

SafeDiSTL: Decomposing Joint Constraints for Safe Multi-Agent Reinforcement Learning

Hongrui Zheng Aditya Singh Rahul Mangharam

University of Pennsylvania, Philadelphia, USA
 {hongruiz,singhadi,rahulm}@engineering.upenn.edu

Abstract

Cooperative multi-agent tasks often impose constraints on a shared objective or collective safety. A team level violation cost function can measure this constraint outcome, but does not distinguish which agents' actions most strongly influence that violation. We present SafeDiSTL, a constrained multi-agent reinforcement learning framework that allows for credit assignment even with joint constraint measurements. We use Signal Temporal Logic (STL) robustness to measure episode-level violation through running-maximum cost increments. Smooth full-episode robustness and differentiable dynamics provide action sensitivities, which allocate each cost increment across agents at the same decision time. These sensitivities account for delayed effects when weighting agents' shares of each cost increment. We study the framework in cooperative payload transport, connected navigation, and networked traffic control tasks. SafeDiSTL achieves 1.6-5.1x the safe-success rate in transport, improves navigation safe-success by 11.1%, and delivers 50-68% more throughput in traffic when compared to constrained baselines. Results demonstrate a strong improvement of the learned policy in dealing with shared constraint while retaining the performance of unconstrained policies.

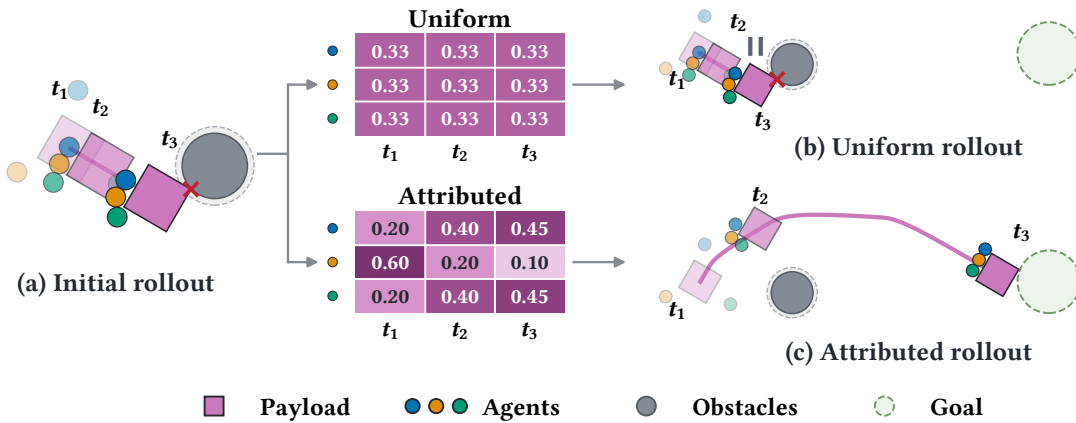


Figure 1: An illustration of the responsibility attribution mechanism. (a) A shared payload approaches an obstacle and incurs a constraint violation. (b) Uniform responsibility allocation produces conflicting corrections and agents stay deadlocked. (c) Agent-specific attribution enables coordinated obstacle avoidance.

1 Introduction

Recent developments in multi-agent autonomous systems and their applications in aerial swarms, intelligent transportation networks, and automated warehouses have made multi-agent coordination a central problem. In these environments, multiple agents must operate in shared spaces and achieve collective objectives, including collaborative manipulation, routing, and formation control, while satisfying system constraints such as connectivity and collision avoidance. Constraints in single-agent RL are commonly formulated with Constrained Markov Decision Processes (CMDPs) [2]. Policies are then optimized by reformulating the problem as a primal-dual optimization between reward and costs. Constrained Policy Optimization (CPO) [1] balances updates with more reward or cost consideration using a Lagrangian multiplier. Multi-agent Reinforcement Learning (MARL) is a popular paradigm for learning decentralized agent policies in multi-agent problems. Existing work extends CMDP formulations to MARL [11], although generally assuming well-defined, decoupled cost functions for individual agents. However, a shared constraint definition is important in multi-agent settings. For example, if multiple robots carry a shared rigid payload, a crash between the payload and obstacles is hard to attribute to individual agents. Consider the example shown in Figure 1: three warehouse robots collaboratively transport a large payload to a designated delivery location. At a given point t_3 , one robot might be required to push in a direction such that the payload avoids the obstacle (green agent pushing up), while the other two maintain progress toward the goal (yellow and blue push right). When encountering a violation such as that in Figure 1.(a), assigning equal responsibility to all agents leads to a deadlock (Figure 1.(b)) without addressing the asymmetrical desired updates to individual agents. This shows that a shared constraint definition alone is not enough. It tells the team of agents how much they violated, but without explicit learning signals to individual agents on how to update their policies. Without careful decomposition, the introduction of a uniform penalty for all agents will hinder task learning progress. If an agent makes significant contribution towards the team’s task progress but not towards violating constraint at a timestep, that individual transition should not incur a penalty in the policy learning process.

Credit assignment is an active area of research in MARL. Counterfactual Multi-Agent Policy Gradients (COMA) [9] established counterfactual reasoning for agents on task reward. Existing work [11, 14, 17] considers both individual and shared constraints in MARL problems, but leaves much to be desired in the effective decomposition of the constraint objective. This motivates the need for a principled mechanism to decompose the shared constraint among agents while accounting for the changing individual responsibilities to enable effective collaboration. Signal Temporal Logic (STL) [16] is a compact formal specification language that expresses logical and temporal requirements over trajectories. The quantitative measure called robustness [7, 8] assigns the margin of success to a trajectory instead of a binary signal. The *biggest advantage* of using STL is that differentiating its robustness through the full trajectory captures how individual actions influence subsequent constraint outcomes. We use these gradients to allocate cost increments across agents at its original timestep, while cost advantages propagate learning signals to earlier decisions. For the example in Figure 1, we express the shared responsibility as "Always no collision between payload and obstacle". Differentiable dynamics help achieve this by propagating individual agents’ influence to joint state space trajectories, and allowing back propagation from the margin of violation to decisions. As shown in Figure 1.(c), the green agent learns to subtly change its behavior by pushing *upward* rather than to the right, resulting in a successful avoidance of the obstacle. In summary, the contributions of this work are as follows:

1. We introduce an agent- and time-specific attribution mechanism that decomposes shared

team constraints using STL robustness gradients while preserving the total cost.

2. We develop SafeDiSTL, a constrained MARL framework that incorporates these attributed costs into policy optimization for decentralized execution.
3. We evaluate SafeDiSTL on three tasks, comparing task performance and constraint satisfaction with other baselines.

2 Related Work

Constrained MARL: Gu et al. [11] introduce MACPO and MAPPO-Lagrangian, using constrained trust region updates and Lagrangian policy optimization, respectively. Like MACPO, IsCPO [17] coordinates separate agent policies through sequential updates. IsCPO considers global costs and updates agents in random order, passing optimization information to subsequent agents. Thus, later agents account for the effects of earlier updates on the shared constraint. Both MACPO and IsCPO coordinate policy updates, rather than an explicit allocation of a trajectory’s violation cost according to each agent’s influence. Ling et al. [14] directly address constraint credit assignment through fictitious MARL in a collective setting. Its formulation exploits anonymous interactions represented by aggregate counts of agents in local states and actions, rather than individual identities. This structure supports scalable reward and constraint credit assignment, but is not assumed in our setting, where shared constraints can depend on coupled geometric configurations and action-dependent dynamics. SafeDiSTL instead obtains agent-specific cost allocations from the gradient sensitivity of a shared trajectory specification.

Multi-agent Credit Assignment: Credit assignment methods for multi-agent systems extract individual learning signals from shared team outcomes. COMA [9] uses a centralized critic and a counterfactual baseline that marginalizes one agent’s action while holding the others fixed. QPD [21] applies integrated gradients along trajectory paths to decompose global action values into individual credits. STAS [5] first redistributes episodic return over time and then uses Shapley attribution between agents. These methods establish counterfactual, gradient-based, and temporal attribution for cooperative reward optimization. Our focus is the attribution of the shared constraint: we differentiate STL robustness through dynamics rather than attribute a learned return prediction.

Responsibility Allocation: Wang et al. [20] use STL robustness to construct multi-agent rewards incorporating task and safety specifications. Zheng et al. [24] uses STL robustness as a learning signal for coordination of multiple agents. Lyu et al. [15] dynamically distribute collision avoidance responsibility using a control barrier function (CBF)-inspired risk measure. Cosner et al. [6] learn scene dependent allocations for responsibility aware control barrier functions. These approaches demonstrate the value of asymmetric safety responsibility in decentralized control. SafeDiSTL addresses the related problem of allocating learning signals for constraint violations during policy optimization, using STL robustness gradients to construct agent specific update signals rather than allocating online safety filter obligations.

3 Preliminaries

3.1 Signal Temporal Logic

Syntax: Signal Temporal Logic (STL) [16] allows us to create spatiotemporal requirements over trajectories τ , which encodes the global constraint in our learning problem. For example, we can express “Always avoid a collision between payload and obstacles” in a formal specification:

$\phi = \Box_{[0,5]} \neg \mu_{\text{col}}(p_{\text{payload}}, p_{\text{obs}})$, where μ_{col} is a predicate function that measures whether the two objects described by positions p are in collision. The STL syntax is

$$\phi := \top \mid \mu^g \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathbf{U}_{[t_1, t_2]} \phi_2. \quad (1)$$

$\top$ is the truth value, $\mu^g := g(\tau(t)) \geq 0$ is an atomic predicate, its truth value being determined by the value of the function $g : \mathbb{R}^n \rightarrow \mathbb{R}$. $\wedge$ and $\neg$ denote the “conjunction” and “negation” Boolean operators, respectively. $\mathbf{U}$ denotes the “until” temporal operator, which specifies that formula ϕ_1 holds until ϕ_2 holds in interval $[t_1, t_2]$. Disjunction is defined as $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$; “Eventually” is defined as $\Diamond_{[t_1, t_2]} \phi := \top \mathbf{U}_{[t_1, t_2]} \phi$; “Always” is defined as $\Box_{[t_1, t_2]} \phi := \neg \Diamond_{[t_1, t_2]} \neg\phi$; And “Implication” is defined as $\phi_1 \implies \phi_2 := \neg\phi_1 \vee \phi_2$.

