that one agent can observe other agents or obstacles only when they are within its sensing radius. Agents use LiDAR to detect obstacles. For each agent, there are 32 evenly-spaced LiDAR rays originating from each agent measuring the relative location of obstacles. To reduce the size of the multi-agent graph, the 8 shortest LiDAR rays are returned.

We use directed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent the LiDAR environments. $\mathcal{V}$ is the set of nodes containing the objects in the environments, including agents $\mathcal{V}_a$, goals/landmarks V_g, and the hitting points of LiDAR rays (obstacles) $\mathcal{V}_o$. The edges $\mathcal{E} \subseteq \{(i,j) \mid i \in \mathcal{V}_a, j \in \mathcal{V}\}$ denote the information flow from a sender node j to a receiver node (agent) i. An edge (i,j) exists only if the distance between node i and j are smaller than the sensing radius R. The neighbor nodes of agent i is defined as $\mathcal{N}_i \coloneqq \{j \mid (i,j) \in \mathcal{E}\}$, so that the information flow happens between the agents and their neighbors. The node features v_i include the state of the node x_i and a one-hot encoding of the type of the node i (e.g., agent, goal/landmark, LiDAR hitting points). The edge features e_{ij} include the information passed from node j to node i, including the relative positions and velocities.

In all LiDAR environments, we include 3 rectangle-shaped obstacles, and the agents need to avoid inter-agent collision and agent-obstacle collisions.

We consider 4 LiDAR environments: Target, Spread, Line, and Bicycle:

Target: The agents need to reach their pre-assigned goals (Figure 2a).

Spread: The agents need to collectively cover a set of goals without having access to an assignment (Figure 2b).

Line: The agents need to form a line between two given landmarks (Figure 2c).

Bicycle: The agents follow the more difficult bicycle dynamics. The task here is the same as Target (Figure 2d).

The agents in the first 3 environments follow the double integrator dynamics. The state of agent i is $x_i = [p_i^x, p_i^y, v_i^x, v_i^y]^\top$, where $[p_i^x, p_i^y]^\top \coloneqq p_i \in \mathbb{R}^2$ is the position of agent i, and $[v_i^x, v_i^y]$ is its velocity. The control inputs are $u_i = [a_i^x, a_i^y]^\top$, which are the acceleration along the x-axis and y-axis. The agents follow the dynamics

$$\dot{x}_i = \begin{bmatrix} v_i^x & v_i^y & a_i^x & a_i^y \end{bmatrix}^\top. \tag{101}$$

We limit the control inputs of the agents to be within $[-1, 1]$, and also the velocities to be within $[-10, 10]$. In the Bicycle environment, the state of agent i is defined with

$x_i = [p_i^x, p_i^y, \cos\theta, \sin\theta, v]^\top$, where θ is the heading and v is the speed. The control inputs are $u_i = [\delta, a]^\top$, including the steering angle δ and the acceleration a. The agents follow the bicycle dynamics given by

$$\dot{x}_i = [v\cos\theta \quad v\sin\theta \quad -v\sin\theta\tan\delta \quad v\cos\theta\tan\delta \quad a]^\top. \tag{102}$$

The control inputs are limited by $v \in [-10, 10]$ and $\delta \in [-1.47, 1.47]$. In all LiDAR environments, we use a simulation time step of 0.03 seconds and a total horizon of 128 time steps. For all LiDAR environments, the edge features are defined as the relative positions and relative velocities between the nodes.

