Distributed Safe Multi-agent Control Using Neural Graph Control Barrier Functions

Anonymous Author(s)
Affiliation
Address
`email`

Abstract: We consider the problem of distributed safe multi-agent control in large-scale environments with potentially moving obstacles, where a large number of agents are required to maintain safety using only local information and reach their goals. This paper addresses the problem of safety, scalability, and generalizability by introducing graph control barrier functions (GCBFs) for distributed control. The newly introduced GCBF is based on the well-established CBF theory for safety guarantees but utilizes a graph structure for scalable and generalizable decentralized control. We use graph neural networks to learn both neural a GCBF certificate and distributed control. We also extend the framework from handling state-based models to directly taking point clouds from LiDAR for more practical robotics settings. We demonstrated the efficacy of GCBF in a variety of numerical experiments, where the number, density, and traveling distance of agents, as well as the number of unseen and uncontrolled obstacles increase. Empirical results show that GCBF can constantly outperform leading methods such as MAPPO and multi-agent distributed CBF (MDCBF). Trained with only 16 agents, GCBF can achieve up to 3 times improvement of success rate (agents reach goals and never encountered in any collisions) on < 500 agents, and still maintain more than 50% success rates for >1000 agents when other methods completely fail.

1 Introduction

Multi-agent systems (MAS) can complete much more complex tasks efficiently as compared to single-agent systems such as reconnaissance or sensor coverage of a large unexplored area. Safety of MAS, in terms of collision and obstacle avoidance, is a non-negotiable requirement in the numerous autonomous robotics applications (see [1] for an overview) such as a swarm of drones flying in a dense forest [2, 3], multi-object configuration and manipulation in warehouses [4, 5, 6] and autonomous driving [7, 8, 9]. In addition, the agents are required to either follow a pre-defined path or reach a destination for completing their individual or team objectives. With the increase in the number of robots in the MAS, it becomes difficult to design control policies for all the agents for such a multi-task problem as the computational complexity grows exponentially with the MAS scale [10].

Common multi-agent motion planning methods include but are not limited to solving mixed integer linear programs (MILP) for computing safe paths for agents [11, 12] and RRT-based methods [13]. However, they are not scalable to large-scale MAS. Multi-agent Reinforcement Learning (MARL)-based approaches, e.g., Multi-agent Proximal Policy Optimization (MAPPO) [14], have also been adapted to solve multi-agent motion planning problems. However, most of the MARL works model safety as a penalty rather than a hard constraint and thus, cannot guarantee safety. In recent years, safety constraints have been handled via control barrier functions (CBFs) [15]. Particular for MAS, generally a CBF is assigned for each safety constraint, and then an approximation method is used for accounting for the multiple constraints [16, 17, 18, 19]. The issue with such methods is that it is very difficult to construct a handcrafted CBF for large-scale MAS consisting of highly nonlinear dynamics.

The Multi-agent Decentralized CBF (MDCBF) framework in [20] uses a neural network-based CBF designed for MAS, but they do not encode a method of distinguishing between other controlled agents and *uncontrolled* agents such as static and dynamic obstacles. Furthermore, they use a discrete approximation of the time derivative of the CBF but do not account for changing graph topology in their approximation, which can lead to a wrong evaluation of the CBF constraints and consequently, failure. The Control Admissiblity Models (CAM)-based framework in [21] also attempts to address a similar problem. However, one of the limitations of their approach is that it involves sampling control actions from a set defined by CAM. However, such a sampling method cannot always find a feasible control input that satisfies the safety constraint.

To overcome these limitations, in this paper, we present a novel Graph CBF for large-scale MAS to address the problem of safety, scalability, and generalizability. We propose a learning-based control policy to achieve a higher safety rate in practice. We use graph neural networks (GNN) to better capture the changing graphical topology of distance-based inter-agent communications. We also also use LiDAR-based observations for handling unseen and potentially unstationary obstacles in real-world environments. With these technologies, our proposed framework can generalize well to many challenging settings, including more crowded environments and unseen obstacles.

We consider a 2D car environment and a 3D drone environment in our numerical experiments. In the obstacle-free case, we train with 16 agents and test with over 1000 agents. In particular, for < 500 agents, the proposed method achieves a threefold improvement in safely reaching tasks, while for large-scale experiments (> 1000 agents) where the existing approaches achieve close to 0 success rate, our approach achieves $50\% - 100\%$ success rate. In the obstacle environment, we consider only 16 point-sized obstacles in training, while in testing, we consider up to 32 large-sized obstacles. We see over 15% improvement in success rate as compared to baselines. The experiments corroborate that the proposed method outperforms the existing methods in successfully completing the tasks in a variety of 2D and 3D environments. Our contributions are summarized below:

- We introduce Graph CBF (GCBF), a new kind of barrier function for MAS to encode and enforce the safety constraint and to handle different types of agents and obstacles.
- We use GNNs to jointly learn a GCBF and a distributed controller which is robust to the changes of neighbors, and a LiDAR-based observation model for obstacles.
- Empirical performance shows a significant improvement by our GCBF over other leading approaches, especially in difficult settings.

