

# Solving Multi-Agent Safe Optimal Control with Distributed Epigraph Form MARL

Songyuan Zhang<sup>\*†</sup>, Oswin So<sup>\*†</sup>, Mitchell Black<sup>‡</sup>, Zachary Serlin<sup>‡</sup> and Chuchu Fan<sup>†</sup>

<sup>†</sup>Department of Aeronautics and Astronautics, MIT, Cambridge, Massachusetts, USA

Email: {szhang21, oswinso, chuchu}@mit.edu

<sup>‡</sup>MIT Lincoln Laboratory, Lexington, Massachusetts, USA

Email: {mitchell.black, Zachary.Serlin}@ll.mit.edu

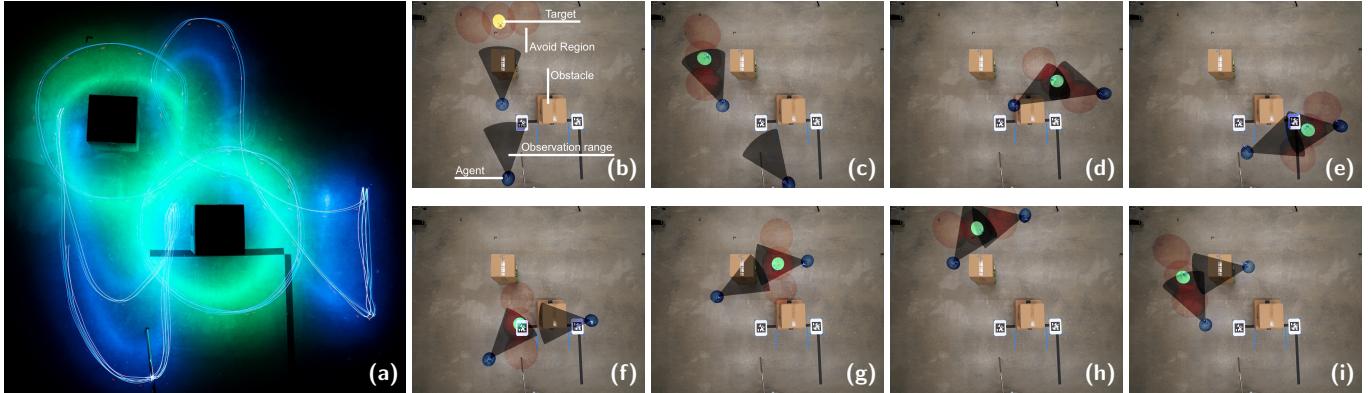


Fig. 1: **Two agents using Def-MARL to safely and collaboratively inspect a moving target.** We propose a novel safe MARL algorithm, Def-MARL, that solves the multi-agent safe optimal control problem. Def-MARL translates the original problem to its epigraph form to avoid unstable training and extends the epigraph form to the CTDE paradigm for distributed execution. (a): Long exposure photo of the trajectories of the drones. The trajectory of the target is shown in green and that of the agents is shown in blue. (b)-(i): Snapshots of the agents’ policy. Using Def-MARL, the agents learn to collaborate to maintain visual contact with the target at all times, with each agent being responsible only when the target is on their side.

**Abstract**—Tasks for multi-robot systems often require the robots to collaborate and complete a team goal while maintaining safety. This problem is usually formalized as a constrained Markov decision process (CMDP), which targets minimizing a global cost and bringing the mean of constraint violation below a user-defined threshold. Inspired by real-world robotic applications, we define safety as zero constraint violation. While many safe multi-agent reinforcement learning (MARL) algorithms have been proposed to solve CMDPs, these algorithms suffer from unstable training in this setting. To tackle this, we use the epigraph form for constrained optimization to improve training stability and prove that the centralized epigraph form problem can be solved in a distributed fashion by each agent. This results in a novel centralized training distributed execution MARL algorithm named Def-MARL. Simulation experiments on 8 different tasks across 2 different simulators show that Def-MARL achieves the best overall performance, satisfies safety constraints, and maintains stable training. Real-world hardware experiments on Crazyflie quadcopters demonstrate the ability of Def-MARL to safely coordinate agents to complete complex collaborative tasks compared to other methods.<sup>1</sup>

## I. INTRODUCTION

Multi-agent systems (MAS) play an integral role in our aspirations for a more convenient future with examples such

as autonomous warehouse operations [37], large-scale autonomous package delivery [45], and traffic routing [76]. These tasks often require designing distributed policies for agents to complete a team goal collaboratively while maintaining safety. To construct distributed policies, multi-agent reinforcement learning (MARL) under the centralized training distributed execution (CTDE) paradigm [84, 25] has emerged as an attractive method. To incorporate the safety constraints, most MARL algorithms either choose to carefully design their objective function to incorporate *soft* constraints [69, 58, 79, 74, 54, 57], or model the problem using the constrained Markov decision process (CMDP) [3], which asks for the mean constraint violation to stay below a user-defined threshold [32, 38, 18, 41, 26, 90]. However, real-world robotic applications always require *zero* constraint violation. While this can be addressed by setting the constraint violation threshold to zero in the CMDP, in this setting the popular Lagrangian methods experience training instabilities which result in sharp drops in performance during training, and non-convergence or convergence to poor policies [66, 35, 24, 36].

These concerns have been identified recently, resulting in a series of works that enforce *hard* constraints [82, 89, 66, 24, 86] using techniques inspired by Hamilton-Jacobi reachability [71, 44, 49, 46, 5] in deep reinforcement learning (RL) for the *single-agent* case and have been shown to improve safety

<sup>\*</sup>Equal contribution.

<sup>1</sup>Project website: <https://mit-realm.github.io/def-marl/>.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

compared to other safe RL approaches significantly. However, to the best of our knowledge, theories and algorithms for safe RL are still lacking for the *multi-agent* scenario, especially when policies are executed in a *distributed* manner. While single-agent RL methods can be directly applied to the MARL setting by treating the MAS as a centralized single agent, the joint action space grows exponentially with the number of agents, preventing these algorithms from scaling to scenarios with a large number of agents [33, 69, 23].

To tackle the problem of zero constraint violation in multi-agent scenarios with *distributed* policies<sup>2</sup> while achieving high collaborative performance, we propose **Distributed epigraph form MARL** (**Def-MARL**) (Fig. 1). Instead of considering the CMDP setting, **Def-MARL** directly tackles the multi-agent safe optimal control problem (MASOCP), whose solution satisfies zero constraint violation. To solve the MASOCP, **Def-MARL** uses the epigraph form technique [10], which has previously been shown to yield better policies compared to Lagrangian methods in the single-agent setting [66]. To adapt to the multi-agent setting we consider in this work, we prove that the centralized epigraph form of MASOCP can be solved in a distributed fashion by each agent. Using this result, **Def-MARL** falls under the CTDE paradigm.

We validate **Def-MARL** using 8 different tasks from 2 different simulators, multi-particle environments (MPE) [40] and Safe Multi-agent MuJoCo [32], with varying numbers of agents, and compare its performance with existing safe MARL algorithms using the penalty and Lagrangian methods. The results suggest that **Def-MARL** achieves the best performance while satisfying safety: it is as safe as conservative baselines that achieve high safety but sacrifice performance, while matching the performance of unsafe baselines that sacrifice safety for high performance. In addition, while the baseline methods require different choices of hyperparameters to perform well in different environments and suffer from unstable training because of zero constraint violation threshold, **Def-MARL** is stable in training using the same hyperparameters across all environments, indicative of the algorithm’s robustness to environmental changes.

We also perform real-world hardware experiments using the Crazyflie (CF) drones [27] on two complex collaborative tasks and compare **Def-MARL** with both centralized and decentralized model predictive control (MPC) methods [29]. The results indicate that **Def-MARL** finishes the tasks with 100% safety rates and success rates, while the MPC methods get stuck in local minima or have unsafe behaviors.

To summarize, our **contributions** are presented below:

- Drawing on prior work that addresses the training instability of Lagrangian methods in the zero-constraint violation setting, we extend the epigraph form method from single-agent RL to MARL, improving upon the training instability of existing MARL algorithms.

<sup>2</sup>In this paper, the policies are distributed if each agent makes decisions using local information/sensor data and information received via message passing with other agents [25], although this setting is sometimes called “decentralized” in MARL [83].

- We present theoretical results showing that the outer problem of the epigraph form can be decomposed and solved in a distributed manner during online execution. This allows **Def-MARL** to fall under the CTDE paradigm.
- We illustrate through extensive simulations that, without any hyperparameter tuning, **Def-MARL** achieves stable training and is as safe as the most conservative baseline while simultaneously being as performant as the most aggressive baseline across all environments.
- We demonstrate on Crazyflie drones in hardware that **Def-MARL** can safely coordinate agents to complete complex collaborative tasks. **Def-MARL** performs the task better than centralized/decentralized MPC methods and does not get stuck in suboptimal local minima or exhibit unsafe behaviors.

## II. RELATED WORK

**Unconstrained MARL.** Early works that approach the problem of safety for MARL focus on navigation problems and collision avoidance [14, 13, 21, 64], where safety is achieved by a sparse collision penalty [39], or a shaped reward penalizing getting close to obstacles and neighboring agents [14, 13, 21, 64]. However, adding a penalty to the reward function changes the original objective function, so the resulting policy may not be optimal for the original constraint optimization problem. In addition, the satisfaction of collision avoidance constraints is not necessarily guaranteed by even the optimal policy [47, 21, 39].

**Shielding for Safe MARL.** One popular method that provides safety to learning-based methods is using *shielding* or a *safety filter* [25]. Here, an unconstrained learning method is paired with a shield or safety filter using techniques such as predictive safety filters [88, 50], control barrier functions [11, 55], or automata [19, 77, 48, 7]. Such shields are often constructed before the learning begins and are used to modify either the feasible actions or the output of the learned policy to maintain safety. One benefit is that safety can be guaranteed during both training and deployment since the shield is constructed before training. However, they require domain expertise to build a valid shield, which can be challenging in the single-agent setting and even more difficult for MAS [25]. Other methods can automatically synthesize shields but face scalability challenges [48, 20]. Another drawback is that the policy after shielding might not consider the same objective as the original policy and may result in noncollaborative behaviors or deadlocks [56, 85, 87].

**Constrained MARL.** In contrast to unconstrained MARL methods, which change the constraint optimization problem to an unconstrained problem, constrained MARL methods explicitly solve the CMDP problem. For the single-agent case, prominent methods for solving CMDPs include primal methods [78], primal-dual methods using Lagrange multipliers [8, 70, 35, 36], and trust-region-based approaches [1, 35]. These methods provide guarantees either in the form of asymptotic convergence guarantees to the optimal (safe) solution [8, 70] using stochastic approximation theory [59, 9], or

recursive feasibility of intermediate policies [1, 60] using ideas from trust region optimization [61]. The survey [31] provides an overview of the different methods of solving safety-constrained single-agent RL. In multi-agent cases, however, the problem becomes more difficult because of the non-stationary behavior of other agents, and similar approaches have been presented only recently [32, 38, 18, 41, 26, 90, 15]. However, the CMDP setting they handle makes it difficult for them to handle hard constraints, and results in poor performance with zero constraint violation threshold [24].

**Model predictive control.** Distributed MPC methods have been proposed to handle MAS, incorporating multi-agent path planning, machine learning, and distributed optimization [75, 72, 92, 22, 42, 16, 52]. However, the solution quality of nonlinear optimizers used to solve MPC when the objective function and constraints are nonlinear highly depends on the initial guess [73, 29]. Moreover, the real-time nonlinear optimizers typically require access to (accurate) first and second-order derivatives [53, 29], which present challenges when trying to solve tasks that have non-differentiable or discontinuous cost functions and constraints such as the ones we consider in this work.

### III. PROBLEM SETTING AND PRELIMINARIES

#### A. Multi-agent safe optimal control problem

We consider the multi-agent safe optimal control problem (MASOCP) as defined below. Consider a homogeneous MAS with  $N$  agents. At time step  $k$ , the global state and control input are given by  $x^k \in \mathcal{X} \subseteq \mathbb{R}^n$  and  $u^k \in \mathcal{U} \subseteq \mathbb{R}^m$ , respectively. The global control vector is defined by concatenation  $u^k := [u_1^k; \dots; u_N^k]$ , where  $u_i^k \in \mathcal{U}_i$  is the control input of agent  $i$ . We consider the general nonlinear discrete-time dynamics for the MAS:

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

where  $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$  is the global dynamics function. We consider the *partially observable* setting, where each agent has a limited communication radius  $R > 0$  and can only communicate with other agents or observe the environment within its communication region. Denote  $o_i^k = O_i(x^k) \in \mathcal{O} \subseteq \mathbb{R}^{n_o}$  as the vector of the information observed by agent  $i$  at the time step  $k$ , where  $O_i : \mathcal{X} \rightarrow \mathcal{O}$  is an encoding function of the information shared from neighbors of agent  $i$  and the observed data of the environment. We allow multi-hop communication between agents, so an agent may communicate with another agent outside its communication region if a communication path exists between them.

Let the avoid/unsafe set of agent  $i$  be  $\mathcal{A}_i := \{o_i \in \mathcal{O} : h_i(o_i) > 0\}$ , for some function  $h_i : \mathcal{O} \rightarrow \mathbb{R}$ . The global avoid set is then defined as  $\mathcal{A} := \{x \in \mathcal{X} : h(x) > 0\}$ , where  $h(x) = \max_i h_i(o_i) = \max_i h_i(O_i(x))$ . In other words,  $\exists i, \text{ s.t. } o_i \in \mathcal{A}_i \iff x \in \mathcal{A}$ . Given a global cost function  $l : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$  describing the task for the agents to accomplish<sup>3</sup>,

<sup>3</sup>The cost function  $l$  is **not** the cost in CMDP. Rather, it corresponds to the negation of the *reward* in CMDP.

we aim to find distributed control policies  $\pi_i : \mathcal{O} \rightarrow \mathcal{U}_i$  such that starting from any given initial states  $x^0 \notin \mathcal{A}$ , the policies keep the agents outside the avoid set  $\mathcal{A}$  and minimize the infinite horizon cost. In other words, denoting  $\pi : \mathcal{X} \rightarrow \mathcal{U}$  as the joint policy such that  $\pi(x) = [\pi_1(o_1); \dots; \pi_N(o_N)] = [\pi_1(O_1(x)); \dots; \pi_N(O_N(x))]$ , we aim to solve the following infinite-horizon MASOCP for a given initial state  $x^0$ :

$$\min_{\{\pi_i\}_{i=1}^N} \sum_{k=0}^{\infty} l(x^k, \pi(x^k)) \quad (2a)$$

$$\text{s.t. } h_i(O_i(x^k)) \leq 0, \quad \forall i \in \{1, \dots, N\}, k \geq 0, \quad (2b)$$

$$x^{k+1} = f(x^k, \pi(x^k)), \quad k \geq 0. \quad (2c)$$

Note that the safety constraint (2b) differs from the average constraints considered in CMDPs [3]. Consequently, instead of allowing safety violations to occur as long as the mean constraint violation is below a threshold, this formulation disallows *any* constraint violation. From hereon, we omit the dynamics constraint (2c) for conciseness.

#### B. Epigraph form

Existing methods are unable to solve (2) well. This has been observed previously in the single-agent setting [82, 89, 66, 24]. We show later that the poor performance of methods that tackle the CMDP setting to the constrained problem (2) also translates to the multi-agent setting, as we observe a similar phenomenon in our experiments (Section V). Namely, although unconstrained MARL can be used to solve (2) using the penalty method [51], this does not perform well in practice, where a small penalty results in policies that violate constraints, and a large penalty results in higher total costs. The Lagrangian method [32] can solve the problem theoretically, but it suffers from unstable training and has poor performance in practice when the constraint violation threshold is zero [66, 24]. In this section, we introduce a new method of solving (2) that can mitigate the above problems by extending prior work [66] to the multi-agent setting.

