

---

# HMARL-CBF – Hierarchical Multi-Agent Reinforcement Learning with Control Barrier Functions for Safety-Critical Autonomous Systems

---

**H. M. Sabbir Ahmad<sup>1</sup>, Ehsan Sabouni<sup>1</sup>, Alexander Wasilkoff<sup>1</sup>, Param Budhraja<sup>1</sup>, Zijian Guo<sup>1</sup> Songyuan Zhang<sup>2</sup>, Chuchu Fan<sup>2</sup>, Christos Cassandras<sup>1</sup>, Wenchao Li<sup>1</sup>**

<sup>1</sup>Boston University, <sup>2</sup>Massachusetts Institute of Technology

{sabbir92, esabouni, awasiloff, paramb, zjguo, cgc, wenchao}@bu.edu  
{szhang21, chuchu}@mit.edu

## Abstract

We address the problem of safe policy learning in multi-agent safety-critical autonomous systems. In such systems, it is necessary for each agent to meet the safety requirements at all times while also cooperating with other agents to accomplish the task. Toward this end, we propose a safe Hierarchical Multi-Agent Reinforcement Learning (HMARL) approach based on Control Barrier Functions (CBFs). Our proposed hierarchical approach decomposes the overall reinforcement learning problem into two levels — learning joint cooperative behavior at the higher level and learning safe individual behavior at the lower or agent level conditioned on the high-level policy. Specifically, we propose a skill-based HMARL-CBF algorithm in which the higher-level problem involves learning a joint policy over the skills for all the agents and the lower-level problem involves learning policies to execute the skills safely with CBFs. We validate our approach on challenging environment scenarios whereby a large number of agents have to safely navigate through conflicting road networks. Compared with existing state-of-the-art methods, our approach significantly improves the safety achieving near perfect (within 5%) success/safety rate while also improving performance across all the environments.

## 1 Introduction

Safety-critical multi-agent systems are ever-growing with applications spanning self-driving cars, unmanned aerial vehicles (UAVs), swarm robotics, soft robotics, autonomous underwater vehicles (AUVs), to name only a few. As these systems operate in complex and often unpredictable environments, failure can result in catastrophic outcomes, including risks to human lives, environmental damage, or severe economic consequences. It is, therefore, necessary to learn cooperative policies which can achieve the task while ensuring safety between agents and with respect to the environment.

The problem is further exacerbated in partially observable settings. Learning a flat policy using existing Multi-Agent Reinforcement Learning (MARL) [1, 2, 3] algorithms by adjoining the safety constraints to the reward function provides one possible approach; this, however, suffers from scalability and high sample complexity with the number of agents, and, most importantly, lacks safety guarantees. The problem of safety has been addressed in single and multi-agent RL using constrained Markov Decision Processes (CMDP) [4, 5, 6]. However, this formulation considers the discounted sum of the constraints over a trajectory, thus enforcing safety in a statistical sense over trajectories as opposed to pointwise time constraints enforced over each entire trajectory, pertinent to safety-critical systems. A Hierarchical approach as proposed in [7, 8, 8, 9] can alleviate sample complexity but existing methods do not tackle safety. Toward this end, we propose the following:

1. We propose HMARL - CBF - a novel CTDE Safety Guaranteed Hierarchical Multi-Agent Reinforcement Learning approach based on CBFs for safety-critical cooperative multi-agent partially observable systems. These systems have safety constraints that have to be satisfied at all times.
2. Our approach adopts a hierarchical structure based on skills/options, where the high-level policy leverages skills to optimize the cooperative behavior among agents, while the low-level policy is dedicated to learning and executing safe skills. It is worth emphasizing that this hierarchical design guarantees safety during both the training phase and real-world deployment.
3. We validate our proposed approach on challenging conflicting environment scenarios where a large number of agents must each travel safely from its origin to its destination without colliding with other agents. Simulation results demonstrate superior performance and safety compliance achieving near perfect ( $\geq 95\%$ ) success/safety rate compared to existing benchmark methods.

## 2 Related Works

**Multi-Agent RL.** MARL provides a solution for cooperative multi-agent games, commonly following the Centralized Training with Decentralized Execution (CTDE) scheme to improve coordination [1, 2, 3, 10]. Value decomposition methods (e.g., VDN [3], QMIX [2], QTRAN [11]) improve credit assignment by factorizing the joint value function, while policy gradient methods (e.g., MADDPG [1], COMA [12]) use centralized critics to stabilize learning. However, safety remains unaddressed, as existing methods do not consider real-time constraints.

**Hierarchical Multi-Agent RL (HMARL).** Hierarchical RL structures decision-making hierarchically, accelerating learning [13]. Classical methods include HAM [14], MAXQ [15], options [16, 17], and feudal architectures [18]. Deep HRL extends this with subgoal-based [19, 20], option-based [21, 22], and skill-based frameworks [23, 24]. HMARL introduces hierarchy in MARL for enhanced coordination and exploration. Notable methods include value decomposition (QTRAN [11], MAXQ [15], HAVEN [8], [9]), feudal structures [25], and option-based hierarchies [26]. Temporal abstraction is explored in [7], while meta-policy learning appears in [19]. Skill discovery in HRL has explored mutual information maximization [27], trajectory clustering [28], and hierarchical skill identification [29], option learning [30], imitation [31], and adaptive refinement via Skill-Critic [32]. However, these works define skills as fixed length or variable length sequences lack interpretable semantics and safety guarantees, limiting applicability to safety-critical systems.

**Safe RL** Safety in multi-agent systems is commonly addressed using the Constrained Markov Decision Process (CMDP) framework [4, 5, 6, 33]. Approaches include primal methods [34, 35, 36] and primal-dual methods [37, 38, 39]. While CMDPs ensure safety over trajectories, they do not consider pointwise time constraints critical for safety-critical systems. Multi-agent scenarios further complicate this, as the number of constraints scales with agents. Primal-dual methods [35, 40, 41] also face issues with training instability, slow convergence, and hyperparameter sensitivity. Additionally, works addressing reach avoid problems [42, 43] are primarily limited to single agent fully observable settings. Previous works on RL-CBFs have harnessed exact or approximate models [44, 45, 46] to learn safe policies. The existing methods tackle safety for single-agent systems [46, 47], and multi-agent systems [48, 49]. [44] considers a multi-agent cooperative setting conditioned on a fixed coordination policy which can be suboptimal. In [46], solved constrained RL problem with CBF-based chance constraints using the augmented Lagrangian, which nullifies the merits of CBF filters. Additionally, existing works require a nominal policy which is learned [47, 50], however this, can be challenging for MAS. Finally, [51] proposed a method for mulit-agent systems, but such methods suffer from high sample complexity and scalability due to the flat policy structure.

## 3 Background

### 3.1 Semi-Markov Decision Process

A Semi-Markov Decision Process (SMDP) is a generalization of a MDP that allows actions to take variable amounts of time steps before transitioning to the next state. Building on that, similar to [52], we model a Multi-Agent Semi-Markov Decision Process (MSMDP) as a tuple  $(\mathcal{N}, \mathcal{S}, \mathcal{Z}, P, R, \gamma, \mathcal{T})$

described as follows:  $\mathcal{N}$  is a finite set of  $N$  agents, with each agent  $i \in \mathcal{N}$  having an individual action set  $\mathcal{Z}^i$ . The joint action space  $\mathcal{Z} = \prod_{i=1}^n \mathcal{Z}^i$  consists of joint actions  $\mathbf{z} = (z^1, \dots, z^n)$ , representing the joint actions of the agents. The joint state space is denoted by  $\mathcal{S} = \prod_{i \in \mathcal{N}} \mathcal{S}^i \times \mathcal{S}^e$  where  $\mathcal{S}^e$  is the environment state. The reward function is denoted by  $R : \mathcal{S} \times \mathcal{Z} \times \mathbb{N} \rightarrow \mathbb{R}$  where  $R(\mathbf{s}, \mathbf{z}, k)$  corresponds to the reward accrued executing action  $\mathbf{z}$  in state  $\mathbf{s}$  in  $k$  steps,  $\gamma$  is the discount factor, and the multistep transition probability function  $P : \mathcal{S} \times \mathbb{N} \times \mathcal{S} \times \mathcal{Z} \rightarrow [0, 1]$  gives the probability of transitioning from state  $\mathbf{s}$  to state  $\mathbf{s}'$  in  $k$  time steps as a result of the joint action  $\mathbf{z}$ . Since the joint actions may have varying termination times,  $P$  is influenced by the duration of executing an action from the state it is initiated until transitioning to the next state (also referred to as the decision epoch) and a termination scheme  $\mathcal{T}$ .

Three termination strategies for the temporally extended joint actions,  $\tau_{\text{any}}$ ,  $\tau_{\text{all}}$ , and  $\tau_{\text{continue}}$ , are analyzed in [53, Figure 1]. First, we define decision epoch as the time when either a subset or all the agents select new actions. Synchronous schemes,  $\tau_{\text{any}}$  and  $\tau_{\text{all}}$ , set decision epochs when either any or all actions within a joint action terminate, with all agents selecting new actions. In contrast, the asynchronous scheme  $\tau_{\text{continue}}$  sets decision epoch when any of the actions within the joint action terminate. Then let those actions that did not terminate naturally continue running while initiating new actions for the agents that completed their action. Our decentralized approach adopts this asynchronous action update strategy.

The equation corresponding to the state value function  $V^\pi(\mathbf{s})$  and state-action value function  $Q^\pi(\mathbf{s}, \mathbf{z})$  are defined below for a MSMDP:

$$V^\pi(\mathbf{s}) = \mathbb{E}_\pi [Q^\pi(\mathbf{s}, \mathbf{z})] = \mathbb{E}_{\mathbf{z}, k, \mathbf{s}'} [R(\mathbf{s}, \mathbf{z}, k) + \gamma^k V^\pi(\mathbf{s}')] \quad (1)$$

$$Q^\pi(\mathbf{s}, \mathbf{z}) = \mathbb{E}_{k, \mathbf{s}'} [R(\mathbf{s}, \mathbf{z}, k) + \gamma^k V^\pi(\mathbf{s}')] = \mathbb{E}_{k, \mathbf{s}'} [R(\mathbf{s}, \mathbf{z}, k) + \gamma^k \mathbb{E}_{\mathbf{z}'} [Q^\pi(\mathbf{s}', \mathbf{z}')]] \quad (2)$$

where  $k$  represents the time interval until agent(s) update their action, and  $R(\mathbf{s}, \mathbf{z}, k)$  represents the reward obtained by taking the joint action  $\mathbf{z}$  in state  $\mathbf{s}$  over the random time interval  $k$ . The learning objective is to find the joint policy  $\pi$  for all the agents that maximizes:  $\mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^{\sum_{t'=0}^{t-1} k_{t'}} R(\mathbf{s}_t, \mathbf{z}_t, k_t)]$ .

### 3.2 Control Barrier Functions

Consider the following agent dynamics (with the agent index dropped for simplicity):

$$\dot{\mathbf{s}} = f(\mathbf{s}) + g(\mathbf{s})\mathbf{a}, \quad (3)$$

where  $\mathbf{s} \in \mathbb{R}^n$  is the state of the agent,  $\mathbf{a} \in \mathcal{U} \subseteq \mathbb{R}^q$  are the primitive actions, and  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times q}$  are locally Lipschitz.

**Definition 3.1** (Class  $\mathcal{K}$  function). A continuous function  $\alpha : [0, \beta) \rightarrow [0, \infty]$ ,  $\beta > 0$  is said to belong to class  $\mathcal{K}$  if it is strictly increasing and  $\alpha(0) = 0$ .

**Definition 3.2.** A set  $C$  is forward invariant for system (3) if for every  $\mathbf{s}(0) \in C$ , we have  $\mathbf{s}(t) \in C$ , for all  $t \geq 0$ .

**Definition 3.3** (Control barrier function [54]). Given a continuously differentiable function  $b : \mathbb{R}^n \rightarrow \mathbb{R}$  and the set  $C := \{\mathbf{s} \in \mathbb{R}^n : b(\mathbf{s}) \geq 0\}$ ,  $b(\mathbf{s})$  is a candidate control barrier function (CBF) for the system (3) if there exists a class  $\mathcal{K}$  function  $\alpha$  such that

$$\sup_{\mathbf{a} \in \mathcal{U}} [L_f b(\mathbf{s}) + L_g b(\mathbf{s})\mathbf{a} + \alpha(b(\mathbf{s}))] \geq 0, \quad (4)$$

for all  $\mathbf{s} \in C$ , where  $L_f, L_g$  denote the Lie derivatives along  $f$  and  $g$ , respectively. If we can find a control barrier function  $b$  then the set  $C$  is forward invariant. Thus when  $C$  is the set of safe states a control barrier function provides us with a safety certificate. CBFs can be extended to higher order resulting in HOCBFs.

**Theorem 3.4** ([54]). *Given a constraint  $b(\mathbf{s}(t))$  with the associated sets  $C_i$ 's as defined in (6), any Lipschitz continuous controller  $\mathbf{a}(t)$ , that satisfies (7)  $\forall t \geq t_0$  renders the sets  $C_i$  (including the set corresponding to the actual safety constraint  $C_1$ ) forward invariant for control system (3).*

**Definition 3.5** (Relative degree). The relative degree of a (sufficiently many times) differentiable function  $b : \mathbb{R}^n \rightarrow \mathbb{R}$  with respect to system (3) is the number of times it needs to be differentiated along its dynamics until the control  $\mathbf{a}$  explicitly appears in the corresponding derivative.

For a constraint  $b(\mathbf{s}) \geq 0$  with relative degree  $m$ ,  $b : \mathbb{R}^n \rightarrow \mathbb{R}$ , and  $\zeta_0(\mathbf{s}) := b(\mathbf{s})$ , we define a sequence of functions  $\zeta_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i \in \{1, \dots, m\}$  :

$$\zeta_i(\mathbf{s}) := \dot{\zeta}_{i-1}(\mathbf{s}) + \alpha_i(\zeta_{i-1}(\mathbf{s})), \quad i \in \{1, \dots, m\}, \quad (5)$$

where  $\alpha_i(\cdot)$ ,  $i \in \{1, \dots, m\}$  denotes a  $(m-i)^{\text{th}}$  order differentiable class  $\mathcal{K}$  function. We further define a sequence of sets  $C_i$ ,  $i \in \{1, \dots, m\}$  associated with (5) which take the following form,

$$C_i := \{\mathbf{s} \in \mathbb{R}^n : \zeta_{i-1}(\mathbf{s}) \geq 0\}, \quad i \in \{1, \dots, m\}. \quad (6)$$

**Definition 3.6** (High Order CBF (HOCBF) [55, 56]). Let  $C_1, \dots, C_m$  be defined by (6) and  $\zeta_1(\mathbf{s}), \dots, \zeta_m(\mathbf{s})$  be defined by (5). A function  $b : \mathbb{R}^n \rightarrow \mathbb{R}$  is a High Order Control Barrier Function (HOCBF) of relative degree  $m$  for system (3) if there exists  $(m-i)^{\text{th}}$  order differentiable class  $\mathcal{K}$  functions  $\alpha_i$ ,  $i \in \{1, \dots, m-1\}$  and a class  $\mathcal{K}$  function  $\alpha_m$  such that

$$\sup_{\mathbf{a} \in \mathcal{U}} [L_f^m b(\mathbf{s}) + L_g L_f^{m-1} b(\mathbf{s}) \mathbf{a} + \Omega(b(\mathbf{s})) + \alpha_m(\zeta_{m-1}(\mathbf{s}))] \geq 0 \quad (7)$$

for all  $\mathbf{s} \in \bigcap_{i=1}^m C_i$ . In (7),  $L_f^m$  and  $L_g$  denotes derivative along  $f$  and  $g$   $m$  times and one time respectively, and  $S(\cdot)$  denotes the remaining Lie derivative along  $f$  with degree less than or equal to  $m-1$  (omitted for simplicity, see [55]).

