GCBF+: A Neural Graph Control Barrier Function Framework for Distributed Safe Multiagent Control

Songyuan Zhang, *Student Member, IEEE*, Oswin So, *Graduate Student Member, IEEE*, Kunal Garg, *Member, IEEE*, and Chuchu Fan, *Member, IEEE*

Abstract*—Distributed, scalable, and safe control of large-scale multiagent systems is a challenging problem. In this article, we design a distributed framework for safe multiagent control in large-scale environments with obstacles, where a large number of agents are required to maintain safety using only local information and reach their goal locations. We introduce a new class of certificates, termed graph control barrier function (GCBF), which are based on the well-established control barrier function theory for safety guarantees and utilize a graph structure for scalable and generalizable distributed control of MAS. We develop a novel theoretical framework to prove the safety of an arbitrary-sized MAS with a single GCBF. We propose a new training framework GCBF+ that uses graph neural networks to parameterize a candidate GCBF and a distributed control policy. The proposed framework is distributed and is capable of taking point clouds from LiDAR, instead of actual state information, for real-world robotic applications. We illustrate the efficacy of the proposed method through various hardware experiments on a swarm of drones with objectives ranging from exchanging positions to docking on a moving target without collision. In addition, we perform extensive numerical experiments, where the number and density of agents, as well as the number of obstacles, increase. Empirical results show that in complex environments with agents with nonlinear dynamics (e.g., Crazyflie drones), GCBF+ outperforms the hand-crafted CBF-based method with the best performance by up to 20% for relatively small-scale MAS with up to 256 agents, and leading reinforcement learning (RL) methods by up to 40% for MAS with 1024 agents. Furthermore, the proposed method does not compromise on the performance, in terms of goal reaching, for achieving high safety rates, which is a common tradeoff in RL-based methods. Project website: https://mit-realm.github.io/gcbfplus/

***Index Terms*—Machine learning, multi-agent systems, neural certificates, robotics and automation, robot safety.**

I. Introduction

A. Background

MULTIAGENT systems (MAS) have received tremendous attention from scholars in different disciplines, including computer science and robotics, as a means to solve complex problems by subdividing them into smaller tasks [1]. MAS applications include but are not limited to warehouse operations [2], [3], self-driving cars [4], [5], [6], [7], coordinated navigation of a swarm of drones in a dense forest for search-and-rescue missions [8], [9]; interested reader is referred to [10] for an overview of MAS applications. For such safety-critical MASs, it is important to design controllers that not only guarantee safety in terms of collision and obstacle avoidance but are also scalable to large-scale multiagent problems.

Common MAS motion planning methods include but are not limited to solving mixed integer linear programs for computing safe paths for agents [11], [12] and sampling-based planning methods such as rapidly exploring random tree (RRT) [13]. However, such *centralized-in-execution* approaches, where the complete MAS state information is used, are not scalable to large-scale MAS. The recent work [14] performing distributed trajectory optimization proposes a scalable method for MAS control. However, this approach cannot take into account changing neighborhoods and environments, limiting its applications. There has been quite a lot of development in the learning-based methods for MAS control in recent years; see [15] for a detailed overview of learning-based methods for safe control of MAS.

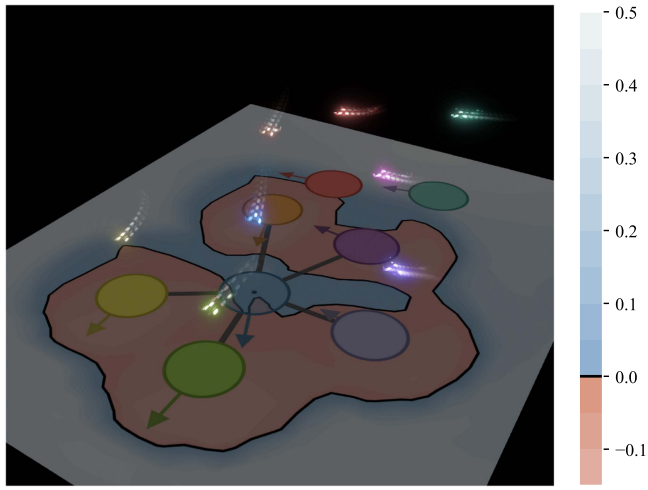


Fig. 1. *8-CF swapping with GCBF:* We learn a distributed GCBF for an 8-agent swapping task on the CF hardware platform. We visualize the learned GCBF for the *blue agent* and draw the edges with its neighboring agents in grey. The learned GCBF can handle arbitrary graph topologies and, hence, can scale to an arbitrary number of agents without retraining.

Received 21 October 2024; accepted 20 December 2024. Date of publication 15 January 2025; date of current version 21 February 2025. This work was supported in part by the National Science Foundation (NSF) CAREER Award under grant CCF-2238030, in part by the MIT Lincoln Lab under the Safety in Aerobatic Flight Regimes (SAFR) program, and in part by the MIT-DSTA Program. This article was recommended for publication by Associate Editor J. Alonso-Mora and Editor M. Schwager upon evaluation of the reviewers' comments. *(Corresponding author: Songyuan Zhang.)*

The authors are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: szhang21@mit.edu; oswinso@gmail.com; kgarg@umich.edu; chuchu-fan1990@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3530348, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3530348

Multiagent reinforcement learning (MARL)-based approaches, e.g., multiagent proximal policy optimization (MAPPO) [16], have also been adapted to solve multiagent motion planning problems. However, when it comes to reinforcement learning (RL), one main challenge for safety, particularly in multiagent cases, is the tradeoff between the practical performance and the safety requirement because of the conflicting reward-penalty structure [15]. We argue that our proposed framework does not suffer from a similar tradeoff and can automatically balance satisfying safety requirements and performance criteria through a carefully designed training loss.

In the past few years, control barrier functions (CBFs) have become a popular tool to encode safety requirements for robotic systems [17]. For MAS, safety is generally formulated pair-wise. Therefore, a CBF is assigned for each pair-wise safety constraint, and approximation methods are used to combine multiple constraints [18], [19], [20], [21]. While hand-crafted CBF-based quadratic programs (CBF-QP) have shown promising results for single-agent systems [22] and *simple* (i.e., linear systems or systems with relative degree one) small-scale MASs [23], it is difficult to find a CBF when it comes to highly complex nonlinear systems and large-scale MAS. Another major challenge with such approaches is constructing a CBF in the presence of input constraints, e.g., actuation limits. There are some recent developments on this topic for MAS, see, e.g., [24]; however, as noted by the authors in [25], finding a barrier function that satisfies safety conditions in the presence of input bounds is a complex problem.

B. Our Contributions

To overcome these limitations, in this article, we first introduce the notion of graph control barrier function (GCBF) (Fig. 1), for large-scale MAS to address the problems of safety, scalability, and generalizability. The proposed GCBF can account for an arbitrary and changing number of neighbors, and hence, computed for a small-scale MAS, the same GCBF generalizes to large-scale MAS. We provide a new safety result using the notion of GCBF that certifies the safety of MAS of arbitrary size. This is the first such result that shows the safety of MAS of any size using one barrier function. Next, we introduce GCBF+, a novel *centralized training distributed execution* framework to learn a candidate GCBF along with a distributed control policy. We use graph neural networks (GNNs) to capture the changing graphical topology of distance-based observation information flow. We propose a novel loss function formulation that accounts for safety, goal-reaching as well as actuation limits, thereby addressing the limitations of hand-crafted CBF-QP methods. Furthermore, the proposed algorithm can work with LiDAR-based point-cloud observations to handle obstacles in real-world environments. With these technologies, our proposed framework can generalize well to many challenging settings, including crowded and unseen obstacle environments.

To corroborate the practical applicability of the proposed method, we perform various hardware experiments on a swarm of Crazyflie (CF) drones. The hardware experiments consist of the drones safely exchanging positions in a crowded workspace in the presence of static and moving obstacles and docking on a moving target while maintaining safety. We also perform extensive numerical experiments and provide empirical evidence of the improved performance of the proposed GCBF+ framework compared to the prior version of the algorithm (GCBFv0) in [26], a state-of-the-art MARL method (InforMARL) [27], MPC method from [28], and hand-crafted CBF-QP methods from [23]. We consider three 2-D environments and two 3-D environments in our numerical experiments consisting of linear and nonlinear systems. In the obstacle-free case, we train with 8 agents and test with over 1000 agents. In the linear cases, the performance improvement (in terms of safety rate) is about 5%, while in the nonlinear cases, the performance improvement is more than 30%. In the obstacle environment, we consider 8 obstacles in training, while up to 128 obstacles are considered in testing. These experiments corroborate that the proposed method outperforms the baseline methods in successfully completing the tasks in various 2-D and 3-D environments.

C. Differences From Conference Version

This article builds on the conference paper [26], which presented the GCBFv0 algorithm. We propose a new algorithm GCBF+, which improves upon GCBFv0 in the following ways.

1) *Algorithmic Modifications:* In the prior work, the control policy was learned to account only for the safety constraints, and a CBF-based switching mechanism was used for switching between a goal-reaching nominal controller and a safe neural controller. This led to undesirable behavior and deadlocks in certain situations. Furthermore, an online policy refinement mechanism was used in [26] when the learned controller could not satisfy the safety requirements, which required agents to communicate their control actions for the policy update, adding computational overhead. We modify how the training loss is defined so that the safety and goal-reaching requirements do not conflict, making it possible for the training loss to go to zero. In this way, we can use a single controller for both safety and goal-reaching, without an online policy refinement step for higher safety rates.
2) *Actuation Limits:* Another major limitation of GCBFv0 is that it does not account for actuator limits, which may result in undesirable behavior when implemented on real-world robotic systems. In contrast, in the proposed method in this work, the learned controller considers actuator limits through a look-ahead mechanism for approximation of the safe control invariant set. This mechanism ensures that the learned controller satisfies the actuation limits while keeping the system safe.
3) *Theoretical Results on Generalization:* While Zhang et al. [26] proved that a GCBF certifies safety for a specific size of the MAS, it does not prove that the same GCBF certifes safety when the number of agents changes. We advance this theoretical result to prove that a GCBF can certify the safety of an MAS of *any* size. This brings the theoretical understanding of the algorithm closer to the empirical results, where we observe our GCBF+ algorithm

scaling to over 1000 agents while being trained with 8 agents.

4) *Hardware Experiments:* We include various hardware experiments on a swarm of CF drones, thereby demonstrating the practicality of the proposed framework.
5) *Additional Numerical Experiments:* We also include various new numerical experiments as compared to the conference version. In particular, we perform experiments with more realistic system dynamics, such as the 6DOF CF drone, in contrast to simpler dynamics used in the numerical experiments in [26]. Furthermore, we provide comparisons to new baselines: InforMARL from [27], which is a better RL-based method for safe MAS control, MPC from [28], and centralized and distributed handcrafted CBF-based methods from [23].
6) *Better Performance:* We illustrate that the new GCBF+ algorithm proposed in this article has much better performance than the original GCBFv0 algorithm in the conference version. In particular, in complex environments consisting of agents with nonlinear dynamics, GCBFv0 has a success rate of less than 10% while GCBF+ has a success rate of over 55%. Furthermore, in crowded 2-D obstacle environments, GCBFv0 has a success rate close to 20% while GCBF+ has a success rate of over 95%.

D. Related Work

Graph-based methods: Graph-based planning approaches such as *prioritized* multiagent path finding (MAPF) [29] and conflict-based search for MAPF [30] can be used for multiagent path planning for known environments. However, MAPF does not take into account system dynamics and does not scale to large-scale systems due to computational complexity. Another line of work for motion planning in obstacle environments is based on the notion of velocity obstacles [31] defined using collision cones for velocity. Such methods can be used for large-scale systems with safety guarantees under mild assumptions. However, the current frameworks under this notion assume single or double integrator dynamics for agents. The work in [32] scales to large-scale systems, but it only considers a discrete action space and, hence, does not apply to robotic platforms that use more general continuous input signals.

Model predictive control (MPC): To tackle MAS, distributed MPC methods have been proposed, incorporating multiagent path planning and machine learning (ML) [33], [34], [35], [36], [37] with distributed optimization [38], [39], [40]. Although the computation is distributed, many MPC works require a central node to perform global information exchange. In addition, the computation complexity of MPC methods impedes their scalability.

Centralized CBF-based methods: For systems with relatively simple dynamics, such as single integrator, double integrator, and unicycle dynamics, it is possible to use a distance-based CBF [17]. For systems with polynomial dynamics, it is possible to use the sum-of-squares (SoS) [41] method to compute a CBF [42]. The key idea of SoS is that the CBF conditions consist of a set of inequalities, which can be equivalently expressed as checking whether a polynomial function is SoS. In this manner, a CBF can be computed through convex optimization [42], [43], [44]. However, the SoS-based approaches suffer from the curse of dimensionality (i.e., the computational complexity grows exponentially with the degree of polynomials involved) [45].

Distributed CBF-based methods: While centralized CBF is an effective shield for small-scale MAS, due to its poor scalability, it is difficult to use it for large-scale MAS. To address the scalability problem, distributed CBFs have been developed [23], [26], [46], [47], [48], [49]. In contrast to centralized CBF where the state of the MAS is used, for a distributed CBF, only the local observations and information available from communication with neighbors are used, reducing the problem dimension significantly. However, similar to a centralized CBF, it is difficult to hand-craft distributed CBF for agents with nonlinear dynamics and input constraints. Most of the works on the safety of MAS consider CBF between each pair of agents, but the resulting control set that satisfies all the pair-wise CBF conditions along with input constraints can be empty.

Learning CBFs: One way of navigating the challenge of handcrafting a CBF is to use neural networks (NNs) for learning a CBF [50]. In the past few years, ML-based methods have been used to learn CBFs for complex systems [47], [51], [52], [53], [54]. However, it is challenging for them to balance safety and task performance for multitask problems, and some methods are not scalable to large-scale multiagent problems. The multiagent decentralized CBF (MDCBF) framework in [47] uses an NN-based CBF designed for MAS. However, they do not encode a method of distinguishing between other controlled agents and *uncontrolled* agents such as static and dynamic obstacles. This can lead to either conservative behaviors if all the neighbors are treated as noncooperative obstacles, or collisions if the obstacles are treated as cooperative, controlled agents. Furthermore, the method in [47] does not account for changing graph topology in their approximation, which can lead to an incorrect evaluation of the CBF constraints and consequently, failure.

Multiagent RL: The review paper [55] provides a good overview of the recent developments in MARL with applications in safe control design (see [56], [57], [58]). There is also a lot of work on MARL-based approaches with focuses on motion planning [16], [46], [59], [60], [61], [62]. However, these approaches do not provide safety guarantees due to the reward structure. One major challenge with MARL is designing a reward function for MAS that balances safety and performance. As argued in [63], MARL-based methods are still in the initial phase of development when it comes to safe multiagent motion planning.

GNN-based methods: Utilizing the permutation-invariance property, GNN-based methods have been employed for problems involving MAS [64], [65], [66]. The control admissiblity models (CAM)-based framework in [64] uses a GNN framework for safe control design for MAS. However, it involves sampling control actions from a set defined by CAM and there are no guarantees that such a set is nonempty, leading to feasibility-related issues of the approach. Works such as [67], [68] use GNNs for generalization to unseen environments and are shown to work on teams of up to a hundred agents. However, in the absence of

an attention mechanism, the computational cost grows with the number of agents in the neighborhood, and hence, these methods are not scalable to very large-scale problems (e.g., a team of 1000 agents) due to the computational bottleneck.

The rest of this article is organized as follows. We formulate the MAS control problem in Section II. Then, we present GCBF as a safety certificate for MAS in Section III, and the framework for learning GCBF and a distributed control policy in Section IV. Section V presents the implementation details on the proposed method, while Sections VI and VII present numerical and hardware experimental results, respectively. Finally, Section VIII concludes this article.