In LiDAR environments, the constraint function h contains two parts: agent-agent collisions and agent-obstacle collisions. The agent-agent collisions $h^{(1)}$ function is defined as

$$h^{(1)}(o_i) = 2r - \min_{j\in\mathcal{N}_i} \|p_i - p_j\|, \tag{103}$$

and the agent-obstacle collision $h^{(2)}$ function is

$$h^{(2)}(o_i) = r - \min_{j\in\mathcal{N}_i} \|p_i - p_j\|. \tag{104}$$

For the cost functions l, we consider two types of them. The first type is used in the TARGET and the BICYCLE environments, where the agents need to *reach* their pre-assigned goals. We define this type of cost function with

$$l(\mathbf{x}, \mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N}\left(0.01\|p_i - p_i^{\text{goal}}\| + 0.001\text{sign}\left(\text{ReLU}(\|p_i - p_i^{\text{goal}}\| - 0.01)\right) + 0.0001\|u_i\|^2\right), \tag{105}$$

where the first term penalizes the agents if they cannot reach the goal, the second term penalizes the agents if they cannot reach the goal exactly, and the third term encourages small controls. The second type of the cost functions is used in the SPREAD and the LINE environments, where the agents need to *cover* some goals/landmarks. We define this type of cost function with

$$l(\mathbf{x}, \mathbf{u}) = \frac{1}{N}\sum_{j=1}^{N}\min_{i\in\mathcal{V}_a}\Big(0.01\|p_i - p_j^{\text{goal}}\| + 0.001\text{sign}\left(\text{ReLU}(\|p_i - p_j^{\text{goal}}\| - 0.01)\right) + 0.0001\|u_j\|^2\Big). \tag{106}$$

Here, each goal finds its nearest agent and penalizes the whole team with the distance between them. In this way, the optimal policy of the agents is to cover all goals collaboratively.

C.2.2 MUJOCO ENVIRONMENTS

For the TRANSPORT environment, we model the agents as double integrators and control the forces applied to each agent using the MuJoCo simulator (Todorov et al., 2012). We limit the control inputs of the agents to be within $[-1, 1]$. The agents need to collaboratively push a box from the inside so that the box reaches a given goal while avoiding colliding with each other.

We use a similar cost function $l(\mathbf{x}, \mathbf{u})$ as the TARGET environment but only for the box, i.e.,

$$l(\mathbf{x}, \mathbf{u}) = 0.01\|p_{\text{box}} - p_{\text{box}}^{\text{goal}}\| + 0.001\text{sign}\left(\text{ReLU}(\|p_{\text{box}} - p_{\text{box}}^{\text{goal}}\| - 0.01)\right) \tag{107}$$

We usee a single constraint function $h^{(1)}$ for the agent-agent collision, defined as

$$h^{(1)}(o_i) = 2r - \min_{j\in\mathcal{N}_i} \|p_i - p_j\|, \tag{108}$$

where p_i denotes the position of agent i, and r is the radius of the agent.

C.2.3 VMAS ENVIRONMENTS

We use the REVERSETRANSPORT (which we call TRANSPORT2) and WHEEL environments from VMAS (Bettini et al., 2022; 2024). Note that the obstacles in the VMAS environments are not represented using LiDAR but using states including their positions and sizes. In both environments, the agents are modeled as double integrators and control the forces applied to each agent. We limit the control inputs of the agents to be within $[-1, 1]$.

TRANSPORT2: In this environment, N agents are placed in a square red package and must push the package from the inside to a goal location. The red package and the goal location are randomly sampled. Unlike the original REVERSETRANSPORT environment, we include the following two additional safety constraints:

- **Inter-agent collision avoidance:** The agents must avoid colliding with each other.
- **Package collision avoidance:** The center of the package must avoid colliding with three randomly placed circular obstacles.

We use a similar cost function $l(\mathbf{x}, \mathbf{u})$ as the TARGET environment but only for the package, i.e.,

$$l(\mathbf{x}, \mathbf{u}) = 0.01\|p_{\text{package}} - p^{\text{goal}}_{\text{package}}\| + 0.001\text{sign}\left(\text{ReLU}(\|p_{\text{package}} - p^{\text{goal}}_{\text{package}}\| - 0.01)\right) \tag{109}$$

We usee a two constraint functions $h^{(1)}$, $h^{(2)}$ for the agent-agent collision and package-obstacle collisions respectively. The agent-agent collision function $h^{(1)}$ is defined as

$$h^{(1)}(o_i) = 2r - \min_{j \in \mathcal{N}_i} \|p_i - p_j\|, \tag{110}$$

where p_i denotes the position of agent i, and r is the radius of the agent. The package-obstacle collision function $h^{(2)}$ is defined as

$$h^{(2)}(o_i) = r_{\text{obs}} - \min_{q \in \{1,2,3\}} \|p_{\text{package}} - p_q\|. \tag{111}$$