Related work Sampling-based path planning approaches such as *prioritized* multi-agent path finding [22], conflict-based search for MAPF [23] can be used for multi-agent path planning for known environments, but do not generalize to new unseen environments. The work in [24] scales to large-scale systems, but it only considers discrete action space and hence does not apply to robotic platforms that use more general continuous input signals. Works such as [25, 26, 27] address this problem using GNNs for generalization to unseen environments and are shown to work on teams of up to a hundred agents. However, they are not scalable to very large-scale problems (e.g., a team of 1000 agents) due to the computational bottleneck. In recent years, the most commonly employed method of solving safe motion planning problems involves neural CBF-based approaches [20, 28, 29, 30]. Machine learning (ML)-based approaches have shown promising results in designing CBF-based controllers for complex safety-critical systems [28, 29, 30]. The NN-CBF framework consists of model-based learning [31, 30, 32, 33] or model-free learning [34, 35, 36]. Our approach uses a model-based learning framework, and in contrast to the aforementioned works, applies to MAS. Utilizing the permutation-invariance property, GNN-based methods have been employed for problems involving MAS [26, 25, 37, 21, 38, 39, 40, 27, 41]. These prior work only consider static obstacles in the environment, or do not consider the presence of obstacles or *uncontrolled* agents at all. On the other hand, there is also a lot of work on MARL-based approaches with focuses on motion planning [42, 43, 44, 45, 46, 47, 14, 48, 49, 50]. But these approaches cannot provide safety guarantees due to the reward structure and as argued in [51], MARL-based methods are still in the initial phase of development when it comes to safe multi-agent motion planning.

2 Problem formulation

In this work, we consider the problem of designing a distributed control framework for a set of N agents $V_a := \{1, 2, \dots, N\}$ to drive them to their goal locations while maintaining safety. The motion of each agent is governed by general nonlinear dynamics $\dot{x}_i = F_i(x_i, u_i)$, where $x_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^m$ are the state, control input for the i-th agent, respectively and $F_i : \mathbb{R}^n \to \mathbb{R}^m$ is assumed to locally Lipschitz continuous. Here, the vector x_i consists of the position p_i along with other state variables such as speed, orientation, etc. Note that it is possible to consider heterogeneous MAS where the dynamics of agents are different. However, for simplicity, we restrict our paper to the case when all the agents have the same underlying dynamics, i.e., $F_i = F$ for all i. The environment also consists of stationary or dynamic obstacles $\mathcal{O}_k$ for $k \in \{1, 2, \dots, M\}$, where $\mathcal{O}_k$ represents the space occupied by obstacle k. The control objective for each agent is to navigate the obstacle environment to reach its goal location while maintaining safety. We use a LiDAR-based observation model similar to [31], which can be directly used for real-world robotic applications. The observation data consists of n_{rays} evenly-spaced rays originating at the robot and measures the *relative* locations of objects in its sensing radius. The observation data for agent i is denoted by $y_i \in \mathbb{R}^{n_{\text{rays}} \times \mathfrak{n}}$ where $\mathfrak{n} = 2$ (respectively, 3) for 2D (respectively, 3D) environment.

The safety requirement imposes that each pair of agents maintain a minimum safety distance $2r$ where $r > 0$ is the radius of a circle that can contain the entire physical body of each agent. It also requires that each agent maintains a safe distance from other obstacles in the environment. Furthermore, each agent has a limited sensing radius R. We define the neighbor agents of agent i as $\mathcal{N}_i^a = \{j \in V_a \mid \|p_i - p_j\| \leq R, j \neq i\}$, and the neighbor obstacles of agent i as $\mathcal{N}_i^o = \{k \mid \|y_i^k\| \leq R\}$. Therefore, the agents can only sense other agents or obstacles in the set of their neighbors $\mathcal{N}_i = \mathcal{N}_i^a \cup \mathcal{N}_i^o$. The formal statement of the problem considered in this work is given below.

Problem 1 *Given a set of N agents of safety radius r, sensing radius R and a set of non-colliding goal locations $\{p_i^{\text{goal}} \in \mathbb{R}^{\mathfrak{n}}\}_{i=1}^N$, design a distributed control policy $\pi_i = \pi_i(x_i, \bar{x}_i, \bar{y}_i, x_i^{\text{goal}})$ for each agent i, where $\bar{x}_i$ is the conglomerated states of the neighbors $j \in \mathcal{N}_i^a$ and $\bar{y}_i$ the conglomerated observations from $\mathcal{N}_i^o$, such that the following holds for the closed-loop trajectories of the agents:*

- ***Obstacle avoidance**: $\|y_i^j(t)\| > r, \forall j$, i.e., the agents do not collide with the obstacles;*
- ***Inter-agent collision avoidance**: $\|p_i(t) - p_j(t)\| > 2r$ for all $t \geq 0$, $j \neq i$, i.e., the inter-agent distance is greater than the safe distance;*
- ***Liveness**: $\|p_i(t) - p_i^{\text{goal}}\| \to 0$, i.e., each agent eventually reaches its goal location p_i^{goal}.* [1]

3 Methodology

Noticing that the agents, the hitting points of LiDAR rays, and the information flow between them can be naturally modeled as a graph, we propose a novel *graph* CBF (GCBF) which encodes the safety constraint based on the graph structure of MAS. We use a nominal controller for the liveness requirement and use GNNs to learn the GCBF jointly with the safe controller. During application, the GCBF is used to detect unsafe scenarios and switch between the nominal controller and the safe controller to maintain safety. Our GNN architecture is capable of handling a variable number of neighbors and so, it leads to a distributed and scalable solution to the safe MAS control problem.