Quantitative Semantics: STL uses **robustness** $\rho^\phi(\tau) \in \mathbb{R}$ to measure how strongly a given trajectory satisfies or violates a specification ϕ . $\rho > 0$ implies satisfaction and $\rho < 0$ implies violation. We used the standard definition to calculate robustness [7, 8]. We also utilize smooth approximations [12, 13, 18] of max and min operators to enable auto-differentiation of the robustness value of a trajectory to the states in the trajectory. The end-to-end gradient computation from robustness to policy actions can then be enabled with differentiable dynamics.

4 Problem Formulation

4.1 Constrained Cooperative Markov Game

The cooperative multi-agent Markov Decision Process (MDP) is defined by the tuple:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A} = \Pi_{i=1}^N \mathcal{A}_i, P, r, \rho_0, \gamma \rangle,$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the joint action space, $\mathcal{A}_i$ is the action of agent i out of N , P is the transition kernel, r is the shared task reward, ρ_0 is the initial state distribution, and γ is the discount factor. We extend this standard multi-agent MDP to a constrained Markov Game by extending the tuple to:

$$\mathcal{M}_C = \langle \mathcal{M}, c, d \rangle,$$

where $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$ is a non-negative cost function representing the shared responsibility of all agents, and d is a scalar cost budget for each episode. In this setting, we consider decentralized agent policies, where the joint policy factorizes as follows:

$$\pi_\theta(\mathbf{a}_t | o_t) = \Pi_{i=1}^N \pi_\theta(a_{i,t} | o_{i,t}),$$

and the system evolves as:

$$s_{t+1} \sim P(\cdot | s_t, \mathbf{a}_t),$$

where θ is the shared policy parameters, $\mathbf{a}_t$ is the joint action produced at time t , $o_{i,t}$ is the observation of agent i at time t , and s_t is the joint state at time t . The shared task reward objective can be expressed as:

$$J_R(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{H-1} \gamma^t r(s_t, \mathbf{a}_t) \right], \quad (2)$$

where $\tau \sim \pi_\theta$ denotes the trajectories realized under policy π_θ , and H denotes the episode horizon. In addition to the task reward, we also consider the shared cost objective:

$$J_C(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{H-1} \gamma^t c(s_t, \mathbf{a}_t) \right]. \quad (3)$$

Hence, the combined constrained team objective of the agents becomes:

$$\text{maximize}_{\theta} J_R(\theta), \quad \text{subject to } J_C(\theta) \leq d \quad (4)$$

4.2 Lagrangian SafeMARL

A standard Lagrangian formulation of the constrained objective in Equation (4) introduces a non-negative multiplier that trades reward against excess cost [1]:

$$\max_{\theta} \min_{\lambda \geq 0} \mathcal{L}(\theta, \lambda) = J_R(\theta) - \lambda(J_C(\theta) - d) \quad (5)$$

A canonical first-order projected primal–dual update performs ascent in the policy parameters and descent in the multiplier. At iteration k , it takes the form

$$\begin{aligned} \theta_{k+1} &= \theta_k + \eta_{\theta} \nabla_{\theta} \mathcal{L}(\theta_k, \lambda_k), \\ \lambda_{k+1} &= [\lambda_k + \eta_{\lambda} (J_C(\theta_k) - d)]_+, \end{aligned} \quad (6)$$

where $[\cdot]_+ = \max(0, \cdot)$ and $\eta_{\theta}, \eta_{\lambda} > 0$ are learning rates. The multiplier increases when the cost exceeds the budget. Equation (6) is an idealized gradient template: practical algorithms use sampled estimates and may change the surrogate objective, optimizer, or update schedule. Gu et. al. [11] proposes two constrained MARL algorithms MACPO and MAPPO-L: MACPO solves sequential constrained trust-region subproblems rather than using the multiplier recursion in Equation 6; MAPPO-L instead combines reward and cost advantages in a PPO-style surrogate with nonnegative multiplier updates. Our proposed method SafeDiSTL follows the Lagrangian approach but attributes the shared trajectory cost across agents and decision times for the actor update, while retaining the unallocated shared cost for multiplier updates.

5 SafeDiSTL

A shared constraint cost measures the team’s violation but does not distinguish the contributions of individual agents and decisions. SafeDiSTL uses full-episode STL sensitivities to allocate each running-maximum cost increment across agents at the same decision time. Section 5.1 develops this allocation while preserving both the per-step increment and the total episode cost. Section 5.2 incorporates the attributed costs into critic learning and constrained policy optimization, while retaining the original episode cost for the multiplier update.

5.1 Decomposing the Shared Constraint

SafeDiSTL uses an STL specification on the joint state of all agents for both calculating the shared constraint cost and credit assignment. Let $\tau = (s_0, \mathbf{a}_0, \dots, \mathbf{a}_{T_{\tau}-1}, s_{T_{\tau}})$ be a complete episode. Let H denote the maximum episode length and $T_{\tau} \leq H$ the number of transitions in episode τ , which may terminate before reaching H . Let $\rho^{\phi}(\tau)$ denote the episode’s robustness with respect to the shared constraint specification ϕ . We define the episode violation cost and its expectation as

$$\begin{aligned} C^{\phi}(\tau) &= [-\rho^{\phi}(\tau)]_+, \\ J_C^{\phi}(\theta) &= \mathbb{E}_{\tau \sim \pi_{\theta}} [C^{\phi}(\tau)]. \end{aligned} \quad (7)$$

In this paper, we consider the always-conjunction specifications type. This type of specification follows this form:

$$\begin{aligned}\phi &= \bigcap_{[0, T_\tau-1]} \bigwedge_{j=1}^K \mu^{g_j}, \\ \rho^\phi(\tau) &= \min_{0 \leq t < T_\tau} \min_{1 \leq j \leq K} g_j(s_{t+1}),\end{aligned}\tag{8}$$

where K counts shared predicates and $g_j(s) \geq 0$ indicates satisfaction of predicate j . Each predicate function $g(j)$ here is essentially a distance function that describe a part of the shared constraint. For example, it could be the Euclidean distance between a shared payload and an obstacle. Inspired by State-wise Constrained Policy Optimization [23], we accumulate the episode cost through increments in the running maximum. Define the instantaneous post-transition violation of the shared constraint and running maximum by

$$\begin{aligned}c_{\text{ins}}(s_{t+1}) &= \left[- \min_{1 \leq j \leq K} g_j(s_{t+1}) \right]_+, \\ c_{\text{max},t} &= \max\{c_{\text{max},t-1}, c_{\text{ins}}(s_{t+1})\},\end{aligned}\tag{9}$$

with $c_{\text{max},-1} = 0$. The incremental maximum cost is

$$\begin{aligned}\Delta c_t &= c_{\text{max},t} - c_{\text{max},t-1} \\ &= [c_{\text{ins}}(s_{t+1}) - c_{\text{max},t-1}]_+.\end{aligned}\tag{10}$$

Only a violation exceeding the maximum of the previous episode incurs an additional cost. These increments telescope to the episode violation cost:

$$\sum_{t=0}^{T_\tau-1} \Delta c_t = c_{\text{max},T_\tau-1} = C^\phi(\tau).\tag{11}$$

Unlike the generic discounted cost in Equation (3), our shared constraint objective sums without discounting. This preserves the peak violation irrespective of when it occurs, and the correctness of Equation (11). The increments identify when the measured violation increases, but not which decisions influence it. To construct agent- and time-specific learning signals, we differentiate trajectory robustness through the dynamics. Let $\tilde{\rho}^\phi$ denote the differentiable smooth approximation of the robustness and define the action sensitivity

$$z_{i,t} = \left\| \frac{\partial [-\tilde{\rho}^\phi(\tau)]}{\partial a_{i,t}} \right\|_2.\tag{12}$$

Note that we use the 2-norm here, thus our action sensitivity only measures the magnitude of the influence of an agent. The derivative propagates through the entire recorded episode, holding the other sampled actions fixed. Thus, the sensitivities capture delayed effects rather than only immediate effects. The smooth robustness approximation and attribution normalization are detailed in Appendix F. We normalize these sensitivities across agents separately at each decision time:

$$w_{i,t} = \frac{z_{i,t}}{\sum_{j=1}^N z_{j,t}},\tag{13}$$

assuming nonzero sensitivity at time t , with uniform weights when all sensitivities vanish. Thus, $\sum_i w_{i,t} = 1$ for every valid decision time. We allocate the corresponding cost increment using detached attribution weights:

$$\tilde{c}_{i,t} = N \Delta c_t \text{stopgrad}(w_{i,t}). \quad (14)$$

The sensitivity that determines the relative agent weights is from the full episode, while the costs allocated remain local at each timestep. If $\Delta c_t = 0$, no attributed cost is assigned at that time. The factor N accounts for averaging across agents, preserving each increment and hence the episode cost:

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N \tilde{c}_{i,t} &= \Delta c_t, \\ \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_\tau-1} \tilde{c}_{i,t} &= \sum_{t=0}^{T_\tau-1} \Delta c_t = C^\phi(\tau). \end{aligned} \quad (15)$$

5.2 Constrained Policy Optimization

We optimize decentralized policies using a MAPPO-style [22] proximal policy optimization (PPO) algorithm under centralized training. Following the Lagrangian formulation in Section 4.2, the policy update balances the shared task reward with the constraint cost. SafeDiSTL replaces the shared cost signal in this update with the agent-specific attributed costs defined in Section 5.1.