WHEEL: In this environment, N agents must collectively rotate a line anchored to the origin with a large mass. Unlike the original WHEEL environment, we modify the goal to be a target angle that the line must rotate to. We also include the following two additional safety constraints:

- **Inter-agent collision avoidance:** The agents must avoid colliding with each other.
- **Line collision avoidance:** The angle of the line must stay outside of a certain range of angles.

We use the same cost function $l(\mathbf{x}, \mathbf{u})$ as the TARGET environment but on the angle of the line, i.e.,

$$l(\mathbf{x}, \mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N}\left(0.01\|p_i - p_i^{\text{goal}}\| + 0.001\text{sign}\left(\text{ReLU}(\|p_i - p_i^{\text{goal}}\| - 0.01)\right) + 0.0001\|u_i\|^2\right). \tag{112}$$

We usee a two constraint functions $h^{(1)}$, $h^{(2)}$ for the agent-agent collision and line-obstacle collisions respectively. The agent-agent collision function $h^{(1)}$ is defined as

$$h^{(1)}(o_i) = 2r - \min_{j \in \mathcal{N}_i} \|p_i - p_j\|, \tag{113}$$

where p_i denotes the position of agent i, and r is the radius of the agent. The package-obstacle collision function $h^{(2)}$ is defined as

$$h^{(2)}(o_i) = r_{\text{obs}} - |\theta_{\text{line}} - \theta_{\text{obs}}|, \tag{114}$$

where the absolute value on the angle $|\cdot|$ is defined as the minimum angle between the two angles.

C.3 IMPLEMENT DETAILS AND HYPERPARAMETERS

In our experiment, we let all agents share the same parameters for their policies and value functions. More specifically, we parameterize the agent's policy with $\pi_\theta : \mathcal{O} \to \mathcal{U}$, cost-value function with $V^l_\phi : \boldsymbol{\mathcal{X}} \to \mathbb{R}$, and constraint-value function $V^{h^{(m)}}_\psi : \mathcal{O} \to \mathbb{R}^m$ using graph transformers (Shi et al., 2020) with parameters θ, ϕ, and ψ, respectively. Since the cost-value function V^l_ϕ is centralized, after the graph transformer, we compute the average of all node features and pass that to a final layer (multi-layer perceptron) to obtain the global cost value for the MAS.

In Table 1, we provide the value of the common hyperparameters for DGPPO and the baselines. Besides these common hyperparameters, the value of the unique hyperparameters of DGPPO are provided in Table 2.

Table 1: Common hyperparameters of DGPPO and the baselines.

Hyperparameter	Value	Hyperparameter	Value
policy GNN layers	2	RNN type	GRU
massage passing dimension	32	RNN data chunk length	16
GNN output dimension	64	RNN layers	1
number of attention heads	3	number of sampling environments	128
activation functions	ReLU	gradient clip norm	2
GNN head layers	(32, 32)	entropy coefficient	0.01
optimizer	Adam	GAE λ	0.95
discount γ	0.99	clip ϵ	0.25
policy learning rate	3e-4	PPO epoch	1
V^l learning rate	1e-3	batch size	16384
network initialization	Orthogonal	layer normalization	True
V^l GNN layers	2		

Table 2: Unique hyperparameters of DGPPO

Hyperparameter	Value
V^h GNN layers	1
a	0.3
ν	Scheduled. Initialized at $\nu = 1$, then doubled at 0.5 and 0.75 of the total update steps.

C.4 Implementation of the baselines

We implement a JAX (Bradbury et al., 2018) version of the baselines following their original implementations:

- InforMARL: https://github.com/nsidn98/InforMARL (MIT license)
- MAPPO-L: https://github.com/chauncygu/Multi-Agent-Constrained-Policy-Optimisation (MIT License)

C.5 Training curves

Here, we provide the cost and safety rate during training for all algorithms in Figure 8. We can observe that DGPPO achieves stable training in all environments with only one constant set of hyperparameters.