We start by briefly reviewing the notion of CBF commonly used in literature for safety requirements [15]. For a given closed safe set $\mathcal{S} \subset \mathbb{R}^n$, a function $h : \mathbb{R}^n \to \mathbb{R}$ is termed as a CBF if there exists a class-$\mathcal{K}$ function[2] such that the following holds:

$$h(x) > 0 \; \forall x \in \text{int}(\mathcal{S}), \; h(x) < 0 \; \forall x \notin \mathcal{S}, \text{ and } \sup_u L_F h(x, u) \geq -\alpha(h(x)) \; \forall x \in \mathcal{S}, \quad (1)$$

[1] In the rest of the paper, we omit the argument t for the sake of brevity.

[2] A monotonically increasing continuous function $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$ with $\alpha(0) = 0$ is termed as class-$\mathcal{K}$.

where $L_f h(x) := \frac{\partial h}{\partial x} f(x)$ is the Lie derivative of the function h along f, and $\text{int}(S)$ denotes the relative interior of a closed set S. The existence of a CBF implies the existence of a control input u which keeps the system safe. Based on the notion of CBF, we define a new notion of *graph CBF* (GCBF) for encoding safety in MAS. Before formally introducing GCBF, we briefly review the basics of the graph structure. A directed graph is defined as $\mathcal{G} = (V, E)$ where V is the set of nodes and $E = \{(i_1, i_2)\}$ is the set of edges representing the flow of information from node i_2 to i_1. For the considered MAS, the nodes consist of agents V_a and the hitting points V_o of LiDAR rays in their observations, and hence $V = V_a \cup V_o$. The edges are defined between each of the observed points and the observing agent when the distance between them is within the sensing radius R. Since the flow of information is from the observed point to the observing agent, the set of edges $E = V_a \times V$. We use GNN to represent GCBF, so we first define *node* and *edge* features for GCBF.

Node features and edge features The nodes features v_i in GCBF encode the type of the agent with $v_i = 0$ for *controlled* agents (i.e., the agents that operate under the commanded controller) and $v_i = 1$ for *uncontrolled* agents (i.e., the hitting points for LiDAR rays). The edge features e_{ij} are defined as the information shared from node j to agent i, which depends on the states of node j and node i. Since the safety objective depends on the relative positions, one of the edge features is the relative position p_{ij}. The rest of the features can be chosen depending on the underlying system dynamics, e.g., relative velocities for double integrators, and relative headings for Dubin's cars. For brevity, we use $\bar{e}_i = (e_{ij_1}, e_{ij_2}, \dots, e_{ij_{|\mathcal{N}_i|}}, \tilde{e}_{ij_{|\mathcal{N}_i|+1}}, \dots, \tilde{e}_{ij_{N+n_{\text{rays}}}})$ with $\bar{e}_i \in \mathbb{R}^p$ for some $p > 0$, and $\bar{v}_i = (v_{j_1}, v_{j_2}, \dots, v_{j_{|\mathcal{N}_i|}}, \tilde{v}_{j_{|\mathcal{N}_i|+1}}, \dots, \tilde{v}_{j_{N+n_{\text{rays}}}})$ to represent the collected edge and node features for agent i. Here, we use $\tilde{e}_{ij}$ for $j \notin \mathcal{N}_i$ and $\tilde{v}_k$ for the rays $k \notin \mathcal{N}_i$ with constant values so that the sizes of the vectors $\bar{e}_{ij}, \bar{v}_i$ remain fixed.[3] Now we are ready to introduce the notion of GCBF.

Definition 1 (GCBF) *A function $h : \mathbb{R}^p \times \{0,1\}^{N+n_{\text{rays}}} \to \mathbb{R}$ is termed as a Graph CBF (GCBF) if there exists a class-$\mathcal{K}$ function α such that*

$$h(\bar{e}_i, \bar{v}_i) > 0 \; \forall x_i \in \mathcal{S}_i, \quad h(\bar{e}_i, \bar{v}_i) < 0 \; \forall x_i \notin \mathcal{S}_i, \text{ and } \dot{h}(\bar{e}_i, \bar{v}_i) \geq -\alpha(h(\bar{e}_i, \bar{v}_i)) \; \forall \; x_i \in \mathcal{S}_i, \quad (2)$$

where $\mathcal{S}_i = \left\{ x_i \;\middle|\; (\|y_i^k\| > r, \forall k \in n_{\text{rays}}) \bigwedge (\min_{j \in V_a, k \neq i} \|p_i - p_j\| > 2r) \right\}$ *is the safe set for agent i.*

Since the node features are constant, $\dot{h}$ is computed with respect to the edge features as

$$\dot{h}(\bar{e}_i, \bar{v}_i) = \frac{\partial h(\bar{e}_i, \bar{v}_i)}{\partial \bar{e}_i} \frac{\partial \bar{e}_i}{\partial x_i} F(x_i, u_i) + \frac{\partial h(\bar{e}_i, \bar{v}_i)}{\partial \bar{e}_i} \sum_{j \in \mathcal{N}_i} \frac{\partial \bar{e}_i}{\partial x_j} F(x_j, u_j). \quad (3)$$

Note that while choosing the edge features $\bar{e}_i$, it is important to make sure that the time derivates of the features of agent i include the control u_i, so that the input can help keep the system safe. To this end, we assume $\frac{\partial}{\partial u_i} \left(\frac{\partial h(\bar{e}_i, \bar{v}_i)}{\partial \bar{e}_i} \frac{\partial \bar{e}_i}{\partial x_i} F(x_i, u_i) \right) \not\equiv 0$. Note that this is similar to assuming $L_g h \not\equiv 0$ for CBF in (1) where $F(x, u) = f(x) + g(x)u$, which is very common (see [15]). Under this assumption, we can state the following result on the safety of the system under GCBF.