A centralized reward critic $V_R(s_t)$ estimates the expected discounted share task return. For constraint costs, we use an agent-specific critic $V_C^i(o_{i,t}, c_{\max,t-1})$ and a centralized critic $V_C^g(s_t, c_{\max,t-1})$. They estimate the expected remaining cumulative cost assigned to agent i and its team mean, respectively, using the allocation from Section 5.1. The corresponding cost signals are $\tilde{c}_{i,t}$ and

$$\bar{c}_t = \frac{1}{N} \sum_{i=1}^N \tilde{c}_{i,t} = \Delta c_t. \quad (16)$$

Attribution changes the distribution of cost across agents but not its timing or team mean. The centralized cost critic therefore learns from the original increments Δc_t , while the agent-specific critic learns from $\tilde{c}_{i,t}$. Both receive $c_{\max,t-1}$ as training-only context for the constraint history. Then generalized advantage estimation (GAE) connects each signal and critic to its advantage:

$$A_{R,t} = \text{GAE}_t(r, V_R), \quad A_{i,t}^C = \text{GAE}_t(\tilde{c}_i, V_C^i), \quad A_{g,t}^C = \text{GAE}_t(\bar{c}, V_C^g). \quad (17)$$

Here, GAE takes the corresponding signal sequence and rollout-time critic predictions. The critic regression targets are:

$$y_{R,t} = A_{R,t} + V_{R,t}, \quad y_{i,t}^C = A_{i,t}^C + V_{C,t}^i, \quad y_{g,t}^C = A_{g,t}^C + V_{C,t}^g. \quad (18)$$

Here $V_{R,t}$, $V_{C,t}^i$, and $V_{C,t}^g$ denote predictions during rollout. The centralized cost critic supplies a joint-state reference for the local critics: a consistency penalty encourages their mean prediction to agree with it. Therefore, it influences the actor indirectly through agent-specific predictions, rather than through a separate $A_{g,t}^C$ term in the actor objective. We combine the shared reward advantage with each agent’s attributed cost advantage:

$$\begin{aligned} A_{i,t} &= A_{R,t} - \lambda_k A_{i,t}^C, \\ q_{i,t}(\theta) &= \frac{\pi_\theta(a_{i,t} \mid o_{i,t})}{\pi_{\theta_k}(a_{i,t} \mid o_{i,t})}. \end{aligned} \quad (19)$$

Algorithm 1 SafeDiSTL

- 1: Initialize π_θ , critics V_R, V_C^i, V_C^g , and λ_0 .
 - 2: **for** each training iteration k **do**
 - 3: Collect complete episodes using π_{θ_k} .
 - 4: Compute $C^\phi(\tau)$, increments Δc_t , and full-episode sensitivities $z_{i,t}$.
 - 5: Normalize $z_{i,t}$ across agents at each t to obtain $w_{i,t}$.
 - 6: Allocate $\tilde{c}_{i,t} = N \Delta c_t \text{stopgrad}(w_{i,t})$; set $\bar{c}_t = \Delta c_t$.
 - 7: Compute $A_{R,t}, A_{i,t}^C, A_{g,t}^C$ using Equation (17).
 - 8: Fit V_R, V_C^i, V_C^g to $y_{R,t}, y_{i,t}^C, y_{g,t}^C$, with cost consistency regularization.
 - 9: Form $A_{i,t} = A_{R,t} - \lambda_k A_{i,t}^C$.
 - 10: Update θ_k to θ_{k+1} using Equation (20).
 - 11: Update λ_k to λ_{k+1} from the mean $C^\phi(\tau)$ and budget d .
 - 12: **end for**
-

Here $\lambda_k \geq 0$ is the shared constraint multiplier at training iteration k , and $q_{i,t}(\theta)$ is the action likelihood ratio between the updated and rollout policies. The actor is updated using the PPO clipped objective

$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}_{i,t} [\min\{q_{i,t}(\theta)A_{i,t}, \text{clip}(q_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon)A_{i,t}\}], \quad (20)$$

where $\epsilon > 0$ is the clipping threshold. The advantages and attribution weights are held fixed during this update. When individual constraints are also present, their cost advantages and multipliers contribute separate penalty terms and are not included in the shared attribution. For the dual update in Equation (6), we estimate $J_C^\phi(\theta_k)$ using the mean exact episode cost $C^\phi(\tau)$, not the attributed cost of any single agent. The shared multiplier λ_k increases when this mean exceeds d and decreases when it falls below d , subject to its nonnegative range. Thus, attribution changes the agents’ policy-learning signals while the dual update uses the original shared constraint. Algorithm 1 summarizes the training procedure. Cost-advantage estimation, critic losses, and implementation details are given in Appendix G.

6 Experiments

We evaluate SafeDiSTL on three cooperative multi-agent benchmarks with different shared constraints: package–obstacle clearance in transport, communication connectivity in navigation, and core occupancy in traffic control. Across tasks, we examine whether attribution of a shared constraint supports task completion while maintaining constraint satisfaction. We compare against reward-only MAPPO and the constrained baselines MACPO and MAPPO-L from Gu et. al. [11]. Baseline optimization details are provided in Appendix A.

6.1 Experimental Setup

All methods use six training seeds and common evaluation resets within each benchmark. We compare task success, shared constraint satisfaction, and their joint occurrence, as well as task returns. All methods are trained with the same compute budget in each scenario in terms of simulation steps. Transport policies train with 15M sim steps, and evaluates on 500 common resets per obstacle condition. Navigation policies train with 15M sim steps, and evaluates on 128 common resets. Traffic policies train with 20M sim steps and evaluates on 1,000 common resets. In

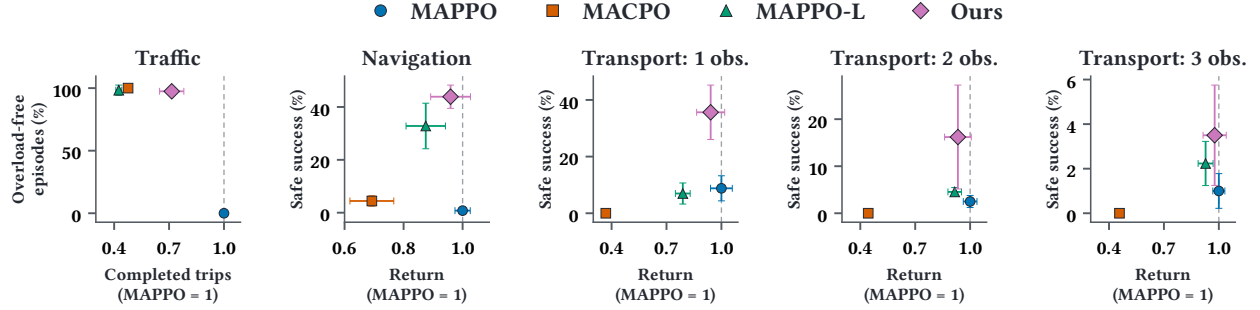


Figure 2: Constraint–performance tradeoffs across benchmarks. Points and error bars show means and sample standard deviations across six training seeds. Traffic reports overload-free episode percentages versus completed trips. Others report safe-success rates where goal was reached without constraint violation versus task return. x -axes metrics are normalized by MAPPO values. Note that the sub-plots zooms into parts of the y -range for better visibility and comparison, and hence have different y -axes ranges.

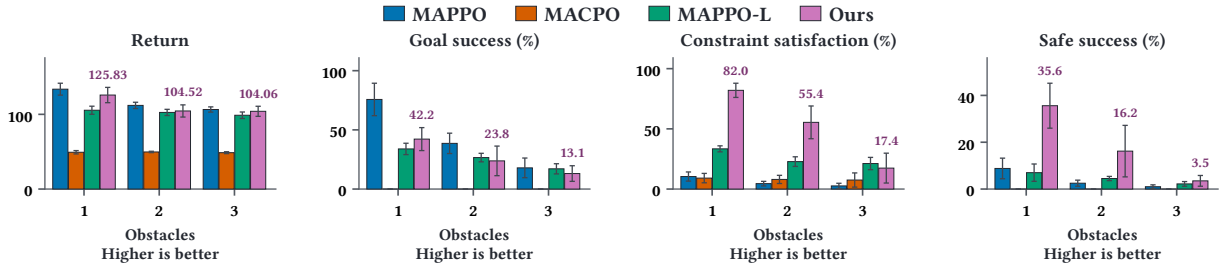


Figure 3: Transport results across obstacle counts (mean $\pm$ SD over six training seeds). Labels are SafeDiSTL mean values.

all benchmarks, SafeDiSTL uses the running max increment cost defined in Section 5.1. All other benchmarks utilizes cost functions equivalent to the per step cost of the same STL specification. For example, in transport, SafeDiSTL’s STL specification says “Always no collision between the payload and the obstacles”, the equivalent per step cost function is the overlapping margin in Euclidean distance between the payload and obstacles’ geometric shapes. Each method then pre-processes the cost functions according to the reference implementation. All experiments are conducted in JAX [4] on GPUs. We primarily examine four key metrics.

1. Task return: episodic sum of task reward that the policies achieve in evaluation.
2. Goal success rate: the percentage of episodes that leads to successful goal reaching.
3. Constraint satisfaction rate: the percentage of evaluation episodes that did not incur any constraint violations.
4. Safe success rate: the percentage of episodes that has the joint occurrence of goal success and constraint satisfaction.

Environment details and policy architectures are provided in Appendices H and I, respectively.

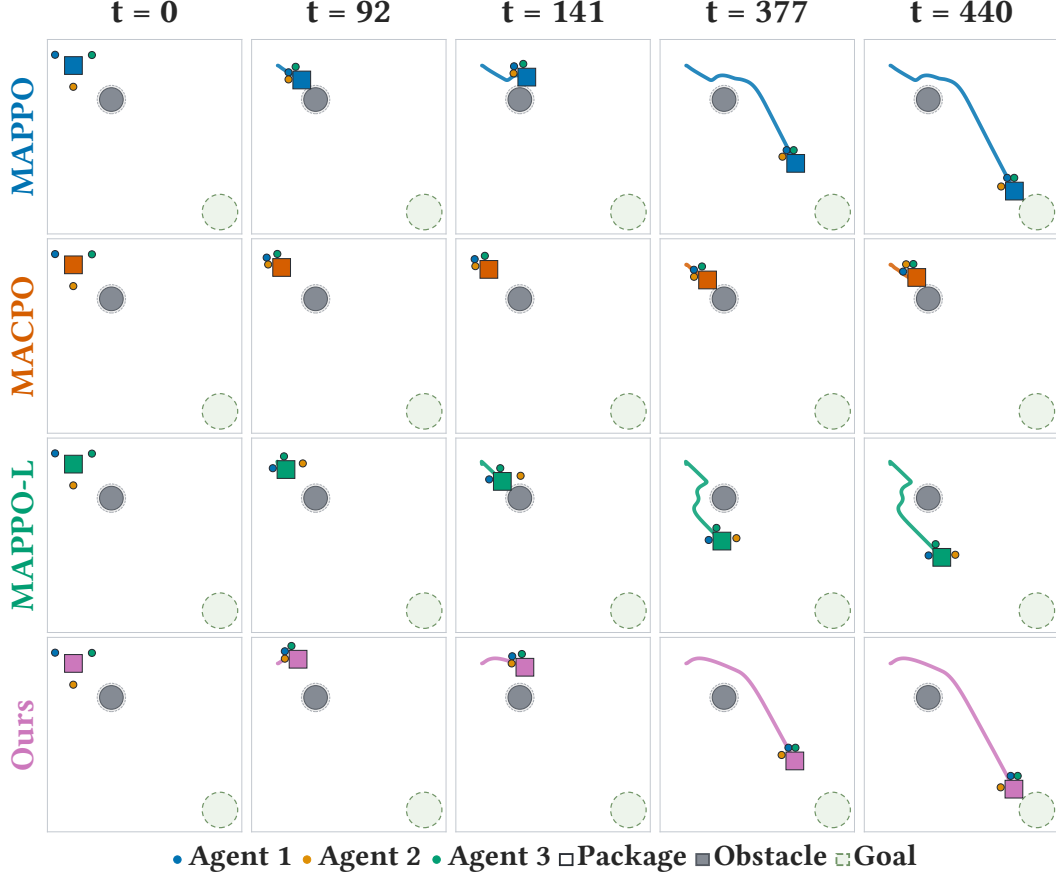


Figure 4: Example rollout comparison with baselines in the 1-obstacle transport scenario with common initial conditions.