C.6 Additional ablation studies

C.6.1 Comparison between the decoupling method and the coupled methods

In Section 4.3, we have introduced a decoupling method to performance gradient descent update following Equation (14). Here, we empirically compare the decoupling method (14) with the CRPO-style coupling method (12) and the CRPO-style coupling method with the constraint being on $C(x, u)$ instead of $\max\{0, C(x, u)\}$ (Equation (10)). We conduct experiments in the Transport2 environment and compare the safety rate and cost of the converged policies with different methods in Figure 9. We can observe that DGPPO achieves a much lower cost compared with the coupling methods. This is because the coupling methods perform gradient descent to minimize $Q^{\boldsymbol{\pi}_\theta}$ only when the *whole trajectory* is safe, i.e., $\mathbb{E}_{\mathbf{x}\sim\rho^{\boldsymbol{\pi}_\theta}}[\tilde{C}_\theta(\mathbf{x})] \leq 0$. This is much more conservative than the safety requirement on the per-transition level used by the decoupling method, especially in the multi-agent case. Therefore, the coupling method has little chance to minimize $Q^{\boldsymbol{\pi}_\theta}$ during training but focuses on safety, resulting in a safe policy with poor performance. On the other hand, if the coupling method is performed with $C(x, u)$ instead of $\max\{0, C(x, u)\}$ (Equation (10)), the learned policy is no longer safe. This matches our discussion in Section 4.3.