Given a constrained optimization problem with objective function  $J$  (e.g.,  $J = \sum_{k=0}^{\infty} l(x^k, \pi(x^k))$  as in (2a)), and constraints  $h$  (e.g., (2b)):

$$\min_{\pi} J(\pi) \quad \text{s.t. } h(\pi) \leq 0, \quad (3)$$

its epigraph form [10] is given as

$$\min_{\pi, z} z \quad \text{s.t. } h(\pi) \leq 0, \quad J(\pi) \leq z, \quad (4)$$

where  $z \in \mathbb{R}$  is an auxiliary variable. In other words, we add a constraint to enforce  $z$  as an upper bound of the cost  $J(\pi)$ , then minimize  $z$ . The solution to (4) is identical to the original problem (3) [10]. Furthermore, (4) is equivalent [66] to

$$\min_z z \quad (5a)$$

$$\text{s.t. } \min_{\pi} J_z(\pi, z) := \max\{h(\pi), J(\pi) - z\} \leq 0 \quad (5b)$$

As a result, the original constrained problem (3) is decomposed into the following two subproblems:

- 1) An unconstrained *inner problem* (5b), where, given an arbitrary desired cost upper bound  $z$ , we find  $\pi$  such that  $J_z(\pi, z)$  is minimized, i.e., best satisfies the constraints  $h \leq 0$  and  $J \leq z$ .
- 2) A 1-dimensional constrained *outer problem* (5a) over  $z$ , which finds the smallest cost upper bound  $z$  such that  $z$  is indeed a cost upper bound ( $J \leq z$ ) and the constraints of the original problem  $h(\pi) \leq 0$  holds.

**Comparison with the Lagrangian method.** Another popular way to solve MASOCP (2) is the Lagrangian method [32]. However, it suffers from unstable training when considering the *zero* constraint violation [66, 35] setting. More specifically, this refers to the case with constraints  $\sum_{k=0}^{\infty} c(x^k) \leq 0$  for  $c : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$  non-negative. Since  $h$  can be negative, we can convert our problem setting (3) to the zero constraint violation setting by taking  $c(x) := \max\{0, h(x)\}$ . Then, (3) reads as

$$\min_{\pi} J(\pi) \quad \text{s.t.} \quad \sum_{k=0}^{\infty} \max\{0, h(x^k)\} \leq 0. \quad (6)$$

The Lagrangian form of (6) is then

$$\max_{\lambda \geq 0} \min_{\pi} J_{\lambda}(\pi, \lambda) := J(\pi) + \lambda \sum_{k=0}^{\infty} \max\{h(x^k), 0\}, \quad (7)$$

where  $\lambda$  is the Lagrangian multiplier and is updated with gradient ascent. However,  $\frac{\partial}{\partial \lambda} J_{\lambda}(\pi, \lambda) = \sum_{k=0}^{\infty} \max\{h(x^k), 0\} \geq 0$ , so  $\lambda$  continuously increases and never decreases. As  $\frac{\partial}{\partial \pi} J_{\lambda}(\pi, \lambda)$  scales linearly in  $\lambda$  when  $h(x^k) > 0$  for some  $k$ , a large value of  $\lambda$  causes a large gradient w.r.t  $x$ , and makes the training unstable. Note that for the epigraph form, since  $z$  does not multiply with the cost function  $J$  but is added to  $J$  in (5b), the gradient  $\frac{\partial}{\partial \pi} J_z(\pi, z)$  does not scale with the value of  $z$  resulting in more stable training. We validate this in our experiments (Section V).

#### IV. DISTRIBUTED EPIGRAPH FORM MULTI-AGENT REINFORCEMENT LEARNING

In this section, we propose the **Distributed epigraph form MARL** (Def-MARL) algorithm to solve MASOCP (2) using MARL. First, we transfer MASOCP (2) to its epigraph form with an auxiliary variable  $z$  to model the desired cost upper bound. The epigraph form includes an inner problem and an outer problem. For distributed execution, we provide a theoretical result that the outer problem can be solved distributively by each agent. This allows Def-MARL to fit the CTDE paradigm, where in centralized training, the agents' policies are trained together given the desired cost upper bound  $z$ , and in distributed execution, the agents distributively find the smallest cost upper bound  $z$  that ensures safety.

##### A. Epigraph form for MASOCP

To rewrite MASOCP (2) into its epigraph form (5), we first define the cost-value function  $V^l$  for a joint policy  $\pi$  using the standard optimal control notation [6]:

$$V^l(x^{\tau}; \pi) := \sum_{k \geq \tau} l(x^k, \pi(x^k)). \quad (8)$$

We also define the constraint-value function  $V^h$  as the maximum constraint violation:

$$\begin{aligned} V^h(x^{\tau}; \pi) &:= \max_{k \geq \tau} h(x^k) = \max_{k \geq \tau} \max_i h_i(o_i^k) \\ &= \max_i \max_{k \geq \tau} h_i(o_i^k) = \max_i V_i^h(o_i^{\tau}; \pi). \end{aligned} \quad (9)$$

Here, we interchange the max to define the *local per-agent* functions  $V_i^h(o_i^{\tau}; \pi) = \max_{k \geq \tau} h_i(o_i^k)$ . Each  $V_i^h$  uses only the agent's local observation and thus is distributed. We now introduce the auxiliary variable  $z$  for the desired upper bound of  $V^l$ , allowing us to restate (2) concisely as

$$\min_{\{\pi_i\}_{i=1}^N} V^l(x^0; \pi) \quad \text{s.t.} \quad V^h(x^0; \pi) \leq 0. \quad (10)$$

The epigraph form (5) of (10) then takes the form

$$\min_z \quad z \quad (11a)$$

$$\text{s.t.} \quad \min_{\{\pi_i\}_{i=1}^N} \underbrace{\max \left\{ \max_i V_i^h(o_i^{\tau}; \pi), V^l(x^{\tau}; \pi) - z \right\}}_{:= V(x^0, z; \pi)} \leq 0. \quad (11b)$$

By interpreting the left-hand side of (11b) as a *new* policy optimization problem, we define the *total* value function  $V$  as the objective function to (11b). This can be simplified as

$$\begin{aligned} V(x^{\tau}, z; \pi) &= \max_i \{V_i^h(o_i^{\tau}; \pi), V^l(x^{\tau}; \pi) - z\} \\ &= \max_i \max \{V_i^h(o_i^{\tau}; \pi), V^l(x^{\tau}; \pi) - z\} \\ &= \max_i V_i(x^{\tau}, z; \pi), \end{aligned} \quad (12)$$

Again, we interchange the max to define  $V_i(x^{\tau}, z; \pi) = \max \{V_i^h(o_i^{\tau}; \pi), V^l(x^{\tau}; \pi) - z\}$  as the *per-agent* total value function. Using this to rewrite (11) then yields

$$\min_z \quad z \quad (13a)$$

$$\text{s.t.} \quad \min \max_{\pi} V_i(x^0, z; \pi) \leq 0. \quad (13b)$$

This decomposes the original problem (2) into an unconstrained inner problem (13b) over policy  $\pi$  and a constrained outer problem (13a) over  $z$ . During offline training, we solve the inner problem (13b): for parameter  $z$ , find the optimal policy  $\pi(\cdot, z)$  to minimize  $V(x^0, z; \pi)$ . Note that the optimal policy of the inner problem depends on  $z$ . During execution, we solve the outer problem (13a) online to get the minimal  $z$  that satisfies constraint (13b). Using this  $z$  in the  $z$ -conditioned policy  $\pi(\cdot, z)$  found in the inner problem gives us the optimal policy for the overall epigraph form MASOCP (EF-MASOCP).

To solve the inner problem (13b), the total value function  $V$  must be amenable to dynamic programming, which we show in the following proposition.

*Proposition 1:* Dynamic programming can be applied to EF-MASOCP (13), resulting in

$$\begin{aligned} V(x^k, z^k; \pi) &= \max \{h(x^k), V(x^{k+1}, z^{k+1}; \pi)\}, \\ z^{k+1} &= z^k - l(x^k, \pi(x^k)). \end{aligned} \quad (14)$$

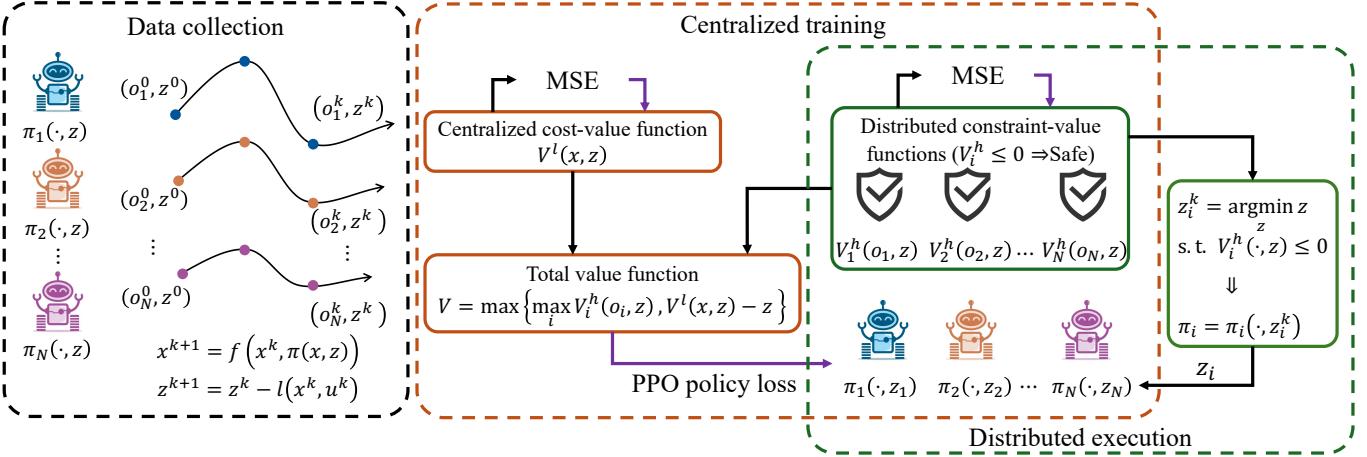


Fig. 2: **Def-MARL algorithm.** Randomly sampled initial states and  $z^0$  are used to collect trajectories in  $x$  and  $z$  using the current policy  $\pi$ . In the centralized training (orange blocks), distributed constraint-value functions  $V_i^h$  and policies  $\pi_i$  and a centralized cost-value function  $V^l$  are jointly trained. During distributed execution (green blocks), the distributed  $V_i^h$  are used to solve the outer problem (15b) to compute the optimal  $z_i$ , which is used in each agent's  $z$ -conditioned policy.

The proof of Proposition 1 is provided in Appendix A following the proof of the single-agent version [66]. In other words, for a given cost upper bound  $z^k$ , the value function  $V$  at the current state  $x^k$  can be computed using the value function at the next state  $x^{k+1}$  but with a *different* cost upper bound  $z^{k+1} = z^k - l(x^k, \pi(x^k))$  which itself is a function of  $z^k$ . This can be interpreted as a “*dynamics*” for the cost upper bound  $z$ . Intuitively, if we wish to satisfy the upper bound  $z^k$  but suffer a cost  $l(x^k, \pi(x^k))$ , then the upper bound at the next time step should be smaller by  $l(x^k, \pi(x^k))$  so that the total cost from  $x^k$  remains upper bounded by  $z^k$ . Additional discussion on Proposition 1 is provided in Appendix C.

*Remark 1 (Effect of  $z$  on the learned policy):* From (12), for a *fixed*  $x$  and  $\pi$ , observe that for  $z$  large enough (i.e.,  $V^l(x; \pi) - z$  is small enough), we have  $V(x, z; \pi) = V^h(x; \pi)$ . Consequently, taking a gradient step on  $V(x, z; \pi)$  equals taking a gradient step on  $V^h(x; \pi)$ , which reduces the constraint violation. Otherwise,  $V(x, z; \pi) = V^l(x; \pi) - z$ . Taking gradient steps on  $V(x, z; \pi)$  equals taking gradient steps on  $V^l(x; \pi)$ , which reduces the total cost.

### B. Solving the inner problem using MARL

Following So and Fan [66], we solve the inner problem using centralized training with proximal policy optimization (PPO) [63]. We use a graph neural network (GNN) backbone for the  $z$ -conditioned policy  $\pi_\theta(o_i, z)$ , cost-value function  $V_\phi^l(x, z)$ , and the constraint-value function  $V_\psi^h(o_i, z)$  with parameters  $\theta$ ,  $\phi$ , and  $\psi$ , respectively. Note that other neural network (NN) structures can be used as well. The implementation details are introduced in Appendix E.

**Policy and value function updates.** During centralized training, the NNs are trained to solve the inner problem (13b), i.e., for a randomly sampled  $z$ , find policy  $\pi(\cdot, z)$  that minimizes the total value function  $V(x^0, z; \pi)$ . We follow MAPPO [80] to train the NNs. Specifically, when calculating

the advantage with the generated advantage estimation (GAE) [62] for the  $i$ -th agent,  $A_i$  [63], instead of using the cost function  $V^l$  [80], we apply the decomposed total value function  $\max\{V_\psi^h(o_i, z), V_\phi^l(x, z) - z\}$ . We perform trajectory rollouts following the dynamics for  $x$  (1) and  $z$  (14) using the learned policy  $\pi_\theta$ , starting from random sampled  $x^0$  and  $z^0$ . After collecting the trajectories, we train the cost-value function  $V_\phi^l$  and the constraint-value function  $V_\psi^h$  via regression and use the PPO policy loss to update the  $z$ -conditioned policy  $\pi_\theta$ .

### C. Solving the outer problem during distributed execution

During execution, we solve the outer problem of EF-MASOCP (13) online. However, the outer problem is still centralized because the constraint (13b) requires the centralized cost-value function  $V^l$ . To achieve a distributed policy during execution, we introduce the following theoretical result:

**Theorem 1:** Assume no two unique values of  $z$  achieves the same unique cost. Then, the outer problem of EF-MASOCP (5a) is equivalent to the following:

$$z = \max_i z_i \quad (15a)$$

$$\begin{aligned} z_i &= \min_{z'} z' \\ \text{s.t. } &V_i^h(o_i; \pi(\cdot, z')) \leq 0, \quad i = 1, \dots, N. \end{aligned} \quad (15b)$$

The proof is provided in Appendix B. Theorem 1 enables computing  $z$  *without* the use of the centralized  $V^l$  during execution. Specifically, each agent  $i$  solves the local problem (15b) for  $z_i$ , which is a 1-dimensional optimization problem and can be efficiently solved using root-finding methods (e.g., [12]) as in [66], then communicates  $z_i$  among the other agents to obtain the maximum (15a). One challenge is that this maximum may not be computable if the agents are not connected. However, in our problem setting, if one agent is not connected, it does not appear in the observations  $o$  of other connected agents. Therefore, it would not contribute to the

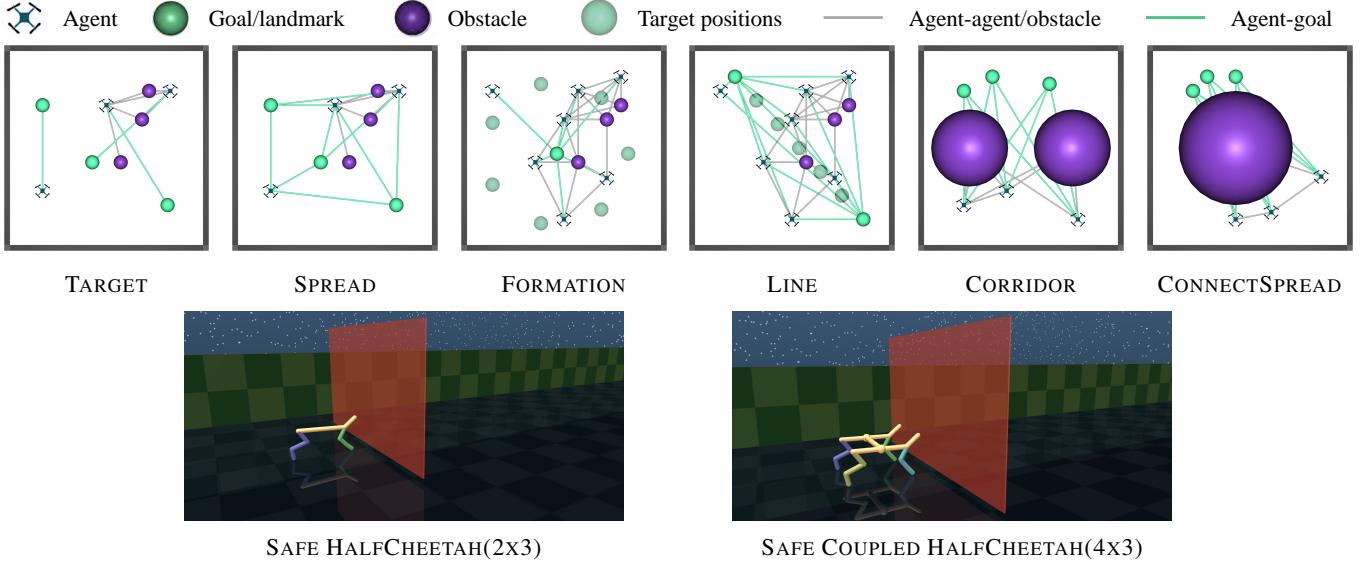


Fig. 3: **Simulation Environments.** Visualization of the (top) *modified* MPE [40] and (bottom) Safe Multi-agent MuJoCo [32] environments we consider.

constraint-value function  $V^h$  of other agents. As a result, it is sufficient for only the connected agents to communicate their  $z_i$ . Furthermore, we observe experimentally that the agents can achieve low cost while maintaining safety even if  $z_i$  is not communicated (see Section V-C). Thus, we do not include  $z_i$  communication for our method. The overall framework of Def-MARL is provided in Fig. 2.