Note that the HOCBF in (7) is a general form of the degree one CBF [54] ( $m = 1$ ) and exponential CBF in [57]. The following theorem on HOCBFs implies the forward invariance property of the CBFs and the original safety set. The proof is omitted (see [55] for the proof).

**Definition 3.7** (Control Lyapunov function (CLF)[54]). A continuously differentiable function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  is a globally and exponentially stabilizing CLF for (3) if there exists constants  $c_i \in \mathbb{R}_{>0}$ ,  $i = 1, 2$ , such that  $c_1 \|\mathbf{s}\|^2 \leq V(\mathbf{s}) \leq c_2 \|\mathbf{s}\|^2$ , and the following inequality holds

$$\inf_{\mathbf{a} \in \mathcal{U}} [L_f V(\mathbf{s}) + L_g V(\mathbf{s}) \mathbf{a} + \eta(\mathbf{s})] \leq e, \quad (8)$$

where  $e$  makes this a soft constraint. If the dynamics are of the form:  $\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{a})$ , we can express it in the form:

$$\dot{\mathbf{s}}' = f'(\mathbf{s}', \mathbf{a}) + g'(\mathbf{s}') \mathbf{a}' \quad (9)$$

where  $\mathbf{s}' = \begin{bmatrix} \mathbf{s} \\ \mathbf{a} \end{bmatrix}$  and  $\mathbf{a}'$  are the augmented system states and new primitive actions of the system respectively,  $f' = \begin{bmatrix} f(\mathbf{s}, \mathbf{a}) \\ \mathbf{0}_{\dim(\mathbf{a})} \end{bmatrix}$ , and  $g' = \begin{bmatrix} \mathbf{0}_{\dim(\mathbf{s}) \times \dim(\mathbf{a}')} \\ \mathbf{B} \end{bmatrix}$ , and  $\mathbf{B} \in \mathbb{R}^{\dim(\mathbf{a}) \times \dim(\mathbf{a}')}$ . Generally,  $\mathbf{B}$  contains ones in its diagonal entries. The redefined dynamics can be used subsequently for defining CBFs.

## 4 Problem Formulation

We consider a cooperative multi-agent setting among a set of agents. Let the set of agents be denoted by  $\mathcal{N} = \{1, \dots, N\}$ . We consider that every agent  $i \in \mathcal{N}$  has dynamics as in (3). We partition the state space into two disjoint spaces corresponding to the agent states and the environment states. The environment state is denoted by  $s^e \in \mathcal{S}^e$ . The state of an agent  $i$  is denoted by  $s^i \in \mathcal{S}^i \subset \mathbb{R}^n$  and the control input vector is denoted by  $\mathbf{a}^i \in \mathcal{A}^i = [\mathbf{a}_{\min}, \mathbf{a}_{\max}] \subset \mathbb{R}^q$  with  $\mathcal{A}^i$  the control input constraint set for the agent  $i$ , and  $\mathbf{a}_{\min}, \mathbf{a}_{\max} \in \mathbb{R}^q$ . Then, the joint state space is denoted by  $\mathcal{S} = \prod_{i \in \mathcal{N}} \mathcal{S}^i \times \mathcal{S}^e$ , and the joint action space of the agents is denoted by  $\mathcal{A} = \prod_{i \in \mathcal{N}} \mathcal{A}^i$ . In our setting, there is a performance related cost function  $l(\mathbf{s}, \mathbf{a})$  associated with the system specification, where  $\mathbf{s} \in \mathcal{S}$  (more specifically  $\prod_{i \in \mathcal{N}} \mathcal{S}^i$ ) and  $\mathbf{a} \in \mathcal{A}$ . Without loss of generality, in our cooperative setting, the stage cost is defined in the form  $l(\mathbf{s}, \mathbf{a}) = \sum_{i \in \mathcal{N}} l^i(s^i, \mathbf{a}^i)$ , where  $l^i(s^i, \mathbf{a}^i)$  denotes the stage cost associated with agent  $i \in \mathcal{N}$ .

We consider a partially observable setting whereby the observation space of agent  $i$  is denoted as  $\mathcal{O}^i \subseteq \mathcal{S}$ , and the joint observation space is  $\mathcal{O} = \prod_{i \in \mathcal{N}} \mathcal{O}^i$ . Each agent  $i$  receives an observation  $o^i \in \mathcal{O}^i$ , which includes its own state  $s^i$  together with those portions of other agents' or environmental states that are observable to it. Associated with every agent  $i$ , we define a set of safety-related functions

$$\{b_j^i(o^i) : j \in \mathcal{N}^i(\mathbf{s}), i \in \mathcal{N}\}, \quad b_j^i : \mathcal{O}^i \rightarrow \mathbb{R},$$

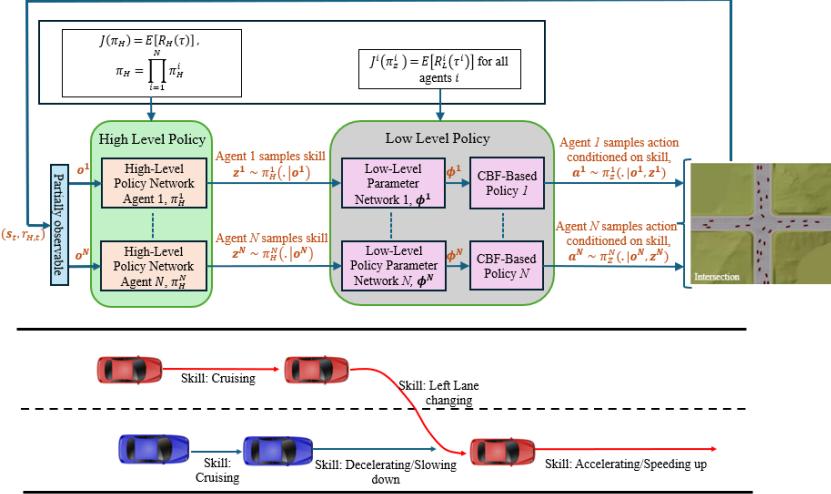


Figure 1: At the higher level, agents choose skills in a decentralized manner based on  $\pi_H$  conditioned on their own observation, whereas at the lower level every agent  $i$  executes these skills by learning a parametric CBF-based policy  $\pi_{z^i}$  conditioned on their observation and the skill. The extrinsic trajectory return  $R_H(\tau)$  is used to jointly learn the cooperative policy for all the agents centrally, and the intrinsic trajectory return  $J^i(\pi_L^i)$  for agent  $i$ 's trajectory  $\tau^i$  is used to learn the policies corresponding to the skills. The high level and low level policy networks are shared among all agents.

where  $b^{i,j}$  encodes the safety condition between agent  $i$  and another agent or environmental entity  $j$  that is included in  $o^i$ . Equivalently, one may write  $b_j^i(s^i, s^j)$  to emphasize the dependence on the agent's own state and the observed state of entity  $j$ . We assume that the constraints are initially satisfied upon their introduction. Thus, our safety-constrained problem is expressed as an Stochastic Optimal Control Problem (SOCP):

$$\min_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t l(s_t, a_t) \right] \quad \text{s.t.} \quad b_j^i(o^i) \geq 0, \quad \forall i \in \mathcal{N}, j \in \mathcal{N}^i(s_t), \forall t \geq 0 \quad (10)$$

where,  $\pi(\cdot | s) \in \Pi$  is the joint state feedback policy of the agents i.e.  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ ,  $\Delta$  is the space of distributions over the support of the random variable. It is noteworthy that this problem requires that the constraints are satisfied pointwise in time for every trajectory generated by  $\pi$ .

**Cooperative setting with pointwise-in-time hard constraints vs. CMDP setting:** As shown in [4], CMDPs address safety in fully cooperative multi-agent settings where agents share a common reward and seek to minimize the expected cumulative cost:

$$\min_{\pi} \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t l(s_t, a_t) \right] \quad \text{s.t.} \quad \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t b_j^i(s_t, a_t^i) \right] \leq \alpha_j^i; \forall j \quad (11)$$

Here,  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$  denotes a state-feedback policy, and  $\tau$  is a trajectory generated by  $\pi$ . While structurally similar to pointwise-in-time constraints in the single-agent setting [40], the multi-agent CMDP formulation requires policies to depend on the joint state  $s_t$ , making decentralized execution infeasible in partially observable environments and linearly increasing constraint complexity with the number of agents. Our formulation overcomes these limitations by enabling decentralized safety guarantees in partially observable, safety-critical multi-agent systems.

## 5 Safe Hierarchical Reinforcement Learning (HMARL-CBF)

We propose a novel Safe Hierarchical Multi-Agent Reinforcement Learning (HMARL) approach illustrated in figure 1 to tackle the aforementioned problem in (10). Specifically, we decompose the problem in (10) into a bi-level RL problem. The higher-level problem objective concerns the optimization of the joint performance of the agents. The lower-level optimization problem addresses

the safety of the agents defined using the second term in the stage cost in (14). Our proposed HMARL method is based on the idea of skills which is defined as follows.

**Safe agent skills.** Inspired by [58, 59], we define a safety-constrained skill  $z^i$  for an agent  $i$  as a seven-tuple:  $(\mathcal{I}_z^i, \mathcal{J}_z^i(\mathbf{o}_{z,init}^i, s_{init}^e), T_{max}, \Phi_z^i, \mathcal{N}^i(\mathbf{o}^i), r_z^i(s^i, \mathbf{a}^i), \pi_z^i)$  where  $\mathcal{I}_z^i \subset \mathcal{S}^i$  is the initiation set for the skill respectively,  $\mathbf{o}_{z,init}^i \in \mathcal{I}_z^i$  and  $s_{init}^e \in \mathcal{S}^e$  are the observation of agent  $i$  and environment state at the time of initiation of the skill,  $\mathcal{J}_z^i(\mathbf{o}_{init}^i, s_{init}^e) \subseteq \mathcal{O}^i$  is the termination set of the skill, and  $\Phi_z^i : \mathcal{O}^i \times \mathbb{N} \rightarrow \{0, 1\}$  is the termination function as defined below:

$$\Phi_z^i(\mathbf{o}^i, t) = \begin{cases} 1, & \text{if } \mathbf{o}^i \in \mathcal{J}_z^i(\mathbf{o}_{z,init}^i, s_{init}^e) \text{ or, } t \geq T_{max} \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

A skill is terminated as a function of the initiation state of the skill  $s_z^i$  in one of the two ways: i. either upon observing that the skill has been executed or, ii. interruption due to a timeout for exceeding the maximum allowed duration of execution,  $T_{max} \in \mathbb{N}$ .  $\mathcal{N}^i(\mathbf{o}^i)$  is the set of safety constraints for agent  $i$  that has to be satisfied during the execution of the skill. We define a reward function  $r_z^i(s^i, \mathbf{a}^i)$  corresponding to action  $\mathbf{a}^i$  in state  $s^i$  which is used to learn the skill. Finally,  $\pi_z^i : \mathcal{O}^i \times \mathcal{J}_z^i \rightarrow \Delta(\mathcal{A}^i)$  is the safe skill policy expressed as a SOCP as follows:

$$\begin{aligned} \max_{\pi_z^i, k} & \mathbb{E}_{\mathbf{s}_t^i, \mathbf{a}_t^i \sim \pi_z^i, k} \left[ \sum_{t=0}^k r_z^i(s_t^i, \mathbf{a}_t^i) | s_0^i = s_{init}^i \right] \\ \text{subject to} & b_i^j(\mathbf{o}_t^i) \geq 0, s_k^i \in \mathcal{J}_z^i, k \leq T_{max}, \forall b_i^j \in \mathcal{N}(\mathbf{o}_t^i), \quad \forall t = 0, \dots, k; \end{aligned} \quad (13)$$

The set of skills for an agent  $i$  is defined as  $\mathcal{Z}^i$ , and the joint set of skills of the  $N$  agents is defined as  $\mathcal{Z} = \cup_{i \in \mathcal{N}} \mathcal{Z}^i$ . We define a high-level policy for the  $N$  agents over their set of skills denoted by  $\pi_H : \mathcal{O} \rightarrow \Delta(\mathcal{Z})$  where  $\pi_H \in \Pi_H$ , which becomes the high-level RL problem. Subsequently, we define  $\pi_z : \mathcal{O} \times \mathcal{J}_z \rightarrow \Delta(\mathcal{A})$  as the joint policy for the joint skill  $z \in \mathcal{Z}$  of the  $N$  agents, where  $\mathcal{J}_z = \cup_{i \in \mathcal{N}} \mathcal{J}_z^i$ . Learning the safe skills of the agents becomes our low-level RL problem. Next, we detail our proposed HMARL-CBF method.