II. Problem Formulation

Notations: In the rest of this article, $\mathbb{R}$ denotes the set of real numbers and $\mathbb{R}_+$ denotes the set of nonnegative real numbers. We use $\|\cdot\|$ to denote the Euclidean norm. A continuous function $\alpha : \mathbb{R} \to \mathbb{R}$ is an extended class-$\mathcal{K}$ function if it is strictly increasing and $\alpha(0) = 0$. We use $[\cdot]_+$ to denote the function $\max(0, \cdot)$. We drop the arguments t, x whenever clear from the context. Unless otherwise specified, given a set of vectors $\{x_i\}$ with $x_i \in \mathcal{X}$ for each $i \in 1, 2, \dots N$ and an index set $\mathcal{I}$, we define $\bar{x}_\mathcal{I} \in \mathcal{X}^{|\mathcal{I}|}$ as the concatenated vector of the vectors x_i with index $i \in \mathcal{I}$ from the index set.

We consider designing a distributed control framework to drive N agents, each denoted with an index from the set $V_a := \{1, 2, \dots, N\}$, to their goal locations in an environment with obstacles while avoiding collisions. The motion of each agent is governed by general nonlinear control affine dynamics

$$\dot{x}_i = f_i(x_i) + g_i(x_i)u_i \tag{1}$$

where $x_i \in \mathcal{X}_i \subset \mathbb{R}^n$ and $u_i \in \mathcal{U}_i \subset \mathbb{R}^m$ are the state, control input for the ith agent, respectively and $f_i : \mathbb{R}^n \to \mathbb{R}^n, g_i : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be locally Lipschitz continuous. For simplicity, we restrict our discussion to the case when all agents have the same underlying dynamics, i.e., where $\mathcal{X}_i = \mathcal{X}$, $\mathcal{U}_i = \mathcal{U}$ and $f_i = f$, $g_i = g$ for all $i \in V_a$. Note that it is also possible to apply our approach to heterogeneous MAS. For convenience, we also define the motion of the entire MAS via the concatenated state vector $\bar{x} := [x_1; x_2; \dots; x_N] \in \mathcal{X}^N$ and $\bar{u} := [u_1; u_2; \dots; u_N] \in \mathcal{U}^N$, such that (1) can equivalently be expressed as

$$\dot{\bar{x}} = \bar{f}(\bar{x}) + \bar{g}(\bar{x})\bar{u} \tag{2}$$

with $\bar{f}$ and $\bar{g}$ defined accordingly.

Let $\mathbb{P} \subset \mathbb{R}^{\mathfrak{n}}$ denote the set of positions in an $\mathfrak{n}$-dimensional environment (i.e., $\mathfrak{n} = 2$ or $\mathfrak{n} = 3$). We assume that each state $x \in \mathcal{X}$ is associated with a position $p \in \mathbb{P}$, and denote by $p_i \in \mathbb{P}$ the first $\mathfrak{n}$ elements of x_i corresponding to the positions of each agent i. For each agent $i \in V_a$, we consider a goal position $p_i^g \in \mathbb{P}$, and define $\bar{p}^g$ as the concatenated goal vector. The observation data consists of $n_{\text{rays}} > 0$ evenly-spaced LiDAR rays originating from each robot and measures the relative location of obstacles within a sensing radius $R > 0$. We assume that R is large enough such that there exists a feasible control input that can keep the agents safe once an obstacle is observed. For mathematical convenience, we denote the jth ray from agent i by $y_j^{(i)} \in \mathcal{X}$, where the first $\mathfrak{n}$ elements of $y_j^{(i)}$ constitute the position of the hitting target $p_j^{(i)} \in \mathbb{P}$ and the last $n - \mathfrak{n}$ elements are zero padding. We then denote the aggregated rays as $\bar{y}_i := [y_1^{(i)}; \dots; y_{n_{\text{rays}}}^{(i)}] \in \mathcal{X}^{n_{\text{rays}}}$. The interagent collision avoidance requirement imposes that each pair of agents maintain a safety distance of $2r$ while the obstacle avoidance requirement dictates that $|y_j^{(i)}| > r$ for all $j = 1, 2, \dots, n_{\text{rays}}$, where $r > 0$ is the radius of a circle that can contain the entire physical body of each agent. The control objective for each agent i is to navigate the obstacle-filled environment to reach its goal p_i^g, as described as follows.

Problem 1: Design a distributed control policy π_i such that, for a set of N agents $\bar{x}$ and noncolliding goal locations $\bar{p}^g$, the following holds for the closed-loop trajectories for each agent $i \in V_a$.

1) Safety (Obstacles): $\|y_j^{(i)}(t)\| > r,\ \forall j = 1, \dots, n_{\text{rays}}, t \geq 0$, i.e., the agents do not collide with the obstacles.
2) Safety (Other Agents): $\|p_i(t) - p_j(t)\| > 2r$ for all $t \geq 0$, $j \neq i$, i.e., the agents do not collide with each other.
3) Liveness: $\inf_{t\geq 0} \|p_i(t) - p_i^g\| = 0$, i.e., each agent eventually reaches its goal location p_i^g.

To solve Problem 1, we consider the existence of a nominal controller that satisfies the liveness property but not necessarily the safety property, and construct a GCBF-based distributed control policy to additionally satisfy the safety property.

III. GCBF: A Safety Certificate for MAS

Based on the algorithm in [26] (GCBFv0), we propose an improved algorithm, termed GCBF+, to train a GCBF that encodes the collision-avoidance constraints based on the graph structure of MAS. We use GNNs to learn a candidate GCBF jointly with the collision-avoidance control policy. Our GNN architecture is capable of handling a variable number of neighbors and, hence, results in a distributed and scalable solution to the safe MAS control problem.

A. Safety for Arbitrary Sized MAS Via Graphs

We first review the notion of CBF commonly used in the literature for safety requirements [17]. Consider a system $\dot{x} = F(x, u)$ where $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, and $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Let $\mathcal{C} \subset \mathcal{X}$ be the 0-superlevel set of a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$, i.e., $\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}$. Then, h is a CBF if there exists an extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that

$$\sup_{u\in\mathcal{U}} \left[\frac{\partial h}{\partial x} F(x, u) + \alpha\left(h(x)\right)\right] \geq 0 \quad \forall x \in \mathcal{X}. \tag{3}$$

Let $\mathcal{S} \subset \mathcal{X}$ denote a safe set with the objective that the system trajectories do not leave this set. If $\mathcal{C} \subset \mathcal{S}$, then the existence of a CBF implies the existence of a control input u that keeps the system safe [22].

Based on the notion of CBF, we define the new notion of a GCBF to encode safety for MAS of any size. To do so, we first define the graph structure we will use in this work.

A directed graph is an ordered pair $G = (V, E)$, where V is the set of nodes, and $E \subset \{(i, j) \mid i \in V_a,\ j \in V\}$ is the set of edges representing the flow of information from a *sender* node (henceforth called a neighbor) j to a *receiver* agent i. Let $\tilde{\mathcal{N}}_i$ denote the set of neighbors for agent $i \in V_a$. For the considered MAS, we define the set of nodes $V = V_a \cup V_o$ to consist of the agents V_a and the hitting points of all the LiDAR rays from all agents denoted as V_o. The edges are defined between each observed point and the observing agent when the distance between them is within a sensing radius $R > 2r > 0$.

Given $\mathfrak{n}$, sensing radius R, safety radius r and n_{rays}, define $M - 1 \in \mathbb{N}$ as the maximum number of sender neighbors that each receiver agent node can have while all the agents in the neighborhood remain safe. For simplicity, define $\mathcal{N}_i \subseteq \tilde{\mathcal{N}}_i$ as the set of M closest neighboring nodes to agent i which also includes agent i[1]. Next, define $\bar{x}_{\mathcal{N}_i} \in \mathcal{X}^M$ as the concatenated vector of x_i and the neighbor node states with fixed size M that is padded with a constant vector if $|\mathcal{N}_i| < M$.

Remark 1: We define M as above so that, for any $i \in V_a$, changes in the neighboring indices $\mathcal{N}_i$ can only occur without collision at a distance R (see Appendix A).

B. Graph Control Barrier Functions

We define the safe set $\mathcal{S}_N \subset \mathcal{X}^N$ of an N-agent MAS as the set of MAS states $\bar{x}$ that satisfy the safety properties in Problem 1, i.e.,

$$\mathcal{S}_N := \left\{ \bar{x} \in \mathcal{X}^N \;\middle|\; \left(\left\| y_j^{(i)} \right\| > r \quad \forall i \in V_a \quad \forall j \in n_{\text{rays}} \right) \bigwedge \left(\min_{i,j \in V_a, i \neq j} \|p_i - p_j\| > 2r \right) \right\}. \tag{4}$$

Then, the unsafe, or avoid, set of the MAS $\mathcal{A}_N = \mathcal{X}^N \setminus \mathcal{S}_N$ is defined as the complement of $\mathcal{S}_N$.

We now introduce the notion of GCBF for encoding safety for MAS. We impose that for a given agent $i \in V_a$, a node j where $\|p_i - p_j\| \geq R$ does not affect the GCBF h so that the resulting h is smooth. Specifically, for any neighborhood set $\mathcal{N}_i$, let $\mathcal{N}_i^{<R}$ denote the set of neighbors in $\mathcal{N}_i$ that are strictly inside the sensing radius R as

$$\mathcal{N}_i^{<R} := \{j : \|p_i - p_j\| < R,\ j \in \mathcal{N}_i\}. \tag{5}$$

Now, we are ready to define GCBF formally.

Definition 1 (GCBF): A continuously differentiable function $h : \mathcal{X}^M \to \mathbb{R}$ is termed as a GCBF if there exists an extended class-$\mathcal{K}$ function α and a control policy $\pi_i : \mathcal{X}^M \to \mathcal{U}$ for each agent $i \in V_a$ of the MAS such that, for all $\bar{x} \in \mathcal{X}^N$ with $N \geq M$

$$\dot{h}(\bar{x}_{\mathcal{N}_i}) + \alpha(h(\bar{x}_{\mathcal{N}_i})) \geq 0 \quad \forall i \in V_a \tag{6}$$

where

$$\dot{h}(\bar{x}_{\mathcal{N}_i}) = \sum_{j \in \mathcal{N}_i} \frac{\partial h(\bar{x}_{\mathcal{N}_i})}{\partial x_j} \left(f(x_j) + g(x_j) u_j \right) \tag{7}$$

with $u_j = \pi_j(\bar{x}_{\mathcal{N}_j})$, and the following two conditions hold.

[1]For breaking ties, the agent with the smaller index is chosen.

1) The gradient of h with respect to nodes R away is 0, i.e.,

$$\frac{\partial h}{\partial x_j}(\bar{x}_{\mathcal{N}_i}) = 0 \quad \forall j \in \mathcal{N}_i \setminus \mathcal{N}_i^{<R}. \tag{8}$$

2) The value of h does not change when restricting to neighbors that are in $\mathcal{N}_i^{<R}$, i.e.,

$$h(\bar{x}_{\mathcal{N}_i}) = h(\bar{x}_{\mathcal{N}_i^{<R}}). \tag{9}$$

Lemma 1: Given a GCBF h, the function $t \mapsto h(\bar{x}_{\mathcal{N}_i(t)}(t))$ is a continuously differentiable function despite $\bar{x}_{\mathcal{N}_i(t)}(t)$ having discontinuities whenever the set of neighboring indices $\mathcal{N}_i(t)$ for agent i changes.

The proof of Lemma 1 is provided in Appendix B.

Remark 2: One way of satisfying the conditions 1) and 2) in Definition 1 is by taking h to be of the form

$$h(\bar{x}_{\mathcal{N}_i}) = \xi_1 \left(\sum_{j \in \mathcal{N}_i} w(x_i, x_j)\, \xi_2(x_i, x_j) \right) \tag{10}$$

where $\xi_1 : \mathbb{R}^\rho \to \mathbb{R}$ and $\xi_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^\rho$ are two encoding functions with ρ the dimension of the feature space, and $w : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a continuously differentiable function such that $w(x_i, x_j) = 0$ and $\frac{\partial w}{\partial x_j}(x_i, x_j) = 0$ whenever $\|p_i - p_j\| \geq R$. In practice, we use graph attention [69], which takes the form (10), to realize conditions 1) and 2) in Definition 1, which we introduce later in Section IV-A.

With Definition 1, a GCBF certifies the forward invariance of its 0-superlevel set under a suitable choice of control inputs. For a GCBF h, let $\mathcal{B}_h \subset \mathcal{X}^M$ denote the 0-superlevel set of h

$$\mathcal{B}_h := \{\tilde{x} \in \mathcal{X}^M \mid h(\tilde{x}) \geq 0\} \tag{11}$$

and define $\mathcal{C}_N \subset \mathcal{X}^N$ as the set of N-agent MAS states where $\bar{x}_{\mathcal{N}_i}$ lie inside $\mathcal{B}_h$ for all $i \in V_a$, i.e.,

$$\mathcal{C}_N := \bigcap_{i=1}^{N} \mathcal{C}_{N,i} \tag{12}$$

where

$$\mathcal{C}_{N,i} := \{\bar{x} \in \mathcal{X}^N \mid \bar{x}_{\mathcal{N}_i} \in \mathcal{B}_h\}. \tag{13}$$

We now state the result of the safety guarantees of GCBF.

Theorem 1: Suppose h is a GCBF following Definition 1 and $\mathcal{C}_N \subset \mathcal{S}_N$ for each $N \in \mathbb{N}$. Then, for any $N \in \mathbb{N}$, the resulting closed-loop trajectories of the MAS with initial conditions $\bar{x}(0) \in \mathcal{C}_N$ under any locally Lipschitz continuous control input $\bar{u} : \mathcal{X}^N \to \mathcal{U}_{\text{safe}}^N$ satisfy $\bar{x}(t) \in \mathcal{S}_N$ for all $t \geq 0$, where

$$\mathcal{U}_{\text{safe}}^N := \left\{ \bar{u} \in \mathcal{U}^N \;\middle|\; \dot{h}(\bar{x}_{\mathcal{N}_i}) + \alpha(h(\bar{x}_{\mathcal{N}_i})) \geq 0 \quad \forall i \in V_a \right\} \tag{14}$$

with the time derivative of h given as in (7).

As a result of Theorem 1, the set $\mathcal{C}_N$, for any $N \in \mathbb{N}$, is a safe control invariant set [70]. An example of a GCBF is shown in Fig. 2.

Unlike traditional methods of proving forward invariance using CBFs [22], the proof of Theorem 1 is more involved as it must handle the dynamics discontinuities that occur when the neighborhood $\mathcal{N}_i$ of agent $i \in V_a$ changes. We make use of

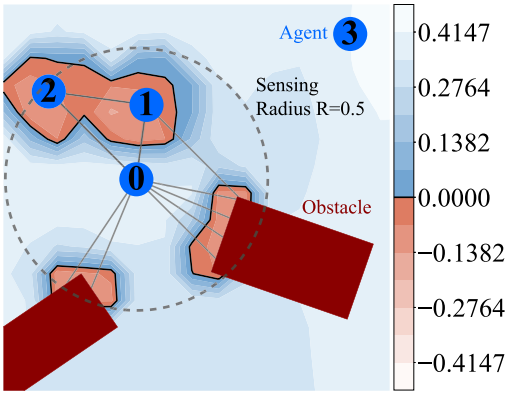


Fig. 2. *GCBF contours:* For a fixed neighborhood $\mathcal{N}_0$ of agent 0, we plot the contours of the learned GCBF h_θ, projected on xy-plane, by varying the position p_0 of agent 0. Since agent 3 is outside of agent 0's sensing radius, i.e., not a neighbor of agent 0, it does not contribute to the value of $h(\bar{x}_{\mathcal{N}_0})$. .