**Dealing with estimation errors.** Since there may be errors estimating  $V^h$  using NN, we can reduce the resulting safety violation by modifying  $h$  to add a buffer region. Specifically, for a constant  $\nu > 0$ , we modify  $h$  such that  $h \geq \nu$  when the constraints are violated and  $h \leq -\nu$  otherwise. We then modify (15b) to  $V_\psi^h(o_i, z_i) \leq -\xi$ , where  $\xi \in [0, \nu]$  is a hyperparameter (where we want  $\xi \approx \nu$  to emphasize more on safety). This makes  $z$  more robust to estimation errors of  $V^h$ . We study the importance of  $\xi$  in Section V-C.

## V. SIMULATION EXPERIMENTS

In this section, we design simulation experiments to answer the following research questions:

- (Q1): Does Def-MARL satisfy the safety constraints and achieve low cost with constant hyperparameters across all environments?
- (Q2): Can Def-MARL achieve the global optimum of the original constrained optimization problem?
- (Q3): How stable is the training of Def-MARL?
- (Q4): How well does Def-MARL scale to larger MAS?
- (Q5): Does the learned policy from Def-MARL generalize to larger MAS?

Details for the implementation, environments, and hyperparameters are provided in Appendix E.

### A. Setup

**Environments.** We evaluate Def-MARL in two sets of simulation environments: *modified* Multi-agent Particle Environments (MPE) [40], and Safe Multi-agent MuJoCo environments [32] (see Fig. 3). In MPE, the agents are assumed to have double integrator dynamics with bounded *continuous* action spaces  $[-1, 1]^2$ . We provide the full details of all tasks in Appendix E. To increase the difficulty of the tasks, we add 3 static obstacles to these environments. For Safe Multi-agent MuJoCo environments, we consider SAFE HALFCHEETAH 2x3 and SAFE COUPLED HALFCHEETAH 4x3. The agents must collaborate to make the cheetah run as fast as possible without colliding with a moving wall in front. To design the constraint function  $h$ , we let  $\nu = 0.5$  in all our experiments and  $\xi = 0.4$  when solving the outer problem.

**Baselines.** We compare our algorithm with the state-of-the-art (SOTA) MARL algorithm InforMARL [51] with a constraint-penalized cost  $l'(x, u) = l(x, u) + \beta \max\{h(x), 0\}$ , where  $\beta \in \{0.02, 0.1, 0.5\}$  is a penalty parameter, and denote this baseline as *Penalty* ( $\beta$ ). We also consider the SOTA safe MARL algorithm MAPPO-Lagrangian [30, 32]<sup>4</sup>. In addition, because the learning rate of the Lagrangian multiplier  $\lambda$  is tiny ( $10^{-7}$ ) in the official implementation of MAPPO-Lagrangian [32], the value of  $\lambda$  during training will be largely determined by the initial value  $\lambda_0$  of  $\lambda$ . We thus consider two  $\lambda_0 \in \{1, 5\}$ . Moreover, to compare the training stability, we consider increasing the learning rate of  $\lambda$  in MAPPO-Lagrangian to  $3 \times 10^{-3}$ .<sup>5</sup> For a fair comparison, we reimplement MAPPO-Lagrangian using the same GNN backbone as

<sup>4</sup>We omit the comparison with MACPO [30, 32] as it was shown to perform similarly to MAPPO-Lagrangian but have significantly worse time complexity and wall clock time for training.

<sup>5</sup>This is the smallest learning rate for  $\lambda$  that does not make MAPPO-Lagrangian ignore the safety constraint. We set  $\lambda_0 = 0.78$  following [32].

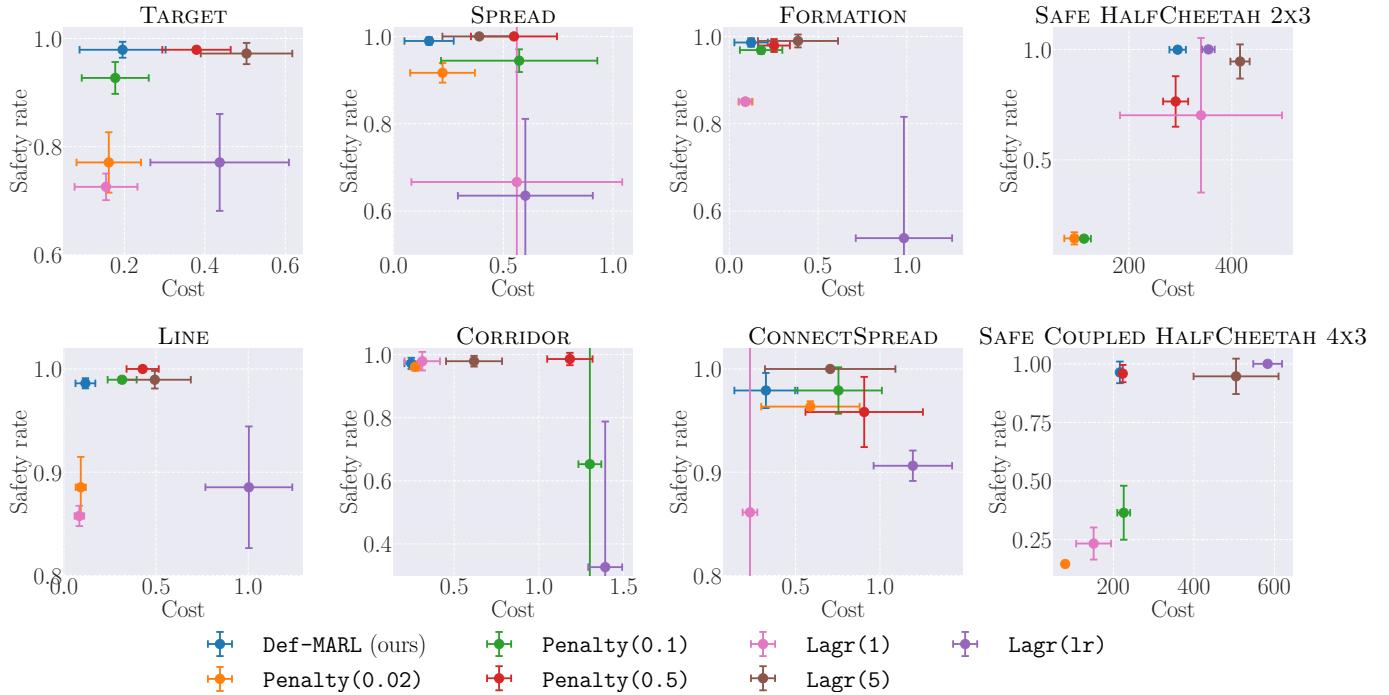


Fig. 4: **Comparison on modified MPE ( $N = 3$ ) and Safe Multi-agent MuJoCo.** Def-MARL is consistently closest to the top-left corner in all environments, achieving low cost with near 100% safety rate. The dots show the mean values and the error bars show one standard deviation.

used in Def-MARL and InforMARL, denoted as  $\text{Lagr}(\lambda_0)$  and  $\text{Lagr}(lr)$  for the increased learning rate one. We run each method for the same number of update steps, which is large enough for all the methods to converge.

**Evaluation criteria.** Following the objective of MASOCP, we use the cost and safety rate as the evaluation criteria for the performance of all algorithms. The *cost* is the cumulative cost over the trajectory  $\sum_{k=0}^T l(x^k, u^k)$ . The *safety rate* is defined as the ratio of agents that remain safe over the entire trajectory, i.e.,  $h_i(o_i^k) \leq 0, \forall k$ , over all agents. Unlike the CMDP setting, we do not report the mean of constraint violations over time but the violation of the hard safety constraints.

## B. Results

We train all algorithms with 3 different random seeds and test the converged policies on 32 different initial conditions. As discussed in Section IV-C, we *disable* the communication of  $z_i$  between agents (investigated in Section V-C). We draw the following conclusions.

**(Q1): Def-MARL achieves the best performance with constant hyperparameters across all environments.** First, we plot the safety rate (y-axis) and cumulative cost (x-axis) for each algorithm in Fig. 4. Thus, the closer an algorithm is to the top-left corner, the better it performs. In both MPE and Safe Multi-agent MuJoCo environments, Def-MARL is always closest to the top-left corner, maintaining a low cost while having near 100% safety rate. For the baselines Penalty and Lagr, their performance and safety are highly sensitive to their hyperparameters. While Penalty with  $\beta = 0.02$

and Lagr with  $\lambda_0 = 1$  generally have low costs, they also have frequent constraint violations. With  $\beta = 0.5$  or  $\lambda_0 = 5$ , they prioritize safety but at the cost of high cumulative costs. Def-MARL, however, maintains a safety rate similar to the most conservative baselines (Penalty(0.5) and Lagr(5)) but has much lower costs. We point out that no *single* baseline method behaves considerably better on *all* the environments: the performance of the baseline methods varies wildly between environments, demonstrating the sensitivity of these algorithms to the choice of hyperparameters. On the contrary, Def-MARL, performs best in *all* environments, using a single set of constant hyperparameters, which demonstrates its insensitivity to the choice of hyperparameters.

**(Q2): Def-MARL is able to reach the global optimum of the original problem.** An important observation is that for Penalty and Lagr with a non-optimal  $\lambda$ , the cost function optimized in their training process is *different* from the original cost function. Consequently, they can have different optimal solutions compared to the original problem. Therefore, even if their training converges, they may not reach the optimal solution to the original problem. In Fig. 5, the converged states of Def-MARL and four baselines are shown. Def-MARL reaches the original problem's global optimum and covers all three goals. On the contrary, the optima of Penalty(0.02) and Lagr(1) are *changed* by the penalty term, so they choose to leave one agent behind to have a lower safety penalty. With an even more significant penalty, the optima of Penalty(0.5) and Lagr(5) are changed dramatically, and they forget the goal entirely and only focus on safety.

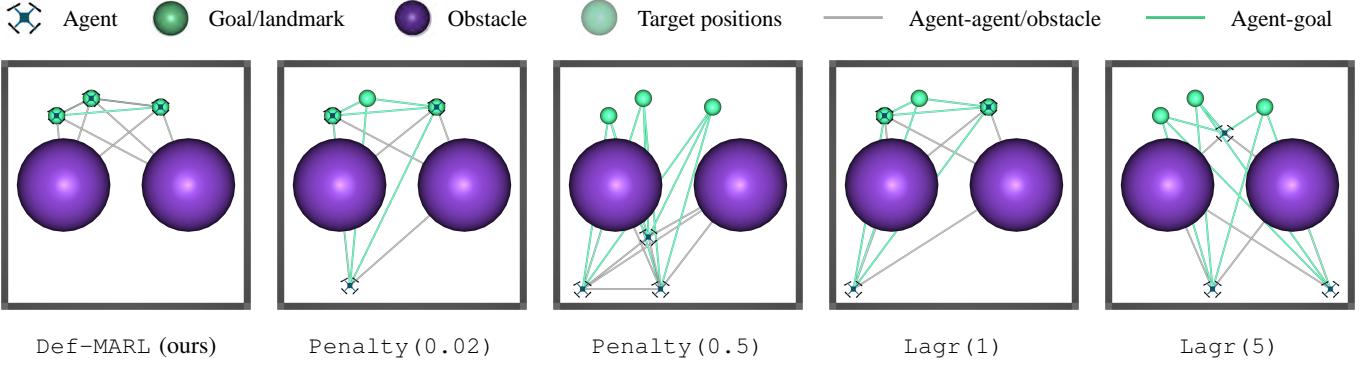


Fig. 5: **Converged states in CORRIDOR.** Def-MARL achieves the global minimum, while other baselines converge to a different optimum (partly) due to training using a *different* cost function.

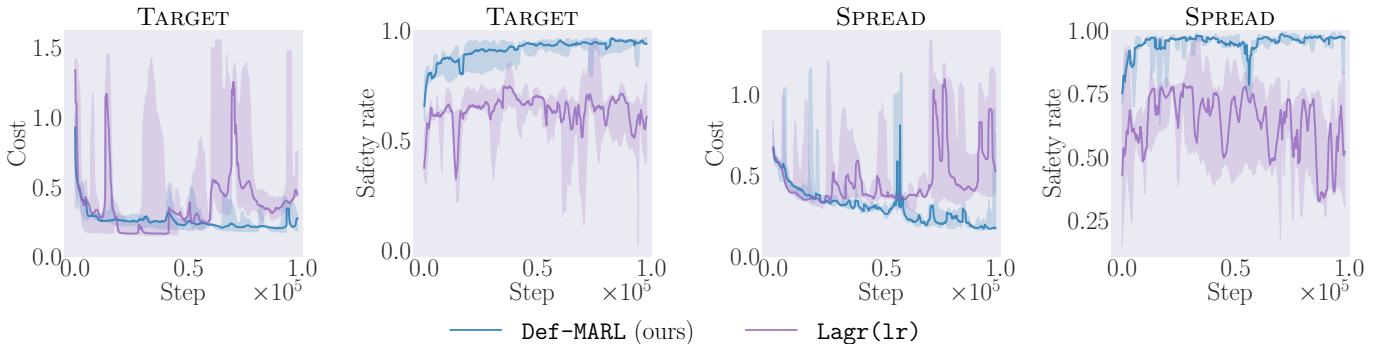


Fig. 6: **Training Curves in TARGET and SPREAD.** Def-MARL has a smoother, more stable training curve compared to Lagr(lr). We plot the mean and shade the  $\pm 1$  standard deviation.

TABLE I: **Policy Generalization.** Testing Def-MARL on TARGET with more agents after training with  $N = 8$  agents.

# Agent	32	128	512
Safety rate	$99.8 \pm 0.2$	$99.6 \pm 0.4$	$99.5 \pm 0.3$
Cost	$-0.387 \pm 0.029$	$-0.408 \pm 0.015$	$-0.410 \pm 0.009$

**(Q3): Training of Def-MARL is more stable.** To compare the training stability of Def-MARL and the Lagrangian method Lagr(lr), we plot their cost and safety rate during training in Fig. 6. Def-MARL has a *smoother* curve compared to Lagr(lr), supporting our theoretical analysis in Section III-B. Due to space limits, the plots for other environments and other baseline methods are provided in Appendix E-D.