Theorem 1 *Given a set of N agents, assume that there exists a GCBF h satisfying (2) for some class-$\mathcal{K}$ function α. Then, the resulting closed-loop trajectories of agents with non-colliding initial conditions under any smooth control input $u_i \in \mathcal{U}_i^{\text{safe}} := \left\{ u \in \mathcal{U}_i \mid \dot{h}(\bar{e}_i, \bar{v}_i) + \alpha(h(\bar{e}_i, \bar{v}_i)) \geq 0 \right\}$ satisfy $x_i(t) \in \mathcal{S}_i$ for all $i \in V_a$ and $t \geq 0$.*

Note that the presence of moving obstacles and controlled agents make the safe set $\mathcal{S}_i$ time-varying. The proof of Theorem 1 is based on the CBF-based forward invariance arguments for time-varying safe sets [19, 52] and is skipped here.

Safe control policy For the multi-objective Problem 1, we use a hierarchical approach for the goal-reaching and the safety objectives. First, we design a nominal controller $u_i^{\text{nom}} = \pi_{\text{nom}}(x_i, x_i^{\text{goal}})$

[3] We use GNN to model GCBF with input $(e_{ij_1}, \dots, e_{ij_{|\mathcal{N}_i|}})$ and $(v_{j_1}, \dots, v_{j_{|\mathcal{N}_i|}})$ since GNN can have variable-size inputs. We use fixed-size input $(\bar{e}_i, \bar{v}_i)$ so that GCBF is mathematically well-defined.

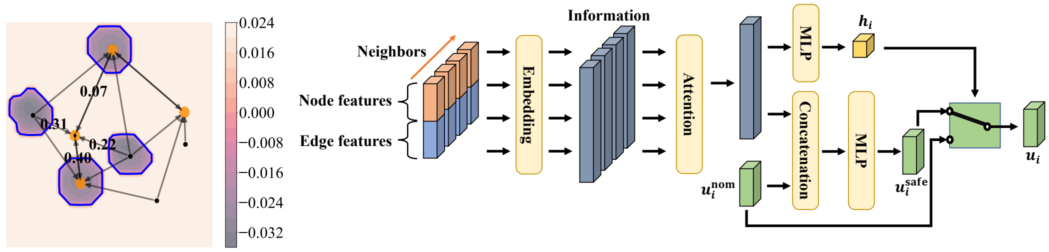

Figure 1: Left: the contours of the learned GCBF of the agent with a diamond mark, where the blue boundary is the 0-level set of the GCBF. Orange circles are agents and black dots are obstacles. The weights on the edges show the attention values. Right: the overview of the proposed framework.

for the goal-reaching objective. In this work, we use LQR and PID-based nominal controllers. Next, using the nominal controller, we design a minimum-norm controller that satisfies the safety constraint using an optimization framework. With GCBF h, a solution to the following optimization problem:

$$\min_{u_i \in \mathcal{U}_i} \quad \|u_i - u_i^{\text{nom}}\|^2, \tag{4a}$$

$$\text{s.t.} \quad \frac{\partial h(\bar{e}_i, \bar{v}_i)}{\partial \bar{e}_i} \frac{\partial \bar{e}_i}{\partial x_i} F(x_i, u_i) + \sum_{j \in \mathcal{N}_i} \frac{\partial h(\bar{e}_i, \bar{v}_i)}{\partial \bar{e}_i} \frac{\partial \bar{e}_i}{\partial x_j} F(x_j, u_j) \geq -\alpha(h(\bar{e}_i, \bar{v}_i)), \tag{4b}$$

keeps agent i in its safety region. Note that (4) is *not a distributed framework* for finding the control policy, since the constraint for computing u_i depends on u_j. Thus, it is not straightforward to solve (4) in a distributed manner, although there is some work on addressing such problems [53]. To this end, we use an NN-based control policy that satisfies the safety constraint and does not require solving a centralized non-convex optimization problem online. Next, we discuss the training setup for jointly learning both GCBF and a distributed safe control policy (see Figure 1).