Lemma 1 to handle such discrete jumps. The proof of Theorem 1 is provided in Appendix C.

Remark 3: Note that Theorem 1 proves that a GCBF can certify the safety of an MAS of *any* size $N \in \mathbb{N}$. This is in contrast to the result in [26], which only proves that a GCBF can certify safety for a specific N. As a result, the GCBF from [26], or other notions of CBFs in the prior work, trained on, say, 8 agents, cannot theoretically certify safety when used for an MAS of 1000 agents. However, the proposed framework allows such theoretical certification to carry over from a smaller-sized MAS to a larger-sized MAS. This brings the theoretical understanding of the proposed algorithm closer to the empirical results, where we observe the new GCBF+ algorithm can scale to over 1000 agents despite being trained with only 8 agents.

Remark 4: Note that the individual $\mathcal{C}_{N,i}$ do not all need to be a subset of $\mathcal{S}_N$ as long the intersection $\mathcal{C}_N \subset \mathcal{S}_N$ in Theorem 1. For example, if $\mathcal{S}_N$ can be written as the intersection of sets $\mathcal{S}_{N,i}$, i.e., $\mathcal{S}_N = \bigcap_{i=1}^N \mathcal{S}_{N,i}$, then it is sufficient that $\mathcal{C}_{N,i} \subseteq \mathcal{S}_{N,i}$ for all $i \in V_a$ to obtain that $\mathcal{C}_N \subseteq \mathcal{S}_N$, since

$$\mathcal{C}_N := \bigcap_{i=1}^N \mathcal{C}_{N,i} \subseteq \bigcap_{i=1}^N \mathcal{S}_{N,i} =: \mathcal{S}_N. \tag{15}$$

In practice, we take this approach and define $\mathcal{S}_{N,i}$ for each agent $i \in V_a$ as

$$\mathcal{S}_{N,i} := \left\{ \bar{x} \in \mathcal{X}^N \;\middle|\; \left(\left\| y_j^{(i)} \right\| > r \quad \forall j \in n_{\text{rays}} \right) \bigwedge \left(\min_{j \in V_a, i \neq j} \|p_i - p_j\| > 2r \right) \right\}. \tag{16}$$

C. Safe Control Policy Synthesis

For the multiobjective Problem 1, in the prior work GCBFv0 [26], we used a hierarchical approach for the goal-reaching and the safety objectives, where a nominal controller was used for the liveness requirement. During training, a term is added to the loss function so that the learned controller is as close to the nominal controller as possible. In execution, GCBFv0 uses a switching mechanism to switch between the nominal controller for goal reaching and the learned controller for collision avoidance. However, the added loss term corresponding to the nominal controller competes with the CBF loss terms for safety, often sacrificing either safety or goal-reaching.

In this work, we use a different mechanism for encoding the liveness property. Given a nominal controller $u_i^{\text{nom}} = \pi_{\text{nom}}(x_i, p_i^g)$ for the goal-reaching objective, we design a controller that satisfies the safety constraint using an optimization framework that minimally deviates from a nominal controller that only satisfies the liveness requirements[2]. Given a GCBF h and an extended class-$\mathcal{K}$ function α, a solution to the following centralized optimization problem:

$$\min_{\bar{u}} \quad \sum_{i \in V_a} \|u_i - u_i^{\text{nom}}\|^2 \tag{17a}$$

$$\text{s.t.} \quad u_i \in \mathcal{U} \quad \forall i \in V_a \tag{17b}$$

$$\sum_{j \in \mathcal{N}_i} \frac{\partial h}{\partial x_j} \left(f(x_j) + g(x_j) u_j \right) \geq -\alpha(h(\bar{x}_{\mathcal{N}_i})) \quad \forall i \in V_a \tag{17c}$$

keeps all agents within the safety region [22]. Note that (17c) is linear in the decision variables $\bar{u}$. When the input constraint set $\mathcal{U}$ is a convex polytope, (17) is a quadratic program (QP) and can be solved efficiently online for robotics applications [17]. We define the policy $\pi_{\text{QP}} : \mathcal{X}^N \to \mathcal{U}^N$ as the solution of the QP (17) at the MAS state $\bar{x}$. Note that (17) is not a distributed framework for computing the control policy, since u_i is indirectly coupled to the controls of all other agents via the constraint (17c). Although there is work on using distributed QP solvers to solve (17) (see, e.g., [71]), these approaches are not easy to use in practice for real-time control synthesis of large-scale MAS. To this end, we use an NN-based control policy that does not require solving the centralized QP online. We present the training setup for jointly learning both GCBF and a distributed safe control policy in the following section.

IV. GCBF+: Framework for Learning GCBF and Distributed Control Policy

A. Neural GCBF and Distributed Control Policy

Drawing on the graph representation of arbitrary sized MAS introduced in Section III-A, we apply GNNs to learn a GCBF h_θ and distributed control policy π_ϕ for parameters θ, ϕ. We transform the MAS graph into *input features* to be used as the GNN input by constructing *node features* and *edge features* corresponding to the nodes and edges of the graph G. To learn a goal-conditioned control policy that can reach different goal positions, we introduce a goal node and an edge between each agent and their goal in the input features.

Node features and edge features: The node features $v_i \in \mathbb{R}^{\rho_v}$ encode information specific to each node. In this work, we take $\rho_v = 3$ and use the node features v_i to one-hot encode the type

[2]In this work, we use simple controllers, such as linear quadratic regulator (LQR) and PID-based nominal controllers in our experiments.

of the node as either an agent node, goal node or LiDAR ray hitting point node. The edge features $e_{ij} \in \mathbb{R}^{\rho_e}$, where $\rho_e > 0$ is the edge dimension, are defined as the information shared from node j to agent i, which depends on the states of the nodes i and j. Since the safety objective depends on the relative positions, one component of the edge features is the relative position $p_{ij} = p_j - p_i$. The rest of the edge features can be chosen depending on the underlying system dynamics, e.g., relative velocities for double integrator dynamics.

GNN structure: Thanks to the ability of GNN to take variable-sized inputs, we do not need to add padding nor truncate the input of the GCBF into a fixed-sized vector when $|\tilde{\mathcal{N}}_i| \neq M$. We define the input of h_θ to be input features $z_i = [z_{i1}^\top, \dots, z_{i|\tilde{\mathcal{N}}_i|}^\top]^\top$, where $z_{ij} = [v_i, v_j, e_{ij}]^\top$. In GNN, we first encode each z_{ij} to the feature space via an MLP ψ_{θ_1} to obtain $q_{ij} = \psi_{\theta_1}(z_{ij})$. Next, we use graph attention [69] to aggregate the features of the neighbors of each node, i.e.,

$$q_i = \sum_{j \in \tilde{\mathcal{N}}_i} \underbrace{\operatorname{softmax}\left(\psi_{\theta_2}(q_{ij})\right)}_{w_{ij}} \psi_{\theta_3}(q_{ij}) \tag{18}$$

where ψ_{θ_2} and ψ_{θ_3} are two NNs parameterized by θ_2 and θ_3. ψ_{θ_2} is often called "gate" NN in the literature [72], and the resulting *attention weights* $w_{ij} \in [0, 1]$ (with $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$) encode how important the sender node j is to agent i. Note that applying attention in GNNs is crucial for satisfying conditions 1) and 2) in Definition 1, which are satisfied in our experiments since the attention weights and their derivatives go to 0 as the interagent distance goes to R without any additional supervision (see Remark 2 and Fig. 4). The aggregated information q_i in (18) is then passed through another MLP ψ_{θ_4} to obtain the output value $h_i = \psi_{\theta_4}(q_i)$ of the GCBF for each agent. We use the same GNN structure for the control policy π_ϕ. Since the input features z_i for agent i only depend on the neighbors $\tilde{\mathcal{N}}_i$, the π_ϕ is distributed unlike (17).

B. GCBF+ Loss Functions

We train the GCBF h_θ and the distributed controller π_ϕ by minimizing the sum of the CBF loss $\mathcal{L}_{\text{CBF}}(\theta, \phi)$ and the control loss $\mathcal{L}_{\text{ctrl}}(\phi)$

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{CBF}}(\theta, \phi) + \mathcal{L}_{\text{ctrl}}(\phi). \tag{19}$$

The CBF loss $\mathcal{L}_{\text{CBF}}(\theta, \phi)$ and the control loss $\mathcal{L}_{\text{ctrl}}(\phi)$ are defined as sums over each agent as

$$\mathcal{L}_{\text{CBF}}(\theta, \phi) := \sum_{i \in V_a} \mathcal{L}_{\text{CBF},i}(\theta, \phi) \tag{20a}$$

$$\mathcal{L}_{\text{ctrl}}(\phi) := \sum_{i \in V_a} \mathcal{L}_{\text{ctrl},i}(\phi). \tag{20b}$$

Denote by $\mathcal{D}_{\mathcal{C},i}, \mathcal{D}_{\mathcal{A},i}$ the set consisting of labeled input features in the safe control invariant region and unsafe region, respectively. The CBF loss $\mathcal{L}_{\text{CBF},i}$ penalizes violations of the GCBF condition (6) and the (sufficient) safety requirement that the 0-superlevel set $\mathcal{C}_{N,i}$ is a subset of $S_{N,i}$ (see Remark 4)

$$\mathcal{L}_{\text{CBF},i}(\theta, \phi) := \eta_{\text{deriv}} \sum_{z_i} \Big[\gamma - \dot{h}_\theta(z_i) - \alpha(h_\theta(z_i))\Big]^+ + \sum_{z_i \in \mathcal{D}_{\mathcal{C},i}} [\gamma - h_\theta(z_i)]^+ + \sum_{z_i \in \mathcal{D}_{\mathcal{A},i}} [\gamma + h_\theta(z_i)]^+ \tag{21}$$

where $\gamma > 0$ is a hyperparameter to encourage strict inequalities and $\eta_{\text{deriv}} > 0$ weighs the GCBF condition (6). We use a class-$\mathcal{K}$ function $\alpha(h) = \bar{\alpha} h$ for a constant $\bar{\alpha} > 0$. From hereafter, we abuse the notation and use α to refer to $\bar{\alpha}$. The control loss $\mathcal{L}_{\text{ctrl},i}$ encourages the learned controller π_ϕ to remain close to the QP controller π_{QP} [the solution to the QP (17) with h being the learned h_θ in the previous learning step], which is the closest control to π_{nom} that maintains safety

$$\mathcal{L}_{\text{ctrl},i}(\phi) := \eta_{\text{ctrl}} \left\| \pi_\phi(z_i) - \pi_{\text{QP},i}(\bar{x}) \right\| \tag{22}$$

where η_{ctrl} is the control loss weight and $\pi_{\text{QP},i}(\bar{x}) \in \mathcal{U}$ is the ith component of $\pi_{\text{QP}}(\bar{x})$. In practice, it is possible that the QP is infeasible during training. To this end, a relaxation term $\zeta \geq 0$ is added to the left-hand side of constraint (17c) along with a penalty term for ζ with a large coefficient added to the objective function (17a). Once the CBF loss (20a) converges to zero, the QP is feasible on the sampled data points.

One of the challenges of evaluating the loss function $\mathcal{L}$ is computing $\dot{h}_\theta$. Similar to [64], we estimate $\dot{h}_\theta$ by $(h_\theta(z_i(t_{k+1})) - h_\theta(z_i(t_k)))/\delta t$, where $\delta t = t_{k+1} - t_k$ is the timestep. Estimating $\dot{h}_\theta$ may be problematic if the set of neighbors $\tilde{\mathcal{N}}_i$ changes between t_k and t_{k+1}. However, the learned attention weights satisfy conditions 1) and 2) in Definition 1 as noted in Section IV-A. Consequently, Lemma 1 implies that h_θ is continuously differentiable, and our estimate of $\dot{h}_\theta$ is well behaved. Note that $\dot{h}_\theta$ includes the inputs from agent i and the neighbor agents $j \in \mathcal{N}_i$. Therefore, when we use gradient descent and backpropagate $\mathcal{L}_i(\theta, \phi)$ during training, the gradients are passed to not only the controller of agent i but also the controllers of all neighbors in $\tilde{\mathcal{N}}_i$[3]. More details on the training process are provided in Section V-A. The training architecture is summarized in Fig. 3.

Remark 5: Benefits of the new loss function: In the prior works [26], [47], the nominal policy π_{nom} is used instead of π_{QP} in $\mathcal{L}_{\text{ctrl}}$. As a result, these approaches suffer from a tradeoff between safety and goal-reaching and often learn a suboptimal policy that compromises safety for liveness, or liveness for safety. In contrast, our proposed method allows the loss to converge to zero, and thus, does not have this tradeoff.

Fig. 5 plots the trajectories of an agent in the presence of an obstacle under learned policies with π_{nom} and π_{QP} in the $\mathcal{L}_{\text{ctrl}}$. When π_{nom} is used, for small values of η_{ctrl}, the learned controller overprioritizes safety leading to conservative behavior (the top left plot). For large η_{ctrl}, the learned policy overprioritizes goal reaching, leading to unsafe behaviors (the top right plot). Using an optimal η_{ctrl}, it is possible to get a desirable behavior as in the top middle plot. In contrast, when π_{QP} is used in the control

[3]We re-emphasize the fact the neighbors' inputs are not required for π_ϕ during testing.

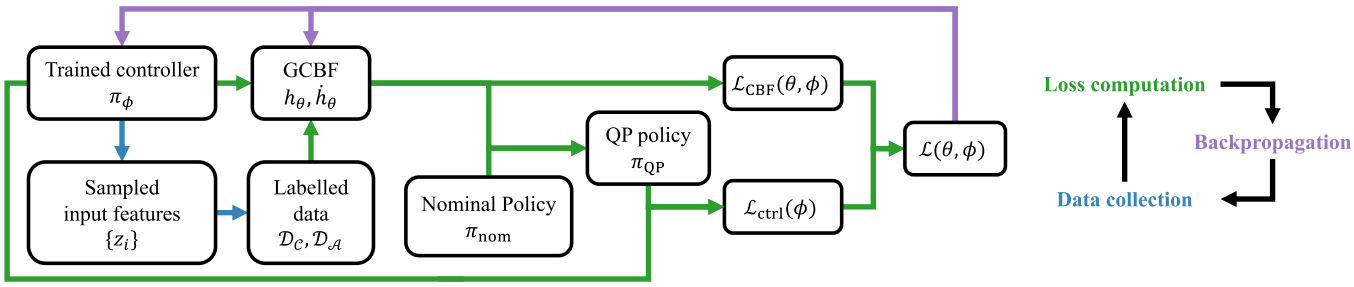


Fig. 3. *GCBF+ training architecture:* The sampled input features are labeled as safe control invariant $\mathcal{D}_\mathcal{C}$ and unsafe $\mathcal{D}_\mathcal{A}$ using the previous step learned control policy π_ϕ. A nominal control policy π_{nom} for goal reaching is used in a CBF-QP with the previously learned GCBF h_θ to generate π_{QP}. Finally, the QP policy and the GCBF conditions are used to define the loss $\mathcal{L}$.