6.2 The Constraint-performance Tradeoff

We first show the key result of the experiments. In Figure 2, we show that SafeDiSTL balances the tradeoff between constraint satisfaction and retaining performance. In each subplot, the y -axes shows constraint satisfaction metrics in percentages, and the x -axes shows performance metrics in normalized returns to those obtained by MAPPO. Our algorithm modifies MAPPO by introducing the dual on constraint and asymmetric cost attribution to individual agents, thus we want to show that SafeDiSTL retains the performance of MAPPO, but does so considering satisfying the shared constraints. In all five scenarios shown in Figure 2, SafeDiSTL balances this trade off by being in the top right quadrant of the plots, while other methods sacrifice MAPPO’s task performance to increase constraint satisfaction. Appendix E examines attribution, cost formulation, and dual-optimizer variants.

6.3 Cooperative Transport

We use a transport task with dynamics and control similar to the one in [3], where three force input agents push a shared package around circular obstacles towards a goal. The shared constraint here is package–obstacle clearance, and safe success requires both the payload reaching the goal and no contact between the payload and the obstacle throughout the entire episode. In this benchmark, we consider three different settings with 1-3 obstacles between the starting position and the payload

goal. Figure 3 shows the comparison between the four methods. SafeDiSTL has the highest mean safe success rate in all three settings: $35.6 \pm 9.6\%$, $16.2 \pm 11.0\%$, and $3.5 \pm 2.3\%$ with one, two, and three obstacles, respectively. MAPPO-L trails closest behind with $7.0 \pm 3.7\%$, $4.5 \pm 0.9\%$, and $2.2 \pm 1.0\%$. With three obstacles, safe success remains low for all methods in this harder condition, yet SafeDiSTL still achieves highest safe success rate there. Although MAPPO has higher mean goal success in all three conditions, SafeDiSTL more often achieves the goal while satisfying the shared clearance requirement. MACPO on the other hand completely fails in all three settings. Full numerical results appear in Table 1. Figure 4 shows an example rollout comparison between the methods. The same trend shows up here as well: MAPPO reaches the goal, the two constrained baselines has degraded goal reaching capability, while SafeDiSTL manages to go around the obstacle and reaches the goal.

6.4 Connected Navigation

In this benchmark, we control four point-mass agents navigating through two staggered obstacle passages to a shared destination. The shared constraint is buffered full episode communication connectivity: at every timestep, the longest edge of the agents’ Euclidean minimum-spanning tree must not exceed 3.325 m. Goal success requires all agents to remain in the destination region during the final 20 control steps. Figure 6 reports the comparison between all four methods. SafeDiSTL has the highest mean goal success rate ($71.88 \pm 6.13\%$) and safe-success rate ($43.88 \pm 4.40\%$) among the four main methods. MAPPO-L attains slightly higher full episode constraint satisfaction rate than SafeDiSTL ($56.64 \pm 10.61\%$ versus $54.69 \pm 8.33\%$). MACPO reaches only $4.43 \pm 1.96\%$ safe successes. MAPPO has the highest mean return but almost always violates the constraint. These results show that in the connected navigation benchmark, considering the constraint effectively also helps with task success rate since grouping together through the narrow passage reduces accidental collision with the obstacle that might slow down goal reaching. Full numerical results appear in Table 2. Figure 5 also shows the same conclusion in matched rollout trajectories. SafeDiSTL agents grouped closely together and navigated smoothly towards the goal without any disconnects in the graph. The agents in all other methods disconnected at least once throughout the episode.

6.5 Networked Traffic Control

We consider a seven region Networked Macroscopic Fundamental Diagram (NMFD) traffic task [10, 19]. Six peripheral regions exchange destination labeled traffic through a shared central core. Each region meters its outgoing traffic, while core exits are uncontrolled and randomized at every time step. The shared constraint is on the core region’s occupancy. The flow rate of the core region is dictated by the cubic polynomial shown in Figure 7. The flow rate first increases as the number of vehicles in the region increases. After the peak, it drops down to the congestion limit, where the flow rate drops below 1 vehicle per second. Thus the shared constraint is set as $n_{\text{core}} \leq n_{\text{safe}} \simeq 8786.415$. Vehicle admissions to the core therefore trade off completed trips against core region congestion, while restrictive metering can increase the wait time in the system.

The traffic in the system is randomly spawned as source-destination pairing of two peripheral zones into the queue. All vehicles must travel through the core on their way to their destinations. The agents control their own regions’ flow rate for vehicles with all other peripheral regions as destination. Then the system revolves following the controlled core region in-flow rate, and the uncontrolled, randomized out-flow rate. The agents observe the randomize exit rate, and should react to changes for decisions on their control. For this task, since there isn’t a clear measure for success, we use these key metrics instead:

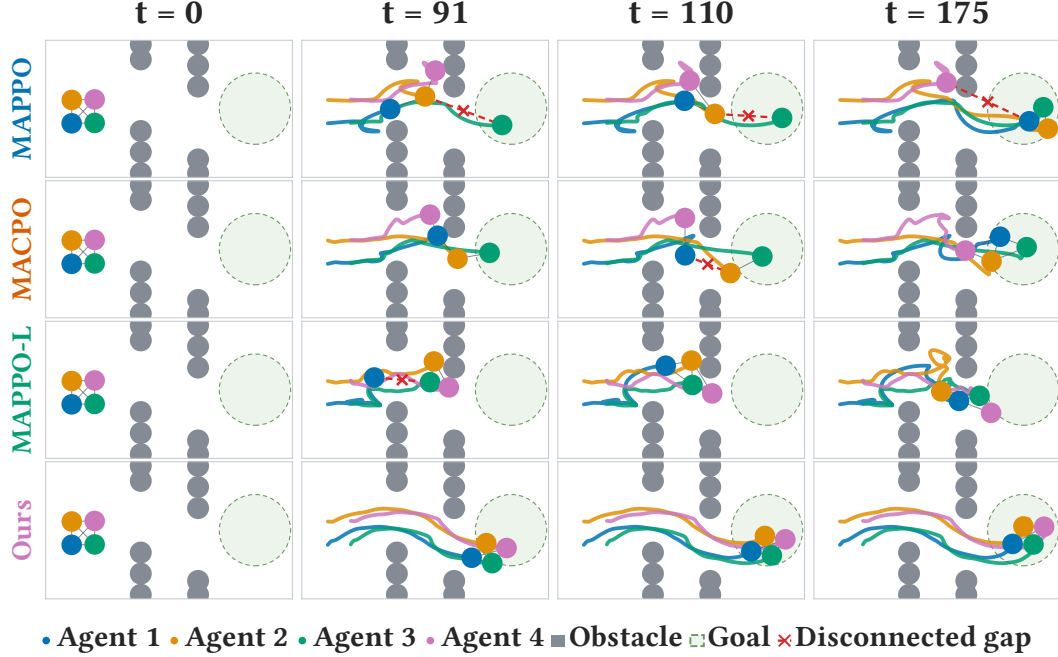


Figure 5: Example rollout comparison with baselines in the connected navigation benchmark with common initial conditions.

1. Completed trips: this is the equivalent to task return, where the policy uses as training reward.
2. Overload-free episode rate: the percentage of full episodes without overloading the core region.
3. Core vehicle-hours: the time that vehicles spends on average at the core region, this measures the system delay.
4. Time above core limit: the time that core regions has occupancy higher than the limit.

Figure 8 compares all four methods. The training reward for all methods here are completed trips. The goal of this benchmark is to test the trade off between throughput and core congestion. Since core region exit rates are randomized, a smart team of agents should learn to coordinate when to let each destination’s vehicles go without overloading the core. SafeDiSTL retains 71.4% of MAPPO throughput while reducing the core vehicle-hours to 8.8% of MAPPO and the time above core limits to 0.3% of MAPPO. Although MACPO and MAPPO-L also learn to reduce constraint violation to the same level, they only retain 47.6% and 42.5% of the MAPPO throughput, respectively. Full numerical results appear in Table 3.