GCBF and distributed control policy training We parameterize the GCBF as NNs with parameters θ, denoted as h_θ. The NN contains a GNN component and a multilayer perceptron (MLP) component. In the GNN component, each connected edge $\{i, j\}$ first goes through an MLP layer f_{θ_1}, which encodes the edge feature e_{ij} and the node features v_j to the latent space, i.e., $q_{ij} = f_{\theta_1}(e_{ij}, v_j)$. Then, we use the attention mechanism [54] to aggregate the information of the neighbors, i.e., $q_i = \sum_{j \in \mathcal{N}_i} \text{softmax}(f_{\theta_2}(q_{ij})) f_{\theta_3}(q_{ij})$, where f_{θ_2} and f_{θ_3} are two NNs parameterized by θ_2 and θ_3. The function f_{θ_2} is often called "gate" NN in literature [55], and the output of $\text{softmax}(f_{\theta_2}(q_{ij}))$ is called "attention", which is a scatter value between 0 and 1 for each agent $j \in \mathcal{N}_i$ represents how critical agent j is to agent i. We discuss later the necessity of applying the attention mechanism. After the GNN component, aggregated information is processed by another MLP with parameters θ_4 to get the GCBF value for each agent, i.e., $h_i = f_{\theta_4}(q_i)$.

We design the safe distributed controller as $u_i^{\text{safe}} = \pi_\phi(\bar{e}_i, \bar{v}_i, \pi_{\text{nom}}(x_i, x_i^{\text{goal}}))$. The distributed control policy π_ϕ is an NN with a similar structure as GCBF, designed for collision and obstacle avoidance. The GNN component of π_ϕ is the same as the GNN component of h_θ, except that π_ϕ also uses π_{nom} as its feature. This helps the NN controller to learn how to modify the agent's behavior given a nominal policy to keep it safe. Thus, we concatenate the nominal control signal u_i^{nom} with the output of the GNN component as the input to the MLP component of π_ϕ (see Figure 1). Note that the input to the control policy is only the local information $(\bar{e}_i, \bar{v}_i)$, and unlike (4), it does not require knowledge of neighbors' inputs. In this way, the controller is fully distributed, and thanks to GNN's ability to handle variable sizes of inputs, π_ϕ generalizes to larger graphs with much more neighbors. We train the GCBF and the distributed controller by minimizing the empirical loss $\mathcal{L} = \sum_{i \in V_a} \mathcal{L}_i$, where $\mathcal{L}_i$ is the loss for agent i defined as

$$\begin{aligned}\mathcal{L}_i(\theta, \phi) = &\sum_{x_i \in \mathcal{S}_i} \left[\gamma - \dot{h}_\theta(\bar{e}_i, \bar{v}_i) - \alpha(h_\theta(\bar{e}_i, \bar{v}_i))\right]^+ + \sum_{x_i \in \mathcal{S}_i} [\gamma - h_\theta(\bar{e}_i, \bar{v}_i)]^+ \\ &+ \sum_{x_i \notin \mathcal{S}_i} [\gamma + h_\theta(\bar{e}_i, \bar{v}_i)]^+ + \eta \left\| \pi_\phi(\bar{e}_i, \bar{v}_i, \pi_{\text{nom}}(x_i, x_i^{\text{goal}})) - \pi_{\text{nom}}(x_i, x_i^{\text{goal}}) \right\|,\end{aligned} \tag{5}$$

where $[\cdot]^+ = \max(\cdot, 0)$, and $\gamma > 0$ is a hyper-parameter to encourage strict inequalities. The first three terms in the loss correspond to the GCBF conditions in (2), while the last term encourages small controller deviation from π_{nom} so that the control input $\pi_\phi(\bar{e}_i, \bar{v}_i, \pi_{\mathrm{nom}}(x_i, x_i^{\mathrm{goal}}))$ can have better goal-reaching performance with $\eta > 0$ as a hyper-parameter to balance the weight of the GCBF constraint losses and the norm of the resulting input. Note that $\dot{h}_\theta(\bar{e}_i, \bar{v}_i)$ is calculated using (3). Therefore, during training, when we use gradient descent and backpropagate $\mathcal{L}_i^{\mathrm{GCBF}}(\theta, \phi)$, the gradients are passed to not only the controller of agent i but also the controllers of all neighbors in $\mathcal{N}_i$.[4] For the class-$\mathcal{K}$ function α, we simply use $\alpha(h) = \alpha \cdot h$, where $\alpha > 0$ is a positive constant. During training, we use the on-policy strategy and collect data by executing the learned controller π_ϕ.

One of the challenges of evaluating the loss function $\mathcal{L}$ is how to estimate $\dot{h}_\theta$. Similar to [21], we estimate $\dot{h}_\theta$ by $\left(h_\theta(\bar{e}_i(t_{k+1}), \bar{v}_i(t_{k+1})) - h_\theta(\bar{e}_i(t_k), \bar{v}_i(t_k))\right)/\delta t$, where $\delta t = t_{k+1} - t_k$ is the simulation timestep. However, the discretized approximation may cause an issue if the graph connections change between any two consecutive time steps. Fortunately, the attention mechanism we use naturally addresses this problem. During training, the agents learn to pay more attention (i.e., close to 1) to nodes that are near while the attention value is close to 0 for the nodes that are at the boundary of the sensing region. Therefore, if an edge breaks in between time steps and a node gets out of the sensing radius, the CBF value does not change significantly. In this manner, the estimation of $\dot{h}_\theta$ does not encounter large errors due to changes in edges in between time steps.