## 5.1 Multi-Agent High-Level Policy Learning

As mentioned above, the high-level policy is formulated as a joint centralized optimization problem involving all the agents over their set of skills. The optimization problem can be solved using existing CTDE schemes, either using policy gradients or value decomposition. The higher level problem can be modeled as an MSMDP with the corresponding extrinsic reward for state  $s$  and skill  $z$  executed in  $k$  steps using policy  $\pi_z$  is defined as:  $R(s, z, k) = \mathbb{E}_{\mathbf{a}_t \sim \pi_z} [\sum_{t=0}^{k-1} \gamma^t r_H(s_t, \mathbf{a}_t) | s_0 = s]$ . We call this extrinsic as it is task-specific and, hence, chosen to be the environment reward. This is revised by incorporating safety corresponding to (10) rendering the following extrinsic reward:

$$r_H(s, \mathbf{a}) = - \left[ l(s, \mathbf{a}) - \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}^i(s)} p_i^j \mathbb{I}(b_j^i(s^i, s^j)) b(s^i, s^j) \right] \quad (14)$$

where  $s \in \mathcal{S}$ ,  $\mathbf{a} \in \mathcal{A}$ ,  $p_i^j \in \mathbb{R}_{>0}$  and  $\mathbb{I}$  is an indicator function defined as follows.

$$\mathbb{I}(x) = \begin{cases} 1, & \text{if } x < 0 \text{ for some } i, \\ 0, & \text{otherwise.} \end{cases}$$

The minimization problem is thus mapped to a maximization problem of accumulated reward. Then, the problem in (10) can be mapped to the following problem:

$$\begin{aligned} J(\pi_H, \pi_z) &= \mathbb{E}_{\tau \sim (\pi_H \circ \pi_z)} [R_H(\tau)] = \mathbb{E}_{\tau \sim (\pi_H \circ \pi_z)} \left[ \sum_{t'=0}^{\infty} \gamma^{k_{t'}} R(s_{t'}, z_{t'}, k_{t'}) | z_{t'} \sim \pi_H \right] \\ &= \mathbb{E} \left[ \sum_{t'=0}^{\infty} \mathbb{E}_{s_t, \mathbf{a}_t \sim \pi_{z_{t'}}, k_{t'}} \left[ \sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} r_H(s_t, \mathbf{a}_t) \right] \right] \\ \pi_H^* &= \arg \max_{\pi_H \in \Pi_H} J(\pi_H, \pi_z) \end{aligned} \quad (15)$$

where  $\Pi_H$  is the space of the high-level policies and  $\tau$  is a trajectory sampled from the joint composition of  $\pi_H$  and  $\pi_z$ . The policy  $\pi_H^*$  given a joint skills policy of the agents  $\pi_z$  optimizes the trajectories with respect to the reward in (14). This problem can be solved using any existing on-policy or off-policy MARL algorithm [60, 61, 62]. If we parameterize the policy with  $\theta$ , the policy gradient of the parametrized policy  $\pi_{H_\theta}$ , as derived in [63] is given by:

$$\nabla_\theta J(\pi_{H_\theta}, \pi_z) = \mathbb{E}_{\tau \sim (\pi_{H_\theta} \circ \pi_z)} \left[ \sum_{t=0}^{\infty} \nabla_\theta \log \pi_{H_\theta}(z_t | s_t) \left( \sum_{t'=0}^{\infty} \gamma^{t'} R(s_{t'}, z_{t'}, k_{t'}) \right) \right] \quad (16)$$

Under a CTDE scheme, in order to execute the policies de-centrally, the agent policies/actor network (parameters) can be learned individually agent-wise conditioned on their observation (and history  $h_{t-1}^i$ ) i.e.,  $\pi_{H_\theta}^i(o_t^i, h_{t-1}^i)$ , using a centralized critic network. Alternatively, the policy  $\pi_{H_\theta}$  can be learnt by minimizing the following (Q-learning) loss using a parametrization  $Q_\theta$ :

$$\mathcal{L}(\theta) = \mathbb{E}_{s_{t'}, z_{t'}, s_{t'+1}, k_{t'}} [(R(s_{t'}, z_{t'}, k_{t'}) + \gamma^{k_{t'}} \max_{z \in \Pi_z} Q_\theta^{tot}(s_{t'+1}, z_{t'+1}) - Q_\theta^{tot}(s_{t'}, z_{t'}))^2] \quad (17)$$

where  $Q_\theta^{tot}$  is the total state-action value as defined in (2) which can be learned using value decomposition based on Individual Global Max principle proposed in [64, 65].

## 5.2 CBF-Based Individual Safe Skill Learning

The low-level policy is associated with execution of the skills selected by the agents based on the high-level policy. We define an intrinsic step reward conditioned on any skill  $z^i$  of any agent  $i$  as  $r_L^i(s^i, a^i | z^i)$  to incorporate intrinsic motivation. Notice, the safety constraints are included in the extrinsic step reward  $r_H$ . Additionally, to ensure safe execution of the skill, we formulate the low-level policy as a Constrained Optimal Control Problem (COPC) as follows:

$$\begin{aligned} & \max_{\pi_{z_{t'}}^i \in \Pi_{z_{t'}}^i, k_{t'}} \mathbb{E}_{\pi_{z_{t'}}^i, s_{t'}, \dots, k_{t'}, k_{t'}} \left[ \sum_{t=t'}^{k_{t'}} r_L^i(s_t^i, a_t^i | z_{t'}^i) \right] \\ & \text{subject to } b_i^j(s_t^i, s_{t'}^j) \geq 0 \quad \forall s_t^i, s_{t'}^j, j \in \mathcal{N}^i(s_t), t \in [t', \dots, k_{t'}] \\ & s_{t'}^i \in \mathcal{S}^i, s_{k_{t'}} \in \mathcal{J}_{z_{t'}}^i. \end{aligned} \quad (18)$$

where  $t'$  is the time the skill was initiated by agent  $i$ . In order to solve the problem we formulate a *parameterized* constrained optimization problem based on CBFs at each time step  $t$  that gives us the deterministic policy as a function of the observation conditioned on the skill  $\mu^i(o_t^i | z_t^i; \phi^i)$ . We include control Lyapunov functions (CLFs)  $V_i^j$  as functions of the state  $s_t^i$  and the terminal state  $s_{k_{t'}}^i \in \mathcal{J}_{z_{t'}}^i$  to assist in learning the optimal trajectory corresponding to the skill and terminate by converging to the termination set with  $T_{\max}$ . This is similar to stabilizing the system at the terminal state. Note that  $s_{k_{t'}}$  need not be a terminal state, but any state-related Lyapunov function that aids in learning the skill. This results in an optimization problem with a quadratic cost function and linear constraint which makes it a Quadratic Program (QP) that requires low compute. The optimization problem corresponding to the deterministic policy  $\mu^i(o_t^i | z_t^i; \phi^i)$  is given below.

$$\min_{a_t^i \in \mathcal{A}_t^i, e} a_t^{i\top} H(\phi_H^i) a_t^i + F^\top(\phi_F^i) a_t^i + \phi_e^\top e \quad (19)$$

subject to

$$\begin{aligned} L_f^m b_i^j(s_t^i, s_{t'}^j) + L_g L_f^{m-1} b_i^j(s_t^i, s_{t'}^j) a_t^i \Omega(b_i^j(s_t^i, s_{t'}^j); \phi_b^{i,j}) + \alpha_m(\zeta_{m-1}(s_t^i, s_{t'}^j); \phi_b^{i,j}) & \geq 0; \quad \forall j \in \mathcal{N}^i(s_t) \\ L_f V_i^j(s_t^i, s_{k_{t'}}^i) + L_g V_i^j(s_t^i, s_{k_{t'}}^i) a_t^i + \eta(s_t^i, s_{k_{t'}}^i; \phi_v^{i,j}) & \leq e^{i,j}; \quad \forall j = 1, 2, \dots \end{aligned}$$

where  $\phi^i = [\phi_H^i, \phi_F^i, \phi_b^{i,1}, \dots, \phi_b^{i,|\mathcal{N}^i(s_t)|}, \phi_v^{i,1}, \phi_v^{i,2}, \dots]$  are the parameters of the optimization problem such that each term in the vector  $\phi^i$  belongs to  $\mathbb{R}_{>0}$ ,  $H \in \mathbb{R}^{\dim(\mathcal{A}_i) \times \dim(\mathcal{A}_i)}$ ,  $F \in \mathbb{R}^{\dim(\mathcal{A}_i)}$  and  $e$  is the slack term from the CLF constraints. The chosen quadratic objective, combined with the affine CBF constraints, forms a Quadratic Program (QP) at each time step  $t$ , which is well suited for real-time safety-critical control applications.

In our case,  $H$  is chosen to be positive definite as the objective generally includes tracking error/control effort minimization, which makes the problem strictly convex. The parameters  $\phi^i$  are learned as

a function of the agent  $i$ 's observation  $\mathbf{o}^i$  and parameterized by  $\nu^i$ :  $\phi^i = \Gamma_{\nu^i}(\mathbf{o}^i | z^i)$ , where  $\Gamma$  is differentiable w.r.t to  $\nu^i$ . The function can be a deterministic or stochastic function of a continuous random variable  $\epsilon$  using the reparameterization trick in [66]. With the reparameterization trick, the control policy becomes stochastic, thus allowing for use of stochastic policy gradient algorithms.

An agent's trajectory  $\tau^i = \{\mathbf{s}_0^i, \mathbf{a}_0^i, \mathbf{s}_1^i, \mathbf{a}_1^i, \mathbf{s}_2^i, \mathbf{a}_2^i, \dots\} = \{\mathbf{s}_0^i, \mathbf{z}_0^i, \mathbf{s}_{k_0}^i, \mathbf{z}_{k_0}^i, \mathbf{s}_{k_1}^i, \mathbf{z}_{k_1}^i, \dots\}$  can be represented as a sequence of states and actions indexed by time as well as states and skills indexed by time on a different scale. Additionally, the forward invariant property of CBFs can guarantee that the trajectory resides in the safe set. In order to jointly learn with the high-level policy from the sampled trajectories, we map the safe skill learning problem for any agent  $i$  in (18) as a trajectory optimization problem with the policy  $\pi_{z_\nu}^i$  given by the following objective:

$$J^i(\pi_H, \pi_z^{-i}, \pi_{z_\nu}^i) = \mathbb{E}_{\tau^i \sim (\pi_H \circ (\pi_z^{-i}, \pi_{z_\nu}^i))}[R_L^i(\tau^i)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_L^i(\mathbf{s}_t^i, \mathbf{a}_t^i) | \mathbf{z}_t^i \sim \pi_H^i, \mathbf{a}_t^i \sim \pi_{z_t \nu}^i\right] \quad (20)$$

We drop the policy parameters  $\phi$  and superscript  $i$  from  $\nu$  to avoid overloading the symbols. Here,  $\pi_z^{-i}$  is the joint skills policy of the other agents except agent  $i$ . we solve the optimization problem in (20) by learning the parameters of the QP in (19) that maximizes the total reward. The problem in (20) can be solved using policy gradient (PG) methods as detailed in Appendix ??.

**Theorem 5.1.** *The satisfaction of the CBF constraints  $b_j^i$  in (19) guarantees safe execution of the skills, thus guaranteeing the safety of our proposed HMARL approach.*

*Proof.* Given, the constraints  $b_i^j$ 's are satisfied at the initial time, the satisfaction of the CBF constraints makes the corresponding safety sets forward invariant per Theorem (3.4), thus guaranteeing their satisfaction at all future times. This makes the policies corresponding to the skills of the agents safe, thus, guaranteeing the safety of our proposed HMARL approach.  $\square$

It should be noted that robust CBFs can be used in the presence of model uncertainty following [67] to achieve similar safety guarantees. Besides that, we detail how the high-level task specific reward can be incorporated to the low-level skill learning to align the low level learning to the task.

We follow [68] by applying the Karush-Kuhn-Tucker condition on the Lagrangian to derive the gradient of the state feedback policy with respect to its parameters. Specifically, let  $\lambda$  denote the dual variables on the HOCBF and CLF constraints, let  $D(\cdot)$  create a diagonal matrix from a vector, and let  $\mu^*, \lambda^*$  denote the optimal solutions of  $\mu$  and  $\lambda$ , respectively. We can then write  $d_\mu$  and  $d_\lambda$  in the form:

$$\begin{bmatrix} d_\mu \\ d_\lambda \end{bmatrix} = \begin{bmatrix} H & G^T D(\lambda^*) \\ G & D(G\mathbf{a}^* - h) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}_{\dim(\mathcal{A}_i)} \\ 0 \end{bmatrix} \quad (21)$$

where  $G, h$  are concatenated by  $G_b^j, G_v^j, h_b^j, h_v^j$ ,

$$\begin{aligned} G_b^j &= -L_g L_f^{m-1} b_i^j(\mathbf{s}^i, \mathbf{s}^j); \quad j \in \mathcal{N}^i(\mathbf{s}) \\ G_v^j &= -L_g V_i^j(\mathbf{s}^i, \mathbf{s}^j; \phi_v^{i,j}); \quad j = 1, 2, \dots \\ h_b^j &= L_f^m b_i^j(\mathbf{s}^i, \mathbf{s}^j) + \Omega(b_i^j(\mathbf{s}^i, \mathbf{s}^j; \phi_b^{i,j})) + \alpha_m(\zeta_{m-1}(\mathbf{s}^i, \mathbf{s}^j; \phi_b^{i,j})); \quad j \in \mathcal{N}^i(\mathbf{s}) \\ h_v^j &= L_f V_i^j(\mathbf{s}^i, \mathbf{s}^j) + \eta(\mathbf{s}^i, \mathbf{s}^j; \phi_v^{i,j}); \quad j = 1, 2, \dots \end{aligned} \quad (22)$$

As the control bounds in (19) are not trainable, they are not included in  $G$  and  $h$ . Then, the relevant gradients with respect to all the parameters are given by

$$\nabla_H \mu = \frac{1}{2} (d_u \mathbf{u}^T + \mathbf{u} d_u^T), \quad \nabla_F \mu = d_u \quad (23)$$

$$\nabla_G \mu = D(\lambda^*) (d_\lambda \mathbf{u}^T + \lambda d_u^T), \quad \nabla_h \mu = -D(\lambda^*) d_\lambda \quad (24)$$

With the hierarchical composition of the policies  $(\pi_H \circ \pi_z)$ , we define the augmented state  $\hat{\mathbf{s}} = [\mathbf{s}^i, \mathbf{s}^{-i}, \mathbf{z}, \mathbf{s}_{z,init}]$  where  $\mathbf{s}^{-i}$  is the state of the agents other than agent  $i$  and  $\mathbf{s}_{z,init}$  is the joint state of the agents at the time of the initiation of the joint skill  $\mathbf{z}$ . The initiation state is included as the joint termination of the skill depends on the initiation state. The state and action value functions for agent

$i$  corresponding to reward  $r_L^i$ , skill policy  $\pi_{z^i}$  (given the high level policy  $\pi_H$  and the skill policy other agents  $\pi_{\mathbf{z}^{-i}}$ ) are defined as follows:

$$\begin{aligned} V^{\pi_{z^i}}(\hat{s}) &= \mathbb{E}[r_L^i(\hat{s}_0, \mathbf{a}_0^i | \hat{s}_0 = \hat{s}) + \mathbb{E}_{\hat{s}_1}[Q(\hat{s}_1, \mathbf{a}_1^i)]] \\ &= \mathbb{E}[r_L^i(\mathbf{s}_0^i, \mathbf{a}_0^i | \mathbf{s}_0^i = \mathbf{s}^i, z^i) + \mathbb{E}_{\hat{s}_1}[Q(\hat{s}_1, \mathbf{a}_1^i)]] \end{aligned} \quad (25)$$

The state-action value function for agent  $i$ , given the hierarchical policy composition  $\pi_H \circ \pi_{\mathbf{z}}$ , is defined as:

$$Q^{\pi_{z^i}}(\hat{s}, \mathbf{a}^i) = \mathbb{E}[r_L^i(\hat{s}, \mathbf{a}^i) + \gamma \mathbb{E}_{\hat{s}'}[V^{\pi_{z^i}}(\hat{s}')]] \quad (26)$$

The corresponding advantage function is given by:

$$A^{\pi_{z^i}}(\hat{s}, \mathbf{a}^i) = Q^{\pi_{z^i}}(\hat{s}, \mathbf{a}^i) - V^{\pi_{z^i}}(\hat{s}) \quad (27)$$

Based on the above definition of value functions and advantage function, as well the gradient of QP controller output with respect to its parameters, standard on-policy and off-policy algorithms can be used to derive the policy gradient to learn the controller parameters.