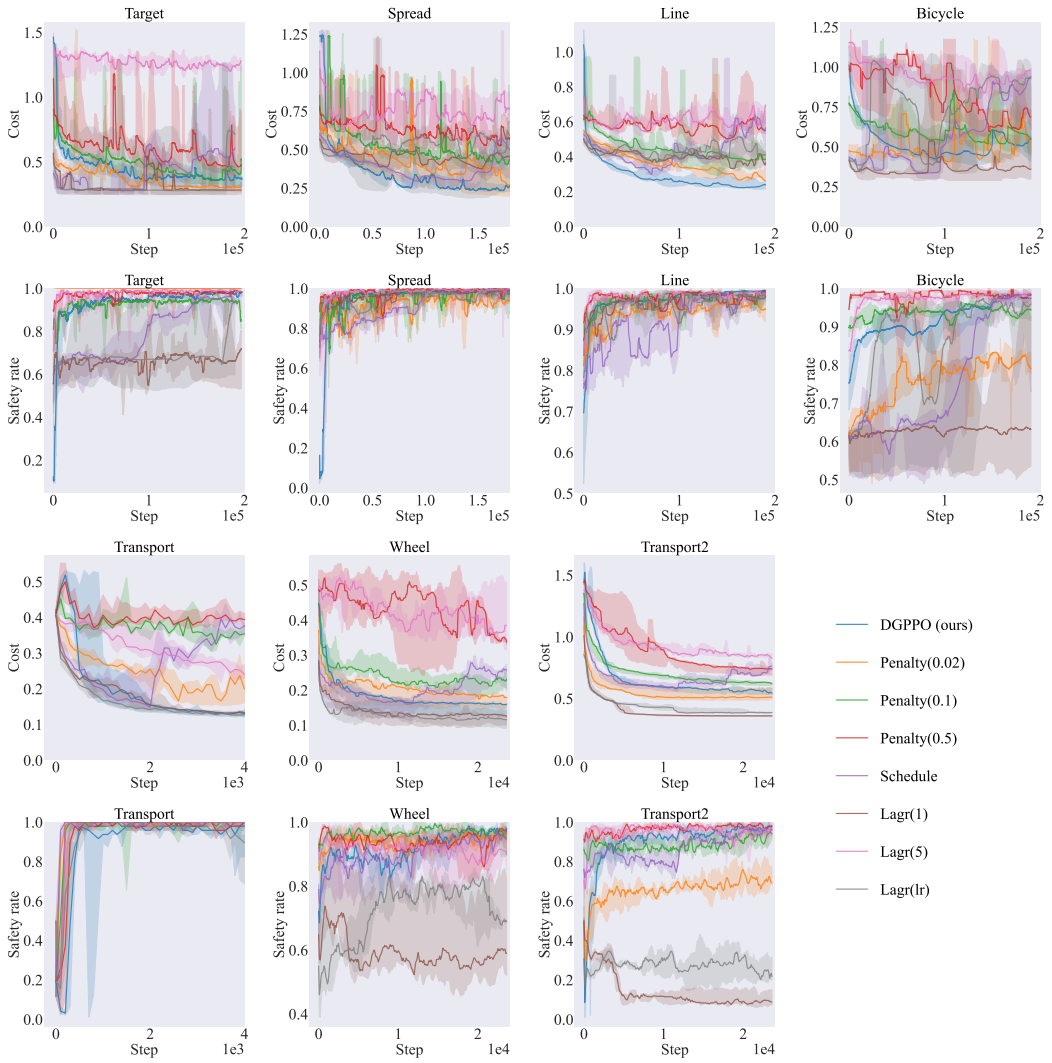


Figure 8: Costs and safety rates of DGPPO and the baselines during training.

C.6.2 Provide the baselines with more data

In Section 4.5, we introduced that DGPPO requires sampling with both a stochastic and a deterministic policy. This means that DGPPO requires twice as much data per update step compared with the baselines, although the update steps are the same. Here, we answer the question that *what if the baselines are provided with double data*? We choose the Wheel environment and select the three best baselines in Figure 3, namely Penalty(0.02), Penalty(0.1), and Schedule. To double the data, we consider two situations: doubling the batch size and keeping the training steps (similar to DGPPO), and doubling the training steps and keeping the batch size. The results are shown in Figure 10, where the semi-transparent colors show the original performance of the baselines, and the non-transparent colors show the performance of baselines after doubling the data. We can observe that after doubling the data, it is uncertain how the performance of the baselines changes. For example, Penalty(0.02) performs better than before with doubling the batch size in Figure 10a, but performs worse with doubling training steps in Figure 10b. On the contrary, DGPPO consistently outperforms all baselines.

C.6.3 Comparison with constrained optimization without a CBF

As the proposed algorithm DGPPO is based on CBF, one natural question to ask is *what if we do not learn the CBF* but directly perform constrained optimization? To answer the question, we consider

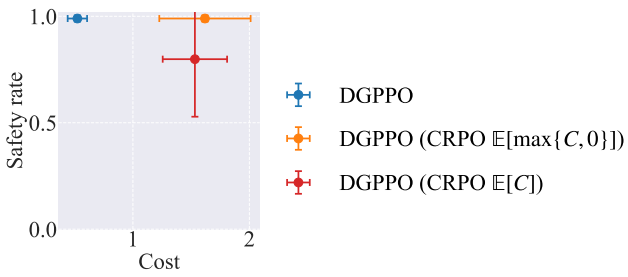
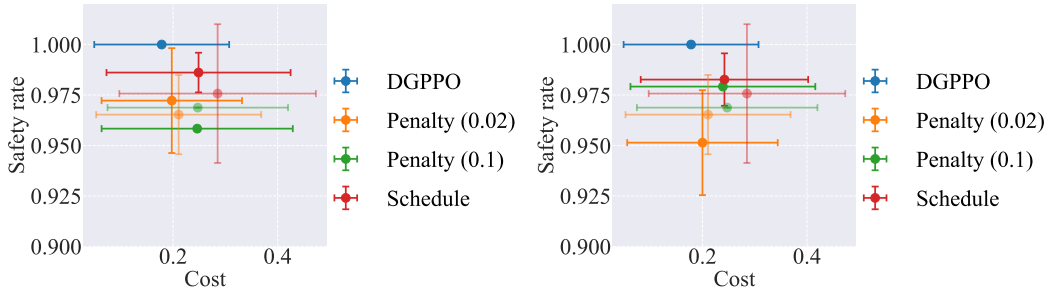


Figure 9: Comparison between DGPPO with the decoupling method and the CRPO-style update with $\mathbb{E}[\max\{0, C\}]$ and with $\mathbb{E}[C]$.