GCBF detector and online policy refinement When the training finishes, we can execute our controller in a fully distributed manner. To achieve better goal-reaching performance, we use the learned GCBF as a detector to detect unsafe scenarios and use a switching control policy to reduce potential conservatism due to only using *safe* policy π_ϕ. In particular, we define the control assignment for each agent as $u_i = u_i^{\mathrm{nom}}$ if $u_i^{\mathrm{nom}} \in \mathcal{U}_i^{\mathrm{safe}}$ and $u_i = u_i^{\mathrm{safe}}$, otherwise. Namely, at each time step, the system uses the nominal controller if the GCBF conditions (2) are satisfied with the nominal controller u_i^{nom}. If not, it switches to the learned policy u_i^{safe} to ensure safety.

While the control policy π_ϕ is designed to satisfy the GCBF conditions (2), the GCBF conditions can still be violated because of various reasons, such as distribution shift in testing and difficulty in exploring the state-space in high-dimensional and large-scale MAS during training. To this end, similar to [20], we use an online policy refinement technique to make the learned policy *safer*. At a given time instant, if the learned policy π_ϕ does not satisfy the GCBF conditions (2), we compute the residue $\delta(u_i^{\mathrm{safe}}) = \max\Big(0, \gamma - \dot{h}_\theta(\bar{e}_i, \bar{v}_i) - \alpha(h_\theta(\bar{e}_i, \bar{v}_i))\Big)$ and use gradient descent to update the control policy π_ϕ until $\delta(u_i^{\mathrm{safe}}) = 0$ or the maximum iteration is reached.

4 Experiments

Environments We conduct experiments on three different environments consisting of a SimpleCar modeled under double-integrator dynamics, a DubinsCar model, and a Drone modeled under linearized drone dynamics (see Appendix A.2 for more details). Both car environments are 2D while the drone environment is 3D. The parameters in the 2D car environments are $R = 1, r = 0.05, u_M = 0.8$ where u_M denotes the maximum speed of each agent. For the 3D drone environment are $R = 0.5, r = 0.05, u_M = 0.6$. The workspace $\mathcal{X} = l^{\mathrm{n}}$ of each of the environments is a hyper-rectangle of side-length $l > 0$. The total timesteps of experiments are 2500 for 2D environments and 2000 for 3D environments.

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 time period over all agents. The reaching rate is defined as the ratio of agents reaching their goal location by the end of the experiment time period. The success rate is defined as the ratio of agents that are both safe and goal-reaching. We note that the safety metric in [21] is slightly misleading as they measure the portion of collision-free states for safety rate. For each environment, we evaluate the performance over 16 instances of randomly

[4]We re-emphasize on the fact that during testing, the neighbors' inputs are not required for π_ϕ.

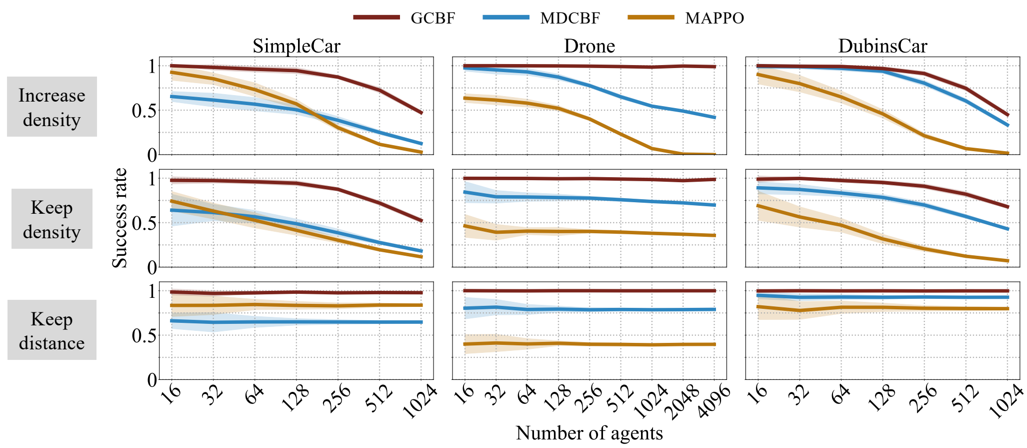

Figure 2: Success rate of GCBF, MDCBF, and MAPPO algorithms across the three environments and the three sets of experiments, namely, increasing density of the agents in a fixed workspace, increasing the size of the workspace to keep the density same, and increasing the size of the workspace but limiting the average distance traveled by agents.

chosen initial and goal locations from the workspace $\mathcal{X}$ for 3 policies trained with different random seeds. Here, we report the mean success rate and their standard deviations for the 16 instances for each of the 3 policies. We report the safety rate, reaching rate, and ablation results in Appendix B.

Baselines We use MDCBF [20] and MAPPO [14] as the baselines for comparisons. MDCBF learns pair-wise CBFs between agents and takes the minimum on one agent as the CBF value of this agent. Furthermore, it considers each neighbor equally important without attention and does not use CBF as a detector but directly uses the learned controller. MAPPO is a MARL-based algorithm that learns to be safe and goal-reaching by maximizing the expected reward. For fair comparisons, we re-implement the algorithm from [14] using GNN. We do not perform comparisons with other MARL-based methods due to two main reasons: first, we perform comparisons with MDCBF which is already illustrated to outperform MARL-based methods, and second, it takes a lot of computational resources and cost to re-implement, train and test numerous baselines.