**High-level task specific reward to low level learning.** The low-level policies should be such that they optimize the entire trajectory given by the problem in (10). Hence, in addition to introducing intrinsic motivation through the low-level reward, inspired by [23], we introduce an advantage term to the low-level reward resulting in the following revised low-level reward:

$$\tilde{r}_L^i(\mathbf{s}_t^i, \mathbf{a}_t^i | z_{t'}^i) = \lambda \frac{A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'})}{\mathcal{N}} + (1 - \lambda)r_L^i(\mathbf{s}_t^i, \mathbf{a}_t^i | z_{t'}^i) \quad (28)$$

where  $\lambda \in [0, 1]$  and  $t' \leq t$  is the time when the joint skill at time  $t$  was initiated. The high-level advantage  $A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{a}_{t'})$  is defined as below:

$$A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) = Q^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) - V^{\pi_H}(\mathbf{s}_{t'}) \quad (29)$$

where  $Q^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'})$  and  $V^{\pi_H}(\mathbf{s}_{t'})$  are as defined in (2) and (1) respectively. The parameter  $\lambda$  can be used to balance between skill learning and optimization of the joint behavior of the agents. From [23] we have that for high level policies the following holds:

$$J(\tilde{\pi}_H) = J(\pi_H) + \mathbb{E}_{\mathbf{s}_{t'}, \mathbf{z}_{t'} \sim \pi_H} \left[ \sum_{t'=0} \gamma^{t'} A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) \right] \quad (30)$$

**Proposition 5.2.** Given the low level reward in (28), the revised low level learning objective  $\tilde{J}^i$  for agent  $i$  can be expressed as:

$$\tilde{J}^i(\pi_H, \pi_{\mathbf{z}^{-i}}, \pi_{z^i}) \approx \frac{\lambda}{\mathcal{N}(1-\gamma)} \mathbb{E} \left[ \sum_{t'=0} \gamma^{t'} A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) \right] + (1 - \lambda) J^i(\pi_H, \pi_{\mathbf{z}^{-i}}, \pi_{z^i}) \quad (31)$$

*Proof.* Based on 28, we can write our low level policy's learning objective for agent  $i$  as the following:

$$\begin{aligned} \tilde{J}^i(\pi_H, \pi_{\mathbf{z}^{-i}}, \pi_{z^i}) &= \mathbb{E} \left[ \sum_{t'=0} \mathbb{E}_{\mathbf{s}_t^i, \mathbf{a}_t^i \sim \pi_{z_t^i}} \left[ \sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} \tilde{r}_L^i(\mathbf{s}_t^i, \mathbf{a}_t^i | z_{t'}^i) \right] \right] \\ &= \mathbb{E} \left[ \sum_{t'=0} \mathbb{E}_{\mathbf{s}_t^i, \mathbf{a}_t^i \sim \pi_{z_t^i}} \left[ \sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} \left( \lambda \frac{A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'})}{\mathcal{N}} + (1 - \lambda)\tilde{r}_L^i(\mathbf{s}_t^i, \mathbf{a}_t^i | z_{t'}^i) \right) \right] \right] \\ &= \frac{\lambda}{\mathcal{N}} \mathbb{E} \left[ \sum_{t'=0} \frac{\gamma^{t'}(1 - \gamma^{k_{t'}})}{1 - \gamma} A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) \right] \\ &\quad + (1 - \lambda) \mathbb{E} \left[ \sum_{t'=0} \mathbb{E}_{\mathbf{s}_t^i, \mathbf{a}_t^i \sim \pi_{z_t^i}} \left[ \sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} r_{L_A}^i(\mathbf{s}_t^i, \mathbf{a}_t^i | z_{t'}^i) \right] \right] \\ &\approx \frac{\lambda}{\mathcal{N}(1-\gamma)} \mathbb{E} \left[ \sum_{t'=0} \gamma^{t'} A^{\pi_H}(\mathbf{s}_{t'}, \mathbf{z}_{t'}) \right] + (1 - \lambda) J^i(\pi_H, \pi_{\mathbf{z}^{-i}}, \pi_{z^i}) \end{aligned} \quad (32)$$

□

Similarly, the low level reward for every agent can be revised as above to align skill learning process with the task learning process, as the first term is related to the high level policy return.

## 6 Experiments

### 6.1 Environment Details

We validate our proposed method on traffic scenarios involving complex road networks with multiple agents that must ensure safety concerning neighboring agents while navigating toward their destinations. We use the MetaDrive simulator [69] to implement, validate, and compare our method against existing baselines in a cooperative setting, where agents (autonomous vehicles) collaborate to improve the overall traffic efficiency.

The simulator includes five environments for multi-agent experiments. We implement our algorithm in four of the originally proposed environments and design a new environment to simulate an on-ramp merging roadway, replacing the parking lot environment. Since safety is less challenging in parking lots due to low vehicle speeds, the problem can be formulated and solved as a static optimization problem with a fixed number of vehicles and parking spaces. The five environments from the METADRIVE simulator are illustrated in the figure 2. As mentioned these environments involve regions where agents can cross path with other agents, Hence, it is necessary for the agents to cooperate but at the safe time stay safe with each other to achieve the task i.e. reach destination successfully and timely. In all five environments used in our experiments, the vehicles are initialized at random spawn points, and assigned destinations to each of them randomly. The agents are terminated in three situations: reaching the destinations (deemed as successful), crashing with others or driving out of the road (deemed as failure). Once an existing vehicle is terminated new vehicles are spawned immediately to ensure continuous traffic flow. The terminated vehicles remains static for 10 steps (2s) in the original positions, creating impermeable obstacles. If a vehicle crashes into other terminated vehicles before it is removed from the environment, it is also considered as failure.

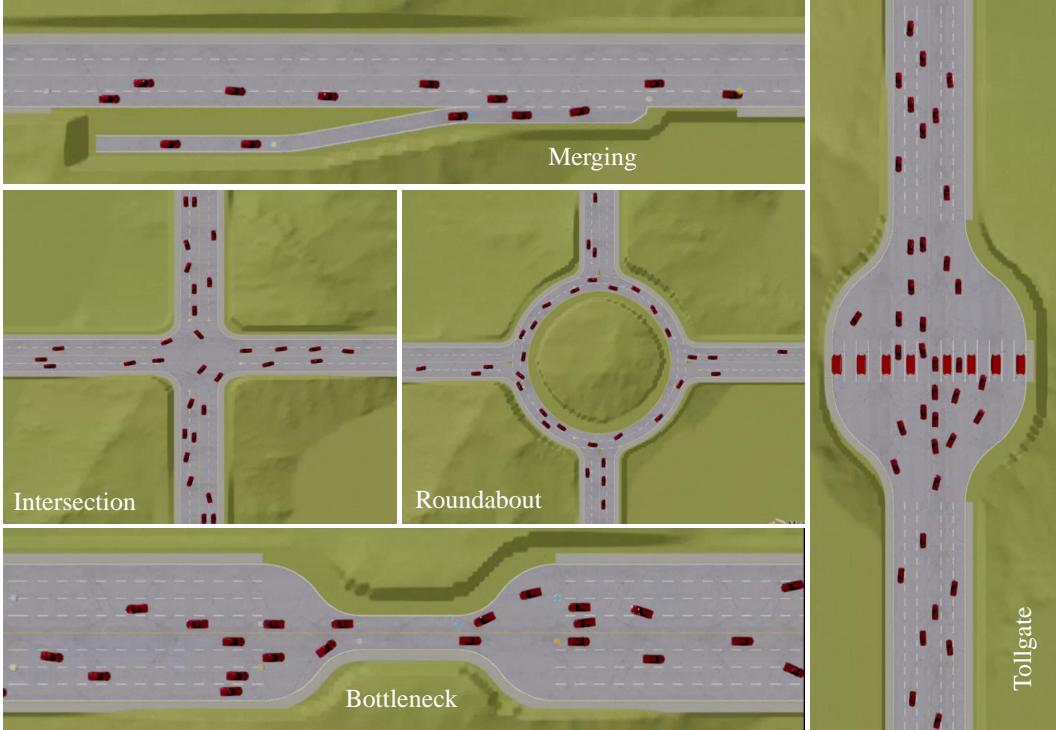


Figure 2: Illustration of the various METADRIVE environments.

In addition, we evaluate our method on a suite of lidar-based environments introduced in [51], as illustrated in Figure 3. All environments are partially observable, with agents equipped with lidar sensors. In the *Target* environments, each agent is assigned a fixed goal and must reach it while avoiding collisions with other agents and obstacles. The primary distinction between the *Target* and *Target-Bicycle* environments lies in the underlying agent dynamics; further details are provided in

Figure 3. In the *Spread* environment, agents must cooperate to cover all goal locations while avoiding both obstacles and one another.

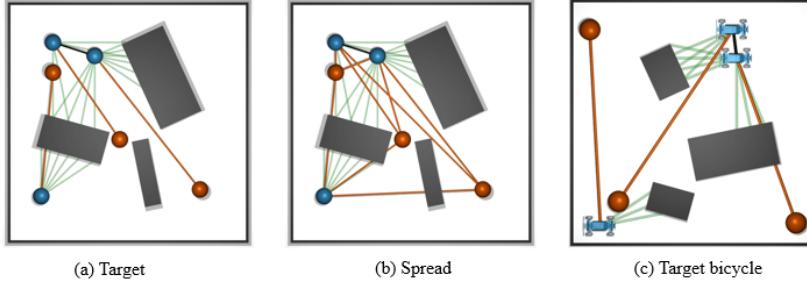


Figure 3: Illustration of the lidar based environments which include: (a) target environment, (b) spread environment and (c) target bicycle environment.

## 6.2 Experimental Details and Additional Results

We begin this section by presenting the details of the training, evaluation and ablation experiments. Besides that, we include results for the generalization experiments used to validate our proposed method.

## 6.3 Implementation Details

We have implemented our proposed method using both on-policy and off-policy algorithms. The on-policy implementation was done using MAPPO [61] with PPO [70] for the skills. The off-policy implementation used QMIX [64] for the higher level policy and SAC [71] for learning the agent skills. The extrinsic reward was chosen to be same as that defined in MetaDrive (which includes the energy, time, driving reward, lane deviation, collision penalty) of all the AVs in the environment. Besides that we include some regularization terms corresponding to our skills to avoid overfitting to any specific skill(s). The algorithms were implemented using RLLib; an open-source framework for ML. Further details about the implementation can be found in 6.2.

### 6.3.1 Algorithm details

Our method can be implemented using both existing on-policy and off-policy RL algorithms. The bilevel RL problems presented in (15) and (20), respectively, should be solved sequentially. However, as previous works illustrated [72], jointly optimizing both policies by updating their corresponding parameters alternatively renders similar performances, which is consistent with our observation from the experiments. Based on this observation, the algorithm is presented in Algorithm 1. The high-level policy  $\pi_{H_\theta}$  is decomposed into agent-level parameterized policies  $\pi_{H_{\theta^i}}^i$  to learn decentralized policies.

We implement the algorithm using two different approaches inspired by existing works in the area of MARL. Notably, we implemented the algorithm in the following ways. It is important to note that under all these approaches, the hierarchical policy is learned agent-wise to allow for decentralized execution.

1. Joint policy learning: This approach involves solving (15) by optimizing the high-level policies of the agents jointly by learning the policy parameters  $\theta^i$ 's.
2. Group policy learning: This approach is inspired by the idea of learning the value function by decomposing the joint value function into the value function of groups of agents. In our fully cooperative setting, the total step reward can be composed as a sum of the rewards of the individual agents, i.e.  $r_H(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^N r_H^i(\mathbf{s}^i, \mathbf{a}^i)$ . Let  $\{G^j\}_{j=1}^{N_G}$  represent the groups with each  $G^j \subset \mathcal{N}$  such that,  $G^j \cap G^{j'} = \emptyset$  and  $\mathcal{N} = \cup_{i=1}^{N_G} G^i$ . We can define the group reward as the sum over the individual reward of the agents in the group Then, under this approach we can express the objective in (15) as below (We only include the necessary symbols in the expectation to avoid

---

**Algorithm 1** Hierarchical Multi-Agent Reinforcement Learning with CBFs

---

```

1: Initialize randomly the high-level policy parameters  $\theta$  and parameters  $\nu$  of the low-level policy
   parameters  $\phi$  //network parameter sharing done between agents.
2: Set Grouping flag.
3: for each episode do
4:    $z_0^i \sim \pi_{H\theta}(z^i|o_0^i), \forall i \in \mathcal{N}$  // sample initial skill
5:   Set data buffer  $\mathcal{D} = \{\}$ 
6:   for  $t \leftarrow 0, \dots, T$  do
7:     Get action for agent  $i$  based on the selected skill  $a_t^i = \mu^i(o_t^i|z_t^i; \phi_\nu) \forall i \in \mathcal{N}$  // sample
       primitive action
8:     Get  $s_{t+1}$  and  $r_H$  from environment and  $r_L^i \forall i \in \mathcal{N}$ 
9:     Sample skill for agent  $i$ ,  $z_{t+1}^i \sim \pi_{H\theta}^i(z^i|o_t^i), \forall i \in \mathcal{N}_z(t)$  //  $\mathcal{N}_z(t)$  is the list of agents who
       have to update their skills based on termination scheme used.
10:    if ~Grouping then
11:       $\mathcal{D} = \mathcal{D} \cup \{s_t, \{o_t^i, z_t^i, a_t^i, \phi_t^i, r_L^i, o_{t+1}^i\}_{i \in \mathcal{N}}, r_H, s_{t+1}\}$ 
12:    else
13:       $\mathcal{D} = \mathcal{D} \cup \{s_t^{G_j}, \{o_t^i, z_t^i, a_t^i, \phi_t^i, r_L^i, o_{t+1}^i\}_{i \in G^j}, r_H^{G_j}, s_{t+1}^{G_j}\}_{\forall j \in \mathcal{N}_G}$ 
14:    end if
15:  end for
16:  Update high level policy network  $\theta$  according to (16), or, (17), using sampled data.
17:  Update low level policy network parameters  $\nu$  based on (23) and (24), using sampled data
18: end for

```

---

symbol overload):

$$\begin{aligned}
J(\pi_H, \pi_z) &= \mathbb{E} \left[ \sum_{t'=0}^{\infty} \mathbb{E}_{a_t \sim \pi_{z_{t'}}} \left[ \sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} r_H(s_t, a_t) \right] \right] \\
&= \mathbb{E} \left[ \sum_{t'=0}^{\infty} \mathbb{E}_{a_t \sim \pi_{z_{t'}}} \left[ \sum_{t=0}^{k_{t'}-1} \underbrace{\gamma^{t+k_{t'}} \sum_{j=1}^{N_G} \sum_{i=1}^N r_H^i(s_t^i, a_t^i) \mathbb{I}[i \in G^j]}_{=r_H^{G^j}(s_t^{G^j}, a_t^{G^j})} \right] \right] \\
&= \sum_{j=1}^{N_G} \mathbb{E} \left[ \sum_{t'=0}^{\infty} \mathbb{E}_{a_t^{G^j} \sim \pi_{z_{t'}}} \left[ \sum_{t=0}^{k_{t'}-1} r_H^{G^j}(s_t^{G^j}, a_t^{G^j}) \right] \right]
\end{aligned} \tag{33}$$

Here,  $s^{G^j}$  and  $a^{G^j}$  are the states and actions of the agents in group  $G^j$ . The agent skill policies are learnt independently of each other. Hence, the joint skills policies can be written as  $\pi_z = \prod_{i=1}^N \pi_{z^i} = \prod_{j=1}^{N_G} \prod_{i \in G^j} \pi_{z^i} = \prod_{j=1}^{N_G} \pi_{z^{G^j}}$  where  $\pi_{z^{G^j}}$  is the joint skills policy of the agents in the group  $G^j$ . Then, (33) can be written as:

$$\begin{aligned}
J(\pi_H, \pi_z) &= \sum_{j=1}^{N_G} \mathbb{E} \left[ \sum_{t'=0}^{\infty} \mathbb{E}_{a_t^{G^j} \sim \pi_{z_{t'}^{G^j}}} \left[ \sum_{t=0}^{k_{t'}-1} r_H^{G^j}(s_t^{G^j}, a_t^{G^j}) \right] \right] \\
&= J^1(\pi_H^{G^1}) + \dots + J^{N_G}(\pi_H^{G^{N_G}})
\end{aligned}$$

where,  $k_{t'}$  now becomes the time at which any agent in the group changes selects/switches to a new skill. We drop the low-level policy corresponding to skills for brevity. This approach enables better scalability as the overall problem is decomposed into one that learns policies for the groups; however, this comes at the expense of performance as the original optimization problem involves all the agents.