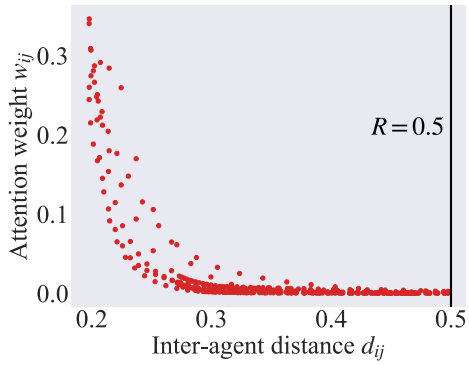


Fig. 4. *Satisfaction of Definition 1 in practice:* The attention weights w_{ij} in (18) are plotted against interagent distances d_{ij} for sensing radius $R = 0.5$ from multiple trajectories. The weight w_{ij} approaches 0 as the interagent distance d_{ij} approaches R without explicit supervision, showing that GCBF+ automatically learns to satisfy conditions 1) and 2) in Definition 1 via the approach outlined in Remark 2.

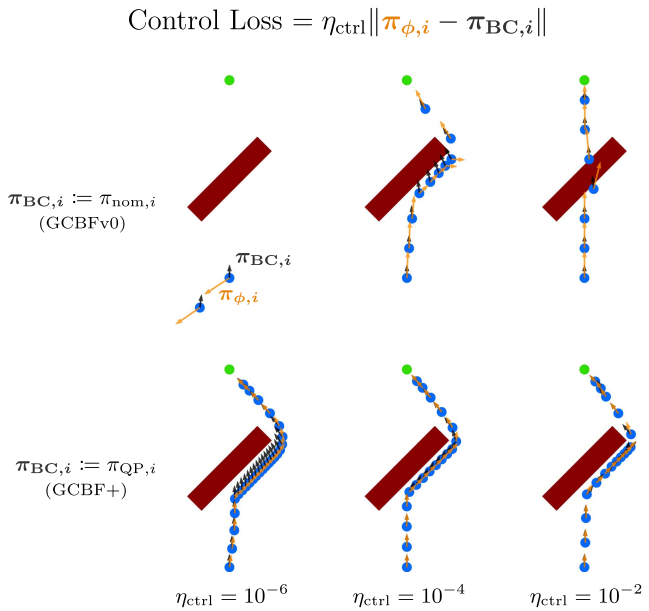


Fig. 5. *Comparison of the choice of control loss:* The learned policy $\boldsymbol{\pi_\phi}$ is sensitive to the choice of η_{ctrl} when using π_{nom} as in GCBFv0 [26] (top) since a learned policy $\boldsymbol{\pi_\phi}$ that is close to π_{nom} may be unsafe and not satisfy the GCBF conditions (6). Consequently, choosing η_{ctrl}, which controls the relative weight between $\mathcal{L}_{\text{CBF}}$ and $\mathcal{L}_{\text{ctrl}}$, becomes a balancing act of staying close to π_{nom} while remaining safe. In contrast, by definition of π_{QP} (17), the control $\boldsymbol{\pi_\phi}$ is already safe. Hence, the safety of $\boldsymbol{\pi_\phi}$ is not sensitive to η_{ctrl} when using π_{QP} for the control loss, as in GCBF+.

loss, the goal-reaching and the safety losses do not compete with each other and it is possible to get a desirable behavior without extensive hyperparameter tuning as can be seen in the bottom plots in Fig. 5. Note that when π_{nom} is used, even with the optimal value of η_{ctrl}, the learned input (shown with black arrows) does not align with the nominal input (shown with orange arrows), meaning that the total loss may not converge to zero in such formulations unless the nominal policy is also safe. On the other hand, when π_{QP} is used, the two control inputs have much more similar values, which allows for the total loss to converge to 0.

C. Data Collection and Labeling

The training data $\{z_i\}$ is collected over multiple scenarios and the loss is calculated by evaluating the CBF conditions on each sample point. We use an on-policy strategy to collect data by periodically executing the learned controller π_ϕ, which helps align the train and test distributions.

When labeling the training data as $\mathcal{D}_{\mathcal{C},i}$ or $\mathcal{D}_{\mathcal{A},i}$, it is important to note that an input feature z_i that is not in a collision may be unable to prevent an inevitable collision in the future under actuation limits. For example, under acceleration limits, an agent that is moving too fast may not have enough time to stop, resulting in an inevitable collision. Therefore, we cannot naïvely label all the input features as $\mathcal{D}_{\mathcal{C},i}$ if they are not in any collision at the current time step unless there exists a control policy that can keep them safe in the future. However, as noted in [53], [73], [74], and [75], computing an infinite horizon control invariant set for high-dimensional nonlinear systems is computationally challenging, and often, various approximations are used for such computations. In this work, we use a finite-time reachable set as an approximation. At any given learning step, given a graph G, and the corresponding input features $z_i, i \in V_a$, we use the learned policy from the previous iteration to propagate the system trajectories for T timesteps. If the entire trajectory remains in the safe set $S_{N,i}$ for agent i, then z_i is added to the set $\mathcal{D}_{\mathcal{C},i}$. If there exist collisions in z_i, it is added to the set $\mathcal{D}_{\mathcal{A},i}$. Otherwise, it is left unlabeled. As $T \to \infty$, this recovers

the infinite horizon control invariant set, but is not tractable to compute. Instead, we choose a large but finite value of T as a hyperparameter in our numerical experiments.

The training data can be collected and labeled efficiently as follows. With the learned policy from the previous iteration, we sample feature trajectories $\{z_i^0, \dots, z_i^{\bar{T}}\}$ for some $\bar{T} \geq T$. If any feature z_i^t is unsafe, it is added to $\mathcal{D}_{\mathcal{A},i}$. If T features $z_i^{t+1}, \dots, z_i^{t+T}$ after a feature z_i^t including itself are all safe, then z_i^t is added to $\mathcal{D}_{\mathcal{C},i}$. Otherwise, it remains unlabeled.

Remark 6: Importance of control-invariant labels: Note that GCBFv0 [26] does *not* use the concept of the safe control invariant set during training. Instead, similar to prior works [47], [51], the learned CBF is enforced to be positive on the entire safe set $\mathcal{S}_{N,i}$, even for states that are not control invariant. Prior works attempt to alleviate these issues by estimating the control-invariant using a shrunk safe set and introducing a fixed margin between the safe and unsafe sets. However, for high relative degree dynamics, this margin is state-dependent. Therefore, this estimation of the control-invariant set does not result in a true control-invariant set. As noted in [53], if the safe set $\mathcal{S}_{N,i}$ is not control-invariant, then no valid CBF exists that is positive on $\mathcal{S}_{N,i}$. We later investigate the importance of the quality of the control-invariant labels (as controlled by T) in Section VI-C and find that poor approximations of the control-invariant set $\mathcal{C}_{N,i}$ via small values of T leads to large drops in the safety rate. This provides some insight into the performance improvements of GCBF+ over GCBFv0.

V. Experiments: Implementation Details

In this section, we introduce the details of the experiments, including the implementation details of GCBF+ and the baseline algorithms, and the details of each environment.

Environments: We conduct experiments on five different environments consisting of three 2-D environments (SingleIntegrator, DoubleIntegrator, DubinsCar) and two 3-D environments (LinearDrone, CrazyflieDrone). See Appendix D1 for details. The parameters are $R = 0.5, r = 0.05$ in all environments. The total time steps for each experiment is 4096.

Evaluation criteria: We use safety rate, reaching rate, and success rate as the evaluation criteria for the performance of a chosen algorithm. The *safety rate* is defined as the ratio of agents not colliding with either obstacles or other agents during the experiment period over all agents. As an example, in a scenario consisting of 100 agents, if 15 agents undergo a collision at any time step, then the safety rate will be 85%. The goal *reaching rate*, or simply, the reaching rate, is defined as the ratio of agents that reach their goal location by the end of the experiment period[4]. The *success rate* is defined as the ratio of agents that are safe and reach their goals. For each environment, we evaluate the performance over 32 instances of randomly chosen initial and goal locations for 3 policies trained with different random seeds. We report the mean rates and their standard deviations for the 32 instances for each of the 3 policies (i.e., average performance over 96 experiments).

[4]Note that we do not consider collision dynamics in our experiments and the agents continue moving after a collision.

TABLE I
Size of Layers of the MLP Used in the GNN

MLP	Hidden layer size	Output layer size
ψ_{θ_1}	256×256	128
ψ_{θ_2}	128×128	1
ψ_{θ_3}	256×256	128
ψ_{θ_4}	256×256	1 for h_θ and m for π_ϕ

TABLE II
Hyperparameters Used in Our Training

Environment	T	η_{ctrl}	lr policy	lr GCBF
SingleIntegrator	1	10^{-4}	10^{-5}	10^{-5}
DoubleIntegrator	32	10^{-4}	10^{-5}	10^{-5}
DubinsCar	32	10^{-5}	3×10^{-5}	3×10^{-5}
LinearDrone	32	10^{-3}	10^{-5}	10^{-5}
CF Drone	32	3×10^{-5}	10^{-5}	10^{-4}

A. Implementation Details

Our learning framework contains two NN models: the GCBF h_θ and the controller π_ϕ. The sizes of the MLP layers in them are shown in Table I. The resulting trained control policy is loaded on each agent locally and requires 1.5 MB of memory. To make the training easier, we define $\pi_\phi = \pi_\phi^{\text{NN}} + \pi_{\text{nom}}$, where π_ϕ^{NN} is the NN controller, so that π_ϕ^{NN} only needs to learn the deviation from π_{nom}.

We use Adam [76] to optimize the NNs for 1000 steps in training. The training time is around 3 h on a 13th Gen Intel(R) Core(TM) i7-13700KF CPU @ 3400 MHz and an NVIDIA RTX 3090 GPU. We choose the hyperparameters following Table II, where "lr cbf" and "lr policy" denote the learning rate for h_θ and π_ϕ, respectively. We set $\alpha = 1.0$, $\gamma = 0.02$, and $\eta_{\text{deriv}} = 0.2$ for all the environments.

B. Baseline Methods

We compare GCBF+ with GCBFv0 [26], InforMARL [27], MPC [28], and centralized and distributed variants of handcrafted CBF-QPs [23]. We use a modified version of the GCBFv0 introduced in [26] where we remove the online policy refinement step since it requires multiple rounds of interagent communications to exchange control inputs during execution and does not work in the presence of actuation limits.

The InforMARL algorithm is a variant of MAPPO [16] that uses a GNN architecture for the actor and critic networks. We use a reward function that consists of three terms. First, we penalize deviations from the nominal controller, i.e.,

$$R_{\text{nom},i} := -\frac{1}{2} \left\| u_i - u_i^{\text{nom}} \right\|^2 . \tag{23}$$

To improve performance, we use a sparse reward term for reaching the goal, i.e.,

$$R_{\text{goal},i} := \begin{cases} 1.0, & \|p_i - p_i^g\| \leq 2r \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

Safety is incorporated by adding the following term to the reward function to penalize collisions, similar to [77], [78]

$$R_{\text{col},i} := \max\left\{\max_{j\in\tilde{N}_i} R_{\text{col,agent},ij}, \max_{j\in n_{\text{rays}}} R_{\text{col,obs},ij}\right\} \tag{25}$$

where

$$R_{\text{col,agent},ij} := \begin{cases} -1 & \|p_i - p_j\| < 2r \\ \frac{\|p_i-p_j\|}{2r} - 2 & 2r \le \|p_i - p_j\| \le 4r \\ 0 & 4r < \|p_i - p_j\| \end{cases} \tag{26}$$

for interagent collisions, and

$$R_{\text{col,obs},ij} := \begin{cases} -1 & \left\|p_i - p_j^{(i)}\right\| < r \\ \frac{\left\|p_i-p_j^{(i)}\right\|}{2r} - 2 & r \le \left\|p_i - p_j^{(i)}\right\| \le 3r \\ 0 & 4r < \left\|p_i - p_j^{(i)}\right\| \end{cases} \tag{27}$$

for agent-obstacle collisions. The final reward function is a sum of the abovementioned terms weighted by λ_{nom}, λ_{goal}, and $\lambda_{\text{col}} > 0$.

$$R_i := \lambda_{\text{nom}} R_{\text{nom},i} + \lambda_{\text{goal}} R_{\text{goal},i} + \lambda_{\text{col}} R_{\text{col},i}. \tag{28}$$

We use a distributed MPC that does not assume interagent communication [28], similar to the other baselines. At each time step, since the control actions of neighbor agents are unknown, they are assumed to follow a constant velocity model, and each agent solves the following H-horizon optimal control problem

$$\min_{u_i} \quad \|u_i - u_i^{\text{nom}}\|^2 \tag{29a}$$

$$\text{s.t.} \quad u_i \in \mathcal{U} \tag{29b}$$

$$x_i^k \in S_{N,i}, \quad k = 0, \ldots, H-1 \tag{29c}$$

where we use a constant velocity prediction of other agents in the definition of the safe set $S_{N,i}$ in (29c). In the SingleIntegrator environment, we estimate the current velocity using the past two position observations, while in other environments, velocity is included in agents' states. The MPC baseline is implemented in CasADi [79] using the SNOPT [80] optimizer.

For the hand-crafted CBF-QPs, we define a pairwise higher order CBF [81] between each pair of agents $(i.j)$[5]

$$h_0 = \sum_{l\in\mathbb{P}} (p_i^l - p_j^l)^2 - (2r)^2 \tag{30}$$

$$h = \dot{h}_0 + \alpha_0 h_0 \tag{31}$$

where $\mathbb{P} = \{x, y\}$ for 2-D environments and $\mathbb{P} = \{x, y, z\}$ for 3-D, $\alpha_0 \in \mathbb{R}_+$ is a constant and r is the radius of the agent. In the LiDAR environments, we also use a pairwise CBF between each agent i and its LiDAR hitting points j, defined as

$$h_0' = \sum_{l\in\mathbb{P}} \left(p_i^l - p_j^{(i),l}\right)^2 - r^2 \tag{32}$$

$$h' = \dot{h}_0' + \alpha_0 h_0' \tag{33}$$

where $p_j^{(i),l}$ is the lth position dimension of LiDAR hitting point $p_j^{(i)}$. We consider two CBF-QP frameworks [23].

1) *Centralized CBF:* In this framework, inputs of all the agents are solved together by setting up a centralized QP containing CBF constraints of all the agents. In this case, the CBF condition is

$$\dot{h} + \alpha h \ge 0, \quad \dot{h}' + \alpha h' \ge 0. \tag{34}$$

2) *Decentralized CBF:* In this framework, each agent computes its control input but the CBF condition is *shared* between the neighbors as in [23]. Let $\dot{x}_{ij} = f_{ij}(x_{ij}) + g_{ij,i}(x_{ij})u_i + g_{ij,j}(x_{ij})u_j$ denote the combined dynamics of the pair (i, j) where $f_{ij}, g_{ij,i}, g_{ij,j}$ can be obtained from combining the agent dynamics. Then, the constraint used in the agent is QP is

$$\frac{\partial h}{\partial x_{ij}} g_{ij,i}(x_{ij})u_i \ge -\frac{1}{2}\left(\alpha h(x_{ij}) + \frac{\partial h}{\partial x_{ij}} f_{ij}(x_{ij})\right) \tag{35}$$

while that in agent js QP is

$$\frac{\partial h}{\partial x_{ij}} g_{ij,j}(x_{ij})u_j \ge -\frac{1}{2}\left(\alpha h(x_{ij}) + \frac{\partial h}{\partial x_{ij}} f_{ij}(x_{ij})\right) \tag{36}$$

so that the sum of constraints (35) and (36) recovers the CBF condition (34). Since the obstacles are not controlled, for the agent-LiDAR pair, the constraint used in agent is QP is

$$\frac{\partial h'}{\partial x_{ij}} g_{ij,j}(x_{ij})u_j \ge -\left(\alpha h'(x_{ij}) + \frac{\partial h'}{\partial x_{ij}} f_{ij}(x_{ij})\right). \tag{37}$$

For both centralized and decentralized approaches, we design two baselines with $\alpha = 1.0$ and $\alpha = 0.1$, respectively. The resulting 4 baselines are named CBF1.0, CBF0.1, DecCBF1.0, and DecCBF0.1, respectively.