Experiment settings We conduct four sets of experiments for demonstrating the scalability, generalizability, and reliability of the proposed method. First, we fix the workspace size $\mathcal{X}$ where the agent trajectories evolve. In this experiment, we use $\mathcal{X} = 32 \times 32$ for 2D car environments and $\mathcal{X} = 16 \times 16 \times 16$ for the 3D drone environment and perform experiments with up to 1024 agents for the 2D environments and up to 4096 agents for the 3D environment. In the second set of experiments, we keep the per-unit agent density constant. To this end, we increase the size of $\mathcal{X}$ as the number of agents increases from 16 to 4096 (see Appendix A.2 for workspace sizes). In the third set of experiments, we further constrain the maximum traveling distance to 4.0 units for each agent while increasing the size of the workspace to keep the per-unit agent density constant. In the fourth set of experiments, we introduce moving obstacles where we perform experiments in the DubinsCar environment with up to 32 obstacles and 64 agents in a workspace $\mathcal{X} = 12 \times 12$. The obstacles are assumed to be moving with a bounded, constant, unknown speed up to 0.2 units and the size of the obstacle varies between 0 to 0.5 units. Agents use LiDAR to detach obstacles. Each agent generates equally-spaced 32 rays with a maximum sensing radius $R = 1.0$ unit. For the first three experiments, we train all the algorithms with 16 agents, and for the fourth experiment, we train with 64 agents and with 16 randomly generated point-sized obstacles to model LiDAR observations.

Results Figure 2 shows the performance of the proposed framework (GCBF) against the baselines MDCBF and MAPPO. In all the experiments, the success rate of GCBF is higher than that of the considered baselines. Particularly as the number of agents increases, the decrement in the success rate of MAPPO and MDCBF is very high. For the SimpleDrone environment, we notice that there is

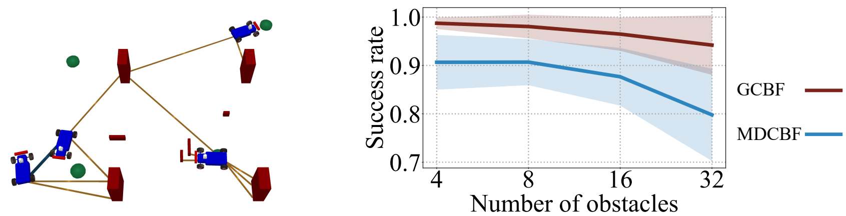

Figure 3: Left: Illustration of the DubinsCar environment with obstacles. The green circles are goal points and the red rectangles are obstacles. The solid blue line shows the connection between agents and the orange lines show LiDAR rays. Right: Success rate plots for GCBF and MDCBF.

almost no drop in the success rate with an increase in the number of agents. We speculate that this is because the agents in 3D have more degrees of freedom to move to avoid collisions and hence, achieve a very high safety rate (see the individual safety and goal-reaching plots in Appendix B). For the first two sets of experiments, the success rate drop is primarily because the inter-agent interactions are increasing. In the first set of experiments, it is clear with an increase in the density of agents for a fixed workspace, the inter-agent interactions increase. For the second set of experiments, although the per-unit agent density is the same, with an increase in the workspace size, the average distances traveled by the agents in randomly generated initial and goal location instances also increase. Thus, the inter-agent interaction increases. We designed the third set of experiments to further analyze the effect of traveling distance on success rate. In the third set of experiments, not only the density but also the average distance traveled by each agent is fixed, which keeps the number of inter-agent interactions constant. It can be observed that in this case, the success rate of GCBF remains very close to 1 in all three environments. Thus, we can conclude on the basis of these experiments that the main deciding factor for success rate is the average inter-agent interactions. Figure 3 illustrates that the proposed method using GCBF achieves a higher success rate across obstacle environments as compared to MDCBF since it can deal with different types of neighbors. The success rate of MAPPO with obstacles is consistently lower than 0.1, so we do not include it in the plot.

5 Limitations

In the current framework, there is no cooperation among the controlled agents, which leads to conservative behaviors. In certain scenarios, this non-cooperation can also lead to deadlocks or oscillatory behavior. Another limitation is the assumption of knowledge of the neighbors' velocities. From a practical point of view, measuring relative position is possible using LiDAR or other sensors, but accurate estimation of other agents' velocities and accelerations is not possible. Similar to any other NN-based control policy, 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 neural networks. These limitations inform our future line of work on relaxation of the assumption on available information, introducing cooperation among agents to reduce conservatism, and looking into methods of verification of the correctness of the control policy.

6 Conclusions

In this paper, we introduce a new notion of GCBF to encode inter-agent collision and obstacle avoidance in control for large-scale multi-agent systems with LiDAR-based observations, and jointly learn it with a distributed controller using GNNs. The proposed control framework is completely distributed as each agent only uses local information in its sensing region, and thus, is scalable to large-scale problems. Experimental results demonstrate that even when trained on small-scale MAS, the proposed method can achieve higher success rates in completing goal-reaching tasks while maintaining safety for large-scale MAS even in the presence of dynamic obstacles.

References

[1] C. Ju, J. Kim, J. Seol, and H. I. Son. A review on multirobot systems in agriculture. *Computers and Electronics in Agriculture*, 202:107336, 2022.

[2] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How. Search and rescue under the forest canopy using multiple uavs. *The International Journal of Robotics Research*, 39(10-11): 1201–1221, 2020.