Note that all these implementations retain the safety guarantee as it is offered by our safe skill-learning approach.

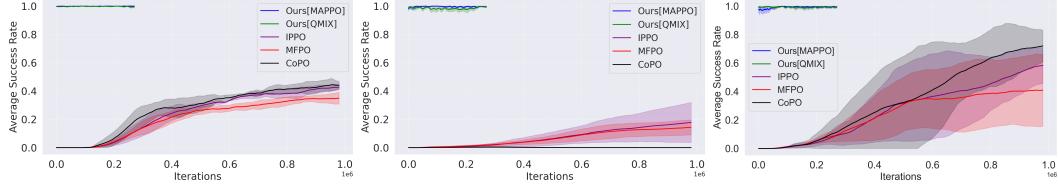


Figure 4: From left to right: success rate in merging, tollgate, and bottleneck environments.

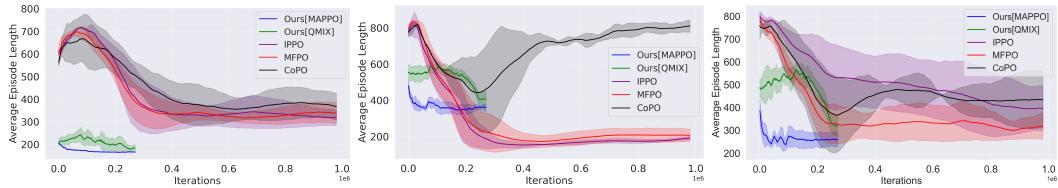


Figure 5: From left to right: average episode length in merging, tollgate, and bottleneck environments.

#### 6.4 Results and Baseline Comparison

We compare our algorithms against the state-of-the-art baselines in METADRIVE environment namely, CoPo [73], the independent policy optimization (IPPO) method [74] using PPO [70] as the individual learners, and Mean Field MARL with centralized critic [75] and form the mean field policy optimization (MFPO). The curriculum learning (CL) [76] is also a baseline, where the training was divided into 4 phases and gradually the number of initial agents in each episode from 25% to 100% of the target number. We ran our algorithm and the benchmark algorithms for 5 seeds over 300k and 1M iterations respectively. It is important to mention that the benchmark algorithms use 2000+ hours of individual driving experience in addition to the training iterations to achieve the illustrated. *On the other hand, we jointly optimize both the RL policies from same data, thus not introducing any additional sample complexity due to the hierachal decomposition.* The training results include the success rate and average episode length to illustrate the efficacy and validate our proposed approach. Notably, the efficiency metric defined by the authors of CoPo [73]—calculated as the inverse of the average episode length divided by the difference between the number of successful and unsuccessful agents—further supports the validity of our chosen metrics. The *success rate* is the fraction of the successful agents to the total number of spawned agents.

We trained with 15 vehicles for merging, 30 vehicles for intersection and roundabout, 25 vehicles for bottleneck and 40 vehicles for tollgate respectively. As mentioned previously, the extrinsic reward corresponding to (14) was set to the environment reward, besides the regularization terms. The intrinsic reward is described in subsection 6.6. We present the results of training for the Merging, Bottleneck and Tollgate environments in Figures 4 and 5 respectively. The key observation is that our algorithm achieves almost 100 % success rate with an approximate model of the vehicle (see appendix 6.6) from the beginning of training, while also optimizing the joint policy evident from the average episode length. Although several baselines achieve better average episode length in the tollgate environment, the success rate is very low, implying most agents crashes out resulting in the remaining agents to finish faster, returning a lower average episode length. With the hierarchical decomposition, our algorithms converge faster ( $\leq 300k$ ) compared to the baselines which we run for 1M iterations, thus confirming the claim about expediting convergence.

We present the training results for the remaining three environments from figure 5 i.e. merging, roundabout and intersection in figure 6. As can be noticed, our algorithm achieves a near perfect safety rate with both on-policy and off-policy algorithms during training, and, keeping consistent with the tollgate and bottleneck environments are trained for 300k iterations. However, in order to encourage exploration in roundabout and intersection environments for learning the cooperative behavior, we use curriculum learning by varying the arrival rates, while keeping the traffic density constant. The curriculum transitions happened at 30k, 50k, 80k and 125k iterations during training.

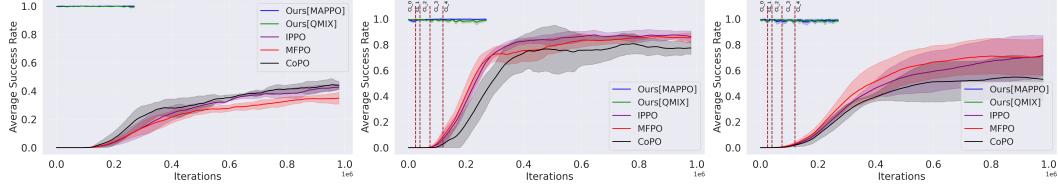


Figure 6: Plot of the success rate of the agents with our algorithm vs the baselines in merging, roundabout, and intersection environments respectively. CL refers to curriculum learning, four different curriculum were used with varying traffic densities. The benchmark methods use 2000+ hours of individual driving data prior to the training.

#### 6.4.1 Evaluation Setup and Results

**METADRIVE Simulator.** Next, we evaluate the efficacy of our algorithm in optimizing the task objective against the baseline algorithms. We use the success-weighted metrics for evaluation and comparison with the baseline, ablation, and generalization experiments. The reason for weighting the metrics by success rate is, because, poorly performing algorithms with very low success rate (agents crashing and getting removed from the episode rapidly) can achieve superior average energy and time which is debunked through this. Besides *success rate*, we use *success weighted average energy* and *success weighted average time* which are the average energy and episode length of all agents during an episode divided by the success rate. The evaluation results are presented in table 1.

Table 1: Summary of results for Merging, Intersection, Roundabout, Bottleneck, and Tollgate environments.  $\uparrow$ : higher is better;  $\downarrow$ : lower is better. **Bold**: the best performance for each metric.

Environments	Metrics	IPPO	CL	MFPO	CoPo	HMARL-CBF (on-policy)	HMARL-CBF (on-policy w/ Grouping)	HMARL-CBF (off-policy)	HMARL-CBF (off-policy w/ Grouping)
Merging	Success $\uparrow$	0.43 $\pm$ 0.02	0.32 $\pm$ 0.04	0.43 $\pm$ 0.09	0.48 $\pm$ 0.01	<b>0.99<math>\pm</math>0.01</b>	<b>0.99<math>\pm</math>0.00</b>	<b>0.99<math>\pm</math>0.01</b>	<b>0.99<math>\pm</math>0.01</b>
	Time $\downarrow$	772.51 $\pm$ 112.00	1231.72 $\pm$ 44.53	762.32 $\pm$ 47.97	669.46 $\pm$ 72.08	<b>164.52<math>\pm</math>2.64</b>	187.91 $\pm$ 64.65	169.60 $\pm$ 7.61	166.70 $\pm$ 9.26
	Energy $\downarrow$	48.16 $\pm$ 1.23	31.31 $\pm$ 0.19	14.44 $\pm$ 2.51	13.81 $\pm$ 1.25	14.10 $\pm$ 0.23	<b>12.14<math>\pm</math>1.83</b>	14.73 $\pm$ 0.76	13.92 $\pm$ 0.09
Intersection	Success $\uparrow$	0.72 $\pm$ 0.13	0.62 $\pm$ 0.13	0.59 $\pm$ 0.16	0.73 $\pm$ 0.09	0.97 $\pm$ 0.04	0.93 $\pm$ 0.05	<b>0.96<math>\pm</math>0.03</b>	<b>0.96<math>\pm</math>0.01</b>
	Time $\downarrow$	599.16 $\pm$ 162.50	695.93 $\pm$ 155.30	725.39 $\pm$ 222.51	687.69 $\pm$ 141.90	317.41 $\pm$ 33.92	388.27 $\pm$ 64.92	<b>264.28<math>\pm</math>32.90</b>	280.11 $\pm$ 32.00
	Energy $\downarrow$	5.90 $\pm$ 0.62	6.64 $\pm$ 0.56	6.52 $\pm$ 0.27	5.68 $\pm$ 0.39	5.56 $\pm$ 0.04	<b>3.62<math>\pm</math>0.68</b>	5.78 $\pm$ 0.24	5.11 $\pm$ 0.20
Roundabout	Success $\uparrow$	0.85 $\pm$ 0.05	0.82 $\pm$ 0.04	0.85 $\pm$ 0.04	0.80 $\pm$ 0.03	<b>0.99<math>\pm</math>0.01</b>	<b>0.99<math>\pm</math>0.04</b>	0.98 $\pm$ 0.03	0.98 $\pm$ 0.02
	Time $\downarrow$	447.06 $\pm$ 6.41	332.89 $\pm$ 5.70	404.00 $\pm$ 17.88	515.05 $\pm$ 18.83	403.18 $\pm$ 21.66	375.67 $\pm$ 29.31	370.21 $\pm$ 67.04	<b>326.26<math>\pm</math>24.13</b>
	Energy $\downarrow$	11.74 $\pm$ 0.52	12.69 $\pm$ 0.18	12.07 $\pm$ 0.67	12.31 $\pm$ 0.26	9.89 $\pm$ 0.23	<b>7.75<math>\pm</math>0.39</b>	10.15 $\pm$ 0.82	11.41 $\pm$ 0.69
Bottleneck	Success $\uparrow$	0.09 $\pm$ 0.03	0.34 $\pm$ 0.15	0.48 $\pm$ 0.16	0.75 $\pm$ 0.07	<b>0.99<math>\pm</math>0.00</b>	<b>0.99<math>\pm</math>0.02</b>	<b>0.99<math>\pm</math>0.01</b>	<b>0.99<math>\pm</math>0.01</b>
	Time $\downarrow$	2644.44 $\pm$ 353.44	939.35 $\pm$ 147.21	679.83 $\pm$ 158.33	651.54 $\pm$ 63.52	241.16 $\pm$ 8.33	262.94 $\pm$ 59.10	313.80 $\pm$ 22.00	<b>240.47<math>\pm</math>28.52</b>
	Energy $\downarrow$	69.66 $\pm$ 9.77	11.61 $\pm$ 2.02	9.95 $\pm$ 1.39	6.93 $\pm$ 0.61	7.37 $\pm$ 0.00	<b>4.86<math>\pm</math>0.68</b>	7.29 $\pm$ 0.08	7.49 $\pm$ 0.03
Tollgate	Success $\uparrow$	0.08 $\pm$ 0.37	0.05 $\pm$ 0.02	0.11 $\pm$ 0.04	0.00 $\pm$ 0.00	<b>0.99<math>\pm</math>0.02</b>	0.98 $\pm$ 0.02	0.98 $\pm$ 0.01	<b>0.99<math>\pm</math>0.01</b>
	Time $\downarrow$	2809.04 $\pm$ 364.04	3211.06 $\pm$ 511.27	1884.46 $\pm$ 319.82	-	369.32 $\pm$ 5.82	427.02 $\pm$ 32.43	377.42 $\pm$ 9.99	<b>307.82<math>\pm</math>8.09</b>
	Energy $\downarrow$	74.64 $\pm$ 10.47	114.25 $\pm$ 27.02	53.63 $\pm$ 7.72	-	8.40 $\pm$ 0.05	<b>5.17<math>\pm</math>0.44</b>	8.93 $\pm$ 0.11	8.99 $\pm$ 0.01

As can be seen, our method achieves almost 100% success rate which is significantly better than the baselines. Also, we achieve better time and energy efficiency compared to the baselines across all environments as evident from the results. This goes to highlight that our methods learns a better cooperative policy in addition to enhancing the safety of the agents.

In addition to the presented results in table 1, we run the baseline algorithms by changing the penalty factors for the safety related terms (both inter-agent and environment safety) in the environment reward for the bottleneck environment. The results are presented in table 2. The primary takeaway is that the added emphasis on safety does not render better results for the baseline algorithms compared to our method. Besides that, different algorithms perform differently with increase in penalty factor, thus not exhibiting any trend between penalty and success rate.

**Lidar Environment Suite.** We compare our method with DGPO [51], InforMARL [77] and MAPPO-Lagrangian [78], three state of the art methods for multi-agent safe control. For InforMARL, we evaluate fixed and scheduled penalty variants. Further details about the baselines can be found in [51]. For MAPPO-Lagrangian, we test two settings for the Lagrange multiplier and one learning rate. Performance is better if the plot is located in top right corner. As can be seen, our method achieves a high success rate while also achieving high reward compared to the baselines in [51]. Finally, we present the results for the bicycle target environment in figure 8 which shows that our method scales with number of agents.

Table 2: Summary of baseline comparison using various penalty factors for Bottleneck environment.  
 $\uparrow$ : higher is better;  $\downarrow$ : lower is better. **Bold**: the best performance for each metric.