**(Q4): Def-MARL can scale to more agents while maintaining high performance and safety, but is limited by GPU memory due to centralized training.** We test the limits of Def-MARL on the number of agents during training by comparing all methods on  $N = 5, 7$  with FORMATION and LINE, and  $N = 8, 12, 16$  with TARGET (Fig. 7). We were unable to increase  $N$  further due to GPU memory limitations due to the use of *centralized* training. For this experiment, we omit Lagr(lr) as it has the worst performance in MPE with  $N = 3$ . Def-MARL is closest to the upper left corner in all environments, and its performance does not decrease with an

increasing number of agents.

**(Q5): The trained policy from Def-MARL generalizes to much larger MASs.** To test the generalization capabilities of Def-MARL, we test a policy trained with  $N = 8$  on much larger MASs with up to  $N = 512$  on TARGET (Table I) with the same agent density to avoid distribution shift. Def-MARL maintains a high safety rate and low costs despite being applied on a MAS with 64 times more agents.

### C. Ablation studies

Here we do ablation studies on the communication of  $z_i$ , and study the hyperparameter sensitivity of Def-MARL.

**Is communicating  $z_i$  necessary?** As introduced in Section IV-C, theoretically, all connected agents should communicate and reach a consensus on  $z = \max_i z_i$ . However, we observe in Section V-B that the agents can perform well even if agents take  $z \leftarrow z_i$  without communicating to compute the maximum. We perform experiments on MPE ( $N = 3$ ) to understand the impact of this approximation in Table II and see that using the approximation does not result in much performance difference compared to communicating  $z_i$  and using the maximum.

**Varying  $\xi$  in the outer problem.** To robustify our approach against estimation errors in  $V^h$ , we solve for a  $z_i$  that is slightly more conservative by modifying (15b) to  $V_\psi^h(o_i, z_i) \leq -\xi$  (Section IV-C). We now perform experiments

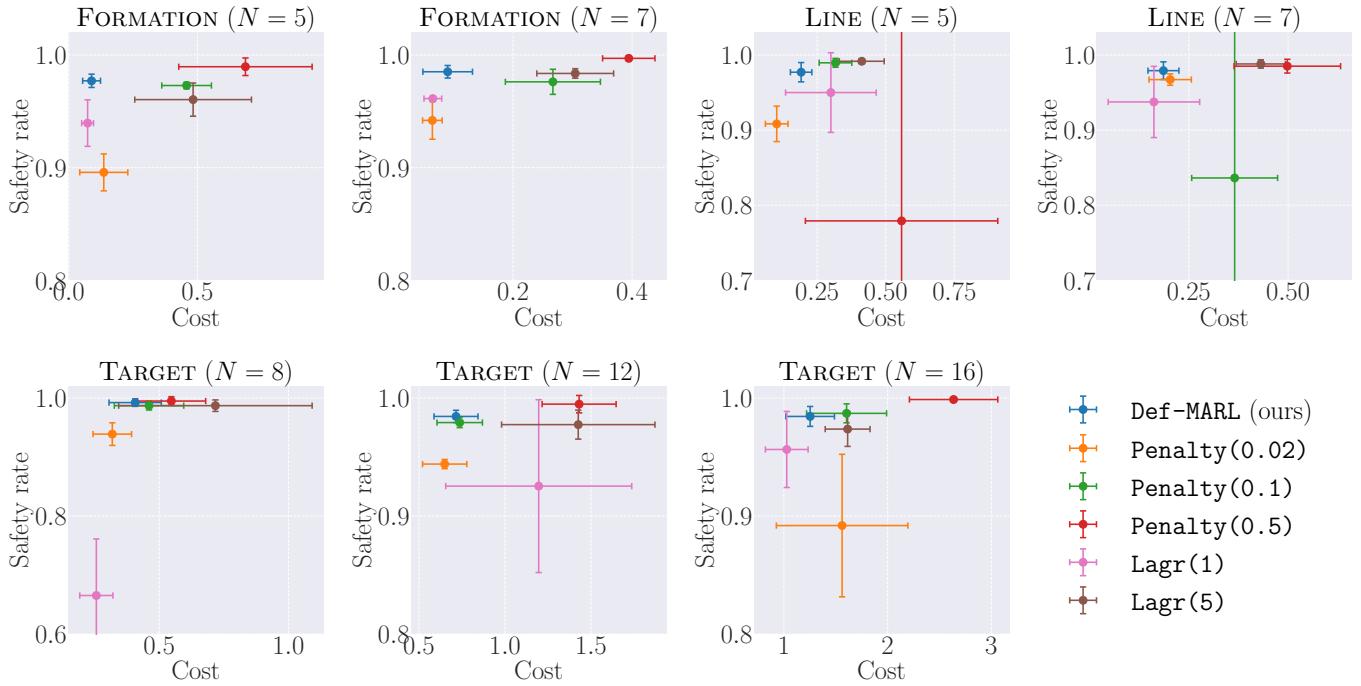


Fig. 7: **Comparison on larger-scale modified MPE.** Def-MARL remains in the top-left corner even when the number of agents increases. The dots show the mean and the error bars show one standard deviation.

TABLE II: Effect of  $z_i$  communication (Section IV-C) in different environments.

Environment	No communication ( $z \leftarrow z_i$ )		Communication ( $z = \max_i z_i$ )	
	Safety rate	Cost	Safety rate	Cost
TARGET	97.9 $\pm$ 1.5	0.196 $\pm$ 0.108	96.9 $\pm$ 3.0	0.214 $\pm$ 0.141
SPREAD	99.0 $\pm$ 0.9	0.162 $\pm$ 0.144	98.6 $\pm$ 1.3	0.171 $\pm$ 0.128
FORMATION	98.3 $\pm$ 1.0	0.123 $\pm$ 0.940	98.3 $\pm$ 1.8	0.126 $\pm$ 0.100
LINE	98.6 $\pm$ 0.5	0.117 $\pm$ 0.540	98.3 $\pm$ 0.5	0.121 $\pm$ 0.630
CORRIDOR	97.9 $\pm$ 1.8	0.247 $\pm$ 0.390	98.6 $\pm$ 1.9	0.255 $\pm$ 0.470
CONNECTSPREAD	97.9 $\pm$ 1.7	0.324 $\pm$ 0.187	99.0 $\pm$ 0.8	0.339 $\pm$ 0.201

to study the effect of different choices of  $\xi$  (Table III) on LINE ( $N = 3$ ). The results show that higher values of  $\xi$  result in higher safety rates and slightly higher costs, while the reverse is true for smaller  $\xi$ . This matches our intuition that modifying (15b) can help improve constraint satisfaction when the learned  $V^h$  has estimation errors. We thus recommend choosing  $\xi$  close to  $\nu$ . We also provide the sensitivity analysis on more hyperparameters in Appendix E.

## VI. HARDWARE EXPERIMENTS

Finally, we conduct hardware experiments on a swarm of Crazyflie (CF) drones [27] to demonstrate Def-MARL’s ability to safely coordinate agents to complete complex collaborative tasks in the real world. We consider the following two tasks.

- CORRIDOR. A swarm of drones collaboratively gets through a narrow corridor and reaches a set of goals without explicitly assigning drones to goals.
- INSPECT. Two drones collaborate to maintain direct visual contact with a target drone that follows a path shaped like an eight while staying out of the target drone’s avoid

TABLE III: Effect of varying  $\xi$  (Section IV-C) for LINE ( $N = 3$ ) with fixed  $\nu = 0.5$ .

$\xi$	Safety rate	Cost
0.5	100.0 $\pm$ 0.0	0.127 $\pm$ 0.061
0.4	98.6 $\pm$ 0.5	0.117 $\pm$ 0.540
0.2	96.5 $\pm$ 0.5	0.108 $\pm$ 0.044
0.0	93.4 $\pm$ 0.020	0.102 $\pm$ 0.035

zone. Visual contact only occurs when the line of sight to the target drone is not blocked by obstacles.

For both tasks, all drones have collision constraints with other drones and static obstacles. We visualize the tasks in Fig. 8.

**Baselines.** Def-MARL has demonstrated superior performance among RL methods in simulation experiments. Consequently, we do not consider RL methods as baselines for the hardware experiments. Instead, we present a comparative analysis between Def-MARL and model predictive control (MPC), a widely employed technique in practical robotic applications. Notably, MPC necessitates model dynamics knowledge, whereas Def-MARL does not. We compare Def-MARL against the following two MPC baselines.

- *Decentralized.* We consider a decentralized MPC method (DMPC), where each drone tries to individually minimize the total cost and prevents collision with other controlled drones by assuming a constant velocity motion model using the current measurement of their velocity.
- *Centralized.* We also test against a centralized MPC (CMPC) method to better disentangle phenomena related



(a) CORRIDOR

(b) INSPECT

Fig. 8: **Hardware tasks.** We perform hardware experiments using a swarm of CF drones on the CORRIDOR and INSPECT tasks. In CORRIDOR, the team must cross a narrow corridor and cover a set of goals collectively without prior assignment. In INSPECT, the team must maintain visual contact with the target drone while staying out of the avoid zone around the target.

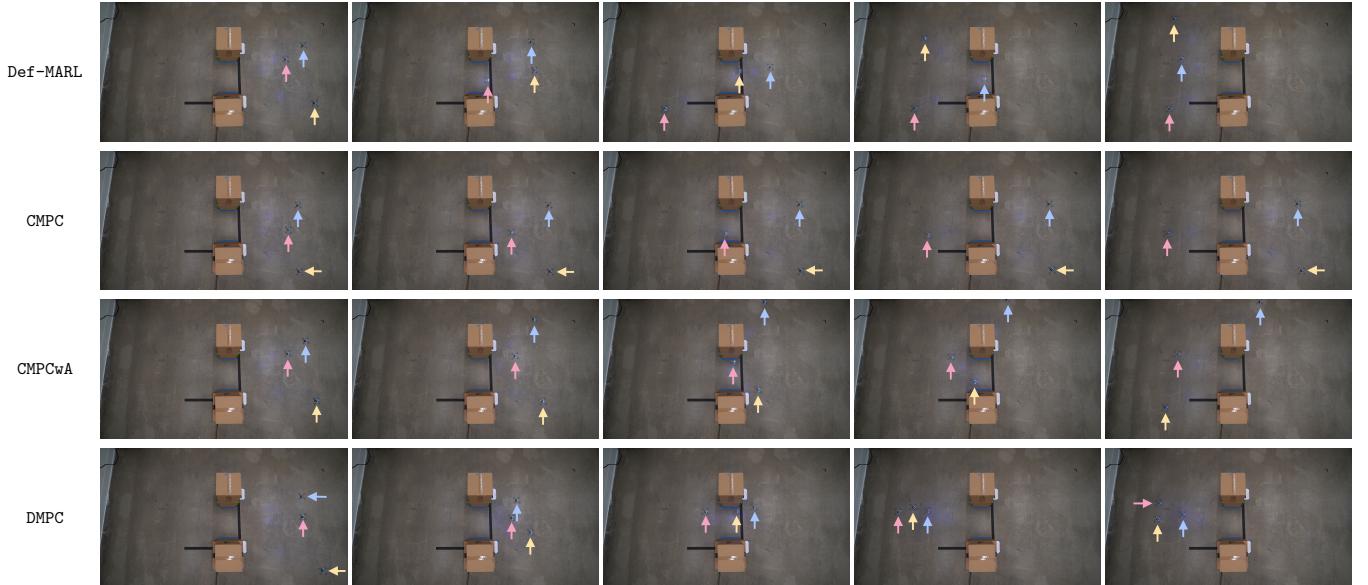


Fig. 9: **Hardware Results on CORRIDOR ( $N = 3$ ).** Left to right: key frames of the trajectories generated by different algorithms. Arrows with different colors indicate the positions of different drones. Def-MARL (top) finishes the task with 100% success rate because the drones learn to cross the corridor one by one. CMPC and CMPCwA (middle) sometimes get stuck in local minima and cannot finish the task because of the highly nonconvex cost function. DMPC (bottom) has unsafe cases where the agents cross the unsafe radius so they cannot reach the goals because the MPC problem becomes infeasible.

to numerical nonlinear optimization and performing decentralized control. This method uses the same cost function used by Def-MARL.

Both MPC methods are implemented in CasADi [4] with the SNOPT nonlinear optimizer [28]. Details for the hardware setup and experiment videos are provided in Appendix F.

#### A. CORRIDOR

We run each algorithm from 16 random initial conditions and use the task *success rate* to measure their performance. The task is defined to be successful if all the goals are covered by the agents and all agents stay safe during the whole task.

Accordingly, the success rate is defined as the number of successful tasks divided by 16. In our tests, the success rates of Def-MARL, CMPC, and DMPC are 100%, 0%, and 62.5%. To analyze this result, we visualize the trajectory of Def-MARL and some failure cases of the baselines in Fig. 9.

**CMPC is prone to local minima.** We first compare Def-MARL with CMPC. Since we sum the distance from each goal to the closest drone, the cost function in this task is very nonconvex. Consequently, CMPC results in very suboptimal solutions, where only the closest drone to the goals attempts to reach the goals, with the remaining drones left on the other side. To alleviate this issue, although the original task does



Fig. 10: **Hardware Results of Def-MARL on CORRIDOR ( $N = 7$ )**. Even in this crowded environment, Def-MARL maintains a success rate of 100%.



Fig. 11: **Hardware Results on INSPECT**. The CF drone overlaid with the yellow/green sphere is the target. The sphere turns green when the target is observed and yellow otherwise. The CF drones overlaid with blue spheres are agents, which turn red if the agents become unsafe. The red spheres around the target show the avoid zone that agents cannot enter. Def-MARL finishes the task with safe and collaborative behaviors. For example, they learn to wait on two sides of the obstacles and take turns to observe the target. CMPC gets stuck in local minima and only moves the closest drone to the target, leaving the other drone stationary. DMPC makes both agents chase after the target without collaboration, and even has unsafe cases.

not have an explicit goal allocation, we provide a handicap to the CMPC methods and rerun the experiments with explicitly assigned goals for each drone. This simplifies the optimization problem by removing the discrete nature of goal assignment. We name this baseline CMPCwA (CMPC with assignment). However, even with explicit goal assignments, we still see that sometimes one of the drones in the team gets stuck behind the corridor, resulting in a success rate of 87.5%. In contrast, Def-MARL does not succumb to this local minimum and completes the task with a success rate of 100%.

**DMPC has unsafe cases.** As DMPC without goal assignment will also suffer from similar issues as CMPC, we choose to assign goals for each agent in this baseline. This results in a simpler problem, as explained above. However, unlike CMPCwA, the agents using DMPC do not know the *actual* actions of the other agents and can only make predictions based on their observations because of the decentralized nature of DMPC. Therefore, collisions may occur if the other agents behave significantly differently from the predictions. In the DMPC row of Fig. 9, the agents collide<sup>6</sup> in the middle of the tasks, causing the MPC optimization problem to become

infeasible and preventing the agents from reaching their goals.<sup>7</sup>

**Def-MARL can scale up to 7 agents.** We also test the scalability of Def-MARL with 7 CF drones in the same environment. Notably, the size of the environment remains unchanged, so the environment becomes much more crowded and thus more challenging. We test Def-MARL with 9 different random initial conditions, and it maintains a success rate of 100%. We visualize one of the trajectories in Fig. 10. Note that we were limited to only 7 drones here due to only having 7 drones available. However, given the simulation results, we are hopeful that Def-MARL can scale to larger hardware swarms.