7 Limitations and Conclusions

We propose SafeDiSTL, a constrained MARL framework that allows for credit assignment under shared team constraints. Using STL robustness gradients, we derive a cost attribution scheme that decomposes the shared team constraints while preserving the total cost. These attributed costs are then incorporated in policy optimization for centralized training and decentralized execution. Across three diverse case studies on cooperative payload transport, connected navigation, and traffic

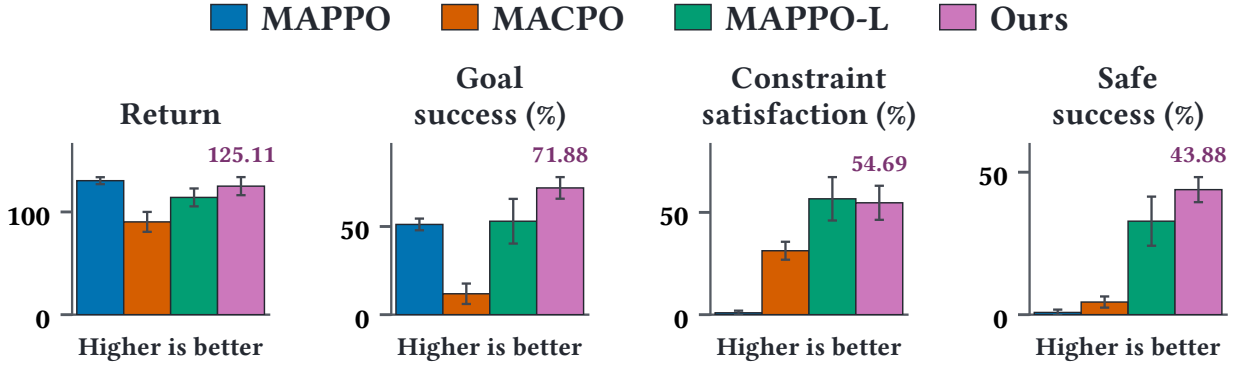


Figure 6: Navigation results (mean $\pm$ SD over six training seeds). Labels are SafeDiSTL mean values.

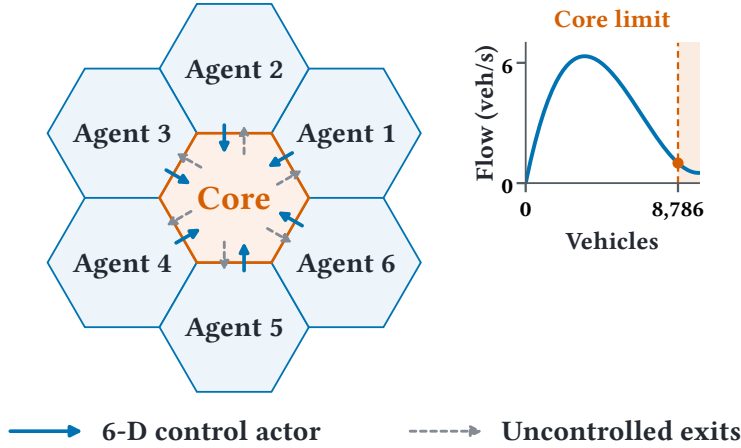


Figure 7: Seven region traffic network with a shared core occupancy constraint. Controls for each agent is a 6-D vector that controls the flow rate for all 6 destinations (0 always for self).

network control, SafeDiSTL shows a strong improvement in handling shared constraints over the other state-of-the art MARL baselines.

First, while SafeDiSTL has shown an exceptional safety performance tradeoff in benchmarks, it relies on the gradients of the robustness signal through the system dynamics. The dynamics gradients might be hard to come by in contact-rich scenarios. Even with non-differentiable dynamics, we can still estimate via finite differences by using more calls to the simulation. Future extensions should aim to mitigate this issue by devising gradient-free signals for STL robustness decomposition. Second, our sensitivity metric used to allocate costs is the magnitude of influence. Future work could also consider how to incorporate the direction of influence. Third, we did not show benchmarks on higher fidelity robotic scenarios, and potential transfer of learned policies to real robots. Future works will include validation of the proposed method on real-world robotic platforms.

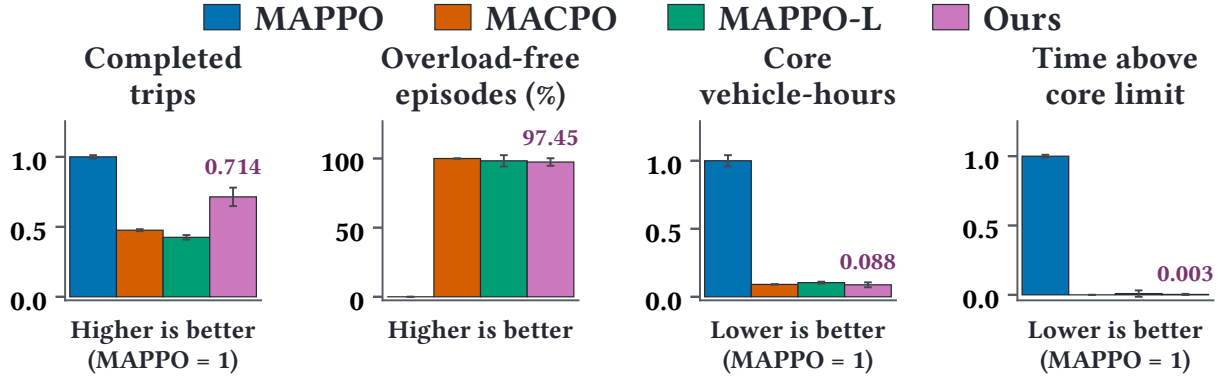


Figure 8: Traffic results (mean $\pm$ SD over six training seeds). Labels are SafeDiSTL mean values.

References

- [1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. 22–31.
- [2] Eitan Altman. 1995. *Constrained Markov Decision Processes*. Technical Report RR-2574. INRIA.
- [3] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. 2022. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems* (2022).
- [4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs*. <http://github.com/jax-ml/jax>
- [5] Sirui Chen, Zhaowei Zhang, Yaodong Yang, and Yali Du. 2023. STAS: Spatial-Temporal Return Decomposition for Multi-agent Reinforcement Learning. *arXiv preprint arXiv:2304.07520* (2023). <https://arxiv.org/abs/2304.07520>
- [6] Ryan K. Cosner, Yuxiao Chen, Karen Leung, and Marco Pavone. 2023. Learning Responsibility Allocations for Safe Human-Robot Interaction with Applications to Autonomous Driving. In *2023 IEEE International Conference on Robotics and Automation*.
- [7] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Formal Modeling and Analysis of Timed Systems*, Krishnendu Chatterjee and Thomas A. Henzinger (Eds.). Springer, Berlin, Heidelberg, 92–106. doi:10.1007/978-3-642-15297-9_9
- [8] Goergios Fainekos. 2008. *Robustness of Temporal Logic Specifications*. PhD Thesis. University of Pennsylvania. http://www.public.asu.edu/~gfaineko/pub/fainekos_thesis.pdf

- [9] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2974–2982.
- [10] JW Godfrey. 1969. The mechanism of a road network. *Traffic Engineering & Control* 8, 8 (1969).
- [11] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. 2023. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence* 319 (June 2023), 103905. doi:10.1016/j.artint.2023.103905
- [12] Parv Kapoor, Kazuki Mizuta, Eunsuk Kang, and Karen Leung. 2025. STLCCG++: A Masking Approach for Differentiable Signal Temporal Logic Specification. *IEEE Robotics and Automation Letters* 10, 9 (Sept. 2025), 9240–9247. doi:10.1109/LRA.2025.3588389
- [13] Karen Leung, Nikos Aréchiga, and Marco Pavone. 2023. Backpropagation through Signal Temporal Logic Specifications: Infusing Logical Structure into Gradient-Based Methods. *The International Journal of Robotics Research* 42, 6 (2023), 356–370. doi:10.1177/02783649221082115
- [14] Jiajing Ling, Arambam James Singh, Nguyen Duc Thien, and Akshat Kumar. 2023. Constrained Multiagent Reinforcement Learning for Large Agent Population. In *Machine Learning and Knowledge Discovery in Databases*, Massih-Reza Amini, Stéphane Canu, Asja Fischer, Tias Guns, Petra Kralj Novak, and Grigorios Tsoumakas (Eds.). Springer Nature Switzerland, Cham, 183–199. doi:10.1007/978-3-031-26412-2_12
- [15] Yiwei Lyu, Wenhao Luo, and John M. Dolan. 2023. Risk-Aware Safe Control for Decentralized Multi-Agent Systems via Dynamic Responsibility Allocation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 9737–9744.
- [16] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Proceedings of FORMATS-FTRTFT (LNCS, Vol. 3253)*. 152–166.
- [17] Yoshihiro Okawa, Hayato Dan, Natsuki Morita, and Masatoshi Ogawa. 2023. Multi-Agent Reinforcement Learning with Information-sharing Constrained Policy Optimization for Global Cost Environment. *IFAC-PapersOnLine* 56, 2 (Jan. 2023), 1558–1565. doi:10.1016/j.ifacol.2023.10.1854
- [18] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*. 1235–1240. doi:10.1109/CCTA.2017.8062628
- [19] Renukanandan Tumu, Wenceslao Shaw Cortez, Ján Drgoňa, Draguna L. Vrabie, and Sonja Glavaski. 2024. Differentiable Predictive Control for Large-Scale Urban Road Networks. arXiv:2406.10433 [eess.SY]. doi:10.48550/arXiv.2406.10433
- [20] Jiangwei Wang, Shuo Yang, Ziyang An, Songyang Han, Zhili Zhang, Rahul Mangharam, Meiyi Ma, and Fei Miao. 2025. Multi-Agent Reinforcement Learning Guided by Signal Temporal Logic Specifications. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 6048–6054. doi:10.1109/IRoS60139.2025.11246629
- [21] Yaodong Yang, Jianye Hao, Guangyong Chen, Hongyao Tang, Yingfeng Chen, Yujing Hu, Changjie Fan, and Zhongyu Wei. 2020. Q-value Path Decomposition for Deep Multiagent

- Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. 10706–10715. <https://proceedings.mlr.press/v119/yang20d.html>
- [22] Chao Yu, Akash Velu, Eugene Vinitisky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*. doi:10.52202/068431-1787
- [23] Weiye Zhao, Rui Chen, Yifan Sun, Feihan Li, Tianhao Wei, and Changliu Liu. 2024. State-wise Constrained Policy Optimization. *Transactions on Machine Learning Research* (2024).
- [24] Hongrui Zheng, Zirui Zang, Ahmad Amine, Cristian Ioan Vasile, and Rahul Mangharam. 2026. STL-SVPIO: Signal Temporal Logic guided Stein Variational Path Integral Optimization. arXiv:2603.13333 [cs.RO]. doi:10.48550/arXiv.2603.13333

A Baseline Optimization Details

This appendix gives the optimization details summarized in Section 4.2. Equation 6 is a first-order Lagrangian template, not the update rule of every constrained policy optimizer. CPO [1] distinguishes this stateful multiplier update from its constrained trust-region subproblem, whose multipliers are solved anew at each iteration. SafeMARL extends constrained policy optimization to cooperative Markov games through MACPO and MAPPO-Lagrangian [11].