Penalty Factors	Metrics	IPPO	CL	MFPO	CoPo	HMARL-CBF (on-policy)
1	Success $\uparrow$	$0.09 \pm 0.03$	$0.34 \pm 0.15$	$0.48 \pm 0.16$	$0.75 \pm 0.07$	<b><math>0.99 \pm 0.00</math></b>
	Time $\downarrow$	$2644.44 \pm 353.44$	$939.35 \pm 147.21$	$679.83 \pm 158.33$	$651.54 \pm 63.52$	<b><math>241.16 \pm 8.33</math></b>
	Energy $\downarrow$	$69.66 \pm 9.77$	$11.61 \pm 2.02$	$9.95 \pm 1.39$	$6.93 \pm 0.61$	<b><math>7.37 \pm 0.00</math></b>
5	Success $\uparrow$	$0.58 \pm 0.11$	$0.395 \pm 0.01$	$0.615 \pm 0.08$	$0.716 \pm 0.02$	<b><math>0.99 \pm 0.00</math></b>
	Time $\downarrow$	$568 \pm 56$	$646.5 \pm 43.16$	$713 \pm 273.7$	$529.13 \pm 79$	<b><math>241.16 \pm 8.33</math></b>
	Energy $\downarrow$	$30 \pm 41.5$	$10.8 \pm 0.24$	$8.64 \pm 0.65$	$7.69 \pm 0.02$	<b><math>7.37 \pm 0.00</math></b>
10	Success $\uparrow$	$0.18 \pm 0.08$	$0.55 \pm 0.11$	$0.42 \pm 0.23$	$0.655 \pm 0.04$	<b><math>0.99 \pm 0.00</math></b>
	Time $\downarrow$	$1292.17 \pm 647$	$606.49 \pm 204.89$	$1414 \pm 973$	$514 \pm 126.92$	<b><math>241.16 \pm 8.33</math></b>
	Energy $\downarrow$	$40.97 \pm 27$	$9.14 \pm 0.93$	$22.12 \pm 22.08$	$8.37 \pm 0.04$	<b><math>7.37 \pm 0.00</math></b>

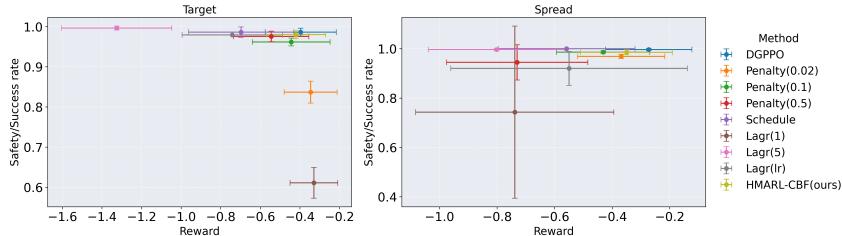


Figure 7: Success/safety rate vs reward for Target and Spread environment.

## 6.5 Ablation Results

In this subsection, we present ablation studies to further highlight the merit of our proposed approach using the HMARL-CBF on-policy implementation. The results are presented in Table 3. First, we remove the hierarchical structure to implement a flat MAPPO policy using various penalties on the safety constraints. However, a flat policy fails to learn in these environments. Next, we replace the cooperative high-level policy with a uniform random selection of skills across agents. This causes a reduction in success rates as well as performance across all environments. Subsequently, we investigate the necessity of pointwise-in-time constraints via random constraints dropout—masking road boundary constraints in the low-level policy with a Bernoulli distributed random variable (success Probability 0.5)—which uniformly reduces success rates and exhibits the largest performance decline in environments where agents cross paths across several zones in the environment (i.e. safety-critical). Next, we substitute the low-level skill policy with an parameterized LQR-based policy (removing the CBFs) within the hierarchical MARL framework. This causes a significant decline in success rate as well as in success-weighted time and energy efficiency. Finally, we examine fixed low-level policy parameterization by selecting parameters at the boundaries of the feasible space without prior knowledge, which further diminishes performance across all metrics across the environments.

We also perform generalization experiments to assess how well our policies to various environmental changes. The results for the experiments can be found in appendix 6.5.1.

### 6.5.1 Generalization Results

Besides the evaluation experiments, we conducted a series of experiments to assess the generalizability of our proposed approach. Firstly we systematically vary the lane widths in the environments. The corresponding success rates are reported in Table 4. Our analysis indicates that reducing lane width has a minimal impact on success rates in the roundabout and bottleneck environments. However, a more comparatively greater decline is observed in the intersection environment. This performance degradation is primarily attributed to the default intersection configuration (Figure 1), where two lanes are available in each direction. Agents are trained to execute lane changes from both lanes depending on their assigned routes. When lane width is reduced, agents attempting to perform lane changing from the inside lane at times gets stuck (and, eventually get eliminated from the episode

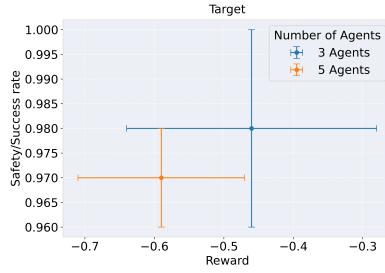


Figure 8: Success/safety rate vs reward for Bicycle Target environment.

Table 3: Summary of ablation results for all the environments (Pf is penalty factor).

Environments	Metrics	Flat Policy (Pf = 1)	Flat Policy (Pf = 5)	Flat Policy (Pf = 10)	Uniform high-level	Random Constraints Dropout	Hierarchical MARL with LQR	Fixed low level parameterization	<b>HMARL-CBF</b>
Merging	Success ↑	0	0	0	0.94 <sub>±0.09</sub>	0.91 <sub>±0.04</sub>	0	0.52 <sub>±0.06</sub>	<b>0.99<sub>±0.01</sub></b>
	Energy ↓	-	-	-	14.17 <sub>±0.49</sub>	15.30 <sub>±0.31</sub>	-	22.30 <sub>±0.12</sub>	<b>14.10<sub>±0.23</sub></b>
	Time ↓	-	-	-	211.70 <sub>±23.80</sub>	183.91 <sub>±1.49</sub>	-	-	<b>164.52<sub>±2.64</sub></b>
Intersection	Success ↑	0	0	0	0.86 <sub>±0.04</sub>	0.80 <sub>±0.04</sub>	0.14 <sub>±0.02</sub>	0	<b>0.97<sub>±0.04</sub></b>
	Energy ↓	-	-	-	5.75 <sub>±0.08</sub>	5.00 <sub>±0.11</sub>	50.00 <sub>±0.71</sub>	-	<b>5.56<sub>±0.04</sub></b>
	Time ↓	-	-	-	574.30 <sub>±60.60</sub>	320.50 <sub>±68.17</sub>	921.42 <sub>±40.71</sub>	-	<b>317.41<sub>±33.92</sub></b>
Roundabout	Success ↑	0	0	0	0.68 <sub>±0.08</sub>	0.60 <sub>±0.08</sub>	0	0	<b>0.99<sub>±0.01</sub></b>
	Energy ↓	-	-	-	13.23 <sub>±1.08</sub>	8.50 <sub>±0.74</sub>	-	-	<b>9.89<sub>±0.23</sub></b>
	Time ↓	-	-	-	907.35 <sub>±50.98</sub>	600.03 <sub>±34.67</sub>	-	-	<b>403.18<sub>±21.66</sub></b>
Bottleneck	Success ↑	0	0	0	0.65 <sub>±0.03</sub>	0.55 <sub>±0.09</sub>	0	0	<b>0.99<sub>±0.00</sub></b>
	Energy ↓	-	-	-	10.03 <sub>±0.10</sub>	12.03 <sub>±0.04</sub>	-	-	<b>7.37<sub>±0.00</sub></b>
	Time ↓	-	-	-	923.07 <sub>±61.73</sub>	965.45 <sub>±24.18</sub>	-	-	<b>241.16<sub>±8.33</sub></b>
Tollgate	Success ↑	0	0	0	0.91 <sub>±0.03</sub>	0.92 <sub>±0.03</sub>	0	0	<b>0.99<sub>±0.02</sub></b>
	Energy ↓	-	-	-	8.68 <sub>±0.03</sub>	9.35 <sub>±0.02</sub>	-	-	<b>8.40<sub>±0.05</sub></b>
	Time ↓	-	-	-	713.00 <sub>±24.15</sub>	423.40 <sub>±15.60</sub>	-	-	<b>369.32<sub>±5.82</sub></b>

due to timeout), or, hit road boundary which causes the drop in success rate. Conversely, increasing lane width generally maintains or enhances success rates across all environments.

Secondly, we investigated skill transferability by training low-level skills in an intersection environment and subsequently using them to train a high-level policy across multiple environments. Namely, we investigate the transferability in the merging and roundabout environments. This is because the observation is different and distinct for the Bottleneck and Tollgate environments compared to the other environments. The results are presented in table 5. This analysis reveals that skills learned in one environment can generalize effectively to others producing comparatively similar performance. In the roundabout environment, we observe a minor decline in the success rate and increase in the weighted average energy and time; yet our approach outperforms the baselines presented in table 3.

Finally, we test the generalizability of our method for varying traffic density/ number of agents. The results for this test are presented in tables 6 and 7 respectively. For every environment we trained for the default density as shown in the table and tested for lower and higher values. From the results, we can conclude that the success rate is unaffected by traffic density. The average time of the agents increases with increasing traffic density as expected. Finally, the energy values show a decreasing trend with increase in traffic density across most environments. This was anticipated because energy has a negative correlation with time.

### 6.5.2 Computer Details.

We conducted our experiments on a desktop equipped with a single NVIDIA RTX A5000 GPU and 32 CPU cores. Given that our environments involve a high number of agents (up to 45 agents in the tollgate environment), the primary computational cost is attributed to data collection. Training and convergence are achieved efficiently, as demonstrated in the training and evaluation results. Furthermore, we utilize shared neural network architectures across agents, which significantly reduces the overall computational cost and training time.

Table 4: Success rate across different environments for varying lane widths.

Lane Width (m)	Success Rate				
	Merging	Intersection	Roundabout	Bottleneck	Tollgate
3.25	0.99 <sub>±0.01</sub>	0.90 <sub>±0.03</sub>	0.98 <sub>±0.02</sub>	0.97 <sub>±0.02</sub>	0.99 <sub>±0.001</sub>
3.5 (default)	0.99 <sub>±0.01</sub>	0.97 <sub>±0.04</sub>	0.99 <sub>±0.01</sub>	0.99 <sub>±0.003</sub>	0.99 <sub>±0.02</sub>
4	0.99 <sub>±0.01</sub>	0.97 <sub>±0.015</sub>	0.99 <sub>±0.01</sub>	0.99 <sub>±0.001</sub>	0.99 <sub>±0.02</sub>

Table 5: Summary of skill transfer results (\* indicates runs done by transferring skills).

Experiments	Success	Energy	Time
Merging	0.99 <sub>±0.01</sub>	14.24 <sub>±0.23</sub>	166.18 <sub>±2.64</sub>
Merging*	0.96 <sub>±0.016</sub>	14.49 <sub>±0.58</sub>	178.34 <sub>±5.90</sub>
Roundabout	0.99 <sub>±0.01</sub>	9.89 <sub>±0.23</sub>	403.18 <sub>±21.66</sub>
Roundabout*	0.96 <sub>±0.016</sub>	11.39 <sub>±0.08</sub>	404.16 <sub>±18.6</sub>

Table 6: Summary of results for varying number of agents in Merging, Intersection, and Roundabout environments.

Metric	Merging			Intersection			Roundabout		
	10	15 (default)	20	20	30 (default)	40	20	30 (default)	40
Success	0.997 <sub>±0.0</sub>	0.99 <sub>±0.01</sub>	0.98 <sub>±0.02</sub>	0.955 <sub>±0.02</sub>	0.97 <sub>±0.04</sub>	0.94 <sub>±0.04</sub>	0.99 <sub>±0.01</sub>	0.99 <sub>±0.0</sub>	0.98 <sub>±0.0</sub>
Energy	14.03 <sub>±0.36</sub>	14.24 <sub>±0.23</sub>	14.47 <sub>±0.18</sub>	5.85 <sub>±0.07</sub>	5.82 <sub>±0.04</sub>	5.94 <sub>±0.05</sub>	10.48	10.32	9.95
Time	165.18 <sub>±2.15</sub>	166.18 <sub>±2.64</sub>	169.55 <sub>±2.15</sub>	318.54 <sub>±19</sub>	332.3 <sub>±36</sub>	361.8 <sub>±20.46</sub>	389.34	404.38	415.03

Table 7: Summary of results for varying number of agents in the Bottleneck and Tollgate environments.

Metric	Bottleneck			Tollgate		
	20	25 (default)	30	30	40 (default)	45
Success	0.99 <sub>±0.01</sub>	0.99 <sub>±0.003</sub>	0.978 <sub>±0.02</sub>	0.99 <sub>±0.02</sub>	0.99 <sub>±0.02</sub>	0.99 <sub>±0.01</sub>
Energy	7.47 <sub>±0.03</sub>	7.44 <sub>±0.001</sub>	7.43 <sub>±0.02</sub>	8.51 <sub>±0.02</sub>	8.51 <sub>±0.048</sub>	8.50 <sub>±0.02</sub>
Time	227.22 <sub>±7.77</sub>	243.59 <sub>±8.33</sub>	244.62 <sub>±6.62</sub>	371.53 <sub>±6.78</sub>	371.05 <sub>±5.82</sub>	374.02 <sub>±8.48</sub>

## 6.6 Algorithm Implementation

### 6.6.1 Implementation Details: METADRIVE Simulator

Metadrive environments illustrated in figure 2 by default, include 3 basic sensors: lidar, sideDetector and laneLineDetector which are used for detecting moving objects, sidewalks/solid lines, and broken/solid lines respectively. The lidar state observation returns a state vector containing necessary information for navigation tasks. The details of the observation can be found here <sup>1</sup>. The environmental safety constraints include i. the road boundaries and ii. the booth for tollgate environment only. And, each agent has to stay safe to all neighboring agents which are within the lidar sensing radius. It is important to mention that we only use a dynamic bicycle model for the CBF constraints, the environment uses pybullet physics engine for simulation. The kinematic bicycle model of the vehicle is provided in (34).

$$\begin{aligned} \dot{p} &= \frac{v \cos(\psi)}{1 - d\kappa(p)}, \\ \dot{d} &= v \sin(\psi), \\ \dot{\psi} &= \frac{v}{L} \tan(\delta) - \kappa(p)\dot{p}, \\ \dot{v} &= a. \end{aligned} \tag{34}$$

<sup>1</sup><https://metadrive-simulator.readthedocs.io/en/latest/obs.html>

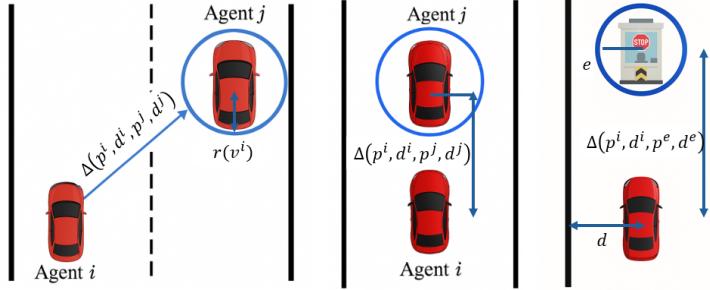


Figure 9: From left to right: illustration of i. inter agent safety constraints on another lane, ii. inter agent safety constraints on same lane and iii. environmental constraints.

where  $p$  represents the position of the vehicle along the reference path,  $d$  is the lateral deviation of the vehicle from the reference path,  $\psi$  denotes the yaw angle of the vehicle (heading angle relative to the path), and  $v$  is the longitudinal velocity of the vehicle. The control inputs are  $\delta$ , which is the steering angle, and  $a$ , which is the acceleration (or deceleration). The parameter  $L$  denotes the wheelbase of the vehicle, and  $\kappa(p)$  is the curvature of the reference path at position  $s$ . It is important to add that, these states are provided by the environment for each ego vehicle/agent in its observation.