## B. INSPECT

We also run each algorithm from 16 different random initial conditions. Note that the agents may not be able to observe the target at their initial positions at the first step. In this environment, the team of two drones should maximize the duration where the goal is observed by at least one drone. Measuring the task performance by the number of timesteps where the target is not visible, we obtain that the performance of Def-MARL, CMPC, and DMPC are  $85.5 \pm 42.9$ ,  $206 \pm 53.2$ , and  $251 \pm 59.1$ , respectively. We also report the *safety rate*,

<sup>6</sup>For our safety, the safety radius of the drones is larger than their actual radius. Here, we mean they have entered each others' safety radius, although they have not collided in reality.

<sup>7</sup>Some MPC-based methods can solve the CORRIDOR environment [43, 68] but assume pre-assigned goals. Additionally, these approaches need additional methods for collision avoidance (e.g., Buffered Voronoi Cells [91], on-demand collision avoidance approaches [42]), which require more domain knowledge.

defined as the ratio of tasks where all agents stay safe, which are 100%, 100%, and 43.75%, respectively. We visualize the trajectories of different methods in Fig. 11.

**Agents using Def-MARL have better collaboration.** The INSPECT is designed such that collaboration between the agents is necessary to observe the target without any downtime. One agent cannot observe the target all the time on its own because the agent’s observation can be blocked by obstacles. It also cannot simply follow the target because of the avoid region. Using Def-MARL, the agents learn collaborative behaviors such as waiting on each side of the obstacles and taking turns to observe the target when the target is on their side. The MPC methods, however, do not have such *global* optimal behavior but get stuck in local minima. For example, CMPC only moves the closest drone to the target, leaving the other drone stationary, while DMPC makes both agents chase after the target without collaboration. Therefore, both MPC methods have small periods of time where neither drone has visual contact with the target. In addition, similar to in CORRIDOR, we observe that DMPC sometimes results in collisions due to the lack of coordination between drones.

## VII. CONCLUSION

To construct safe distributed policies for real-world multi-agent systems, this paper introduces Def-MARL for the multi-agent safe optimal control problem, which defines safety as zero constraint violation. Def-MARL takes advantage of the epigraph form of the original problem to address the training instability of Lagrangian methods in the zero-constraint violation setting. We provide a theoretical result showing that the centralized epigraph form can be solved in a distributed fashion by each agent, which enables distributed execution of Def-MARL. Simulation results on MPE and the Safe Multiagent MuJoCo environments suggest that, unlike baseline methods, Def-MARL uses a constant set of hyperparameters across all environments, and achieves a safety rate similar to the most conservative baseline and similar performance to the baselines that prioritize performance but violate safety constraints. Hardware results on the Crazyflie drones demonstrate Def-MARL’s ability to solve complex collaborative tasks safely in the real world.

## VIII. LIMITATIONS

The theoretical analysis in Section IV-C suggests that the connected agents must communicate  $z$  and reach a consensus. If the communication on  $z$  is disabled, although our experiments show that the agents still perform similarly, the theoretical optimality guarantee may not be valid. In addition, the framework does not consider noise, disturbances in the dynamics, or communication delays between agents. Finally, as a safe RL method, although safety can be theoretically guaranteed under the optimal value function and policy, this does not hold under inexact minimization of the losses. We leave tackling these issues as future work.

## ACKNOWLEDGMENTS

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

© 2025 Massachusetts Institute of Technology.

Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

## REFERENCES

- [1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.
- [2] Akshat Agarwal, Sumit Kumar, Katia Sycara, and Michael Lewis. Learning transferable cooperative behavior in multi-agent team. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS’2020)*. IFMAS, 2020.
- [3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2004.
- [4] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.
- [5] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.
- [6] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.
- [7] Suda Bharadwaj, Roderik Bloem, Rayna Dimitrova, Bettina Konighofer, and Ufuk Topcu. Synthesis of minimum-cost shields for multi-agent systems. In ACC. IEEE, 2019.
- [8] Vivek S Borkar. An actor-critic algorithm for constrained markov decision processes. *Systems & Control Letters*, 54(3):207–213, 2005.
- [9] Vivek S Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*, volume 48. Springer, 2009.
- [10] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [11] Zhiyuan Cai, Huanhui Cao, Wenjie Lu, Lin Zhang, and Hao Xiong. Safe multi-agent reinforcement learning through decentralized multiple control barrier functions. *arXiv preprint arXiv:2103.12553*, 2021.

- [12] Tirupathi R Chandrupatla. A new hybrid quadratic/bisection algorithm for finding the zero of a nonlinear function without using derivatives. *Advances in Engineering Software*, 28(3):145–149, 1997.
- [13] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.
- [14] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 285–292. IEEE, 2017.
- [15] Ziyi Chen, Yi Zhou, and Heng Huang. On the duality gap of constrained cooperative multi-agent reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.
- [16] Christian Conte, Tyler Summers, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. Computational aspects of distributed optimization in model predictive control. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pages 6819–6824. IEEE, 2012.
- [17] Philip Dames, Pratap Tokekar, and Vijay Kumar. Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, 36(13-14): 1540–1553, 2017.
- [18] Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient generalized lagrangian policy optimization for safe multi-agent reinforcement learning. In *Learning for Dynamics and Control Conference*, pages 315–332. PMLR, 2023.
- [19] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. *arXiv preprint arXiv:2101.11196*, 2021.
- [20] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. *AAMAS ’21*, 2021.
- [21] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.
- [22] Giuseppe Fedele and Giuseppe Franzè. A distributed model predictive control strategy for constrained multi-agent systems: The uncertain target capturing scenario. *IEEE Transactions on Automation Science and Engineering*, 2023.
- [23] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [24] Milan Ganai, Zheng Gong, Chennig Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [25] Kunal Garg, Songyuan Zhang, Oswin So, Charles Dawson, and Chuchu Fan. Learning safe control for multi-robot systems: Methods, verification, and open challenges. *Annual Reviews in Control*, 57:100948, 2024.
- [26] Nan Geng, Qinbo Bai, Chenyi Liu, Tian Lan, Vaneet Aggarwal, Yuan Yang, and Mingwei Xu. A reinforcement learning framework for vehicular network routing under peak and average constraints. *IEEE Transactions on Vehicular Technology*, 2023.
- [27] Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42. IEEE, 2017.
- [28] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.
- [29] Lars Grne and Jrgen Pannek. *Nonlinear model predictive control: theory and algorithms*. Springer Publishing Company, Incorporated, 2013.
- [30] Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation. *arXiv preprint arXiv:2110.02793*, 2021.
- [31] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.
- [32] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, 319:103905, 2023.
- [33] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234. Citeseer, 2002.
- [34] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.
- [35] Tairan He, Weiye Zhao, and Changliu Liu. Autocost: Evolving intrinsic cost for zero-violation reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14847–14855, 2023.
- [36] Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [37] Ajay Kattepur, Hemant Kumar Rath, Anantha Simha, and Arijit Mukherjee. Distributed optimization in multi-agent

- robotics for industry 4.0 warehouses. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 808–815, 2018.
- [38] Chenyi Liu, Nan Geng, Vaneet Aggarwal, Tian Lan, Yuan Yang, and Mingwei Xu. Cmix: Deep multi-agent reinforcement learning with peak and average constraints. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pages 157–173. Springer, 2021.
- [39] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.
- [40] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- [41] Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Ba  ar, and Lior Horesh. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8767–8775, 2021.
- [42] Carlos E Luis and Angela P Schoellig. Trajectory generation for multiagent point-to-point transitions via distributed model predictive control. *IEEE Robotics and Automation Letters*, 4(2):375–382, 2019.
- [43] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.
- [44] John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.
- [45] Hang Ma, Jiaoyang Li, TK Kumar, and Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv preprint arXiv:1705.10868*, 2017.
- [46] Kostas Margellos and John Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Transactions on automatic control*, 56(8):1849–1861, 2011.
- [47] Pierre-Fran  ois Massiani, Steve Heim, Friedrich Solowjow, and Sebastian Trimpe. Safe value functions. *IEEE Transactions on Automatic Control*, 68(5):2743–2757, 2023.
- [48] Daniel Melcer, Christopher Amato, and Stavros Tripakis. Shield decentralization for safe multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.
- [49] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton–jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.
- [50] Simon Muntwiler, Kim P Wabersich, Andrea Carron, and Melanie N Zeilinger. Distributed model predictive safety certification for learning-based control. *IFAC-PapersOnLine*, 53(2):5258–5265, 2020.
- [51] Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International Conference on Machine Learning*, pages 25817–25833. PMLR, 2023.
- [52] Angelia Nedi  and Ji Liu. Distributed optimization for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:77–103, 2018.
- [53] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.
- [54] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin B  hmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.
- [55] Marcus A Pereira, Augustinos D Saravatos, Oswin So, and Evangelos A Theodorou. Decentralized safe multi-agent stochastic optimal control using deep fbsdes and admm. *arXiv preprint arXiv:2202.10658*, 2022.
- [56] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2021.
- [57] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.
- [58] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.
- [59] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.
- [60] Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, pages 8502–8511. PMLR, 2020.
- [61] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.
- [62] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- [63] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization

- algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [64] Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.
- [65] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.
- [66] Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2023.
- [67] Oswin So, Cheng Ge, and Chuchu Fan. Solving minimum-cost reach avoid using reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [68] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3(6):545–554, 2021.
- [69] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.
- [70] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.
- [71] Claire J Tomlin, John Lygeros, and S Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.
- [72] Charbel Toumeh and Alain Lambert. Decentralized multi-agent planning using model predictive control and time-aware safe corridors. *IEEE Robotics and Automation Letters*, 7(4):11110–11117, 2022.
- [73] Panagiotis Tsotras, Efstathios Bakolas, and Yiming Zhao. Initial guess generation for aircraft landing trajectory optimization. In *AIAA Guidance, Navigation, and Control Conference*, page 6689, 2011.
- [74] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.
- [75] Peng Wang and Baocang Ding. A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance. *International Journal of Control*, 87(1):52–63, 2014.
- [76] Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(8):8243–8256, 2020.
- [77] Wenli Xiao, Yiwei Lyu, and John Dolan. Model-based dynamic shielding for safe and efficient multi-agent reinforcement learning. *arXiv preprint arXiv:2304.06281*, 2023.
- [78] Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pages 11480–11491. PMLR, 2021.
- [79] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.
- [80] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35: 24611–24624, 2022.
- [81] Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International conference on machine learning*, pages 25636–25655. PMLR, 2022.
- [82] Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2020.
- [83] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *International conference on machine learning*, pages 5872–5881. PMLR, 2018.
- [84] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.
- [85] Songyuan Zhang, Kunal Garg, and Chuchu Fan. Neural graph control barrier functions guided distributed collision-avoidance multi-agent control. In *Conference on Robot Learning*, pages 2373–2392. PMLR, 2023.
- [86] Songyuan Zhang, Oswin So, Mitchell Black, and Chuchu Fan. Discrete GCBF proximal policy optimization for multi-agent safe optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [87] Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. GCBF+: A neural graph control barrier function framework for distributed safe multiagent control. *IEEE Transactions on Robotics*, 41:1533–1552, 2025.
- [88] Wenbo Zhang, Osbert Bastani, and Vijay Kumar. Mamps: Safe multi-agent reinforcement learning via model predictive shielding. *arXiv preprint arXiv:1910.12639*, 2019.
- [89] Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.
- [90] Youpeng Zhao, Yaodong Yang, Zhenbo Lu, Wengang Zhou, and Houqiang Li. Multi-agent first order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 36, 2024.
- [91] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay,

- and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.
- [92] Edward L Zhu, Yvonne R Stürz, Ugo Rosolia, and Francesco Borrelli. Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 6198–6203. IEEE, 2020.

## APPENDIX A PROOF OF PROPOSITION 1

*Proof:* Under the dynamics  $x^{k+1} = f(x^k, \pi(x^k))$ , we have

$$\begin{aligned}
V(x^k, z^k; \pi) &= \max \left\{ \max_{p \geq k} h(x^p), \sum_{p \geq k} l(x^p, \pi(x^p)) - z^k \right\} \\
&= \max \left\{ \max\{h(x^k), \max_{p \geq k+1} h(x^p)\}, \sum_{p \geq k+1} l(x^p, \pi(x^p)) + l(x^k, \pi(x^k)) - z^k \right\} \\
&= \max \left\{ \max\{h(x^k), \max_{p \geq k+1} h(x^p)\}, \sum_{p \geq k+1} l(x^p, \pi(x^p)) - \underbrace{[z^k - l(x^k, \pi(x^k))]}_{:=z^{k+1}} \right\} \\
&= \max \left\{ h(x^k), \max \left\{ \max_{p \geq k+1} h(x^p), \sum_{p \geq k+1} l(x^p, \pi(x^p)) - z^{k+1} \right\} \right\} \\
&= \max \{h(x^k), V(x^{k+1}, z^{k+1}; \pi)\},
\end{aligned} \tag{16}$$

where we have defined  $z^{k+1} = z^k - l(x^k, \pi(x^k))$  in the third equation. ■

## APPENDIX B PROOF OF THEOREM 1

To prove Theorem 1, first, we prove several lemmas:

*Lemma 1:* For any fixed state  $x$ , let  $z^*$  denote the solution of (15), i.e.,

$$\min_z z, \tag{17a}$$

$$\text{s.t. } V^h(x; \pi(\cdot, z)) \leq 0, \tag{17b}$$

and let  $\pi^*$  denote  $\pi(\cdot, z^*)$ , i.e., it is the optimal policy for  $z^*$ :

$$\pi^* = \arg \min_{\pi} V(x, z^*; \pi). \tag{18}$$

Then, no other safe policy  $\tilde{\pi}$  exists that has a strictly lower cost than  $\pi^*$  while satisfying the constraints, i.e.,

$$V^h(x; \tilde{\pi}) \leq 0 \tag{19a}$$

$$V^l(x; \tilde{\pi}) < V^l(x; \pi^*). \tag{19b}$$

In other words,  $\pi^*$  is the optimal solution of the original constrained optimization problem

$$\min_{\pi} V^l(x; \pi), \tag{20a}$$

$$\text{s.t. } V^h(x; \pi) \leq 0. \tag{20b}$$

Before proving this lemma, we first prove the following lemma.

*Lemma 2:* Suppose that such a  $\tilde{\pi}$  exists. Then, there exists a  $z^\dagger := V^l(x; \tilde{\pi}) - V^h(x; \tilde{\pi})$  for which the optimal policy  $\pi^\dagger$  for  $z^\dagger$  satisfies the conditions for  $\tilde{\pi}$  in (19a) and (19b), i.e.,

$$V^h(x; \pi^\dagger) \leq V^h(x; \tilde{\pi}) \leq 0, \tag{21a}$$

$$V^l(x; \pi^\dagger) \leq V^l(x; \tilde{\pi}) < V^l(x; \pi^*). \tag{21b}$$

*Proof:* Since  $\pi^\dagger$  is optimal for  $z^\dagger$ , we have that

$$V(x, z^\dagger; \pi^\dagger) \leq V(x, z^\dagger; \tilde{\pi}). \tag{22}$$

This implies that, by definition of  $z^\dagger$ ,

$$\max \left\{ V^h(x; \pi^\dagger), V^l(x; \pi^\dagger) - z^\dagger \right\} \leq \max \left\{ V^h(x; \tilde{\pi}), V^l(x; \tilde{\pi}) - z^\dagger \right\}, \quad (23)$$

$$= V^h(x; \tilde{\pi}). \quad (24)$$

In particular,

$$V^h(x; \pi^\dagger) \leq V^h(x; \tilde{\pi}), \quad (25)$$

and

$$V^l(x; \pi^\dagger) - \left( V^l(x; \tilde{\pi}) - V^h(x; \tilde{\pi}) \right) \leq V^h(x; \tilde{\pi}), \quad (26)$$

$$\implies V^l(x; \pi^\dagger) \leq V^l(x; \tilde{\pi}). \quad (27)$$

which proves (21a) and (21b).  $\blacksquare$

We are now ready to prove Lemma 1.

*Proof of Lemma 1:* We prove this by contradiction.