A.1 MACPO

Multi-Agent Constrained Policy Optimisation (MACPO) updates decentralized actors sequentially, accounting for the policies of agents already updated in the current iteration [11]. Each actor maximizes a reward surrogate subject to cost constraints and a Kullback–Leibler (KL) trust region. Thus, coordination enters the sequential surrogate rather than a simultaneous update of all actors using a fixed cost penalty.

In the shared-cost notation of Equation 4, the local CPO-style approximation for agent i has the form [1]

$$\begin{aligned} & \max_{\Delta\theta_i} \quad g_i^\top \Delta\theta_i, \\ & \text{subject to} \quad c_i + b_i^\top \Delta\theta_i \leq 0, \\ & \quad \quad \quad \frac{1}{2} \Delta\theta_i^\top H_i \Delta\theta_i \leq \delta_i. \end{aligned} \tag{21}$$

Here g_i and b_i are gradients of the sequential reward and shared-cost surrogates, c_i is the current surrogate cost minus d , H_i is the local Hessian of the mean KL divergence, and $\delta_i > 0$ is the trust-region radius. These quantities are evaluated at the current actor parameters, with preceding actors’ updates accounted for in the surrogates. Linearizing reward and cost while quadratically approximating KL yields a constrained natural-gradient step. For invertible H_i and a positive trust-region multiplier η_i , its direction satisfies

$$\begin{aligned} \Delta\theta_i^* &= \eta_i^{-1} H_i^{-1} (g_i - \nu_i b_i), \\ \theta_i^{\text{new}} &= \theta_i^{\text{old}} + \alpha_i \Delta\theta_i^*, \end{aligned} \tag{22}$$

where $\nu_i \geq 0$ enforces the local cost constraint and $\alpha_i \in (0, 1]$ is selected by backtracking. Unlike the stateful λ_k in Equation 6, these subproblem multipliers are solved anew for each actor update.

MACPO uses a cost-reducing recovery step when the constrained update is infeasible [11]. Multiple budgets add linearized cost constraints to Equation 21. The theoretical sequential procedure provides reward-improvement and cost-feasibility bounds under its assumptions; sampling error, local approximations, and finite line searches mean that practical neural-policy updates do not guarantee constraint satisfaction on every trajectory.

A.2 MAPPO-Style Lagrangian Surrogates

A first-order Lagrangian implementation replaces the exact policy gradient in Equation 6 with a sample-based surrogate using reward and cost advantages. Let $A_{R,t}$ and $A_{C,t}$ be reward and shared-cost advantage estimates obtained from centralized critics. For the decentralized actor of agent i , define

$$\begin{aligned} A_{i,t}^\lambda &= A_{R,t} - \lambda_k A_{C,t}, \\ q_{i,t}(\theta_i) &= \frac{\pi_{\theta_i}(a_{i,t} \mid o_{i,t})}{\pi_{\theta_{i,k}}(a_{i,t} \mid o_{i,t})}. \end{aligned} \tag{23}$$

Table 1: Supplementary Transport results compared to baselines. Entries are mean $\pm$ sample SD across six seeds and 500 common resets per seed. Rates are percentages. Safe success denotes joint occurrence of goal reaching and shared-constraint satisfaction.

Obstacles	Algorithm	Return	Goal success	Constraint satisfaction	Safe success
1	MAPPO	133.68 $\pm$ 7.96	75.7 $\pm$ 13.7	10.4 $\pm$ 3.8	8.8 $\pm$ 4.4
1	MACPO	49.25 $\pm$ 2.23	0.0 $\pm$ 0.0	9.1 $\pm$ 3.9	0.0 $\pm$ 0.0
1	MAPPO-L	105.49 $\pm$ 5.40	33.9 $\pm$ 4.8	33.4 $\pm$ 2.5	7.0 $\pm$ 3.7
1	Ours	125.83 $\pm$ 10.24	42.2 $\pm$ 9.7	82.0 $\pm$ 5.9	35.6 $\pm$ 9.6
2	MAPPO	111.98 $\pm$ 4.19	38.6 $\pm$ 8.6	4.5 $\pm$ 1.9	2.5 $\pm$ 1.3
2	MACPO	49.65 $\pm$ 0.91	0.0 $\pm$ 0.0	8.0 $\pm$ 3.4	0.0 $\pm$ 0.0
2	MAPPO-L	102.56 $\pm$ 4.15	26.6 $\pm$ 3.6	22.8 $\pm$ 4.0	4.5 $\pm$ 0.9
2	Ours	104.52 $\pm$ 8.15	23.8 $\pm$ 12.5	55.4 $\pm$ 13.6	16.2 $\pm$ 11.0
3	MAPPO	106.49 $\pm$ 3.50	17.9 $\pm$ 8.3	2.6 $\pm$ 2.2	1.0 $\pm$ 0.8
3	MACPO	48.69 $\pm$ 1.27	0.0 $\pm$ 0.0	7.5 $\pm$ 5.9	0.0 $\pm$ 0.0
3	MAPPO-L	98.79 $\pm$ 4.35	17.1 $\pm$ 4.3	21.2 $\pm$ 5.1	2.2 $\pm$ 1.0
3	Ours	104.06 $\pm$ 6.74	13.1 $\pm$ 6.6	17.4 $\pm$ 12.4	3.5 $\pm$ 2.3

A MAPPO-style clipped surrogate for a fixed multiplier is

$$\hat{\mathcal{L}}_{\text{clip}}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_{i,t}[\min\{q_{i,t}(\theta_i)A_{i,t}^\lambda, \text{clip}(q_{i,t}(\theta_i), 1 - \epsilon, 1 + \epsilon)A_{i,t}^\lambda\}], \quad (24)$$

where $\epsilon > 0$ is the clipping threshold and the empirical expectation is over valid on-policy samples. Centralized critics may access s_t , while execution uses only $o_{i,t}$. Homogeneous agents can share actor parameters without changing the factorized action sampling. Multiple constraints introduce one nonnegative multiplier per budget and a sum of cost-advantage terms.

The surrogate above illustrates the reward–cost combination rather than specifying every detail of MAPPO-L. The Algorithm-3 MAPPO-L baseline used in our experiments updates independent actors sequentially and resets its nonnegative multipliers at each actor’s optimization phase; its cost objectives and budgets are specified with each benchmark. Reward-only MAPPO omits the cost term. SafeDiSTL uses attributed cost advantages instead of the broadcast shared-cost advantage, as described in Section 5.

Clipping and a finite number of stochastic primal–dual updates do not enforce exact feasibility at each iteration. In particular, this first-order implementation does not inherit CPO’s trust-region bounds or the guarantees of SafeMARL’s theoretical sequential update. Moreover, broadcasting $A_{C,t}$ supplies no explicit differentiation of constraint responsibility across agents. Section 5 replaces this broadcast signal with trajectory-dependent attribution while retaining the exact shared constraint for dual accounting.

B Supplementary Transport Results

Table 1 provides the numerical results plotted in Figure 3.

Table 2: Supplementary Navigation results compared to baselines. Entries are mean $\pm$ sample SD across six seeds and 128 common resets per seed. Goal success, constraint satisfaction, and safe success are percentages; safe success requires both goal achievement and whole-episode connectivity. Safe return assigns zero to unsafe episodes; final 20 step distance is in meters.

Algorithm	Return	Goal success (%)	Constraint satisfaction (%)	Safe success (%)	Safe return	Final 20 Step distance
MAPPO	130.43 $\pm$ 3.38	51.17 $\pm$ 3.31	0.91 $\pm$ 1.04	0.78 $\pm$ 0.99	1.49 $\pm$ 1.72	2.14 $\pm$ 0.16
MACPO	90.32 $\pm$ 9.70	11.85 $\pm$ 5.79	31.25 $\pm$ 4.39	4.43 $\pm$ 1.96	29.24 $\pm$ 3.29	2.59 $\pm$ 0.23
MAPPO-L	114.14 $\pm$ 8.71	52.99 $\pm$ 12.67	56.64 $\pm$ 10.61	32.81 $\pm$ 8.62	69.80 $\pm$ 10.19	1.84 $\pm$ 0.27
Ours	125.11 $\pm$ 8.80	71.88 $\pm$ 6.13	54.69 $\pm$ 8.33	43.88 $\pm$ 4.40	75.31 $\pm$ 10.50	1.83 $\pm$ 0.30

Table 3: Supplementary Traffic results relative to MAPPO (column-wise MAPPO mean = 1). Mean $\pm$ SD across six training seeds; 1,000 common reset scenarios per policy. Higher completed trips and lower values in all other columns are preferable.

Method	Completed trips	Network vehicle-hours	Core vehicle-hours	Episode-max core cost	Time above core limit
MAPPO	1.000 $\pm$ 0.012	1.000 $\pm$ 0.001	1.000 $\pm$ 0.041	1.000 $\pm$ 0.081	1.000 $\pm$ 0.010
MACPO	0.476 $\pm$ 0.007	1.053 $\pm$ 0.001	0.091 $\pm$ 0.002	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
MAPPO-L	0.425 $\pm$ 0.016	1.058 $\pm$ 0.002	0.104 $\pm$ 0.007	0.00074 $\pm$ 0.00180	0.009 $\pm$ 0.023
Ours	0.714 $\pm$ 0.066	1.029 $\pm$ 0.007	0.088 $\pm$ 0.018	0.000061 $\pm$ 0.000053	0.003 $\pm$ 0.004

C Supplementary Navigation Results

Table 2 provides the numerical results plotted in Figure 6, together with safe return and final-20 distance.

D Supplementary Traffic Results

Table 3 retains the full traffic metrics, complementing Figure 8.

E Ablation Studies

We ablate responsibility attribution, the cost formulation, and the dual optimizer. The four variants use uniform attribution with either max-incremental cost or step cost, with fixed-step or Adam multiplier updates. Max-incremental cost is the same incremental maximum cost Δc_t defined for SafeDiSTL in Equation 10; step cost uses positive per-transition violation. Labels without “+ Adam” use fixed-step multiplier updates. Ours retains sensitivity-based attribution with max-incremental cost. The Ours rows are repeated from the main tables, with the same evaluation protocols and seed-level aggregation. Results of ablations are presented in Tables 4, 5 and 6.