In order to incorporate safety with respect to other agents and environmental obstacles, we define simple circles-based constraints as illustrated in figure 9. And, for the road boundaries we use the lateral deviation from the center lane to define safety functions. Let  $p^j$  and  $d^j$  denote the distance along the path and lateral deviation of another agent  $j$  with respect to agent  $i$ , located within within the sensing radius of agent  $i$ . This information is provided in the observation of agent  $i$  i.e.  $(d^i, p^j, d^j) \in \mathcal{O}^i$ . Note that, for an ego vehicle  $p^i$  is always 0 in its body frame. let  $\Delta(p^i, d^i, p^j, d^j)$  denote the euclidean distance between agent  $i$  and  $j$  respectively. Similarly, let  $p^e$  and  $d^e$  denote the longitudinal and lateral distance of an environmental obstacle, like a tollgate booth as illustrated in figure 9 from agent  $i$ , and  $\Delta(p^i, d^i, p^e, d^e)$  be the euclidean distance. Then, the safety constraints corresponding to other agent  $j$ , environmental constraints for an agent  $i$  are given below:

$$\Delta(p^i, d^i, p^j, d^j)^2 \geq r(v^i), \Delta(p^i, d^i, p^e, d^e)^2 \geq e, \text{ and } |d^i| \leq c \quad (35)$$

where  $r(v^i)$  is a monotonically increasing function of  $v^i$ ,  $e \in \mathbb{R}_{>0}$  and  $c \in \mathbb{R}_{>0}$ .

Consider the CBF for the constraint for inter-agent safety defined as  $h(\mathcal{O}^i) = \Delta(p^i, d^i, p^j, d^j) - r(v^i)$  where  $\Delta(p^i, d^i, p^j, d^j) = \sqrt{(p^i - p^j)^2 + (d^i - d^j)^2}$ , then the corresponding CBF constraint is given by:

$$\underbrace{2(p^i - p^j) \left( \frac{v^i \cos(\psi^i)}{1 - d^i \kappa(p^i)} - \dot{p}^j \right) + 2(d^i - d^j)(v^i \sin(\psi^i) - \dot{d}^j) + -2r(v^i) \frac{dr(v^i)}{dv^i} a^i}_{L_f h(\mathcal{O}^i)} \underbrace{+ \alpha ((p^i - p^j)^2 + (d^i - d^j)^2 - r(v^i)^2) \geq 0}_{h(\mathcal{O}^i)} \quad (36)$$

Additionally, we define CLFs for state reference tracking namely velocity and heading reference tracking. As we define termination state set corresponding to each skill, we define CLFs to drive the system to the termination state to be able to successfully execute the skill. However, it is important to note that the CLFs are not defined specific to skills and, hence, are generalized to the task. The CLFs corresponding to velocity and heading reference are given by the equation below. where  $v_t^i$  and  $v_{t_f}^i$  are the velocities at We parameterize the corresponding CLF constraint which in case of the velocity and heading constraints becomes the following:

$$V(v^i, v_{des}^i) = (v^i - v_{des}^i)^2; \quad (37)$$

$$V(\psi^i, \psi_{des}^i) = (\psi^i - \psi_{des}^i)^2 \quad (38)$$

where  $v_{des}$  is the desired velocity and  $\psi_{des}$  is the desired heading of the vehicle.

**Skills description.** We define five skills for each agent for the METADRIVE environments which are: i. cruising, ii. accelerating (speed up), iii. yielding (slowing down), iv. left lane changing, and v. right lane changing.

*Cruising.* As the name suggests, this skill causes the vehicle to move at a constant speed for a fixed distance. The velocity CLF constraint in (37) is incorporated in the skill QP based policy by setting  $v_{des} = 0$ .

*Accelerating/speeding up* This skill corresponds to an agent increasing its longitudinal velocity. Let the initial velocity at the time of the initiation of the skill be  $v_0$ . The skill terminates when the velocity reaches  $v_T = v_0 + dv$ , where  $dv \in \mathbb{R}_{>0}$  denotes the desired velocity increment and  $v_T$  is the velocity of the agent at the time of termination. Hence, the  $v_{des}$  is set to  $v_0 + dv$  for this skill in the low level QP based skill policy. This skill facilitates timely task completion and helps avoid congestion by preventing agents from unnecessarily slowing down the traffic flow.

*Yielding/slowing down.* *Yielding.* This skill enables an agent to reduce its longitudinal velocity. Let  $v_0$  be the velocity at skill initiation. The skill terminates when the velocity decreases to  $v_T = v_0 - dv$ , where  $dv \in \mathbb{R}_{>0}$  is the desired reduction and  $v_T$  is the velocity at the time of termination of the skill. Thus, the velocity CLF in (37) is incorporated to the QP based skill policy by setting  $v_{des} = v_0 - dv$ . Yielding allows agents to safely navigate dense traffic and cooperate by yielding to agents approaching from other directions, thereby facilitating efficient joint task completion.

*Left Lane Changing.* This skill enables an agent to switch to the adjacent left lane. Let  $d_{left,0}$  denote the distance to the left road boundary and  $d_0$  be the lateral deviation from the center lane at the initiation of the skill. Let  $d_{lane}$  denote the lane width, provided by the environment. The target lateral position is computed as  $d_{left,T} = d_{left,0} - d_{lane} + d_0$ . We define the desired heading angle as a function of  $e$  i.e.  $\phi_{des} = f(e)$ , where  $e = d_{left,T} - d_{left,0}$ , and incorporate it into the CLF constraint on heading in (38) via a QP-based low-level skill control policy. This skill is essential for scenarios such as tollgates, bottlenecks, and merging zones, where adaptive lateral movement is required to maintain flow and avoid congestion.

*Right Lane Changing.* This skill enables an agent to switch to the adjacent right lane. Let  $d_{right,0}$  denote the distance to the right road boundary and  $d_0$  be the lateral deviation from the center lane at the initiation of the skill. Let  $d_{lane}$  denote the lane width, provided by the environment. The target lateral position is computed as  $d_{right,T} = d_{right,0} - d_{lane} + d_0$ . We define the desired heading angle as a function of  $e$ , i.e.,  $\phi_{des} = f(e)$ , where  $e = d_{right,T} - d_{right,0}$ , and incorporate it into the CLF constraint on heading in (38) via a QP-based low-level skill control policy. This skill is useful in scenarios such as avoiding obstacles, preparing for highway exits, or facilitating merging from the right, enabling smoother and safer lane coordination.

As mentioned previously we include a timeout to ensure finite time termination of the skills for algorithmic sanity.

**Intrinsic reward** The intrinsic reward serves as an internal shaping signal that guides each agent toward desirable driving behaviors—smoothness, lane-centering, and reference-tracking—Independently of any external task reward. By combining several negative penalties, the agent is discouraged from abrupt maneuvers, drifting off its lane, or deviating from its target speed and heading.

$$r_L^i = -c_1 \|\mathbf{a}^i\|^2 - c_2 \left( \frac{v^i - v_{des}}{v_{des}} \right)^2 - c_3 \left( \frac{\psi^i - \psi_{des}}{\pi} \right)^2 - c_4 \|d^i\|^2 \quad (39)$$

Where  $c_1, c_2, c_3, c_4, c_5$  are the coefficients of each terms. The intrinsic reward combines penalties on aggressive acceleration and steering, deviations from reference speed and heading, and lateral offset from the lane center, with a small forward-progress bonus. Together, these terms encourage smooth, centered, and efficient driving without abrupt maneuvers.

**CBF-Based Safe Skill Implementation.** As described earlier, we parameterize the QP-based deterministic policy  $\mu(\mathbf{o} | z, \phi)$  for each skill  $z$ , as detailed in Section 5.2. The QP formulation is parameterized through the objective, as well as the CBF and CLF constraints. In particular, we introduce parameters for the class- $\mathcal{K}$  function in the CBF constraint, the shape and size of the circular

safety regions, and the convergence rate of the CLF constraint. To ensure validity, all parameters are constrained to be nonnegative using box constraints of the form  $(0, \phi_H]$ , where  $\phi_H$  is a fixed upper bound applied uniformly across all parameters.

### 6.6.2 Implementation Details: Lidar Suite Simulator

**Target.** Agents dynamics in this environment are governed by double integrator dynamics with continuous-time states  $\mathbf{x}_i = [p_x^i, p_y^i, v_x^i, v_y^i]^\top \in \mathbb{R}^4$ , where  $\mathbf{p}_i = [p_x^i, p_y^i]^\top \in \mathbb{R}^2$  denotes position and  $\mathbf{v}_i = [v_x^i, v_y^i]^\top \in \mathbb{R}^2$  denotes velocity; control inputs  $\mathbf{u}_i = [a_x^i, a_y^i]^\top \in \mathbb{R}^2$  represent accelerations along the Cartesian axes, and the agent dynamics follow  $\dot{\mathbf{x}}_i = [v_x^i, v_y^i, a_x^i, a_y^i]^\top$ , with velocity and control constraints bounded in  $[-10, 10]$  and  $[-1, 1]$  respectively.

**Spread.** The underlying dynamics, state representation, control parametrization, and bounds are identical to those in the Target environment, i.e., agents evolve under double integrator dynamics with four-dimensional state vectors comprising positions and velocities, and two-dimensional control inputs that modulate accelerations; the bounded constraints on velocities and controls likewise remain  $[-10, 10]$  and  $[-1, 1]$ , respectively.

**Target Bicycle** In the Target Bicycle environment, agents dynamics are modeled using a bicycle model where the state vector  $\mathbf{x}_i = [p_x^i, p_y^i, \cos \theta^i, \sin \theta^i, v^i]^\top \in \mathbb{R}^5$  encapsulates the agent's position, heading orientation via its sine and cosine, and scalar speed; the control input  $\mathbf{u}_i = [\delta^i, a^i]^\top$  includes the steering angle  $\delta^i$  and linear acceleration  $a^i$ , with dynamics specified as  $\dot{\mathbf{x}}_i = [v \cos \theta, v \sin \theta, -v \sin \theta \tan \delta, v \cos \theta \tan \delta, a]^\top$ , where  $v \in [-10, 10]$  and  $\delta \in [-1.47, 1.47]$  are the admissible bounds on speed and steering angle.

Similar to the METADRIVE environments, we define CBFs based on distance/proximity as in (35). In our formulation, we do not employ a separate heading CLF; instead, we regulate agent behavior using only the velocity CLF by aligning the desired velocity vector with the desired heading direction. Specifically, given a desired heading direction  $\theta_{\text{des}}$ , we compute a target velocity vector as  $\mathbf{v}_{\text{des}} = v_{\text{mag}} [\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$ , where  $v_{\text{mag}}$  is a task-specific magnitude (e.g., maximum safe speed or adaptive goal-directed speed). This  $\mathbf{v}_{\text{des}}$  is then used in the velocity CLF described below to guide both the speed and heading behavior of the agent.

For the Target and Spread environments, where the agent's longitudinal and lateral velocities  $v_x^i, v_y^i$  are explicit components of the state, we define the Control Lyapunov Function (CLF) to track a desired velocity vector  $\mathbf{v}_{\text{des}} = [v_{x,\text{des}}, v_{y,\text{des}}]^\top \in \mathbb{R}^2$  as:

$$V(v_x^i, v_y^i) = \frac{1}{2} ((v_x^i - v_{x,\text{des}})^2 + (v_y^i - v_{y,\text{des}})^2).$$

Given the double integrator dynamics, where the control inputs  $a_x^i$  and  $a_y^i$  directly affect the velocity via  $\dot{v}_x^i = a_x^i$  and  $\dot{v}_y^i = a_y^i$ , the time derivative of the CLF becomes:

$$\dot{V} = (v_x^i - v_{x,\text{des}})a_x^i + (v_y^i - v_{y,\text{des}})a_y^i.$$

To enforce convergence, we include the following CLF constraint in the QP:

$$(v_x^i - v_{x,\text{des}})a_x^i + (v_y^i - v_{y,\text{des}})a_y^i \leq -\alpha ((v_x^i - v_{x,\text{des}})^2 + (v_y^i - v_{y,\text{des}})^2),$$

where  $\alpha > 0$  is a tunable rate parameter.

In the Target Bicycle environment, each agent's velocity is represented as a scalar speed  $v^i \in \mathbb{R}$ , and the longitudinal acceleration  $a^i \in \mathbb{R}$  serves as the control input directly affecting the speed via  $\dot{v}^i = a^i$ . To drive  $v^i$  to a desired speed  $v_{\text{des}} \in \mathbb{R}$ , we define the CLF as:

$$V(v^i) = \frac{1}{2} (v^i - v_{\text{des}})^2.$$

The time derivative of this CLF, using  $\dot{v}^i = a^i$ , is:

$$\dot{V} = (v^i - v_{\text{des}})a^i.$$

To enforce convergence of the velocity to the target, the following CLF constraint is imposed in the QP:

$$(v^i - v_{\text{des}})a^i \leq -\alpha (v^i - v_{\text{des}})^2,$$

where  $\alpha > 0$  is a tunable parameter determining the exponential rate of convergence.

**Skills Description.** We define four interpretable low-level skills for each agent in the LiDAR suite environments, which are: i. speeding up, ii. slowing down, iii. turning left, and iv. turning right.

*Speeding Up.* This skill allows an agent to increase its translational velocity to navigate more efficiently through open space or move rapidly towards its assigned goal. Let the velocity at the start of the skill be  $v_0 \in \mathbb{R}^2$ ; the skill is considered complete once the velocity reaches  $v_T = v_0 + \Delta v$ , where  $\Delta v \in \mathbb{R}^2$ , with  $\|\Delta v\| > 0$ . The CLF constraint associated with velocity tracking is enforced by setting the desired velocity  $v_{\text{des}} = v_0 + \Delta v$  in the QP-based low-level control policy. This skill facilitates faster progression in low-density scenarios while maintaining safety guarantees via CBFs.