Suppose that such a  $\tilde{\pi}$  exists. By Lemma 2, there exists  $z^\dagger$  and  $\pi^\dagger$  that satisfies the conditions for  $\tilde{\pi}$  in (19a) and (19b). Since  $\pi^*$  is optimal for  $z^*$ , this implies that

$$\max \left\{ V^h(x; \pi^*), V^l(x; \pi^*) - z^* \right\} \leq \max \left\{ V^h(x; \pi^\dagger), V^l(x; \pi^\dagger) - z^\dagger \right\}. \quad (28)$$

We now consider two cases depending on the value of the max on the right.

**Case 1** ( $V^h(x; \pi^\dagger) \leq V^l(x; \pi^\dagger) - z^*$ ): For this case,  $\max \left\{ V^h(x; \pi^\dagger), V^l(x; \pi^\dagger) - z^* \right\} = V^l(x; \pi^\dagger) - z^*$ . This implies that

$$V^l(x; \pi^*) - z^* \leq V^l(x; \pi^\dagger) - z^* \iff V^l(x; \pi^*) \leq V^l(x; \pi^\dagger). \quad (29)$$

However, this contradicts our assumption that  $V^l(x; \pi^\dagger) \leq V^l(x; \tilde{\pi}) < V^l(x; \pi^*)$  from (19b).

**Case 2** ( $V^h(x; \pi^\dagger) > V^l(x; \pi^\dagger) - z^*$ ): For this case,  $\max \left\{ V^h(x; \pi^\dagger), V^l(x; \pi^\dagger) - z^* \right\} = V^h(x; \pi^\dagger)$ . This implies that

$$V^h(x; \pi^*) \leq V^h(x; \pi^\dagger) \quad (30)$$

and

$$V^l(x; \pi^*) - z^* \leq V^h(x; \pi^\dagger) \implies V^l(x; \pi^*) - V^h(x; \pi^\dagger) \leq z^*. \quad (31)$$

However, if we examine the definition of  $z^\dagger$ , we have that

$$z^\dagger = V^l(x; \tilde{\pi}) - V^h(x; \tilde{\pi}) \quad (32)$$

$$\leq V^l(x; \tilde{\pi}) - V^h(x; \pi^\dagger) \quad (\text{from (21a) and (30)}) \quad (33)$$

$$< V^l(x; \pi^*) - V^h(x; \pi^\dagger) \quad (\text{from (19b)}) \quad (34)$$

$$\leq z^* \quad (\text{from (31)}). \quad (35)$$

This contradicts our definition of  $z^*$  being the optimal solution of (17), since  $z^\dagger$  satisfies  $V^h(x; \pi(\cdot, z^\dagger)) \leq 0$  but is also strictly smaller than  $z^*$ .  $\blacksquare$

Since both cases lead to a contradiction, no such  $\tilde{\pi}$  can exist.  $\blacksquare$

We also prove the following lemma.

*Lemma 3:* For any fixed state  $x$ , let  $z^*$  denote the solution of (15), i.e.,

$$\min_z z, \quad (36a)$$

$$\text{s.t. } V^h(x; \pi(\cdot, z)) \leq 0, \quad (36b)$$

and let  $\pi_{z^*}$  denote  $\pi(\cdot, z^*)$ , i.e., it is the optimal policy for  $z^*$ :

$$\pi_{z^*} = \arg \min_{\pi} V(x, z^*; \pi). \quad (37)$$

Assuming that there does not exist another  $z$  such that  $V^l(x; \pi_z) = V^l(x; \pi_{z^*})$ . Then for any  $\epsilon \geq 0$ ,

$$V^h(x; \pi_{z^*+\epsilon}) \leq V^h(x; \pi_{z^*}). \quad (38)$$

*Proof:* Since  $\pi_{z^*+\epsilon}$  is optimal for  $z = z^* + \epsilon$ ,

$$\max\{V^h(x; \pi_{z^*+\epsilon}), V^l(x; \pi_{z^*} + \epsilon) - (z^* + \epsilon)\} \leq \max\{V^h(x; \pi_{z^*}), V^l(x; \pi_{z^*}) - (z^* + \epsilon)\}. \quad (39)$$

For the max on the right-hand side, if  $V^h(x; \pi_{z^*}) \geq V^l(x; \pi_{z^*}) - (z^* + \epsilon)$ , then we immediately obtain our desired result

$$V^h(x; \pi_{z^*+\epsilon}) \leq V^h(x; \pi_{z^*}). \quad (40)$$

We thus suppose that  $V^h(x; \pi_{z^*}) \geq V^l(x; \pi_{z^*}) - (z^* + \epsilon)$ , and obtain that

$$V^h(x; \pi_{z^*+\epsilon}) \leq V^l(x; \pi_{z^*}) - (z^* + \epsilon), \quad (41a)$$

$$V^l(x; \pi_{z^*+\epsilon}) \leq V^l(x; \pi_{z^*}). \quad (41b)$$

Now, since  $\pi_{z^*}$  is optimal for  $z = z^*$ ,

$$\max\{V^h(x; \pi_{z^*}), V^l(x; \pi_{z^*}) - z^*\} \leq \max\{V^h(x; \pi_{z^*+\epsilon}), V^l(x; \pi_{z^*+\epsilon}) - z^*\}. \quad (42)$$

We now split into two cases depending on the max on the right-hand side.

**Case 1** ( $V^h(x; \pi_{z^*+\epsilon}) \geq V^l(x; \pi_{z^*+\epsilon}) - z^*$ ): This implies that

$$V^l(x; \pi_{z^*}) - z^* \leq V^h(x; \pi_{z^*+\epsilon}), \quad (43)$$

hence,

$$V^h(x; \pi_{z^*+\epsilon}) \leq V^l(x; \pi_{z^*}) - (z^* + \epsilon) \leq V^h(x; \pi_{z^*+\epsilon}) - \epsilon, \quad (44)$$

which is a contradiction, so this case does not occur.

**Case 2** ( $V^h(x; \pi_{z^*+\epsilon}) < V^l(x; \pi_{z^*+\epsilon}) - z^*$ ): This implies that

$$V^l(x; \pi_z^*) - z^* \leq V^l(x; \pi_{z^*+\epsilon}) - z^*, \quad (45)$$

hence,  $V^l(x; \pi_z^*) \leq V^l(x; \pi_{z^*+\epsilon})$ . Combining this with (41b) gives us that  $V^l(x; \pi_z^*) = V^l(x; \pi_{z^*+\epsilon})$ . However, from our assumption, this implies that  $\pi_z^* = \pi_{z^*+\epsilon}$  and thus our desired result of  $V^h(x; \pi_{z^*+\epsilon}) \leq V^h(x; \pi_{z^*})$ .

■

We can now prove Theorem 1, which follows as a consequence of Lemma 1 and Lemma 3.

*Proof of Theorem 1:* Since  $V^h(x; \pi) = \max_i V_i^h(x_i, o_i; \pi)$ , Lemma 1 implies that Equation (13) is equivalent to

$$z^* := \min_z \{z \mid \max_i V_i^h(x_i, o_i; \pi(\cdot, z)) \leq 0\}. \quad (46)$$

We now show that this is equivalent to the following distributed implementation (equal to (15)):

$$z_i := \min\{z \mid V_i^h(x_i, o_i; \pi(\cdot, z)) \leq 0\}, \quad (47)$$

$$z_{\text{distr}} = \max_i z_i. \quad (48)$$

We will now prove equality via a double inequality proof.

$(z_{\text{distr}} \leq z^*)$ : By definition of  $z^*$ ,

$$V_i^h(x_i, o_i; \pi(\cdot, z^*)) \leq 0, \quad \forall i. \quad (49)$$

However, since  $z_i$  (47) is optimal,  $z_i \leq z^*$ . Hence,

$$z_{\text{distr}} = \max_i z_i \leq z^*. \quad (50)$$

$(z_{\text{distr}} \geq z^*)$ : By definition of  $z_{\text{distr}}$ ,

$$z_{\text{distr}} \geq z_i, \quad \forall i. \quad (51)$$

Using Lemma 3, this implies that

$$V_i^h(x_i, o_i; \pi(\cdot, z_{\text{distr}})) \leq 0, \quad \forall i. \quad (52)$$

Hence,  $z_{\text{distr}}$  satisfies  $\max_i V_i^h(x_i, o_i; \pi(\cdot, z_{\text{distr}})) \leq 0$ . Since  $z^*$  is the smallest  $z$  that still satisfies this constraint,

$$z^* \leq z_{\text{distr}}. \quad (53)$$

We have thus proved that  $z^* = z_{\text{distr}}$ , i.e.,  $z^*$  can be computed in a distributed fashion. ■

## APPENDIX C DISCUSSION ON IMPORTANCE OF PROPOSITION 1

Establishing Proposition 1 is *key* to Def-MARL. Namely,

- 1) Satisfying dynamic programming implies that the value function is *Markovian*. In other words, for a given  $z^0$ , the value at the  $k$ th timestep is *only* a function of  $z^k$  and  $x^k$  instead of the  $z^0$  and the *entire* trajectory up to the  $k$ th timestep.
- 2) Consequently, this implies that the optimal policy will also be Markovian and is *only* a function of  $z^k$  and  $x^k$ .
- 3) Rephrased differently, since the value function is Markovian, this implies that, for a given  $z^0$  and  $x^0$ , the value at the  $k$ th timestep is *equal* to the value (at the initial timestep) of a *new* problem where we start with  $\tilde{z}^0 = z^k$  and  $\tilde{x}^0 = x^k$ .
- 4) Since we relate the value function of consecutive timesteps, given a value function estimator, we can now control the bias-variance tradeoff of the value function estimate by using  $k$ -step estimates instead of the Monte Carlo estimates.
- 5) Instead of only using the  $k$ -step estimates for a single choice of  $k$ , we can compute a weighted average of the  $k$ -step estimates as in GAE to further control the bias-variance tradeoff.

## APPENDIX D ALGORITHM PSEUDOCODE

We describe the centralized training process of Def-MARL in Algorithm 1 and the distributed execution process in Algorithm 2.

---

### Algorithm 1 Def-MARL centralized training

---

**Initialize:** Policy NN  $\pi_\theta$ , cost value function NN  $V_\phi^l$ , constraint value function NN  $V_\psi^h$ .

**while** Training not end **do**

    Randomly sampling initial conditions  $x^0$ , and the initial  $z^0 \in [z_{\min}, z_{\max}]$ .

    Use  $\pi_\theta$  to sample trajectories  $\{x^0, \dots, x^T\}$ , with  $z$  dynamics (14).

    Calculate the cost value function  $V_\phi^l(x, z)$  and the constraint value function  $V_\psi^h(o_i, z)$ .

    Calculate GAE with the total value function (12).

    Update the value functions  $V_\phi^l$  and  $V_\psi^h$  using TD error.

    Update the  $z$ -conditioned policy  $\pi_\theta(\cdot, z)$  using PPO loss.

**end while**

---

## APPENDIX E SIMULATION EXPERIMENTS

### A. Computation resources

The experiments are run on a 13th Gen Intel(R) Core(TM) i7-13700KF CPU with 64GB RAM and an NVIDIA GeForce RTX 3090 GPU. The training time is around 6 hours ( $10^5$  steps) for Def-MARL and Lagr, and around 5 hours for Penalty.

---

**Algorithm 2** Def-MARL distributed execution

---

**Input:** Learned policy NN  $\pi_\theta$ , constraint value function NN  $V_\psi^h$ .  
**for**  $k = 0, \dots, T$  **do**  
    Get  $z_i$  for each agent by solving the distributed EF-MASOCP outer problem (15b).  
    **if**  $z$  communication enabled **then**  
        The connected agents  $j$  communicate  $z_j$  and reach a consensus  $z = \max_j z_j$ .  
        Set  $z_i = z$  for all agents in the connected graph.  
    **end if**  
    Get decentralized policy  $\pi_i(\cdot) = \pi_\theta(\cdot, z_i)$ .  
    Execute control  $u_i^k = \pi_i(o_i^k)$ .  
**end for**

---

### B. Environments

1) *Multi-partical environments (MPE)*: We use directed graphs  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  to represent MPE, where  $\mathcal{V}$  is the set of nodes containing the objects in the multi-agent environment (e.g., agents  $\mathcal{V}_a$ , goals  $\mathcal{V}_g$ , landmarks  $\mathcal{V}_l$ , and obstacles  $\mathcal{V}_o$ ).  $\mathcal{E} \subseteq \{(i, j) \mid i \in \mathcal{V}_a, j \in \mathcal{V}\}$  is the set of edges, denoting the information flow from a sender node  $j$  to a receiver agent  $i$ . An edge  $(i, j)$  exists only if the communication between node  $i$  and  $j$  can happen, which means the distance between node  $i$  and  $j$  should be within the communication radius  $R$  in partially observable environments. We define the neighborhood of agent  $i$  as  $\mathcal{N}_i := \{j \mid (i, j) \in \mathcal{E}\}$ . The node feature  $v_i$  includes the states of the node  $x_i$  and a one-hot encoding of the type of the node  $i$  (e.g., agent, goal, landmark, or obstacle), e.g.,  $[0, 0, 1]^\top$  for agent nodes,  $[0, 1, 0]^\top$  for goal nodes, and  $[1, 0, 0]^\top$  for obstacle nodes. The edge feature  $e_{ij}$  includes the information passed between the sender node  $j$  and the receiver node  $i$  (e.g., relative positions and velocities).

We consider 6 MPE: TARGET, SPREAD, FORMATION, LINE, CORRIDOR, and CONNECTSPREAD. In each environment, the agents need to work collaboratively to finish some tasks:

- TARGET [51]: Each agent tries to reach its preassigned goal.
- SPREAD [17]: The agents are given a set of (not preassigned) goals to cover.
- FORMATION [2]: Given a landmark, the agents should spread evenly on a circle with the landmark as the center and a given radius.
- LINE [2]: Given two landmarks, the agents should spread evenly on the line between the landmarks.
- CORRIDOR: A set of agents and goals are separated by a narrow corridor, whose width is smaller than  $4r_a$  where  $r_a$  is the radius of agents. The agents should go through the corridor and cover the goals.
- CONNECTSPREAD: A set of agents and goals are separated by a large obstacle with a diameter larger than the communication radius  $R$ . The agents should cover the goals without colliding with obstacles or each other while also maintaining the connectivity of all agents.

We consider  $N = 3$  agents for all environments and  $N = 5$  and 7 agents in the FORMATION and LINE environments. To make the environments more difficult than the original ones [51], we add 3 static obstacles in the first 4 environments.

In our modified MPE, the state of the agent  $i$  is given by  $x_i = [p_i^x, p_i^y, v_i^x, v_i^y]^\top$ , where  $[p_i^x, p_i^y]^\top := p \in \mathbb{R}^2$  is the position of agent  $i$ , and  $[v_i^x, v_i^y]$  is the velocity. The control inputs are given by  $u_i = [a_i^x, a_i^y]^\top$ , i.e., the acceleration along each axis. The joint state is defined by concatenation:  $x = [x_1; \dots; x_N]$ . The agents are modeled as double integrators with dynamics

$$\dot{x}_i = [v_i^x \quad v_i^y \quad a_i^x \quad a_i^y]^\top. \quad (54)$$

The agents' control inputs are limited by  $[-1, 1]$ , and the velocities are limited by  $[-1, 1]$ . The agents have a radius  $r_a = 0.05$ , and the communication radius is assumed to be  $R = 0.5$ . The area side length  $L$  is 1.0 for the CORRIDOR and the CONNECTSPREAD environments and 1.5 for other environments. The radius of the obstacles  $r_o$  is 0.4 in the CORRIDOR environment, 0.25 in the CONNECTSPREAD environment, and 0.05 for other environments. All environments use a simulation time step of 0.03s and a total horizon of  $T = 128$ .