F Robustness Differentiation

We write the differentiable dynamics as $s_{t+1} = F(s_t, \mathbf{a}_t)$, with sampled disturbances held fixed. The norm in Equation (12) is over the action coordinates of agent i . Differentiating negative robustness

Table 4: Two-dimensional transport ablations for one through three obstacles. Rates are percentages; entries are mean $\pm$ sample SD across six training seeds, with 500 common resets per seed. All three Ours rows are copied unchanged from Table 1.

Obstacles	Algorithm	Return	Goal success	Shared constraint sat.	Safe success
1	Uniform attribution Max-incremental cost	58.91 ± 8.30	0.5 ± 0.8	74.4 ± 6.3	0.2 ± 0.4
1	Uniform attribution Max-incremental cost + Adam	70.00 ± 26.21	8.4 ± 8.7	75.3 ± 6.5	5.6 ± 7.7
1	Uniform attribution Step cost	46.81 ± 10.14	0.0 ± 0.0	57.7 ± 9.1	0.0 ± 0.0
1	Uniform attribution Step cost + Adam	44.26 ± 5.41	0.0 ± 0.0	58.6 ± 11.7	0.0 ± 0.0
1	Ours	125.83 ± 10.24	42.2 ± 9.7	82.0 ± 5.9	35.6 ± 9.6
2	Uniform attribution Max-incremental cost	59.40 ± 6.96	0.7 ± 1.4	61.8 ± 8.9	0.2 ± 0.3
2	Uniform attribution Max-incremental cost + Adam	80.63 ± 23.60	6.5 ± 5.8	73.3 ± 10.1	4.1 ± 4.0
2	Uniform attribution Step cost	38.41 ± 3.55	0.0 ± 0.0	60.1 ± 18.6	0.0 ± 0.0
2	Uniform attribution Step cost + Adam	44.65 ± 7.49	0.0 ± 0.0	52.1 ± 13.1	0.0 ± 0.0
2	Ours	104.52 ± 8.15	23.8 ± 12.5	55.4 ± 13.6	16.2 ± 11.0
3	Uniform attribution Max-incremental cost	52.54 ± 10.11	0.0 ± 0.0	62.2 ± 4.1	0.0 ± 0.0
3	Uniform attribution Max-incremental cost + Adam	81.79 ± 12.46	1.7 ± 1.1	70.8 ± 6.4	0.6 ± 0.7
3	Uniform attribution Step cost	42.93 ± 3.33	0.0 ± 0.0	50.1 ± 15.6	0.0 ± 0.0
3	Uniform attribution Step cost + Adam	44.66 ± 6.93	0.0 ± 0.0	54.6 ± 15.6	0.0 ± 0.0
3	Ours	104.06 ± 6.74	13.1 ± 6.6	17.4 ± 12.4	3.5 ± 2.3

rather than its rectified violation avoids an additional zero-gradient region when the smoothed trajectory is safe; the allocated cost is nevertheless zero whenever the exact $C^\phi(\tau)$ is zero.

Attribution normalization in implementation. Let $\mathcal{I}_t$ contain the active agents at valid decision time t , and let $Z_t = \sum_{i \in \mathcal{I}_t} z_{i,t}$. The sum in Equation (13) excludes inactive entries. With numerical threshold $\varepsilon_g > 0$, the masked normalization is

$$w_{i,t} = \begin{cases} z_{i,t}/Z_t, & i \in \mathcal{I}_t, Z_t > \varepsilon_g, \\ 1/|\mathcal{I}_t|, & i \in \mathcal{I}_t, Z_t \leq \varepsilon_g, \\ 0, & i \notin \mathcal{I}_t. \end{cases} \quad (25)$$

The uniform fallback handles finite, uninformative sensitivities and preserves $\sum_i w_{i,t} = 1$ at each valid time with at least one active agent. Invalid padding receives zero weight; allocation assumes a nonempty active set at every cost-bearing transition. The factor N in Equation (14) corresponds to the fixed team-size mean, including zero contributions from inactive agents. Nonfinite active derivatives are flagged rather than assigned uniform credit. Complete episodes are required before

Table 5: Navigation ablations. Metrics and evaluation resets match Table 2; Ours is copied unchanged. Entries are mean $\pm$ sample SD across six training seeds. Goal, constraint-satisfaction, and safe-goal rates are percentages; final-20 distance is in meters.

Algorithm	Return	Goal (%)	Safe (%)	Safe goal (%)	Safe return	Final-20 distance
Uniform attribution	126.77 $\pm$ 9.41	54.30 $\pm$ 15.19	2.60 $\pm$ 2.55	2.60 $\pm$ 2.55	4.21 $\pm$ 4.19	2.13 $\pm$ 0.34
Max-incremental cost						
Uniform attribution	128.58 $\pm$ 5.08	50.52 $\pm$ 13.01	0.39 $\pm$ 0.65	0.39 $\pm$ 0.65	0.65 $\pm$ 1.10	2.17 $\pm$ 0.29
Max-incremental cost + Adam						
Uniform attribution	39.60 $\pm$ 22.40	12.63 $\pm$ 9.34	90.36 $\pm$ 4.57	11.98 $\pm$ 9.11	36.42 $\pm$ 20.95	5.29 $\pm$ 1.66
Step cost						
Uniform attribution	129.38 $\pm$ 15.43	61.33 $\pm$ 20.97	11.98 $\pm$ 8.81	10.16 $\pm$ 9.40	19.28 $\pm$ 14.75	1.94 $\pm$ 0.43
Step cost + Adam						
Ours	125.11 $\pm$ 8.80	71.88 $\pm$ 6.13	54.69 $\pm$ 8.33	43.88 $\pm$ 4.40	75.31 $\pm$ 10.50	1.83 $\pm$ 0.30

Table 6: Traffic ablations. Values use the same fixed, column-wise MAPPO mean as Table 3; Ours is copied unchanged. Mean $\pm$ sample SD across six training seeds and 1,000 common resets per policy. Higher completed trips and lower values in all other columns are preferable.

Method	Completed trips	Network vehicle-hours	Core vehicle-hours	Episode-max core cost	Time above core limit
Uniform attribution	0.786 $\pm$ 0.018	1.021 $\pm$ 0.002	0.079 $\pm$ 0.017	0.00031 $\pm$ 0.00058	0.005 $\pm$ 0.009
Max-incremental cost					
Uniform attribution	1.002 $\pm$ 0.010	1.000 $\pm$ 0.001	1.011 $\pm$ 0.076	1.002 $\pm$ 0.111	1.000 $\pm$ 0.010
Max-incremental cost + Adam					
Uniform attribution	0.388 $\pm$ 0.235	1.063 $\pm$ 0.025	0.065 $\pm$ 0.038	0.00000049 $\pm$ 0.00000076	0.000041 $\pm$ 0.000063
Step cost					
Uniform attribution	0.881 $\pm$ 0.033	1.011 $\pm$ 0.003	0.837 $\pm$ 0.050	0.788 $\pm$ 0.082	0.985 $\pm$ 0.012
Step cost + Adam					
Ours	0.714 $\pm$ 0.066	1.029 $\pm$ 0.007	0.088 $\pm$ 0.018	0.000061 $\pm$ 0.000053	0.003 $\pm$ 0.004

full-episode sensitivities and attribution targets can be finalized, even though the cost allocation is time-local. Sensitivity norms discard derivative direction, and attribution depends on the model, action parameterization, and smoothing scale. Conservation of scalar cost does not make the attributed policy gradient an unbiased estimator of the original shared-cost gradient.

The exact minimum can concentrate its derivative on a single predicate and time. For the always-conjunction in Equation (8), we use the log-sum-exp approximation

$$\tilde{\rho}_\beta^\phi(\tau) = -\frac{1}{\beta} \log \sum_{t=0}^{T_\tau-1} \sum_{j=1}^K \exp(-\beta g_j(s_{t+1})), \quad (26)$$

where $\beta > 0$ controls smoothing. Other supported Boolean and temporal compositions use their corresponding smooth minimum or maximum. The surrogate supplies derivatives only: its value and sign do not replace the exact cost or the exact satisfaction test.

G SafeDiSTL Optimization and Implementation Details

G.1 Cost advantages and critic losses

The hybrid cost critic consists of a centralized predictor $V_C^g(s_t, c_{\max, t-1})$ and a shared local-observation predictor $V_C^i(o_{i,t}, c_{\max, t-1})$, evaluated separately for each agent. The local heads predict expected remaining cumulative attributed costs from $\tilde{c}_{i,t}$ and the global head predicts

their team mean from $\bar{c}_t$, defined in Equation (16), with the numerical scale below. Their learned outputs are not an exact decomposition by architecture.

The regression targets in Equation (18) use unnormalized GAE estimates and rollout-time value predictions, and are held fixed during critic fitting. Cost prediction and GAE use the same fixed numerical scale $\kappa_C > 0$ on both $\tilde{c}_{i,t}$ and $\bar{c}_t$. Let b_t be one if the episode continues after transition t , and zero at either termination or the finite-horizon endpoint. With cost discount one, the agent-specific temporal-difference residual and generalized advantage estimate are

$$\begin{aligned}\delta_{i,t}^C &= \kappa_C \tilde{c}_{i,t} + b_t V_C^i(o_{i,t+1}, c_{\max,t}) - V_C^i(o_{i,t}, c_{\max,t-1}), \\ A_{i,t}^C &= \sum_{\ell=0}^{T_\tau-1-t} \eta_C^\ell \delta_{i,t+\ell}^C,\end{aligned}\tag{27}$$

where $\eta_C \in [0, 1]$ is the cost trace-decay parameter. The global cost estimate uses the same recursion with $\bar{c}_t$ and V_C^g . Cost targets do not bootstrap beyond the completed episode. The GAE-based cost targets $y_{i,t}^C$ and $y_{g,t}^C$ are defined in Equation (18), using fixed rollout-time value predictions. The cost critic is trained with clipped squared-error value losses and a consistency penalty:

$$\begin{aligned}\mathcal{L}_C &= \widehat{\mathbb{E}}_t [\ell_v(V_C^g, y_{g,t}^C) + \frac{1}{N} \sum_i \ell_v(V_C^i, y_{i,t}^C) \\ &\quad + \zeta (\frac{1}{N} \sum_i V_C^i - V_C^g)^2].\end{aligned}\tag{28}$$

Here ℓ_v clips changes relative to the rollout-time value prediction, and $\zeta \geq 0$ controls consistency. This regularizer encourages agreement of the predicted mean with the global prediction; Equation 15 holds for the allocated costs independently of critic accuracy, not necessarily for the learned values or generalized advantage targets.