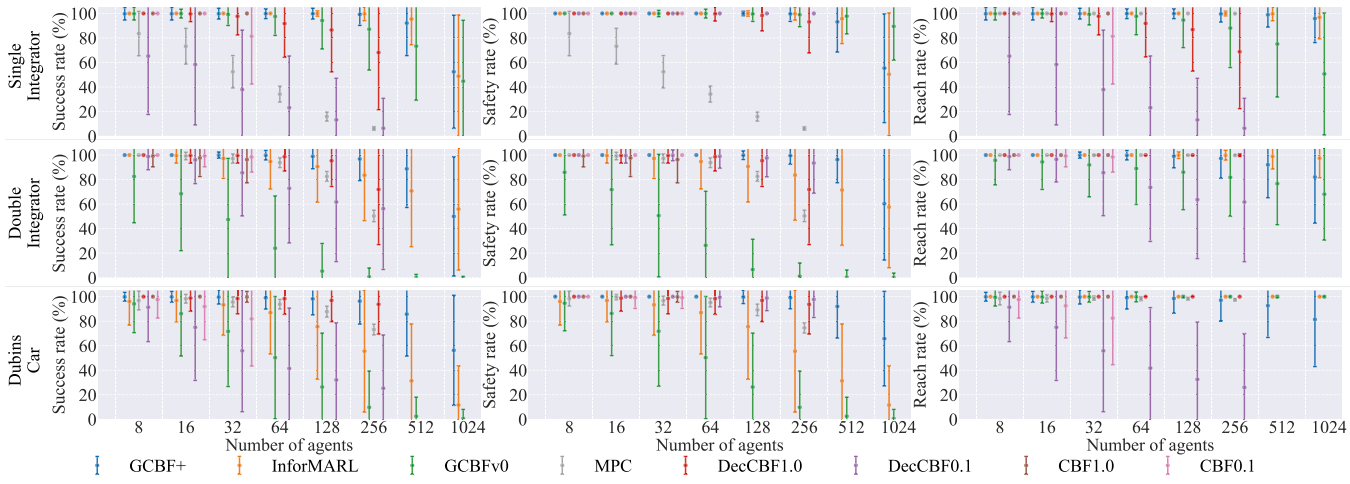
VI. Numerical Experiments: Results

We conduct simulation experiments to examine the scalability, generalizability, and effectiveness of the proposed method. In all experiments, the initial position of the agents and goals are uniformly sampled from the set $\mathbb{P}_0 = [0, l]^{\text{n}}$ for an area width $l > 0$, which we specify for each environment. The *density* of agents can be approximately computed as N/l^{n}. Hence, a smaller value of l results in a higher density of agents and, thus, is more challenging to prevent collisions. We train GCBF+ and GCBFv0 on an environment with 8 agents and 8 obstacles with $l = 4$ for 2-D and $l = 2$ for 3-D environments.

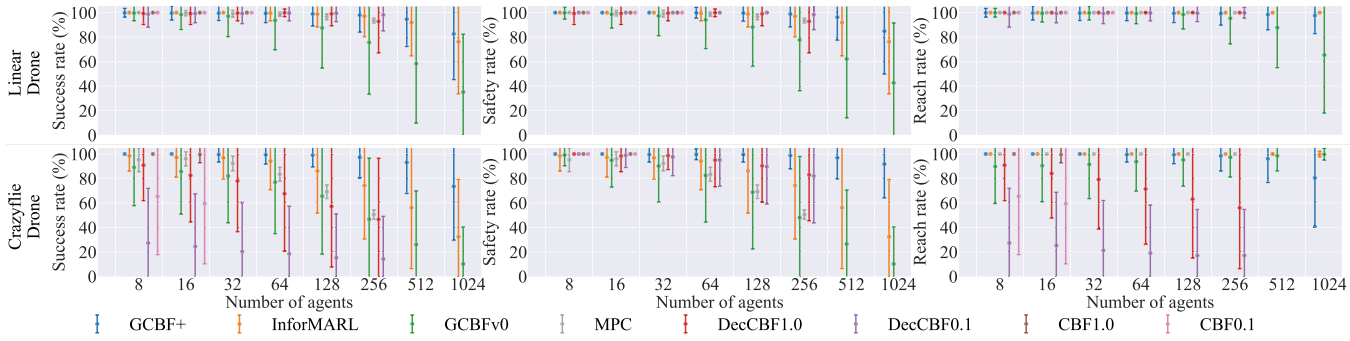
A. Performance Under Increasing Number of Agents

We first perform experiments in an obstacle-free setting where we test the algorithms for a fixed l but increase the number of agents N from 8 to 1024. This tests the ability of each algorithm to maintain safety as the density of agents and goals

[5]Except for the single integrator environment, where we use the same h_0 as in (30) and define $h := h_0$. For the double integrator environment, Wang et al. [23] proposed a CBF that considers the input constraints, which we compare with in Section D2.



(a)



(b)

Fig. 6. Success (left), safety (middle), and reach (right) rates for an increasing number of agents for fixed area width l. (a) 2-D environments. (b) 3-D environments.

in the environment increases by over 100-fold. We use $l = 8$ for SingleIntegrator and DoubleIntegrator environments, $l = 16$ for the DubinsCar environment, and $l = 4$ for both the 3-D environments[6] and show the resulting success rate, safety rate, and reach rate in Fig. 6.

Centralized methods do not scale with increasing number of agents: As expected, the centralized methods (i.e., CBF1.0 and CBF0.1) require increasing amounts of computation time as the number of agents increases. Consequently, we were unable to test CBF1.0 and CBF0.1 for more than 32 agents in all environments due to exceeding computation limits.

Hand-crafted CBFs are overly conservative: The decentralized hand-crafted CBF-QP with $\alpha = 0.1$ has comparable safety performance to GCBF+. However, it is much more conservative than GCBF+ and compromises its goal-reaching ability, as evident from the low reach rates in all environments. For larger α, the decentralized hand-crafted CBF-QP method fails to maintain a high safety rate as the controls become saturated by the control limit. Although the decentralized hand-crafted CBF can be scaled to a large number of agents assuming that each agent can perform computation for its control input locally, in our experiments, we simulate the decentralized controller on one computer node and, hence, are constrained by the memory and computation limits of the computer. Thus, we could not perform experiments for more than 256 agents. However, the downward trend of the reaching rate illustrates that the decentralized hand-crafted CBF-QP method becomes more conservative as the environment gets denser.

MPC has low safety rates because of the noncooperative agents: MPC's success rate drops significantly in all environments as the number of agents increases because the agents do not know the *actual* actions of other agents and only make predictions from observations. This problem is exacerbated in the SingleIntegrator environment where the agent velocities can change instantaneously. This leads to the predicted motions of neighboring agents differing significantly from the actual actions, resulting in low safety rates. On the other hand, for environments with higher relative degrees, the agent velocities do not change instantaneously. Consequently, the constant-velocity model yields relatively more accurate predictions, and hence, improved safety rates. However, the performance of MPC is still considerably poorer than that of GCBF+ in all experiments. This is due to the fact that, while GCBF+ also does not have access

[6]Note that we use more challenging (i.e., smaller) values of l for the 2-D (8, 16 versus 32) and 3-D (4 versus 16) as compared to our previous work [26].

to the control inputs of other agents during *online execution*, however, during *offline centralized training*, GCBF+ policy has access to the control inputs of other agents, which also come from the same GCBF policy being used by the different agents. As a result, the GCBF+ policy can ensure that the GCBF conditions are satisfied. Then, during online execution, the GCBF condition is satisfied if all other agents follow the same GCBF policy that was used during offline training *without* the need for other agents to communicate their control inputs. While distributed MPC can be scaled to more than 256 agents, the computation is carried out on a single computer in our experiments, and the computational requirement of MPC exceeds the capability of one computer. Therefore, we do not test MPC for more than 256 agents. Lastly, since MPC gets even slower with LiDAR data, we do not use MPC as a baseline when conducting experiments with obstacles in Section VI-B.

GCBFv0 struggles with safety for dynamics with high relative degrees: While GCBFv0 performs comparably on the SingleIntegrator environment, the performance deteriorates drastically on all other dynamics. This is because GCBFv0 relies on an accurate hand-crafted safe control invariant set during training, which is difficult to estimate in the presence of control limits for dynamics with high relative degrees. The safe control invariant set is easy to estimate for relative degree 1 environments such as SingleIntegrator, where it can be taken as the complement of the unsafe set. However, for high relative degree dynamics with control limits, the naive estimation method used by GCBFv0 breaks down, causing the safety rate and, thus, success rate to drop significantly. Another potential reason for the poor safety of GCBFv0 is that it uses π_{nom} in the control loss, which forces a tradeoff between safety and goal-reaching (see Remark 5).

GCBF+ performs well on nonlinear dynamics: We observe that all methods have lower success rates in environments that have nonlinear dynamics (DubinsCar, CrazyflieDrone) compared to ones with linear dynamics (SingleIntegrator, DoubleIntegrator, LinearDrone). The performance gap between GCBF+ and other methods is more clear in these challenging environments. On the DubinsCar environment, GCBF+ achieves a 44% higher (compared to InforMARL) and 55% higher (compared to GCBFv0) success rate. On the CrazyflieDrone environment, GCBF+ achieves a 45% higher (compared to InforMARL) and 65% higher (compared to GCBFv0) success rate. Hence, GCBF+ generalizes better than the baseline algorithms, particularly for environments with nonlinear dynamics.

GCBF+ reach rate declines faster than InforMARL and MPC: While the safety rate for GCBF+ is the best among the baselines for denser environments, its reach rate declines as the number of agents increases, while the reach rate for InforMARL and MPC stays consistently near 100% in all environments. The main reason for this decline is that GCBF+ focuses on safety and delegates the liveness (i.e., goal-reaching) requirements to the nominal controller, which is unable to resolve deadlocks. Hence, one potential reason for the lower reach rates of GCBF+ as the density increases is that the learned controller is unable to resolve deadlocks that occur more frequently with increasing density. On the other hand, InforMARL has a sparse reward term for reaching the goal (24), and hence, it is incentivized to learn a controller that can resolve deadlocks at the cost of temporarily deviating from the nominal controller and sacrificing safety, which is evident from the significant drop in the safety rate for InforMARL.

B. Performance Under Increasing Number of Obstacles

In the next set of simulation experiments, we fix the number of agents N and the area width l and vary the number of obstacles present from 0 to 128. For the 2-D DoubleIntegrator environment, we consider $(N = 256, l = 16)$ and $(N = 1024, l = 32)$, where the obstacles are cuboids with side lengths uniformly sampled from $[0.1, 0.5]$ and each agent generates 32 equally spaced LiDAR rays to detect obstacles. For the 3-D LinearDrone environment, we consider $(N = 256, l = 8)$ and $(N = 1024, l = 12)$, where the obstacles are spheres with radius uniformly sampled from $[0.15, 0.3]$ and each agent generates 130 equally spaced LiDAR rays to detect obstacles.

The success rate, safety rate, and reach rate for all cases are shown in Fig. 7. Overall, we observe similar trends as the previous experiment in Section VI-A. In all environments, GCBF+ has the highest success rates compared with the baselines. Trained with just 8 agents and 8 obstacles, GCBF+ can achieve a $> 98\%$ success rate with 256 (and 1024) agents and 128 obstacles. InforMARL performs well but is behind GCBF+. Other baselines have much lower success rates compared with GCBF+ and InforMARL. GCBFv0 does not perform well since it does not account for control limits. The decentralized hand-crafted CBF-QPs perform poorly in the 2-D environment due to their conservatism and in the 3-D environment due to saturation from the control limits.

C. Sensitivity to Hyperparameters

We next perform a sensitivity analysis of our proposed method on the DoubleIntegrator environment to investigate the effect of two hyperparameters: α and T. The α parameter is used to define the CBF derivative condition (21), while T is used to label safe control invariant data and unsafe data (see Section IV-C). We plot the success, reach, and safety rates while varying α from 10^{-2} to 10^{2} in the left plot in Fig. 8. The results showed that using $\alpha = 10^{-2}$ led to a drop in the reach rate, while using $\alpha = 10^{2}$ led to a drop in the safety rate. This behavior can be attributed to the fact that for very small values of α, the CBF condition becomes overly conservative, resulting in poor goal-reaching. For very large values of α, safety can be compromised as the CBF condition allows the system to move towards the unsafe set at a faster rate. This, along with the fact that the control inputs are constrained, may lead to a violation of safety. Note that for $\alpha \in [0.1, 10]$, the performance of GCBF+ does not change much. This implies that GCBF+ is robust to a large range of α.

The right plot in Fig. 8 analyzes the performance of GCBF+ for varying prediction horizons $T \in [4, 64]$ for labeling the data to be safe control invariant or unsafe for training. For a very small horizon $T = 4$, the safety rate drops as the resulting approximation of the safe control invariant set is poor. For a very large horizon $T = 64$, the algorithm becomes too conservative, requiring longer training times to converge. For the chosen fixed

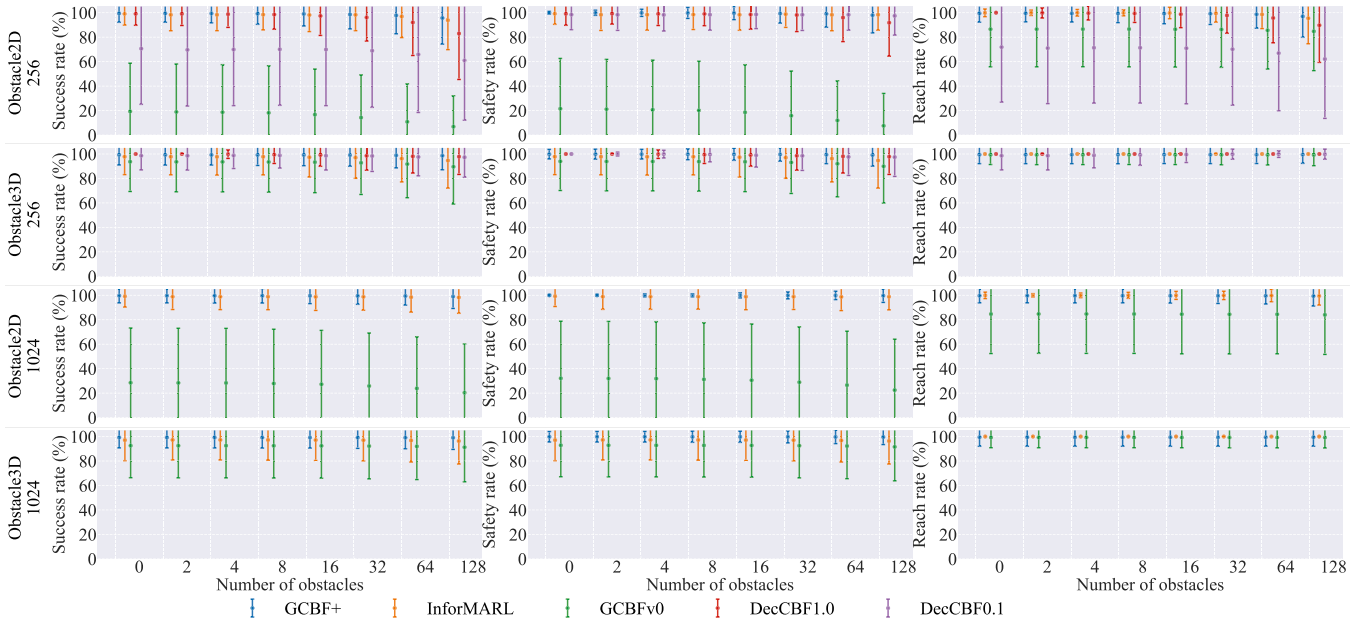


Fig. 7. *Obstacle environment performance:* Success (left), safety (middle), and reach (right) rates for the obstacle environment in one 2-D and one 3-D environment, namely, DoubleIntegrator and LinearDrone environments. In these experiments, the number of agents as well as the area size are kept constant while the number of obstacles are increased. The first set of experiments is conducted on 256 agents with all the baselines. The second set of experiments is conducted with 1024 agents with only GCBF and InforMARL for comparison since we were unable to simulate other baselines for more than 256 agents due to computational limits.