[3] K. A. Ghamry, M. A. Kamel, and Y. Zhang. Multiple uavs in forest fire fighting mission using particle swarm optimization. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1404–1409. IEEE, 2017.

[4] B. Li and H. Ma. Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses. *IEEE Robotics and Automation Letters*, 8(6):3701–3708, 2023. doi:10.1109/LRA.2023.3272272.

[5] A. Kattepur, H. K. Rath, A. Simha, and A. Mukherjee. Distributed optimization in multi-agent robotics for industry 4.0 warehouses. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 808–815, 2018.

[6] A. Krnjaic, J. D. Thomas, G. Papoudakis, L. Schäfer, P. Börsting, and S. V. Albrecht. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. *arXiv preprint arXiv:2212.11498*, 2022.

[7] 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 *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1342–1349. IEEE, 2022.

[8] P. Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[9] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, et al. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Conference on Robot Learning*, pages 264–285. PMLR, 2021.

[10] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41, 2013.

[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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11237–11245, 2021.

[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. *International Journal of Robust and Nonlinear Control*, 30(14):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 *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3597–3602. IEEE, 2021.

[14] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[16] P. Glotfelter, J. Cortés, and M. Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE Control Systems Letters*, 1(2):310–315, 2017.

[17] M. Jankovic and M. Santillo. Collision avoidance and liveness of multi-agent systems with cbf-based controllers. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6822–6828. IEEE, 2021.

[18] 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 *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 777–783. IEEE, 2020.

[19] K. Garg and D. Panagou. Robust control barrier and control lyapunov functions with fixed-time convergence guarantees. In *2021 American Control Conference (ACC)*, pages 2292–2297. IEEE, 2021.

[20] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=P6_q1BRxY8Q.

[21] C. Yu, H. Yu, and S. Gao. Learning control admissibility models with graph neural networks for multi-agent navigation. In *Conference on Robot Learning*, pages 934–945. PMLR, 2023.

[22] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7643–7650, 2019.

[23] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[24] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, and Y. Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[25] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro. Multi-robot coverage and exploration using spatial graph neural networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8944–8950. IEEE, 2021.

[26] Q. Li, F. Gama, A. Ribeiro, and A. Prorok. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11785–11792. IEEE, 2020.

[27] C. Yu and S. Gao. Reducing collision checking for sampling-based motion planning using graph neural networks. *Advances in Neural Information Processing Systems*, 34:4274–4289, 2021.

[28] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[29] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[30] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

[31] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan. Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robotics and Automation Letters*, 7(2):1904–1911, 2022.

[32] M. Saveriano and D. Lee. Learning barrier functions for constrained motion planning with dynamical systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 112–119. IEEE, 2019.

[33] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7139–7145. IEEE, 2020.

[34] Z. Qin, D. Sun, and C. Fan. Sablas: Learning safe control for black-box dynamical systems. *IEEE Robotics and Automation Letters*, 7(2):1928–1935, 2022.

[35] A. Taylor, A. Singletary, Y. Yue, and A. Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, 2020.

[36] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6878–6884. IEEE, 2019.

[37] 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 *2022 International Conference on Robotics and Automation (ICRA)*, pages 8772–8778. IEEE, 2022.

[38] X. Jia, L. Sun, H. Zhao, M. Tomizuka, and W. Zhan. Multi-agent trajectory prediction by combining egocentric and allocentric views. In *Conference on Robot Learning*, pages 1434–1443. PMLR, 2022.

[39] X. Ji, H. Li, Z. Pan, X. Gao, and C. Tu. Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8936–8943. IEEE, 2021.

[40] A. Khan, A. Ribeiro, V. Kumar, and A. G. Francis. Graph neural networks for motion planning. *arXiv preprint arXiv:2006.06248*, 2020.

[41] Q. Li, W. Lin, Z. Liu, and A. Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021.

[42] X. Xiao, B. Liu, G. Warnell, and P. Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022.

[43] 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 Robotics and Automation Letters*, 5(2):3221–3226, 2020.

[44] B. Wang, Z. Liu, Q. Li, and A. Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4): 6932–6939, 2020.

[45] W. Zhang, O. Bastani, and V. Kumar. Mamps: Safe multi-agent reinforcement learning via model predictive shielding. *arXiv preprint arXiv:1910.12639*, 2019.

[46] M. Everett, Y. F. Chen, and J. P. How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[47] 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 *Conference on Robot Learning*, pages 946–955. PMLR, 2023.

[48] 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. *Advances in Neural Information Processing Systems*, 35:27862–27879, 2022.

[49] Z. Cai, H. Cao, W. Lu, L. Zhang, and H. Xiong. Safe multi-agent reinforcement learning through decentralized multiple control barrier functions. *arXiv preprint arXiv:2103.12553*, 2021.

[50] B. Wang, J. Xie, and N. Atanasov. Darl1n: Distributed multi-agent reinforcement learning with one-hop neighbors. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9003–9010. IEEE, 2022.

[51] Y. Wang, M. Damani, P. Wang, Y. Cao, and G. Sartoretti. Distributed reinforcement learning for robot teams: a review. *Current Robotics Reports*, 3(4):239–257, 2022.

[52] L. Lindemann and D. V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, 2018.

[53] 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 *Robotics: Science and Systems*, 2022.

[54] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning*, pages 3835–3845. PMLR, 2019.

[55] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[56] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.