(a) Doubling the batch size.

(b) Doubling the training steps.

Figure 10: Costs and safety rates of DGPPO and three best baselines in the WHEEL environment. We also plot the original performance of the baselines in semi-transparent colors.

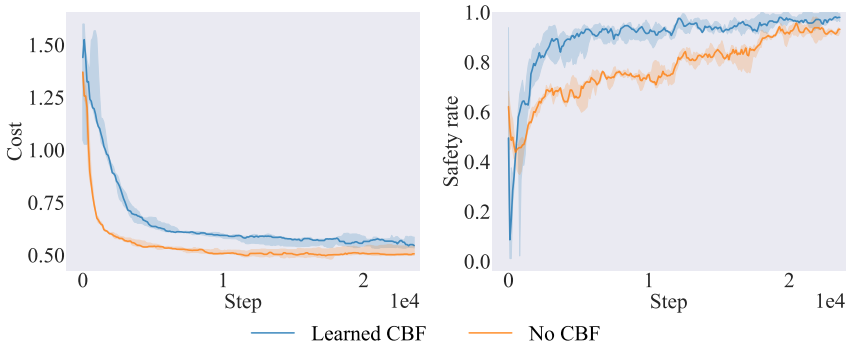


Figure 11: Comparison between DGPPO with doing constrained optimization without a CBF.

another baseline which changes the constraint in Equation (11b) to $\mathbb{E}_{\mathbf{x}\sim\rho^{\pi_\theta}}\left[\max\{0, h^{(m)}(\mathbf{x})\}\right] \leq 0$ while keep all other parts the same as DGPPO. We compare DGPPO (Learned CBF) with this new baseline (No CBF) in the TRANSPORT2 environment. The results are shown in Figure 11, which suggests that No CBF cannot achieve a safety rate that is as high as Learned CBF. Intuitively, it is because CBF not only constrains entering the avoid set, but also constrains the rate at which the agent can approach the safe-unsafe boundary. Therefore, it is more robust to estimation errors than directly doing constrained optimization.

C.6.4 SENSITIVITY ANALYSIS OF ν

In Section 5.3, we show that our proposed ν scheduling method achieves the best result compared with other choices of ν. In particular, $\nu > 2$ results in slower convergence in the cumulative cost. Here, we further perform experiments to demonstrate the influence of ν on DGPPO. We consider the SPREAD environment with $\nu = 1, 2, 4, 6$, and train DGPPO until convergences. The results

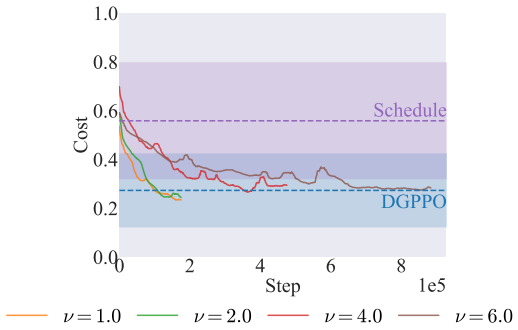


Figure 12: Influence of different ν on the convergence speed and the converged result of DGPPO. The dashed lines show the mean of DGPPO and the baseline with the best performance (Schedule), and the shades show the standard deviation.