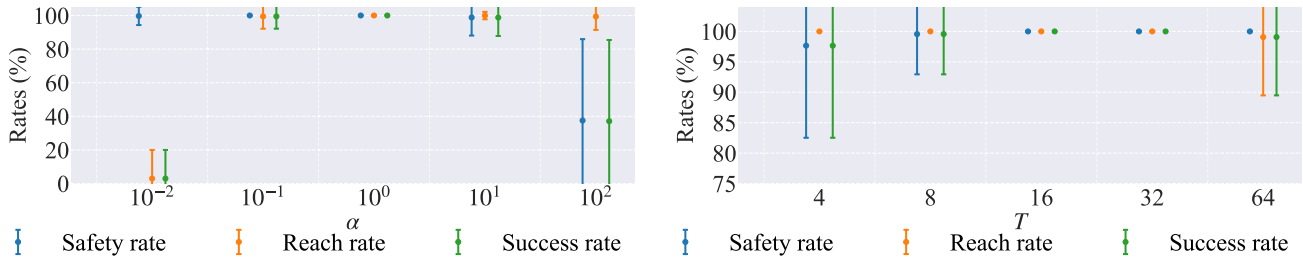


Fig. 8. *Left:* Sensitivity of GCBF+ to the CBF class-$\mathcal{K}$ function parameter α. *Right:* Sensitivity of GCBF+ to prediction horizon T for computation of the control invariant set $\mathcal{S}_{c,i}$.

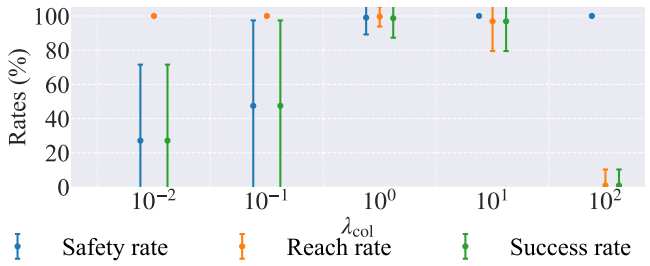


Fig. 9. Sensitivity of InforMARL to the weight λ_{col} in (28).

number of training steps, we observe that the resulting controller, while maintaining 100% safety, achieves 98% goal-reaching rate. However, as observed from the plots, GCBF+ is mildly sensitive to this parameter only at its extreme values, and almost insensitive in the nominal range $T \in [8, 32]$.

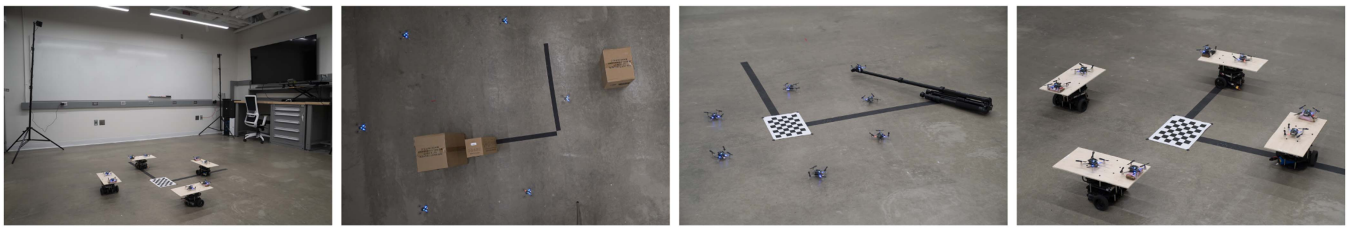
As InforMARL has the best performance among baselines, we analyze its sensitivity as well. Fig. 9 analyzes the sensitivity of the performance of InforMARL to the weight λ_{col} that dictates the penalty for collision in the RL reward function (28). It can be observed that for a relatively small range of $\lambda_{\text{col}} \in [1, 10]$, InforMARL achieves high performance. For smaller values of this weight, the RL-based method overprioritizes goal-reaching, compromising on safety, and for larger values of this weight, the goal-reaching performance is poor due to overprioritization of the system safety.

These experiments illustrate that the proposed algorithm GCBF+ is not as sensitive to its crucial hyperparameters as InforMARL, and hence, does not require fine-tuning of such parameters to obtain desirable results.

VII. Hardware Experiments

We conduct hardware experiments on a swarm of CF 2.1 platform[7] to illustrate the applicability of GCBF+ on real robotic

[7][Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/



(a) (b) (c) (d)

Fig. 10. *Hardware experiment setup:* (a) Overall experimental setup with ground robots, CF drones, and the motion capture system. (b) Setup for the position exchange with large static obstacles. (c) Setup for the position exchange with a drone acting as a moving obstacle, which is mounted on a tripod to be moved around randomly. (d) Setup for tracking and landing on a moving platform.

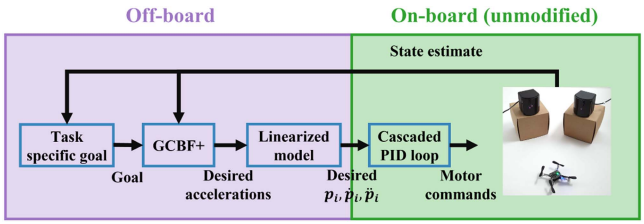


Fig. 11. Hardware control architecture for CFs.

systems. We conduct four sets of experiments as discussed below. The hardware setup is illustrated in Fig. 10. To communicate with the CFs, two Crazyradios are used. Localization is performed using the Lighthouse localization system[8]. Four SteamVR base station 2.0[9] are mounted on tripods and placed on the corners of the flight area.

A. Control Architecture

An overview of the hardware control architecture is shown in Fig. 11. Computation is split into onboard, i.e., on the CF microcontroller unit, and off-board, i.e., a laptop connected to the CFs over Crazyradio. Offboard computation happens on a laptop connected to the CFs over Crazyradios. To communicate with the CFs, we use the crazyswarm2 ROS2 package,[10] which allows for receiving full state estimates from and sending control commands to the CFs. We use a single ROS2 node for the off-board computations at 50 Hz.

Task specific logic: The state estimates are used to compute a task-specific goal position for each of the CFs. For the swapping tasks, the goals do not change. For the docking task, we take the location of the Turtlebots to be the goal position.

GCBF+ controller: The goal positions and the state estimates (position and velocities) are next used to compute target accelerations for each CF using the GCBF+ controller. This GCBF+ controller is trained using double integrator dynamics.

Ideal dynamics model: To track the computed desired accelerations from GCBF+, we make use of the `cmd_full_state`[11] interface in `crazyswarm2`. However, `cmd_full_state` requires set points for the whole state (i.e., also position and velocity) instead of just accelerations. To resolve this problem, we simulate the ideal dynamics model (i.e., double integrator) used for training GCBF+ and take the resulting future positions and velocities that would result from applying the desired accelerations after some duration Δt. We used $\Delta t = 50$ms in our hardware experiments.

We do not modify the onboard computation. The received full state set points are used as set points for a cascaded PID controller, as described in the CF documentation[12].

B. Experimental Results

We conduct the following four sets of hardware experiments.

Position exchange: In this experiment, we arrange the CF drones in a circular configuration with the objective of each drone exchanging position with the diagonally opposite drone. We perform experiments with up to 8 drones. This is a typical experiment setting used for illustration of the capability of an algorithm to maintain safety where there are many interagent interactions. The resulting trajectories of the drones are plotted in Fig. 12(a) and (e). As can be observed from the figure, the CF drones maintain the required safe distance and land safely at their desired location.

To further validate the efficacy of GCBF+ on physical robots, we perform the position exchange experiment with 6 drones in 32 trials. The distribution of the minimum distance between CF drones is shown in Fig. 13. We can observe that the GCBF+ controller successfully avoids collisions in all trials.

Position exchange with static obstacles: In this experiment, we introduce large obstacles to the position exchange experiment as illustrated in Fig. 10(b). Since the CF drones do not have lidar sensors, the lidar observations are simulated using the current position of the drones. We perform experiments with 6 drones, and the experimental results show that the success rate is 100% for all 8 experiments.

Position exchange with moving obstacle: In this experiment, we add a moving obstacle to the setup from the previous experiment to show the generalization ability of GCBF+ to unseen scenarios. The moving obstacle is moved arbitrarily around

[8][Online]. Available: http://tinyurl.com/CFlighthouse
[9][Online]. Available: http://tinyurl.com/lighthouseV2S
[10][Online]. Available: https://github.com/IMRCLab/crazyswarm2
[11][Online]. Available: http://tinyurl.com/CFcmdFullS
[12][Online]. Available: http://tinyurl.com/CFCasCadePID

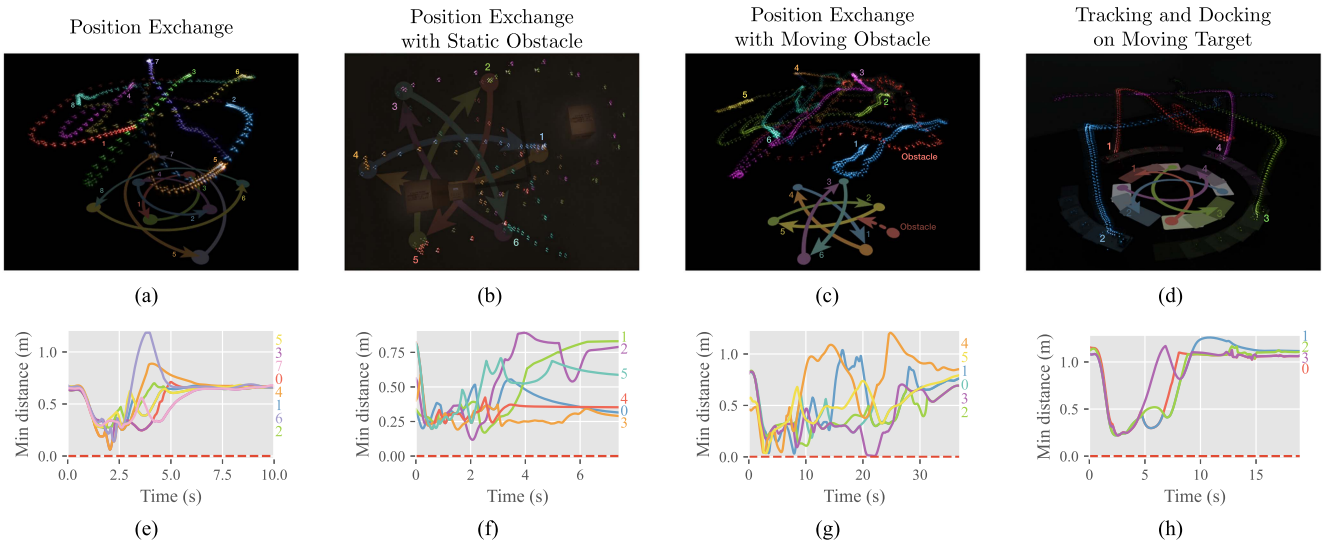


Fig. 12. *Hardware experiment results:* (a), (b), (c), (d) Time-lapse illustration of a CF swarm controlled via GCBF+. (e), (f), (g), (f) Associated minimum distances for each agent to other agents and obstacles.

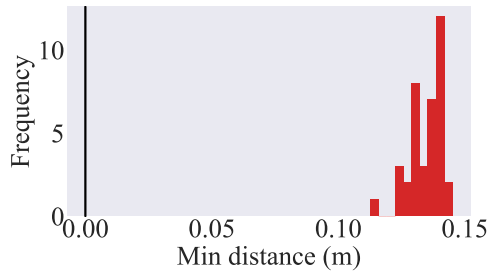


Fig. 13. Distribution of the minimum distance between CFs in the position exchange task.

the environment by a human subject. The path of the moving obstacle is not known beforehand by the controlled CF drones. The moving obstacle is the same size as a CF drone and moves with the same maximum speed as the controlled drones. Fig. 12(c) and (g) illustrates that safety is maintained at all times.

Tracking and landing on moving target: In this experiment, the CF drones are required to track a moving ground vehicle and land on it. For this experiment, instead of a simple LQR controller, a back-stepping controller is used as a nominal controller that can track a moving target with time-varying acceleration. We use four Turtlebot 3 mobile robots[13] as the moving target, equipped with a platform on its top where a CF drone can land. Initially, one CF drone is placed on each of the Turtlebots. The drones take off and start tracking the diagonally opposite moving target, while the Turtlebots move in a circular trajectory. From Fig. 12(d) and (h), the drones successfully land on the moving targets while maintaining safety with each other in this dynamically changing environment. This illustrates the generalizability of GCBF+ to a variety of control problems.

[13][Online]. Available: https://www.turtlebot.com/turtlebot3/

Experiment videos: The experiment videos are available at https://mit-realm.github.io/gcbfplus/.

VIII. Conclusion

In this article, we introduce a new class of CBF, termed GCBF, to encode interagent and obstacle collision avoidance in a large-scale MAS. We propose GCBF+, a training framework that utilizes GNNs for learning a GCBF candidate and a distributed control policy using only local observations. The proposed framework can also incorporate LiDAR point-cloud observations instead of actual obstacle locations, for real-world applications. Numerical experiments illustrate the efficacy of the proposed framework in achieving high safety rates in dense multiagent problems and its superiority over the baselines for MAS consisting of nonlinear dynamical agents. Trained on 8 agents, GCBF+ achieves over 80% safety rate in environments with more than 1000 agents, demonstrating its generalizability and scalability. A major advantage of the GCBF+ algorithm is that it does not have a tradeoff between safety and performance, as is the case with RL-based methods. Furthermore, hardware experiments demonstrate its applicability to real-world robotic systems.

The proposed work has a few limitations. In the current framework, there is no cooperation among the controlled agents, which leads to conservative behaviors. In certain scenarios, this can also lead to deadlocks resulting in a lower reaching rate, as observed in the numerical experiments as well. We are currently investigating methods of designing a high-level planner that can resolve such deadlocks and lead to improved performance. Similar to other NN-based control policies, the proposed method also suffers from difficulty in providing formal guarantees of correctness. In particular, it is difficult, if not impossible, to verify that the proposed algorithm can always keep the system safe via formal verification of the learned NNs

(see [82, Appendix 1] on NP-completeness of NN-verification problem). This informs our future line of work on looking into methods of verification of the correctness of the control policy. Lastly, in this work, we assumed that all the agents have the same underlying dynamics for simplicity. For heterogeneous MAS where agent dynamics are different, the type of agents can be encoded in node features. In addition, edge features can be chosen so that the shared information is the same for different types of nodes. We leave the extension of our method for heterogeneous systems as future work.

Appendix A
Proof of the Claim in Remark 1

For any $i \in V_a$, by definition of M and the continuity of the position of nodes, changes in the neighboring indices $\mathcal{N}_i$ can only occur without collision at a distance R. To see this, we consider the following three cases.

Case 1: $|\tilde{\mathcal{N}}_i| < M$, i.e., the number of neighbors is less than M. In this case, $\mathcal{N}_i = \tilde{\mathcal{N}}_i$, and hence, a node j is added or removed to $\mathcal{N}_i = \tilde{\mathcal{N}}_i$ when it enters or leaves the sensing radius R.