The observation of the agents  $o_i$  includes the node features of itself, its neighbors  $j \in \mathcal{N}_i$ , and the edge features of the edge connecting agent  $i$  and its neighbors. The node features include neighbors' states  $x_j$  and its type ( $[0, 0, 1]$  for agents,  $[0, 1, 0]$  for goals and landmarks, and  $[1, 0, 0]^\top$  for the obstacles). The edge features are the relative states  $e_{ij} = x_i - x_j$ .

The constraint function  $h$  contains two parts for all environments except for CONNECTSPREAD, including agent-agent and agent-obstacles collisions. In the CONNECTSPREAD environment, another constraint regarding the connectivity of the agent graph is considered. For the agent-agent collision, we use the  $h$  function defined as

$$h_a(o_i) = 2r_a - \min_{j \in \mathcal{N}_i} \|p_i - p_j\| + \nu \text{sign} \left( 2r_a - \min_{j \in \mathcal{N}_i} \|p_i - p_j\| \right), \quad (55)$$

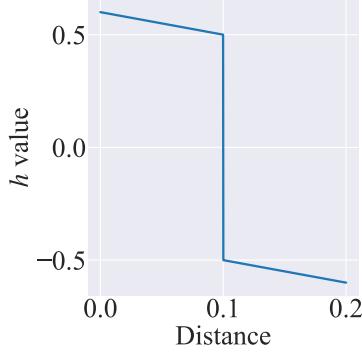


Fig. 12:  $h$  value with respect to distance.

where  $\text{sign}$  is the sign function, and  $\nu = 0.5$  in all our experiments. This represents a linear function w.r.t. the inter-agent distance with a discontinuity at the safe-unsafe boundary (Fig. 12). For the agent-obstacle collision, we use

$$h_o(o_i) = r_a + r_o - \min_{j \in \mathcal{N}_i^o} \|p_i - p_j\| + \nu \text{sign} \left( r_a + r_o - \min_{j \in \mathcal{N}_i^o} \|p_i - p_j\| \right), \quad (56)$$

where  $\mathcal{N}_i^o$  is the observed obstacle set of agent  $i$ . Then, the total  $h$  function is defined as  $h(o_i) = \max\{h_a(o_i), h_o(o_i)\}$  for environments except for CONNECTSPREAD. For CONNECTSPREAD, we also consider the connectivity constraint

$$h_c(o_i) = \max_i \min_{j \in \mathcal{N}_i^c} \|p_i - p_j\| - R' + \nu \text{sign} \left( \max_i \min_{j \in \mathcal{N}_i^c} \|p_i - p_j\| - R' \right), \quad (57)$$

where  $R' = 0.45$  is the required maximum distance for connected agents such that if the distance between two agents is larger than  $R'$ , they are considered disconnected. Note that this cost is only valid with agent number  $N \leq 3$ . For a larger number of agents, the second-largest eigenvalue of the graph Laplacian matrix can be used. Still, since we only use this environment with 3 agents, we use this cost to decrease the complexity. Then, the total  $h$  function of the CONNECTSPREAD environment is defined as  $h(o_i) = \max\{h_a(o_i), h_o(o_i), h_c(o_i)\}$ .

Two types of cost functions are used in the environments. The first type is the *Target* cost used in the TARGET environment, which is defined as

$$l(x, u) = \frac{1}{N} \sum_{i=1}^N \left( 0.01 \|p_i - p_i^{\text{goal}}\| + 0.001 \text{sign} \left( \text{ReLU}(\|p_i - p_i^{\text{goal}}\| - 0.01) \right) + 0.0001 \|u_i\|^2 \right). \quad (58)$$

The first term penalizes the agents if they cannot reach the goal, the second term penalizes the agents if they cannot reach the goal exactly, and the third term encourages small controls. The second type is the *Spread* cost used in all other environments, defined as

$$l(x, u) = \frac{1}{N} \sum_{j=1}^N \min_{i \in \mathcal{V}_a} \left( 0.01 \|p_i - p_j^{\text{goal}}\| + 0.001 \text{sign} \left( \text{ReLU}(\|p_i - p_j^{\text{goal}}\| - 0.01) \right) + 0.0001 \|u_j\|^2 \right). \quad (59)$$

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

2) *Safe multi-agent MuJoCo environments*: We also test on the SAFE HALFCHEETAH(2x3) and SAFE COUPLED HALFCHEETAH(4x3) tasks from the Safe Multi-Agent Mujoco benchmark suite [32]. Each agent controls a subset of joints and must cooperate to minimize the cost (which we take to be the negative of the reward in the original work) while avoiding violating safety constraints. The task is parametrized by the two numbers in the parentheses, where the first number denotes the number of agents, while the second number denotes the number of joints controlled by each agent. The goal for the SAFE HALFCHEETAH and SAFE COUPLED HALFCHEETAH tasks is to maximize the forward velocity but avoid colliding with a wall in front that moves forward at a predefined velocity.

**Note:** Although this is not a homogeneous MAS, since each agent has the same control space (albeit with different dynamics), we can convert this into a homogeneous MAS by *augmenting* the state space with a one-hot vector to identify each agent, then augmenting the dynamics to use the appropriate per-agent dynamics function. This is the approach taken in the official implementation of Safe Multi-Agent Mujoco from Gu et al. [32]. For more details, see Gu et al. [32].

### C. Implementation details and hyperparameters

We parameterize the  $z$ -conditioned policy  $\pi_\theta(o_i, z)$ , cost-value function  $V_\phi^l(x, z)$ , and the constraint-value function  $V_\psi^h(o_i, z)$  using graph transformers [65] with parameters  $\theta$ ,  $\phi$ , and  $\psi$ , respectively. Note that the policy and the constraint-value function are decentralized and take only the local observation  $o_i$  as input, while the cost-value function is centralized. In each layer of the graph transformer, the node features are updated with  $v'_i = W_1 v_i + \sum_{j \in \mathcal{N}_i} \alpha_{ij} (W_2 v_j + W_3 e_{ij})$ , where  $W_i$  are learnable weight matrices, and the  $\alpha_{ij}$  is the attention weight between agent  $i$  and agent  $j$  computed as  $\alpha_{ij} = \text{softmax}(\frac{1}{\sqrt{c}}(W_4 x_i)^\top (W_5 x_j))$ , where  $c$  is the first dimension of  $W_i$ . In this way, the observation  $o_i$  is encoded. If the environment allows  $M$ -hop information passing, we can apply the node feature update  $M$  times so that agent  $i$  can receive information from its  $M$ -hop neighbors. After the information passing, the updated node features  $v'_i$  are concatenated with the encoded  $z$  vector  $W_7 z$ , then passed to another NN or a recurrent neural network (RNN) [34] to obtain the outputs.  $\pi_\theta$  and  $V_\psi^h$  have the same structure as introduced above with different output dimensions because they are decentralized. For the centralized  $V^l(x, z)$ , the averaged node features after information passing are concatenated with the encoded  $z$  and passed to the final layer (NN or RNN) to obtain the global cost value for the whole MAS.

When updating the neural networks, we follow the PPO [63] structure. First, we calculate the target cost-value function  $V_{\text{target}}^l$  and the target constraint-value function  $V_{\text{target}}^h$  using GAE estimation [62], and then backpropagate the following mean-square error to update the value function parameters  $\phi$  and  $\psi$ :

$$\mathcal{L}_{V^l}(\phi) = \frac{1}{M} \sum_{k=1}^M \|V_\phi^l(x^k, z^k) - V_{\text{target}}^l(x^k, z^k)\|^2, \quad (60)$$

$$\mathcal{L}_{V^h}(\psi) = \frac{1}{MN} \sum_{k=1}^M \sum_{i=1}^N \|V_\psi^h(o_i^k, z^k) - V_{\text{target}}^h(o_i^k, z^k)\|^2, \quad (61)$$

where  $M$  is the number of samples. Then, we calculate the advantages  $A_i$  for each agent with the total value function  $V_i(x, z) = \max\{V_\phi^l(x, z) - z, V_\psi^h(o_i, z)\}$  following the same process as in PPO by replacing  $V^l$  with  $V$ , and backpropagate the following PPO policy loss to update the policy parameters  $\theta$ :

$$\mathcal{L}_\pi(\theta) = \frac{1}{MN} \sum_{k=1}^M \sum_{i=1}^N \left[ \min \left\{ \frac{\pi_\theta(o_i^k, z^k)}{\pi_{\text{old}}(o_i^k, z^k)} A_i(x^k, z^k), \text{clip} \left( \frac{\pi_\theta(o_i^k, z^k)}{\pi_{\text{old}}(o_i^k, z^k)}, 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}} \right) A_i(x^k, z^k) \right\} \right]. \quad (62)$$

Most of the hyperparameters of Def-MARL are shared with Penalty and Lagr. The values of the share hyperparameters are provided in Table IV.

TABLE IV: Shared hyperparameters of Def-MARL, Penalty, and Lagr.

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

Apart from the shared hyperparameters, Def-MARL has additional hyperparameters, as shown in Table V. In addition,  $z_{\min}$  and  $z_{\max}$  are the lower and upper bounds of  $z$  while sampling  $z$  in training. Since  $z_{\min}$  represents an estimate of the minimum cost incurred by the MAS, we set it to a small negative number  $-0.5$ . We set  $z_{\max}$  differently depending on the complexity of the environment. For MPE, with maximum simulation timestep  $T$ , we estimate it in the MPE environments using the following equation:

$$z_{\max} = \tilde{l}_{\max} * T, \quad (63)$$

$$\tilde{l}_{\max} = \text{initdist}_{\max} w_{\text{distance}} + w_{\text{reach}} + u_{\max} w_{\text{control}}, \quad (64)$$

where  $\tilde{l}_{\max}$  is a conservative estimate of the maximum cost  $l$ . This is conservative in the sense that this reflects the case where 1) the agents and goals are initialized with the maximum possible distance ( $\text{initdist}_{\max}$ ); 2) the agents do not reach their goal throughout their trajectory; 3) the agents incur the maximum control cost for all timesteps.  $w_{\text{distance}}$ ,  $w_{\text{reach}}$ , and

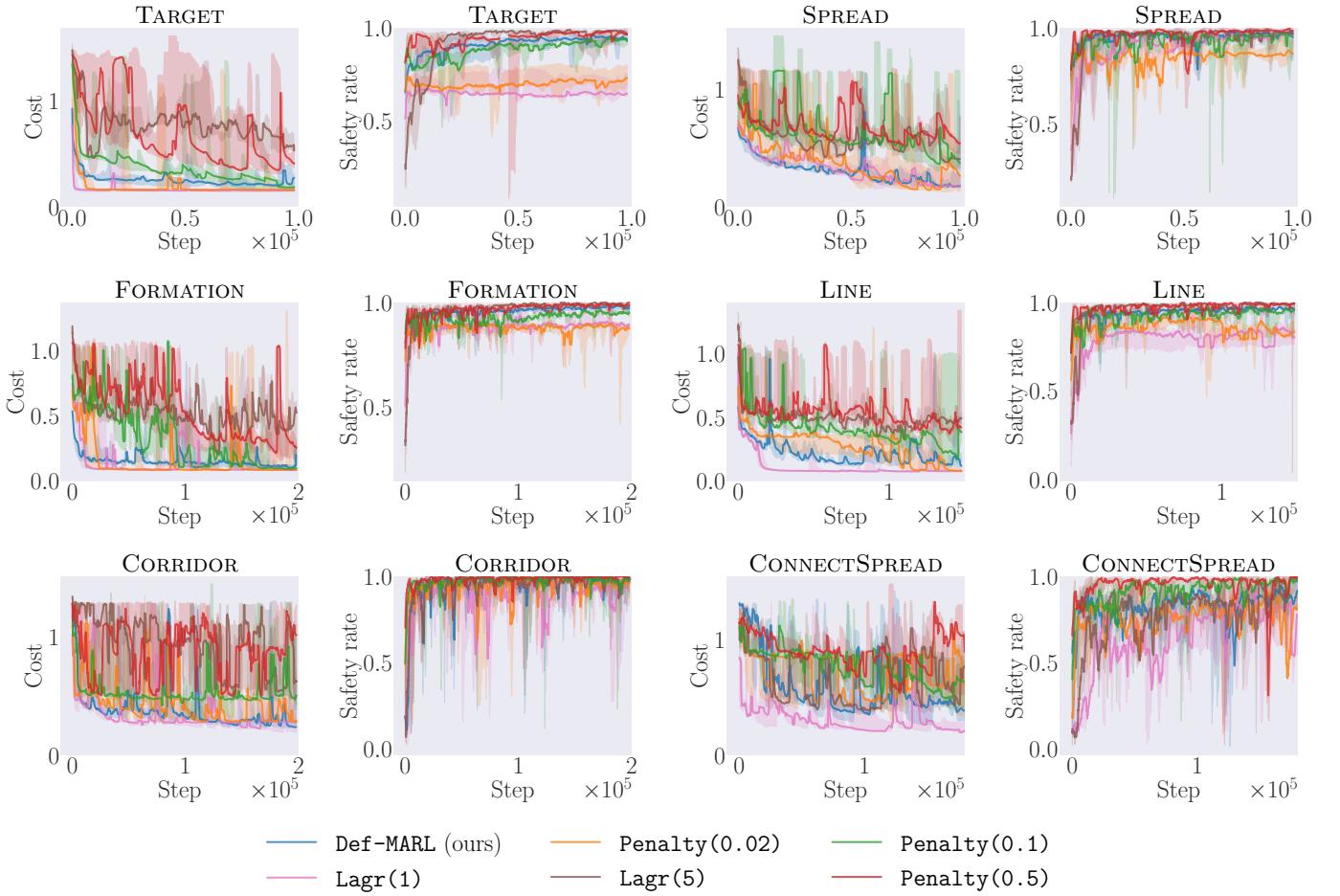


Fig. 13: Cost and safety rate of Def-MARL and the baselines during training in MPE.

$w_{\text{control}}$  denote the corresponding weights of the different cost terms in the cost function  $l$  in (58) and (59). For the multi-agent MuJoCo environments, we first train the agents with (unconstrained) MAPPO with different random seeds, record the largest cost incurred, double it, and then use that as  $z_{\max}$ .

TABLE V: Hyperparameters of Def-MARL.

Hyperparameter	Value
$V^h$ GNN layers	2 for ConnectSpread, 1 for others
$z$ encoding dimension	8
outer problem solver	Chandrupatla's method [12]

All the hyperparameters remain the same in all environments or are pointed out in the tables except for the training steps. The training step is  $10^5$  in the TARGET and the SPREAD environments,  $1.5 \times 10^5$  in the LINE environment, and  $2 \times 10^5$  in other MPE. For the Safe Multi-agent MuJoCo environments, we set the training step to  $7 \times 10^3$ .

#### D. Training curves

To show the training stability of Def-MARL, we have shown the cost and safety rate of Def-MARL and Lagr(1r) during training in the TARGET and SPREAD environments in the main pages (Fig. 6). Due to page limits, we provide the plots for other environments here in Fig. 13, Fig. 14, and Fig. 15. The figures show that Def-MARL achieves stable training in all environments. Specifically, as shown in Fig. 14, while Lagr(1r) suffers from training instability because the constraint violation threshold is zero (as discussed in Section III-B), Def-MARL is much more stable.

#### E. More comparison with the Lagrangian method

In this section, we provide more comparisons between Def-MARL and the Lagrangian method, where we change the constraint-value function of the Lagrangian method from the sum-over-time (SoT) form to the max-over-time (MoT) form.