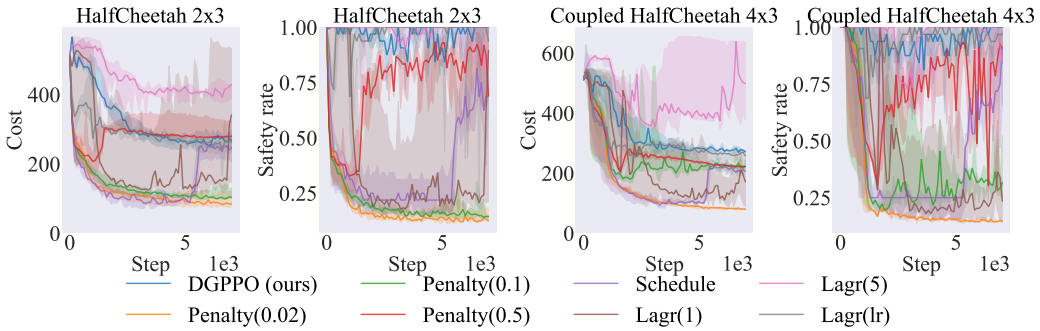


Figure 13: Cost and safety rates of DGPPO and the baselines during training in the Safe multi-agent MuJoCo environments.

are shown in Figure 12, where the mean cost and standard deviation of DGPPO and the baseline with the best performance (Schedule) are shown in dashed lines and shades. We can observe that although the convergence of DGPPO becomes slower with larger ν, *the converged costs are the same*, and are much lower than Schedule. This phenomenon is different from the baselines, where different choices of hyperparameters directly affect the converged costs (See Figure 3).

C.7 Additional environments in the Safe multi-agent MuJoCo environments

To further demonstrate the ability of DGPPO in environments with complex discrete-time dynamics, here we consider another benchmark named safe multi-agent MuJoCo (Gu et al., 2023). We use the Safe HalfCheetah(2x3) and the Safe Coupled HalfCheetah(4x3) tasks, where the agents control different subsets of joints of one or two cheetahs. The first number in the parenthesis denotes the number of agents, while the second number shows the number of joints that each agent controls. The agents need to work collaboratively to maximize the forward velocity but avoid colliding with a wall that moves forward at a predefined velocity. The results are shown in Figure 13, which suggests the same results as the main experiments in Section 5.2, that DGPPO has the best performance with a fixed set of hyperparameters.

C.8 Scalability and Generalizability

Here we test the scalability and generalizability of DGPPO. Following Zhang et al. (2025), we define scalability as the number of agents during training, and generalizability as the ability to be deployed with more agents during test time.

Considering scalability, DGPPO has a similar performance as its based method GCBF+ (Zhang et al., 2025) as we have shown in Section 5.2 that DGPPO can be trained on 7 agents, and GCBF+ is trained with 8 in its original paper. In addition to Figure 5 in the main pages, here we also provide the comparison of the costs and safety rates of the converged policies of each algorithm in Figure 14.

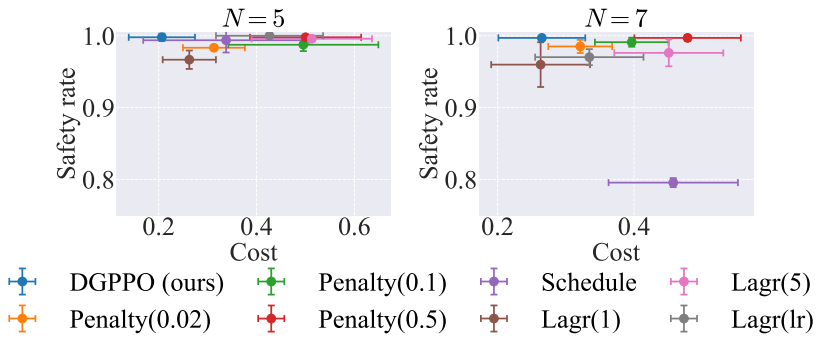


Figure 14: Costs and safety rates of the converged policies of DGPPO and the baselines in environments with $N = 5, 7$.

Table 3: Generalizability test results of DGPPO.