Case 2: $|\tilde{\mathcal{N}}_i| > M$, i.e., the number of neighbors is more than M. In this case, there are more than $M-1$ neighbors of agent i within sensing radius R, which, from the definition of M, implies that the MAS is unsafe.

Case 3: $|\tilde{\mathcal{N}}_i| = M$, i.e., the number of neighbors is M. In this case, if a neighbor is added to $\tilde{\mathcal{N}}_i$ without any other agent leaving the set $\tilde{\mathcal{N}}_i$, then we obtain Case 2 and the MAS is unsafe. If a node j is removed with no other neighbors added to $\tilde{\mathcal{N}}_i$, then this happens when j leaves the sensing radius R. Finally, if a node j is removed at the same time that a node k is added without changing the size of $|\mathcal{N}_i| = M$, by the continuity of the position dynamics there exists a time t where both nodes are at the same distance R from p_i. However, this implies that $|\mathcal{N}_i| = M+1$, and the MAS is unsafe by Case 2.

Consequently, changes in the neighboring indices $\mathcal{N}_i$ can only occur without collision at a distance R.

Appendix B
Proof of Lemma 1

Proof: For convenience, define $\tilde{h} : \mathbb{R} \to \mathbb{R}$ as

$$\tilde{h}(t) := h(\bar{x}_{\mathcal{N}_i(t)}(t)). \tag{38}$$

In order to prove continuous differentiability of $\tilde{h}$ at each t, consider two cases, namely time instants when the neighborhood $\mathcal{N}_i(t)$ changes due to additional or removal of a neighbor node, and time instants when there is no change in $\mathcal{N}_i(t)$. First, consider $t = t_0$ such that the neighborhood set $\mathcal{N}_i(t)$ does not change at t_0, i.e., $\lim_{t\uparrow t_0} \mathcal{N}_i(t) = \lim_{t\downarrow t_0} \mathcal{N}_t(t) = \mathcal{N}_i(t_0)$. Since h and $\bar{x}_{\mathcal{N}_i}$ are continuously differentiable at $t = t_0$, we obtain that $\tilde{h}$ is continuously differentiable at $t = t_0$.

Now, consider the case when $\mathcal{N}_i$ changes at $t = t_0$. For the sake of brevity, denote the neighborhood before the change as $\mathcal{N}_i^-$, and the neighborhood after the change as $\mathcal{N}_i^+$, i.e., $\mathcal{N}_i^- = \lim_{t\uparrow t_0} \mathcal{N}_i(t)$ and $\mathcal{N}_i^+ = \lim_{t\downarrow t_0} \mathcal{N}_i(t)$. Using 2) from Definition 1, $\tilde{h}$ is continuous at $t = t_0$. Moreover, using 2) from Definition 1 on $t < t_0$ and $t \geq t_0$ separately yields the one-sided limits

$$\lim_{t\uparrow t_0} \frac{h(\bar{x}_{\mathcal{N}_i^-}(t)) - h(\bar{x}_{\mathcal{N}_i^+}(t_0))}{t - t_0} = 0 \tag{39}$$

and

$$\lim_{t\downarrow t_0} \frac{h(\bar{x}_{\mathcal{N}_i^+}(t)) - h(\bar{x}_{\mathcal{N}_i^+}(t_0))}{t - t_0} = 0. \tag{40}$$

This, along with 1) from Definition 1 (i.e., the gradient of h is zero at t_0) implies that the derivative of $\tilde{h}$ exists at t_0 and is continuous. Hence, $\tilde{h}$ is continuously differentiable for all t. ■

Appendix C
Proof of Theorem 1

Proof: Consider any $i \in V_a$. We first prove that $\bar{x}_{\mathcal{N}_i}(t) \in \mathcal{B}_h$ for all $t \geq 0$. Define t_k for $k \in \mathbb{N}$ with $t_0 = 0$, such that $\mathcal{N}_i$ is constant on the time segments $[t_k, t_{k+1})$ for all i[14] i.e.,

$$t_k := \inf\{t > t_{k-1} \mid \exists i : \mathcal{N}_i(t) \neq \mathcal{N}_i(t_k)\}, \quad k \geq 1. \tag{41}$$

For each such interval $[t_k, t_{k+1})$, suppose that $h(\bar{x}_{\mathcal{N}_i}(t_k)) \geq 0$. Then, using [83, Lemma 3.4] on the function $t \mapsto h(\bar{x}_{\mathcal{N}_i}(t))$ along with (6) implies that

$$h(\bar{x}_{\mathcal{N}_i}(t)) \geq 0 \quad \forall t \in [t_k, t_{k+1}). \tag{42}$$

Since this holds for all $i \in V_a$, we obtain that

$$\bar{x}(t_k) \in \mathcal{C}_N \subset \mathcal{S}_N \Rightarrow \bar{x}(t) \in \mathcal{C}_N \subset \mathcal{S}_N \quad \forall t \in [t_k, t_{k+1}).$$

Since $t \mapsto h(\bar{x}_{\mathcal{N}_i}(t))$ is continuous from Lemma 1, the limit of this map at t_{k+1} exists. Taking this limit on the left side of (42) implies that $h(\bar{x}_{\mathcal{N}_i(t_{k+1})}) \geq 0$.

Since $h(\bar{x}_{\mathcal{N}_i}(0)) \geq 0$, applying induction thus gives the result that $h(\bar{x}_{\mathcal{N}_i}(t)) \geq 0$, and hence, $\bar{x}_{\mathcal{N}_i}(t) \in \mathcal{C}_i$, for all $t \geq 0$. Since this holds for all $i \in V_a$, by definition of $\mathcal{C}_N$, we have that $\bar{x} \in \mathcal{C}_N \subset \mathcal{S}_N$ for all $t \geq 0$.■

Appendix D
Experiment Details and More Results

A. Environment Details

Here, we provide the details of each experiment environment.

SingleIntegrator: We use single integrator dynamics as the base environment to verify the correctness of the baseline methods and to show the performance of the methods when there are no control input limits. The dynamics is given as $\dot{x}_i = v_i$, where $x_i = [p_i^x, p_i^y]^\top \in \mathbb{R}^2$ is the position of the ith agent and $v_i = [v_i^x, v_i^y]^\top$ its velocity. In this environment, we use $e_{ij} = x_j - x_i$ as the edge information. We use the following reward function weights for training InforMARL:

$$\lambda_{\text{nom}} = 0.1, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 5.0\,. \tag{43}$$

DoubleIntegrator: We use double integrator dynamics for this environment. The state of agent i is given by $x_i = [p_i^x, p_i^y, v_i^x, v_i^y]^\top$, where $[p_i^x, p_i^y]^\top$ is the position of the agent,

[14] A Zeno behavior is not possible because of the smoothness of the control input.

and $[v_i^x, v_i^y]^\top$ is the velocity. The action of agent i is given by $u_i = [a_i^x, a_i^y]^\top$, i.e., the acceleration. The dynamics function is given by

$$\dot{x}_i = [v_i^x, v_i^y, a_i^x, a_i^y]^\top . \tag{44}$$

The simulation time step is $\delta t = 0.03$. In this environment, we use $e_{ij} = x_j - x_i$ as the edge information. We use the following reward function weights for training InforMARL without obstacles.

$$\lambda_{\text{nom}} = 0.1, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 2.0 . \tag{45}$$

and the following reward function weights with obstacles.

$$\lambda_{\text{nom}} = 0.1, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 5.0 . \tag{46}$$

For the hand-crafted CBFs, we use $\alpha_0 = 10$.

DubinsCar: We use the standard Dubin's car model in this environment. The state of agent i is given by $x_i = [p_i^x, p_i^y, \theta_i, v_i]^\top$, where $[p_i^x, p_i^y]^\top$ is the position of the agent, θ_i is the heading, and v_i is the speed. The action of agent i is given by $u_i = [\omega_i, a_i]^\top$ containing angular velocity and longitudinal acceleration. The dynamics function is given by

$$\dot{x}_i = [v_i \cos(\theta_i), v_i \sin(\theta_i), \omega_i, a_i]^\top . \tag{47}$$

The simulation time step is $\delta t = 0.03$. We use $e_{ij} = e_j(x_j) - e_i(x_i)$ as the edge information, where $e_i(x_i) = [p_i^x, p_i^y, v_i \cos(\theta_i), v_i \sin(\theta_i)]^\top$. We use the following reward function weights for training InforMARL

$$\lambda_{\text{nom}} = 0.1, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 1.0 . \tag{48}$$

For the hand-crafted CBFs, we use $\alpha_0 = 5$.

LinearDrone: We use a linearized model for drones in our experiments. The state of agent i is given by $x_i = [p_i^x, p_i^y, p_i^z, v_i^x, v_i^y, v_i^z]^\top$ where $[p_i^x, p_i^y, p_i^z]^\top$ is the 3-D position, and $[v_i^x, v_i^y, v_i^z]^\top$ is the 3-D velocity. The control inputs are $u_i = [a_i^x, a_i^y, a_i^z]^\top$, and the dynamics function is given by

$$\dot{x}_i = \begin{bmatrix} v_i^x \\ w v_i^y \\ v_i^z \\ -1.1 v_i^x + 1.1 a_i^x \\ -1.1 v_i^y + 1.1 a_i^y \\ -6 v_i^z + 6 a_i^z \end{bmatrix} . \tag{49}$$

The simulation time step is $\delta t = 0.03$. We use $e_{ij} = x_j - x_i$ as the edge information. We use the following reward function weights for training InforMARL:

$$\lambda_{\text{nom}} = 1.0, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 1.0 . \tag{50}$$

For the hand-crafted CBFs, we use $\alpha_0 = 3$.

CrazyflieDrone: One of the advantages of GCBF is that it is model-agnostic. To show that GCBF also works for other more realistic dynamics, we test GCBF for CF dynamics. The 6-DOF quadrotor dynamics are given in [84] with $x \in \mathbb{R}^{12}$ consisting of positions, velocities, angular positions, and angular velocities, and $u \in \mathbb{R}^4$ consisting of the thrust at each of four motors

$$\dot{p}_x = (c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi))\, w \tag{51a}$$
$$- (s(\psi)c(\phi) - c(\psi)s(\phi)s(\theta))\, v + u c(\psi)c(\theta) \tag{51b}$$
$$\dot{p}_y = (s(\phi)s(\psi)s(\theta) + c(\phi)c(\psi))\, v \tag{51c}$$
$$- (c(\psi)s(\phi) - s(\psi)c(\phi)s(\theta))\, w + u s(\psi)c(\theta) \tag{51d}$$
$$\dot{p}_z = w\, c(\phi)c(\theta) - u\, s(\theta) + v\, s(\phi)c(\theta) \tag{51e}$$
$$\dot{u} = r\, v - q\, w + g\, s(\theta) \tag{51f}$$
$$\dot{v} = p\, w - r\, u - g\, s(\phi)c(\theta) \tag{51g}$$
$$\dot{w} = q\, u - p\, v + \frac{U_1}{m} - g\, c(\theta)c(\phi) \tag{51h}$$
$$\dot{\phi} = r\frac{c(\phi)}{c(\theta)} + q\frac{s(\phi)}{c(\theta)} \tag{51i}$$
$$\dot{\theta} = q\, c(\phi) - r\, s(\phi) \tag{51j}$$
$$\dot{\psi} = p + r\, c(\phi)t(\theta) + q\, s(\phi)t(\theta) \tag{51k}$$
$$\dot{r} = \frac{1}{I_{zz}} (U_2 - pq(I_{yy} - I_{xx})) \tag{51l}$$
$$\dot{q} = \frac{1}{I_{yy}} (U_3 - pr(I_{xx} - I_{zz})) \tag{51m}$$
$$\dot{p} = \frac{1}{I_{xx}} (U_4 - qr(I_{zz} - I_{yy})) \tag{51n}$$

where $m, I_{xx}, I_{yy}, I_{zz}, k_r, k_t > 0$ are system parameters, $g = 9.8$ is the gravitational acceleration, $c(\cdot), s(\cdot), t(\cdot)$ denote $\cos(\cdot), \sin(\cdot), \tan(\cdot)$, respectively, (p_x, p_y, p_z) denote the position of the quadrotor, (ϕ, θ, ψ) its Euler angles, and $u = (U_1, U_2, U_3, U_4)$ the input vector consisting of thrust U_1 and moments U_2, U_3, U_4[15].

The relation between the vector u and the individual motor speeds is given as

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ -dC_T\sqrt{2} & -dC_T\sqrt{2} & dC_T\sqrt{2} & dC_T\sqrt{2} \\ -dC_T\sqrt{2} & dC_T\sqrt{2} & dC_T\sqrt{2} & -dC_T\sqrt{2} \\ -C_D & C_D & -C_D & C_D \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{52}$$

where ω_i is the angular speed of the ith motor for $i \in \{1, 2, 3, 4\}$, C_D is the drag coefficient and C_T is the thrust coefficient. These parameters are given as: $I_{xx} = I_{yy} = 1.395 \times 10^{-5}$ kg·m^2, $I_{zz} = 2,173 \times 10^{-5}$ kg·m^2, $m = 0.0299$ kg, $C_T = 3.1582 \times 10^{-10}$ N/(r/min)^2, $C_D = 7.9379 \times 10^{-12}$ N/(r/min)^2 and $d = 0.03973$ m (see [84]). We use the following reward function weights for training InforMARL:

$$\lambda_{\text{nom}} = 0.5, \quad \lambda_{\text{goal}} = 0.1, \quad \lambda_{\text{col}} = 2.0 . \tag{53}$$

For the hand-crafted CBFs, we use $\alpha_0 = 3$.

For the CrazyflieDrone environment, we use a two-level control architecture similar to the one used in the hardware experiments described in Section VII. The chosen control algorithm computes a high-level reference velocity and yaw rate as the input, which a low-level LQR controller then tracks.

[15]We noticed that the quadrotor dynamics in [84] (as well as the references they cite) has a couple of typos (in $\dot{p}_z$ and $\dot{p}$). We have fixed those typos using first principles.

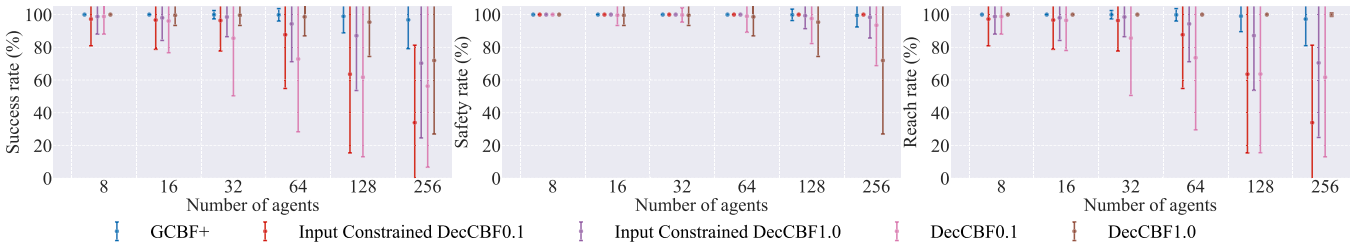


Fig. 14. *Comparison with a CBF that accounts for input constraints:* Success (left), safety (middle), and reach (right) rates for the proposed method (GCBF+), the decentralized CBF, and the decentralized input-constrained CBF in the DoubleIntegrator environment.