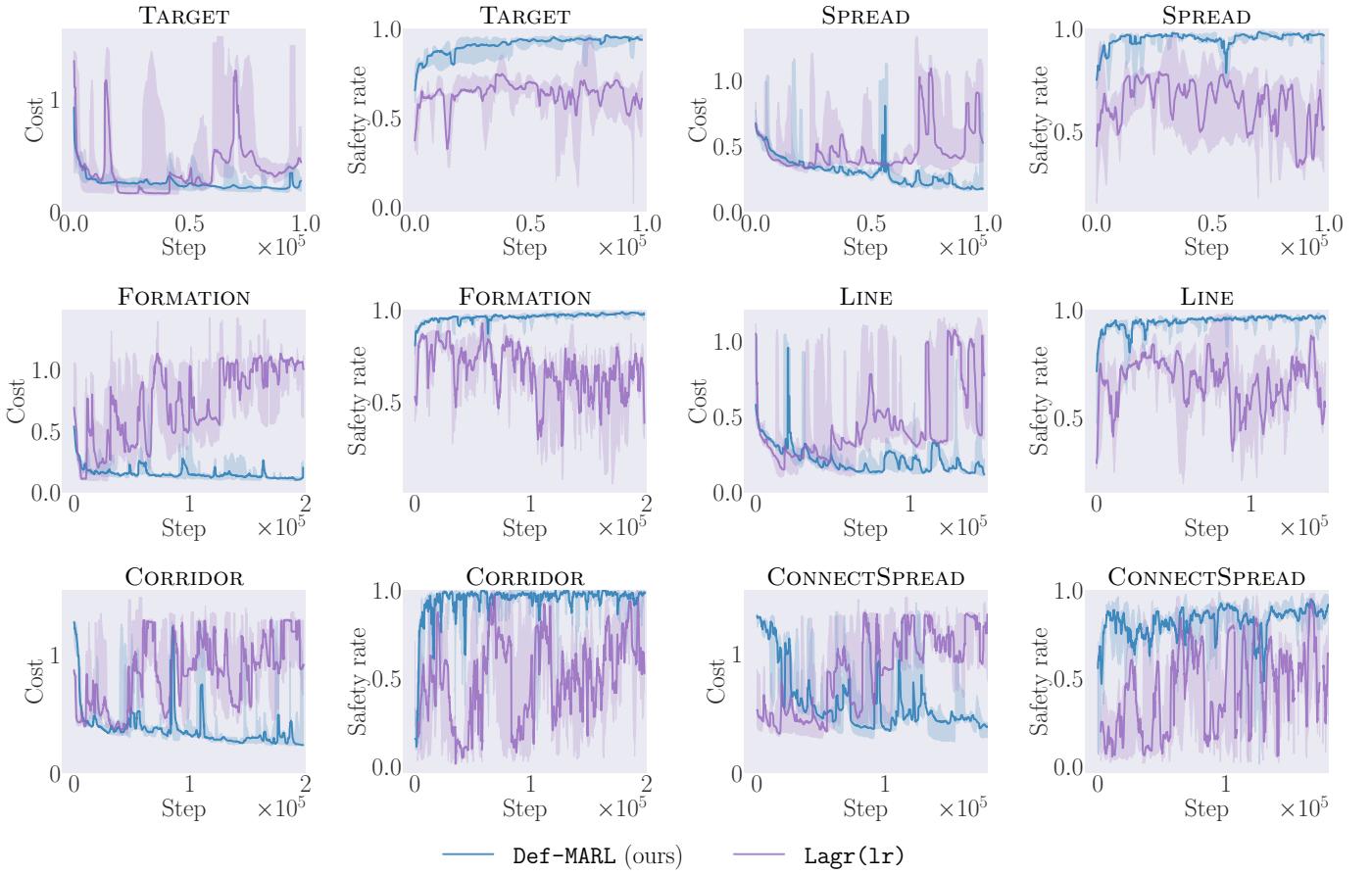


Fig. 14: Cost and safety rate of Def-MARL and Lagr(lr) during training in MPE.

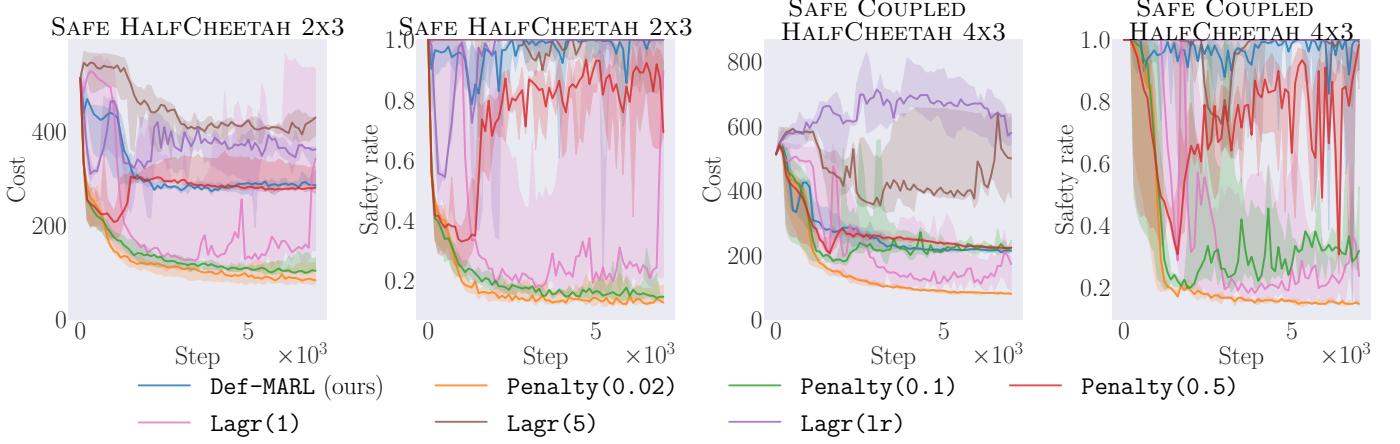


Fig. 15: Cost and safety rate of Def-MARL and all baselines during training in Safe Multi-agent MuJoCo environments.

Using the MoT form, the constraint-value function of the Lagrangian method becomes the same as the one used in Def-MARL (Equation (9)). We create 3 more baselines using this approach with different learning rates (lr) of the Lagrangian multiplier  $\lambda$ , where  $lr(\lambda) \in \{0.1, 0.2, 0.3\}$ . The baselines are called Lagr-MoT. We compare Def-MARL with the new baselines in the TARGET environment, and the results are presented in Fig. 16. We can observe that the Lagrangian method has very different performance with different learning rates of  $\lambda$ . With  $lr(\lambda) = 0.1$ , the learned policy is unsafe, and with  $lr(\lambda) = 0.2$  or  $0.3$ , the training is unstable and the cost of the converged policy is much higher than Def-MARL. In addition, we also plot the  $\lambda$  values during training in Fig. 16. It shows that  $\lambda$  keeps increasing without converging to some value, which also suggests the

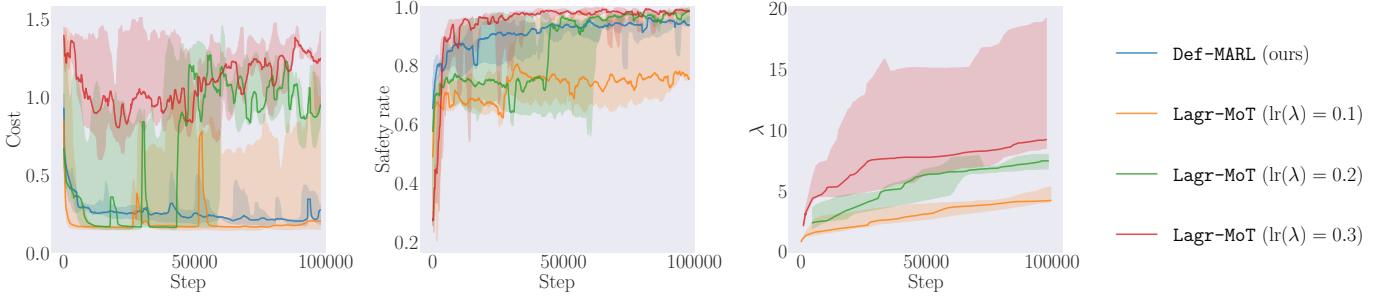


Fig. 16: Cost and safety rate of Def-MARL and Lagr-MoT with different learning rates of  $\lambda$  during training in the TARGET environment, and the  $\lambda$  values during training.

instability of the Lagrangian method.

#### F. Sensitivity analysis on the choice of $z_{\max}$

In Appendix E-C, we have introduced how to determine the sampling interval of  $z$ . Here, we perform experiments in the SPREAD environment to study the sensitivity of Def-MARL on the choice of  $z_{\max}$ . In this experiment, we scale the value of  $z_{\max}$  used for sampling  $z$ , and denote by  $z_{\max,\text{orig}}$  the original value used in the experiments in the main pages, i.e.,  $z_{\max}/z_{\max,\text{orig}} = 1.0$  uses the same value as in the main pages. We report the safety rates and the costs of the Def-MARL policies trained with different  $z_{\max}$  in Table VI. We see that both safety and costs do not change much even when our estimate of the maximum cost  $z_{\max}$  changes by up to 50%. If  $z_{\max}$  is too large (e.g.,  $2 z_{\max,\text{orig}}$ ), the policy becomes too conservative because not enough samples of  $z$  that are near  $z^*$  are observed, reducing the sample efficiency. On the other hand, when  $z_{\max}$  is too small (e.g.,  $0.25 z_{\max,\text{orig}}$ ), there may be states where the optimal  $z^*$  does not fall within the sampled range. This causes the rootfinding step to be inaccurate, as  $V^h$  will be queried at values of  $z$  that were not seen during training, resulting in safety violations.

TABLE VI: Safety and Cost of Def-MARL policies trained with different  $z_{\max}$ .

$z_{\max}/z_{\max,\text{orig}}$	Safety rate	Cost
0.25	$93.8 \pm 2.4$	$0.152 \pm 0.100$
0.5	$98.0 \pm 1.4$	$0.155 \pm 0.104$
1.0	$99.0 \pm 0.9$	$0.162 \pm 0.144$
1.5	$99.0 \pm 0.0$	$0.165 \pm 0.100$
2.0	$99.0 \pm 0.1$	$0.228 \pm 0.109$

#### G. Effect of $z_i$ communication with more agents

To test the effect of  $z_i$  communication with more agents, we increase the number of agents to 512 (at constant density) in TARGET environment during the test (Table VII). Without communication, we only see a 0.3% drop in safety rate going from  $N = 32$  to  $N = 512$ . With  $z$  communication, safety does not decrease with increased  $N$ .

TABLE VII: Effect of  $z_i$  communication in larger scale environments.

$N$	No communication ( $z \leftarrow z_i$ )		Communication ( $z = \max_i z_i$ )	
	Safety rate	Cost	Safety rate	Cost
32	$99.8 \pm 0.2$	$-0.387 \pm 0.029$	$99.8 \pm 0.2$	$-0.416 \pm 0.052$
128	$99.6 \pm 0.4$	$-0.408 \pm 0.015$	$99.8 \pm 0.3$	$-0.491 \pm 0.090$
512	$99.5 \pm 0.3$	$-0.410 \pm 0.009$	$99.9 \pm 0.1$	$-0.608 \pm 0.210$

#### H. Code

The code of our algorithm and the baselines are provided on our project website <https://mit-realm.github.io/def-marl/>.

## APPENDIX F HARDWARE EXPERIMENTS

### A. Implementation details

**Hardware platform.** We perform hardware experiments using a swarm of Crazyflie (CF) 2.1 platform [27]. We use two Crazyradios to communicate with the CFs and use the Lighthouse localization system to perform localization with four SteamVR Base Station 2.0. The computation is split into two parts: Onboard computation is performed on the CF microcontroller unit (MCU), and the offboard computation is performed on a laptop connected to the CFs over Crazyradio. We use the crazyswarm2 ROS2 package to communicate with the CFs, and a single ROS2 node for the off-board computations at 50Hz.

**Control framework.** Given a state estimation of the CFs using the Lighthouse system, the laptop inputs the states to the algorithm models (Def-MARL, CMPC, or DMPC), which then outputs the accelerations. Then, the laptop computes the desired next positions, velocities, and accelerations for the CFs, and sends the information to the CF MCU. After receiving the information, the CF MCU uses the onboard PID controller to track the desired position, velocity, and acceleration, which outputs the motor commands. *Distributed* execution is simulated on the laptop by only giving each agent local information as input to their distributed policies.

### B. Tasks

**CORRIDOR.** The CORRIDOR task uses the same cost and constraint function as the simulation CORRIDOR environment, which has been introduced in Appendix E-B.

**INSPECT.** In this task, two CF drones need to work collaboratively to observe a moving target CF drone while maintaining collision-free with each other and the obstacles. The constraint function  $h$  is defined the same as the simulation environments defined as  $h(o_i) = \max\{h_a(o_i), h_o(o_i)\}$  with  $h_a$  and  $h_o$  defined in (55) and (56). The cost function is given by

$$l(x, u) = l_{\text{visibility}}(x) + 0.1 \min_i l_{\text{dist}}(x_i, x) + \frac{1}{N} \sum_{i=1}^N l_{\text{action}}(u_i), \quad (65)$$

where  $l_{\text{visibility}}(x) = 0$  if the target is observable by at least one agent and  $l_{\text{visibility}}(x) = 0.01$  otherwise. For  $l_{\text{dist}}(x_i, x)$ , we have  $l_{\text{dist}}(x_i, x) = 0$  if the target is observable by agent  $i$ , otherwise, it is defined as the distance between the farthest observed point on the agent-target line and the target. Finally,  $l_{\text{action}}(u_i)$  is defined as  $l_{\text{action}}(u_i) = 0.00001 \|u_i\|^2$ .

### C. Videos

The videos of our hardware experiments are provided in the supplementary materials named ‘Corridor.mp4’ and ‘Inspect.mp4’.

## APPENDIX G CONVERGENCE

In this section, we analyze the convergence of the inner RL problem (13b) to a locally optimal policy.

Since we solve the inner RL problem (13b) in a centralized fashion, it can be seen as an instantiation of single-agent RL, but with a per-agent independent policy. Define the augmented state  $\tilde{x} \in \tilde{\mathcal{X}} := \mathcal{X} \times \mathbb{R}$  as  $[x, z]$ , which follows the dynamics  $\tilde{f} : \tilde{\mathcal{X}} \times \mathcal{U} \rightarrow \tilde{\mathcal{X}}$  defined as

$$\tilde{f}([x^k, z^k], u^k) = [f(x^k, u^k), z^k - l(x^k, u^k)]. \quad (66)$$

The inner RL problem (13b) can then be stated as

$$\min_{\pi} \max_{k \geq 0} h(x^k, \pi(\tilde{x}^k)) \quad (67a)$$

$$\text{s.t. } \tilde{x}^{k+1} = \tilde{f}(\tilde{x}^k, \tilde{\pi}(\tilde{x}^k)), k \geq 0. \quad (67b)$$

This is an instance of a single-agent RL *avoid* problem. Consequently, applying the results from [81, Theorem 5.5] or [67, Theorem 4] gives us that the policy  $\pi$  converges almost surely to a locally optimal policy.

## APPENDIX H ON THE EQUIVALENCE OF THE MASOCP AND ITS EPIGRAPH FORM

In Section III-B, we state that the Epigraph form (5) of a constrained optimization problem is equivalent to the original problem (3). This has been proved in So and Fan [66]. To make this paper more self-contained, we also include the proof here.

*Proof:* For a constrained optimization problem (3), its epigraph form [10, pp 134] is given by

$$\min_{\pi, z} z, \quad (68a)$$

$$\text{s.t. } h(\pi) \leq 0, \quad (68b)$$

$$J(\pi) \leq z, \quad (68c)$$

where  $z \in \mathbb{R}$  is an auxiliary variable. Here, (68b) and (68c) can be combined, which leads to the following problem:

$$\min_{\pi, z} z, \quad (69a)$$

$$\text{s.t. } \max \{h(\pi), J(\pi) - z\} \leq 0. \quad (69b)$$

Using this form, So and Fan [66, Theorem 3] shows that the minimization of  $x$  can be moved into the constraint, which yields

$$\min_z z, \quad (70a)$$

$$\text{s.t. } \min_{\pi} \max \{h(\pi), J(\pi) - z\} \leq 0. \quad (70b)$$

This is the same as (5). ■