*Slowing Down.* This skill enables the agent to reduce its speed when approaching dynamic obstacles, other agents, or goal regions where precise navigation is required. Let the initial velocity be  $v_0 \in \mathbb{R}^2$ , and define the skill termination condition as reaching  $v_T = v_0 - \Delta v$ , where  $\Delta v \in \mathbb{R}^2$  and  $\|\Delta v\| > 0$ . To enforce this behavior, the velocity CLF is specified using  $v_{\text{des}} = v_0 - \Delta v$ . This skill is critical for smooth and safe deceleration, reducing the risk of collisions and improving maneuverability near goals.

*Turning Left.* This skill enables the agent to redirect its heading counterclockwise to avoid obstacles or navigate toward an alternate subgoal. Let the initial heading be  $\theta_0$ , and the desired heading be  $\theta_{\text{des}} = \theta_0 + \Delta\theta$ , where  $\Delta\theta > 0$ . Instead of using a heading CLF, we convert the desired heading into a velocity vector using a fixed target magnitude  $v_{\text{mag}}$ , and set  $v_{\text{des}} = v_{\text{mag}}[\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$ . This desired velocity is then tracked via the standard velocity CLF in the QP. The turning-left skill is useful for curvature adjustment in cluttered or multi-agent scenes.

*Turning Right.* This skill enables clockwise heading correction using the same velocity CLF formulation. Given initial heading  $\theta_0$  and desired adjustment  $\Delta\theta > 0$ , the target heading is  $\theta_{\text{des}} = \theta_0 - \Delta\theta$ , which is converted to a velocity vector as  $v_{\text{des}} = v_{\text{mag}}[\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$ . The agent's translational motion is then regulated by the velocity CLF, enforcing convergence to the desired heading direction through velocity tracking. This skill is particularly effective in obstacle-rich or competitive zones where sharp directional changes are required.

## 6.7 Additional Results

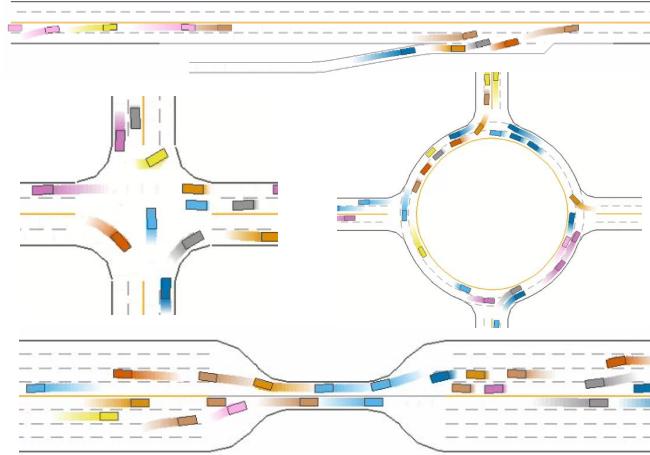


Figure 10: Visualization of agents’ trajectories under the trained model: for each vehicle, its ten most recent positions are shown in a unique hue, with brightness fading from light (earlier points) to dark (later points) to indicate temporal progression.

Here, we present results to further demonstrate the merit of our proposed algorithm. Specifically, we examine the trajectories in detail to analyze the distinct skills exhibited by the agents, offering insight into how our method sustains an extremely high success rate while also outperforming the baselines in terms of energy efficiency and episode duration. To this end, we employ a visualization scheme in

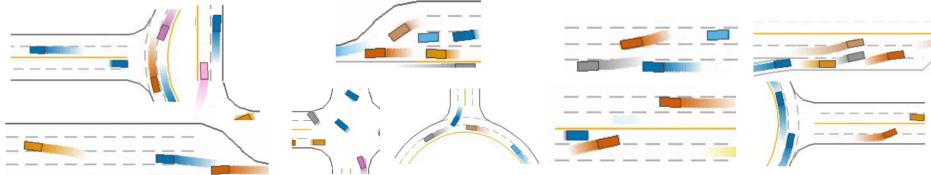


Figure 11: Visualization of skills under the trained model. From left to right: (a) acceleration (speed-up), (b) yielding (slow-down), and (c) lane changing

which each agent’s ten most recent positions are depicted in a distinct color, with brightness gradually diminishing from light (earlier points) to dark (later points) to convey temporal progression. An example of this visualization is depicted in Figure 10. By looking more closely, we can identify some of the skills that the agents exhibit during the episode. For instance, as agents approach their destinations (i.e., exit points of the environment), they execute speed up skill only when safety constraints are satisfied, thereby enhancing road throughput and reducing average travel time. Several such examples are shown in Figure 11a. Yielding (i.e., controlled slow down/deceleration) is another common skill agents employ to maintain a safe distance from other vehicles. By anticipating the need to yield, agents avoid abrupt braking, which both enhances safety and reduces overall episode energy consumption. Several such examples are shown in Figure 11b. Another skill exhibited by agents is lane changing: when executed safely at appropriate locations, it enhances road throughput and consequently reduces the average episode duration. An example of such lane changing is depicted in 11c where the agent with brown color is changing its lane— creating a safe gap into which the grey agent then merges onto the main road.

## 7 Conclusion

In conclusion, we addressed the problem of learning safe policies in multi-agent, safety-critical autonomous systems with pointwise-in-time safety constraints. We proposed HMARL-CBF, a novel hierarchical MARL approach leveraging Control Barrier Functions (CBFs). Our method employs a skill-based hierarchy, learning a high-level policy for selecting skills and a low-level policy for safely executing them. We validated our approach in complex multi-agent environments, demonstrating its effectiveness through extensive empirical results. Future work can explore jointly learning both the set of skills and their corresponding policies, along with the cooperative behavior.

## References

- [1] Lowe, R., Y. Wu, A. Tamar, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [2] Rashid, T., M. Samvelyan, C. Schroeder. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. 2018.
- [3] Sunehag, P., G. Lever, A. Gruslys, et al. Value-decomposition networks for cooperative multi-agent learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [4] Zhao, Y., Y. Yang, Z. Lu, et al. Multi-agent first order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 36, 2024.
- [5] Altman, E. *Constrained Markov decision processes*. Routledge, 2021.
- [6] Gu, S., J. Grudzien Kuba, Y. Chen, et al. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, 319:103905, 2023.
- [7] Tang, H., J. Hao, T. Lv, et al. Hierarchical deep multiagent reinforcement learning with temporal abstraction, 2019.
- [8] Wang, R., K. Wang, F. Xu, et al. Hierarchical cooperative multi-agent reinforcement learning with dual coordination mechanism. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial*

*Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.*

- [9] Ghavamzadeh, M., S. Mahadevan, R. Makar. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13:197–229, 2006.
- [10] Wu, C., A. R. Kreidieh, K. Parvate, et al. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2022.
- [11] Son, K., D. Kim, W. J. Kang, et al. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896. 2019.
- [12] Foerster, J., G. Farquhar, T. Afouras, et al. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2018.
- [13] Pertsch, K., Y. Lee, J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*. 2020.
- [14] Parr, R., S. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, pages 1043–1049. 1997.
- [15] Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
- [16] Sutton, R. S., D. Precup, S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
- [17] Precup, D., R. S. Sutton. Temporal abstraction in reinforcement learning. *PhD thesis, University of Massachusetts Amherst*, 2000.
- [18] Dayan, P., G. E. Hinton. Feudal reinforcement learning. *Advances in Neural Information Processing Systems*, pages 271–278, 1992.
- [19] Vezhnevets, A. S., S. Osindero, T. Schaul, et al. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. 2017.
- [20] Nachum, O., S. Gu, H. Lee, et al. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313. 2018.
- [21] Bacon, P.-L., M. Harb, D. Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31. 2017.
- [22] Harb, M., P.-L. Bacon, M. Klissarov, et al. Waiting for the right time: When to commit to action in hierarchical reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.
- [23] Li, S., R. Wang, M. Tang, et al. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *Advances in Neural Information Processing Systems*, 32, 2019.
- [24] Gehring, J., G. Synnaeve, A. Krause, et al. Hierarchical skills for efficient exploration. *Advances in Neural Information Processing Systems*, 34:11553–11564, 2021.
- [25] Ahilan, S., P. Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning, 2019.
- [26] Tessler, C., S. Givony, T. Zahavy, et al. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI Conference on Artificial Intelligence*, pages 1553–1561. 2017.
- [27] Gregor, K., D. J. Rezende, D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.
- [28] Mankowitz, D. J., T. A. Mann, S. Mannor. Adaptive skills, adaptive partitions (asap). In *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.
- [29] Yang, M., Y. Yang, Z. Lu, et al. Hierarchical multi-agent skill discovery. In *NeurIPS*. 2023.
- [30] Harb, J., P.-L. Bacon, M. Klissarov, et al. When waiting is not an option: Learning options with a deliberation cost. In *AAAI Conference on Artificial Intelligence*. 2018.
- [31] Sharma, A., A. Gupta, S. Levine. Skill-based meta-reinforcement learning. In *International Conference on Machine Learning (ICML)*. 2020.
- [32] Hao, C., C. Weaver, C. Tang, et al. Skill-critic: Refining learned skills for reinforcement learning. *arXiv preprint arXiv:2306.08388*, 2023.

- [33] Zhao, Y., Y. Yang, Z. Lu, et al. Multi-agent first order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 36:39189–39211, 2023.
- [34] Chow, Y., O. Nachum, E. Duenez-Guzman, et al. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.
- [35] Chow, Y., O. Nachum, A. Faust, et al. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.
- [36] Liu, Y., J. Ding, X. Liu. Ipo: Interior-point policy optimization under constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:4940–4947, 2020.
- [37] Liu, M., T. Yang, M. Wang. A natural policy gradient primal-dual method for constrained markov decision processes. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [38] Ding, Z., K. Wei, H. Lu. Upper confidence primal-dual reinforcement learning for cmdps. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [39] Huang, W., J. Ji, B. Zhang, et al. Safedreamer: Safe reinforcement learning with world models. In *The Twelfth International Conference on Learning Representations*. 2024.
- [40] Wang, Y., S. S. Zhan, R. Jiao, et al. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett, eds., *Proceedings of the 40th International Conference on Machine Learning*, vol. 202 of *Proceedings of Machine Learning Research*, pages 36593–36604. PMLR, 2023.
- [41] Gu, S., J. G. Kuba, Y. Chen, et al. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, page 103905, 2023.
- [42] So, O., C. Ge, C. Fan. Solving minimum-cost reach avoid using reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, C. Zhang, eds., *Advances in Neural Information Processing Systems*, vol. 37, pages 30951–30984. Curran Associates, Inc., 2024.
- [43] Yu, D., H. Ma, S. Li, et al. Reachability constrained reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato, eds., *Proceedings of the 39th International Conference on Machine Learning*, vol. 162 of *Proceedings of Machine Learning Research*, pages 25636–25655. PMLR, 2022.
- [44] Sabouni, E., H. Sabbir Ahmad, V. Giammarino, et al. Reinforcement learning-based receding horizon control using adaptive control barrier functions for safety-critical systems\*. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 401–406. 2024.
- [45] Berducci, L., S. Yang, R. Mangharam, et al. Learning adaptive safety for multi-agent systems. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2859–2865, 2023.
- [46] Cheng, Y., P. Zhao, N. Hovakimyan. Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions. In *Learning for Dynamics and Control Conference*, pages 104–115. PMLR, 2023.
- [47] Emam, Y., G. Notomista, P. Glotfelter, et al. Safe reinforcement learning using robust control barrier functions. *IEEE Robotics and Automation Letters*, 10:2886–2893, 2021.
- [48] Pereira, M. A., Z. Wang, I. Exarchos, et al. Safe optimal control using stochastic barrier functions and deep forward-backward sdes, 2020.
- [49] Gao, Z., G. Yang, A. Prorok. Online control barrier functions for decentralized multi-agent navigation. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 107–113. IEEE, 2023.
- [50] Cheng, R., G. Orosz, R. M. Murray, et al. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019.
- [51] Zhang, S., O. So, M. Black, et al. Discrete GCBF proximal policy optimization for multi-agent safe optimal control. In *The Thirteenth International Conference on Learning Representations*. 2025.

- [52] Makar, R., S. Mahadevan, M. Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*, pages 246–253. 2001.
- [53] Rohanianesh, K., S. Mahadevan. Learning to take concurrent actions. In S. Becker, S. Thrun, K. Obermayer, eds., *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002.
- [54] Ames, A. D., K. Galloway, J. W. Grizzle. Control lyapunov functions and hybrid zero dynamics. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6837–6842. 2012.
- [55] Xiao, W., C. Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67(7):3655–3662, 2022.
- [56] Xiao, W., C. G. Cassandras, C. Belta. *Hierarchical Optimal Control with Barrier Functions*, pages 109–120. Springer International Publishing, Cham, 2023.
- [57] Nguyen, Q., K. Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.
- [58] Sutton, R. S., D. Precup, S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.
- [59] Chuck, C., K. Black, A. Arjun, et al. Granger causal interaction skill chains. *Transactions on Machine Learning Research*, 2024.
- [60] Iqbal, S., F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pages 2961–2970. PMLR, 2019.
- [61] Yu, C., A. Velu, E. Vinitsky, et al. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.
- [62] Gupta, S., A. Dukkipati. Probabilistic view of multi-agent reinforcement learning: A unified approach. 2019.
- [63] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- [64] Rashid, T., M. Samvelyan, C. S. De Witt, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.
- [65] Hong, Y., Y. Jin, Y. Tang. Rethinking individual global max in cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 35:32438–32449, 2022.
- [66] Xu, M., M. Quiroz, R. Kohn, et al. Variance reduction properties of the reparameterization trick. In *The 22nd international conference on artificial intelligence and statistics*, pages 2711–2720. PMLR, 2019.
- [67] Choi, J. J., D. Lee, K. Sreenath, et al. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.
- [68] Amos, B., J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.
- [69] Li, Q., Z. Peng, L. Feng, et al. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning, 2022.
- [70] Schulman, J., F. Wolski, P. Dhariwal, et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [71] Haarnoja, T., A. Zhou, P. Abbeel, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [72] Kulkarni, T. D., K. Narasimhan, A. Saeedi, et al. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

- [73] Peng, Z., Q. Li, K. M. Hui, et al. Learning to simulate self-driven particles system with coordinated policy optimization. *Advances in Neural Information Processing Systems*, 34:10784–10797, 2021.
- [74] De Witt, C. S., T. Gupta, D. Makoviichuk, et al. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.
- [75] Yang, Y., R. Luo, M. Li, et al. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.
- [76] Narvekar, S., B. Peng, M. Leonetti, et al. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.
- [77] Nayak, S., K. Choi, W. Ding, et al. Scalable multi-agent reinforcement learning through intelligent information aggregation, 2023.
- [78] Gu, S., J. G. Kuba, M. Wen, et al. Multi-agent constrained policy optimisation, 2022.