B. Comparison With CBF That Accounts for Input Constraints

We compared GCBF+ with generic hand-crafted CBFs that do not account for input constraints in Section VI, as there is no systematic way of hand-crafting a CBF for dynamical MASs with input constraints. However, Wang et al. [23] proposed a CBF-based *hierarchical* control framework that satisfies input constraints for double integrator systems. The pair-wise CBF candidate is given by

$$h_{ij} = \sqrt{4u_{\max}(\|p_i - p_j\| - 2r)} + \frac{(p_i - p_j)^\top}{\|p_i - p_j\|}(v_i - v_j) \tag{54}$$

where $u_{\max}$ is the input constraint. However, this CBF candidate might not be a valid CBF because it does not necessarily satisfy the CBF condition (3) and can result in an infeasible CBF-QP. To address this problem, the authors of [23] develop a hierarchical control framework where whenever the CBF-QP is infeasible for agent i, it decelerates with the maximum acceleration $u_{\max}$. We generate two baselines following this framework with $\alpha = 1.0$ and $\alpha = 0.1$ in the CBF condition (34) and compare GCBF+ with these two baselines (Input Constrained DecCBF α), as well as with the decentralized framework introduced in Section V-B using the CBF in (54). The results are shown in Fig. 14. We can observe that the safety rate of input constrained DecCBF is close to 100%, much higher than DecCBF due to explicitly considering input constraints. In light of the safety guarantees in [23, Th. VI.2], we suspect that the safety violations of input constrained DecCBF occur due to time discretization errors, which are not accounted for in the proof of safety in [23]. However, since the agents need to decelerate with maximum acceleration whenever the CBF-QP is infeasible, the reach rates of the input constrained DecCBF are low, resulting in an overall low success rate. GCBF+, however, considers input constraints during the learning process, and the learned GCBF does not need another hierarchical controller to maintain safety. Moreover, this hybrid controller of [23] only works with double integrator dynamics and cannot be directly used with other dynamics.

C. Analysis of Time Delay

Apart from the goal-reaching rate, we report the distribution of the relative delay ratio, defined as $t_{\text{GCBF+}}/t_{\text{nom}}$ where $t_{\text{GCBF+}}$ and t_{nom} are the timesteps taken by the agents to reach their goals using GCBF+ and the nominal controller, respectively, in the DoubleIntegrator environment with 1024 agents in Fig. 15.

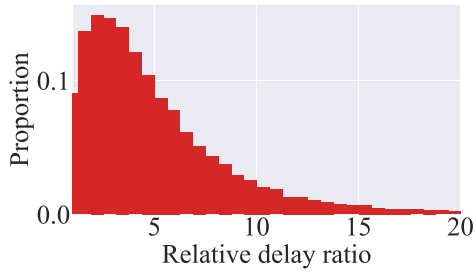


Fig. 15. Histogram of the relative delay ratio, defined as the timestep taken by GCBF+ to reach the goal divided by the timestep taken by the nominal controller to reach the goal in the DoubleIntegrator environment.

We can observe that about 90% of the agents reach their goals within $10\times$ the timesteps taken by the nominal controller.

ACKNOWLEDGMENT

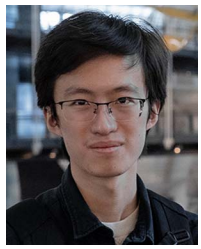
Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.
[2] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3701–3708, Jun. 2023.
[3] A. Kattepur, H. K. Rath, A. Simha, and A. Mukherjee, "Distributed optimization in multi-agent robotics for industry 4.0 warehouses," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.*, 2018, pp. 808–815.
[4] L. M. Schmidt, J. Brosig, A. Plinge, B. M. Eskofier, and C. Mutschler, "An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst.*, 2022, pp. 1342–1349.
[5] P. Palanisamy, "Multi-agent connected autonomous driving using deep reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–7.
[6] M. Zhou et al., "SMARTS: An open-source scalable multi-agent RL training school for autonomous driving," in *Proc. Conf. Robot Learn.*, 2021, pp. 264–285.
[7] S. Zhang, Y. Xiu, G. Qu, and C. Fan, "Compositional neural certificates for networked dynamical systems," in *Proc. Learn. Dyn. Control Conf.*, 2023, pp. 272–285.
[8] Y. Tian et al., "Search and rescue under the forest canopy using multiple uavs," *Int. J. Robot. Res.*, vol. 39, no. 10/11, pp. 1201–1221, 2020.
[9] K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Multiple UAVs in forest fire fighting mission using particle swarm optimization," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 1404–1409.

[10] C. Ju, J. Kim, J. Seol, and H. I. Son, "A review on multirobot systems in agriculture," *Comput. Electron. Agriculture*, vol. 202, 2022, Art. no. 107336.
[11] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, pp. 11237–11245.
[12] R. J. Afonso, M. R. Maximo, and R. K. Galvão, "Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming," *Int. J. Robust Nonlinear Control*, vol. 30, no. 14, pp. 5464–5491, 2020.
[13] J. Netter, G. P. Kontoudis, and K. G. Vamvoudakis, "Bounded rational RRT-QX: Multi-agent motion planning in dynamic human-like environments using cognitive hierarchy and Q-learning," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 3597–3602.
[14] A. D. Saravanos, Y. Aoyama, H. Zhu, and E. A. Theodorou, "Distributed differential dynamic programming architectures for large-scale multiagent control," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4387–4407, Dec. 2023.
[15] K. Garg, S. Zhang, O. So, C. Dawson, and C. Fan, "Learning safe control for multi-robot systems: Methods, verification, and open challenges," *Ann. Rev. Control*, vol. 57, 2024, Art. no. 100948.
[16] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24611–24624, 2022.
[17] A. D. Ames et al., "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.
[18] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Control Syst. Lett.*, vol. 1, no. 2, pp. 310–315, Oct. 2017.
[19] M. Jankovic and M. Santillo, "Collision avoidance and liveness of multi-agent systems with CBF-based controllers," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 6822–6828.
[20] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 777–783.
[21] K. Garg and D. Panagou, "Robust control barrier and control lyapunov functions with fixed-time convergence guarantees," in *Proc. Amer. Control Conf.*, 2021, pp. 2292–2297.
[22] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.
[23] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017.
[24] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 127–132, Jan. 2021.
[25] D. R. Agrawal and D. Panagou, "Safe control synthesis via input constrained control barrier functions," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 6113–6118.
[26] S. Zhang, K. Garg, and C. Fan, "Neural graph control barrier functions guided distributed collision-avoidance multi-agent control," in *Proc. 7th Annu. Conf. Robot Learn.*, 2023, pp. 2373–2392.
[27] S. Nayak, K. Choi, W. Ding, S. Dolan, K. Gopalakrishnan, and H. Balakrishnan, "Scalable multi-agent reinforcement learning through intelligent information aggregation," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 25817–25833.
[28] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *Proc. Eur. Control Conf.*, 2018, pp. 1523–1528.
[29] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 01, pp. 7643–7650.
[30] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.
[31] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8097–8104.
[32] L. Zheng et al., "Magent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 8222–8223.
[33] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *Int. J. Control*, vol. 87, no. 1, pp. 52–63, 2014.
[34] C. Toumieh and A. Lambert, "Decentralized multi-agent planning using model predictive control and time-aware safe corridors," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11110–11117, Oct. 2022.
[35] E. L. Zhu, Y. R. Stürz, U. Rosolia, and F. Borrelli, "Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control," in *Proc. IEEE 59th Conf. Decis. Control*, 2020, pp. 6198–6203.
[36] G. Fedele and G. Franzè, "A distributed model predictive control strategy for constrained multi-agent systems: The uncertain target capturing scenario," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1549–1563, Apr. 2024.
[37] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 375–382, Apr. 2019.
[38] C. Conte, T. Summers, M. N. Zeilinger, M. Morari, and C. N. Jones, "Computational aspects of distributed optimization in model predictive control," in *Proc. IEEE 51st IEEE Conf. Decis. Control*, 2012, pp. 6819–6824.
[39] A. Nedić and J. Liu, "Distributed optimization for control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 77–103, 2018.
[40] P. Mestres and J. Cortés, "Distributed and anytime algorithm for network optimization problems with separable structure," in *Proc. IEEE 62nd Conf. Decis. Control*, 2023, pp. 5463–5468.
[41] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing sostools: A general purpose sum of squares programming solver," in *Proc. IEEE 41st Conf. Decis. Control*, 2002, vol. 1, pp. 741–746.
[42] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1216–1229, Jul. 2018.
[43] M. Srinivasan, M. Abate, G. Nilsson, and S. Coogan, "Extent-compatible control barrier functions," *Syst. Control Lett.*, vol. 150, 2021, Art. no. 104895.
[44] P. Zhao, R. Ghabcheloo, Y. Cheng, H. Abdi, and N. Hovakimyan, "Convex synthesis of control barrier functions under input constraints," *IEEE Control Syst. Lett.*, vol. 7, pp. 3102–3107, 2023.
[45] A. A. Ahmadi and A. Majumdar, "Some applications of polynomial optimization in operations research and real-time decision making," *Optim. Lett.*, vol. 10, pp. 709–729, 2016.
[46] Z. Cai, H. Cao, W. Lu, L. Zhang, and H. Xiong, "Safe multi-agent reinforcement learning through decentralized multiple control barrier functions," 2021, *arXiv:2103.12553*.
[47] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=P6_q1BRxY8Q
[48] V. N. Fernandez-Ayala, X. Tan, and D. V. Dimarogonas, "Distributed barrier function-enabled human-in-the-loop control for multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 7706–7712.
[49] H. Wang, A. Papachristodoulou, and K. Margellos, "Distributed control design and safety verification for multi-agent systems," in *Proc. 62nd IEEE Conf. Decis. Control (CDC),*, Singapore, 2023, pp. 5481–5486, doi: 10.1109/CDC49753.2023.10383473.
[50] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1749–1767, Jun. 2023.
[51] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Proc. Conf. Robot Learn.*, 2022, pp. 1724–1735.
[52] Z. Qin, D. Sun, and C. Fan, "Sablas: Learning safe control for black-box dynamical systems," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1928–1935, Apr. 2022.
[53] O. So et al., "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2024, pp. 11532–11539.
[54] Y. Meng, Z. Qin, and C. Fan, "Reactive and safe road user simulations using neural barrier certificates," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6299–6306.
[55] J. Dinneweth, A. Boubezoul, R. Mandiau, and S. Espié, "Multi-agent reinforcement learning for autonomous vehicles: A survey," *Auton. Intell. Syst.*, vol. 2, 2022, Art. no. 27, doi: 10.1007/s43684-022-00045-z.
[56] W. Zhang, O. Bastani, and V. Kumar, "Mamps: Safe multi-agent reinforcement learning via model predictive shielding," 2019, *arXiv:1910.12639*.
[57] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264–146272, 2019.

[58] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3052–3059.
[59] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: A survey," *Auton. Robots*, vol. 46, no. 5, pp. 569–597, 2022.
[60] Z. Dai, T. Zhou, K. Shao, D. H. Mguni, B. Wang, and H. Jianye, "Socially-attentive policy optimization in multi-agent self-driving system," in *Proc. Conf. Robot Learn.*, 2023, pp. 946–955.
[61] X. Pan, M. Liu, F. Zhong, Y. Yang, S.-C. Zhu, and Y. Wang, "MATE: Benchmarking multi-agent reinforcement learning in distributed target coverage control," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 27862–27879, 2022.
[62] B. Wang, J. Xie, and N. Atanasov, "DARL1N: Distributed multi-agent reinforcement learning with one-hop neighbors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 9003–9010.
[63] Y. Wang, M. Damani, P. Wang, Y. Cao, and G. Sartoretti, "Distributed reinforcement learning for robot teams: A review," *Curr. Robot. Rep.*, vol. 3, no. 4, pp. 239–257, 2022.
[64] C. Yu, H. Yu, and S. Gao, "Learning control admissibility models with graph neural networks for multi-agent navigation," in *Proc. Conf. Robot Learn.*, 2023, pp. 934–945.
[65] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized GNN-based policies," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 8772–8778.
[66] X. Jia, L. Sun, H. Zhao, M. Tomizuka, and W. Zhan, "Multi-agent trajectory prediction by combining egocentric and allocentric views," in *Proc. Conf. Robot Learn.*, 2022, pp. 1434–1443.
[67] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8944–8950.
[68] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792.
[69] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3835–3845.
[70] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.
[71] M. A. Pereira, A. D. Saravanos, O. So, and E. A. Theodorou, "Decentralized safe multi-agent stochastic optimal control using deep FBSDEs and ADMM," in *Proc. Robot.: Sci. Syst.*, 2022.
[72] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. 4th Int. Conf. Learn. Representations*, San Juan, Puerto Rico, May 2016.
[73] K.-C. Hsu, V. Rubies-Royo, C. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Proc. Robot.: Sci. Syst.*, Jul. 2021.
[74] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8550–8556.
[75] L. Schäfer, F. Gruber, and M. Althoff, "Scalable computation of robust control invariant sets of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 69, no. 2, pp. 755–770, Feb. 2024.
[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, May 2014.
[77] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3221–3226, Apr. 2020.
[78] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 285–292.
[79] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, pp. 1–36, 2019.
[80] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, 2005.
[81] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. Amer. Control Conf.*, 2016, pp. 322–328.
[82] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Proc. Comput. Aided Verification: 29th Int. Conf., CAV* 2017, 2017, pp. 97–117.
[83] H. K. Khalil, *Nonlinear Systems*, 3rd ed, vol. 115. Upper Saddle River, NJ: Patience Hall, 2002.
[84] C. Budaciu, N. Botezatu, M. Kloetzer, and A. Burlacu, "On the evaluation of the crazyflie modular quadcopter system," in *Proc. IEEE 24th Int. Conf. Emerg. Technol. Factory Autom.*, 2019, pp. 1189–1195.



Songyuan Zhang (Student Member, IEEE) received the Bachelor of Engineering degree in engineering mechanics [Tsien Excellence in Engineering Program (TEEP)] from Tsinghua University, Beijing, China, in 2021 and the Master of Science degree in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2024.

He is currently a graduate student with the Department of Aeronautics and Astronautics, MIT. His research interests include learning safe and performant controllers for complex large-scale autonomous systems using certificates and reinforcement learning.

Oswin So (Graduate Student Member, IEEE) received the Bachelor of Science degree in computer science from the Georgia Institute of Technology, Atlanta, GA, Georgia, in 2021.

He is currently a graduate student with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include constrained optimization, safety for dynamical systems, multiagent planning, and stochastic optimal control.

Kunal Garg (Member, IEEE) received the Master of Engineering and Ph.D. degrees in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2019 and 2021, respectively, and the Bachelor of Technology degree in aerospace engineering from the Indian Institute of Technology, Mumbai, India, in 2016.

He is currently an Assistant Professor with the Mechanical and Aerospace Engineering Program, School for Engineering of Matter, Transport, and Energy, Arizona State University, Tempe, AZ, USA. Previously, he was a Postdoctoral Associate with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, and before that, with the Department of Electrical Engineering and Computer Science, University of California, Santa Cruz, Santa Cruz, CA, USA. His research interests include robust multiagent path planning, switched and hybrid system-based analysis, and control synthesis for multiagent coordination, finite- and fixed-time stability of dynamical systems with applications to control synthesis for spatiotemporal specifications, and continuous-time optimization.

Chuchu Fan (Member, IEEE) received the Ph.D. degree in computer engineering from the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2019.

She is currently an Associate Professor with the Department of Aeronautics and Astronautics (AeroAstro) and Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology (MIT). Before that, she was a Postdoctoral Researcher with the California Institute of Technology, Pasadena, CA, USA. Her research group, Realm with MIT, works on using rigorous mathematics, including formal methods, machine learning, and control theory, for the design, analysis, and verification of safe autonomous systems.

Dr. Fan was the recipient of an NSF CAREER Award, an AFOSR Young Investigator Program Award, an ONR Young Investigator Program Award, and the 2020 ACM Doctoral Dissertation Award.