Number of agents	Safety rate	Normalized cost
8	1.000 ± 0.000	1.673 ± 0.430
16	0.992 ± 0.088	1.784 ± 0.316
32	0.987 ± 0.112	1.748 ± 0.235
64	0.986 ± 0.118	1.799 ± 0.418
128	0.982 ± 0.133	1.839 ± 0.323
256	0.985 ± 0.122	1.823 ± 0.366
512	0.985 ± 0.123	1.821 ± 0.390

Considering generalizability, GCBF+ can be deployed on 512 agents without significant performance loss after training. Here, we perform a new experiment in the TARGET environment where DGPPO is also trained with 8 agents. In Table 3, we show the test results of DGPPO deployed with larger numbers of agents. We observe that DGPPO maintains high safety rates and low costs when deployed on up to 512 agents.

However, the above results are obtained in environments with the same agent density as the training environment. We cannot deploy DGPPO in environments with significantly higher agent density than the training environment because RL algorithms are sensitive to distribution shifts. We leave handling large distribution shifts to future work.

C.9 CODE

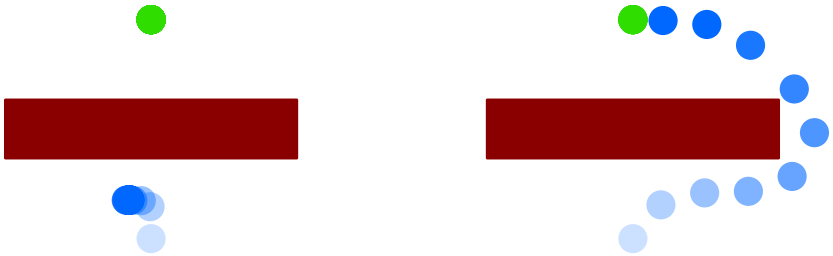
The code of our algorithm and the baselines are provided in the 'dgppo.zip' file in the supplementary materials and online at `https://github.com/MIT-REALM/dgppo`.

D MORE DISCUSSION ABOUT DGPPO

D.1 ADVANTAGES ON NOT DEPENDING ON A NOMINAL POLICY

As discussed in Section 1, one of the drawbacks of the CBF-based methods (Wang et al., 2017; Zhang et al., 2025) is that they require a nominal policy that can achieve high task performance. Here we further discuss why relying on a nominal performant policy is *not* a good idea.

Requirement of Simple or Known Dynamics. Controller design usually requires the dynamics to be simple or requires knowledge of the dynamics. The PID controllers in Zhang et al. (2025) are constructed for the unicycle dynamics. More generally, PID controllers are usually only used with single-input single-output systems. For more complicated systems, one could use LQR or MPC, but this requires full knowledge of the dynamics. In addition, PID controllers are much more difficult to apply in environments with complex *contact dynamics*, for example, our TRANSPORT, WHEEL, and TRANSPORT2 environments.



(a) GCBF+ (Zhang et al., 2025)

(b) DGPPO

Figure 15: Comparison of the converged policies learned using DGPPO and GCBF+ (Zhang et al., 2025) in a TARGET environment, where the agent (blue) needs to avoid the large obstacle (red) and reach the goal (green). The GCBF+ policy is myopic and gets stuck in a deadlock, while the DGPPO policy avoids the obstacle and reaches the goal to minimize the cumulative cost.

Deadlocks. Another drawback is that the CBF-QP approach of Zhang et al. (2025) leads to deadlocks, as discussed in Section VIII of Zhang et al. (2025) or theoretically in e.g. Grover et al. (2021). This is because the safety filter approach of Zhang et al. (2025) only minimizes deviations from the nominal policy at the current time step, even if this leads to a deadlock at a future time step. In contrast, minimizing the cumulative cost directly takes future behavior into account and hence will try to avoid deadlocks. Here we perform an additional experiment to demonstrate this. We apply the converged controller trained with GCBF+ and DGPPO respectively in the TARGET environment shown in Figure 15, where the agent needs to get around a large static obstacle and reach the goal. In the figure, we can observe that the GCBF+ policy gets in a deadlock behind the obstacle because the GCBF+ policy is myopic and only considers safety safe and minimizes the deviation from the reference controller at the *current* timestep. Therefore, stopping behind the obstacle is the optimal solution for GCBF+. On the contrary, the DGPPO policy successfully reaches the goal because it minimizes the *long-horizon* cumulative cost.