G.2 Actor normalization and individual constraints

If agent i also has local constraints indexed by j , let $A_{C,i,j,t}^{\text{loc}}$ and $\lambda_{i,j}^{\text{loc}}$ denote their separately estimated, scaled cost advantages and multipliers. We form

$$A_{i,t}^{\text{norm}} = \mathcal{N} \left(A_{R,t} - \lambda_k A_{i,t}^C - \sum_j \lambda_{i,j}^{\text{loc}} A_{C,i,j,t}^{\text{loc}} \right),\tag{29}$$

where $\mathcal{N}$ denotes normalization of the combined advantage over valid actor samples. The implementation substitutes this normalized advantage into Equation (20) and adds an entropy bonus during optimization. Individual obstacle and workspace costs do not enter the shared robustness gradient or its normalization. They retain their own critics, budgets, and dual updates. Thus, attributing a payload violation cannot erase an individual robot’s constraint.

For a local clearance $\ell_{i,j,t}$ and required buffer Δ_j , the local signal is

$$c_{i,j,t}^{\text{loc}} = [\Delta_j - \ell_{i,j,t}]_+.\tag{30}$$

Each local dual uses the mean raw deficit of its own agent and channel over active transitions, not an average across teammates. These are stepwise budgets, distinct from the shared episode-maximum budget. The local critic uses its configured discounted cost return and bootstraps clean horizon truncations, whereas the completed-episode shared STL return does not. In the three-dimensional suites, both local buffers are 0.05 m, local cost discount and trace decay are 0.99 and 0.95, and local multipliers use projected budget-relative updates with stepsize 10^{-3} and cap two.

G.3 Shared multiplier update

The shared multiplier is updated from the mean exact, unallocated cost of completed episodes. In the budget-relative, fixed-step configuration, the update is

$$\begin{aligned}\hat{C}_k &= \frac{1}{B_k} \sum_{b=1}^{B_k} C^\phi(\tau_b), \\ \lambda_{k+1} &= \text{clip} \left(\lambda_k + \eta_\lambda \frac{\hat{C}_k - d}{d}, 0, \lambda_{\max} \right),\end{aligned}\tag{31}$$

with positive d and B_k completed episodes in the dual-update window. Neither κ_C nor the attribution weights change this physical-unit constraint statistic. The finite multiplier cap is a practical optimization choice, not an exact solution of the unrestricted saddle-point problem.

H Supplementary Experimental Details

H.1 Collaborative Transport

The transport environment contains three force-controlled agents, an unactuated nonrotating square package, and circular obstacles in a planar workspace. The package has side length 0.15 m and mass 50; each agent has mass one and radius 0.03 m, and each obstacle has radius 0.10 m. Each agent selects a normalized two-dimensional action in $[-1, 1]^2$, scaled by 0.6 to produce the applied force. The differentiable VMAS-style simulator uses float64 smooth contact, and four 0.025 s physics substeps per control step. Episodes last 500 control steps.

Resets sample one of eight layouts, apply a random quarter-turn and reflection, jitter each obstacle by at most 0.015 m per coordinate, and randomize the agents’ phase around the package. The layouts obstruct the direct route while providing detours.

The shared constraint margin is the package-box signed distance to each obstacle, minus the obstacle radius and the required 0.02 m clearance. Goal achievement uses the task’s box-distance threshold of approximately 0.1567 m rather than a point-payload center-distance test. The shared reward combines distance-progress shaping scaled by 100 with the campaign’s approach and joint-push coordination terms, held fixed across algorithms. Package contact is logged separately; it neither adds a reward penalty nor terminates the episode.

H.2 Connected Navigation

Four homogeneous planar agents navigate from a compact start formation to a shared circular destination through two staggered obstacle passages. The canonical corridor is 18 m long and 10 m wide; ten static circular obstacles form two five-obstacle barriers. Agent and obstacle radii are 0.75 m and 0.8 m, respectively. Resets randomize barrier locations, opening centers, and opening widths (3.4–3.8 m), perturb the start formation and destination, and apply a global rotation, reflection, and translation. Agents use bounded two-dimensional acceleration commands, maximum speed 1.5 m/s, smooth contact forces, five physics substeps per 0.1 s control step, and a 250-step horizon. The shared reward combines reduction in mean agent-to-goal distance, scaled by $5/R$, with the fraction of agents inside the destination. Goal success requires all four agents to remain within the 2.6 m goal radius throughout the final 20 control steps.

The formal constraint is whole-episode buffered graph connectivity. Let b_u be the longest edge in the Euclidean minimum-spanning tree of the agent positions at checked instant u . For transition

t , let $\mathcal{U}_t$ contain its source and five physics-substep endpoints. With communication radius $R = 3.5$ m and buffer fraction $\xi = 0.05$,

$$\begin{aligned} b_u &= \max_{(i,k) \in \text{MST}(x_u)} \|x_{i,u} - x_{k,u}\|_2, \\ m_t &= \min_{u \in \mathcal{U}_t} \frac{R(1 - \xi) - b_u}{\xi R}. \end{aligned} \tag{32}$$

The buffered check requires $m_t \geq 0$ for every transition, equivalently a checked bottleneck no larger than $R(1 - \xi) = 3.325$ m. The 0.175 m buffer exceeds the maximum relative displacement of 0.06 m between substep checks, satisfying the bounded-speed certificate condition. A safe episode passes every buffered check; a safe goal additionally satisfies the goal criterion. Smooth contacts do not count as connectivity violations, and separate obstacle and boundary cost channels are disabled. Disconnection does not terminate an episode.

H.3 Networked Traffic Control

The traffic task is a seven-region NMFD network with six peripheral regions and a shared core. Direct peripheral-to-peripheral transfers are disabled. Each region uses the cubic production function

$$G(n) = 4.133 \times 10^{-11} n^3 - 8.282 \times 10^{-7} n^2 + 0.0042n, \tag{33}$$

where n is occupancy in vehicles and G is in vehicles/s. Episodes contain 240 control intervals of 30 s, each integrated with ten RK4 substeps. Demand is five times the reference setting: 0.625 vehicles/s per nonlocal peripheral origin–destination pair, with $\pm 20\%$ reset randomization. Three destinations receive low exit baselines of 0.1 and three receive high baselines of 0.9, randomly permuted at reset; temporal noise and incidents vary core exits within $[0, 1]$.

Agent i controls a six-dimensional admission vector $u_i \in [0, 1]^6$. Coordinate u_{id} meters traffic in its region bound for destination d into the core; the self-destination coordinate is masked, leaving five effective controls. If x_{id} is the commodity occupancy and n_i the region total, its core-entry rate is $u_{id}(x_{id}/n_i)G(n_i)$, with zero flow for an empty region. The shared reward is completed trips per interval divided by 100. The core constraint is $n_0 \leq n_{\text{safe}}$, where $n_{\text{safe}} \simeq 8786.415$ is the descending-branch solution of $G(n) = 1$.

I Observations and Policy Network Architectures

This section describes the actor inputs and policy networks used in the reported benchmarks.

I.1 Cooperative Transport

For agent i , the observation concatenates its absolute planar position and velocity (four coordinates), package position minus goal position (two), package position relative to the agent (two), absolute package velocity (two), and a binary package-on-goal indicator (one). It then includes the relative position of each of the K obstacles ($2K$ coordinates) and the relative position and velocity of each of the other two agents (eight coordinates), ordered by agent index. Thus the observation dimension is $19 + 2K$, giving 21, 23, and 25 inputs for the one-, two-, and three-obstacle conditions. The nonrotating package contributes no orientation features, and no assigned push-slot features are appended. Actor inputs are standardized using running means and variances estimated during training and clipped to $[-10, 10]$; evaluation uses the saved normalization statistics without updating them.

The actor is a feed-forward multilayer perceptron with two 64-unit hidden layers, tanh activations, and a linear two-dimensional Gaussian-mean head. A learned, observation-independent logstddev vector specifies the diagonal covariance. Training samples a latent Gaussian action and applies tanh to obtain the normalized force command in $[-1, 1]^2$; deterministic evaluation applies tanh to the mean. The simulator multiplies this command by 0.6. MAPPO, SafeDiSTL, and the uniform-attribution variants share one actor across the three agents; MAPPO-L and MACPO use separate actor parameters for each agent with the same layer sizes and action parameterization.

I.2 Connected Navigation

Each agent receives a 72-dimensional observation. The first 34 coordinates comprise its own planar position and velocity (four), normalized episode time (one), four relative target positions (eight), and three teammate slots (21). The four target entries refer to the shared destination. Each teammate slot contains relative position and velocity from the stored communication packet, a direct-neighbor indicator, a connected-component reachability indicator, and packet age; slots follow the agents’ fixed identifiers within the episode. The remaining 38 coordinates describe the ten obstacles using relative position and radius (30), the four corridor-boundary clearances (four), and the flattened two-dimensional world-rotation matrix (four). Positions, relative distances, obstacle radii, and boundary clearances are divided by 10; velocities are divided by the maximum speed of 1.5; time and packet ages are divided by the 250-step horizon. These features are then standardized using running observation statistics and clipped to $[-10, 10]$, with the saved statistics held fixed during evaluation.

The actor maps these 72 inputs through two 64-unit tanh hidden layers to a linear two-dimensional Gaussian-mean head. It uses a learned, observation-independent diagonal log-standard-deviation vector and tanh-bounded actions in $[-1, 1]^2$. Deterministic evaluation uses the transformed mean. The simulator additionally clips the command’s Euclidean norm to one before applying the acceleration scale. MAPPO, SafeDiSTL, and the uniform-attribution variants share actor parameters across the four agents, whereas MAPPO-L and MACPO use one independently parameterized actor per agent with the same network dimensions. The policies are feed-forward; communication packet contents and ages are explicit observation features, not a recurrent policy state.

I.3 Networked Traffic Control

Each peripheral agent receives 31 observation coordinates: its six destination-specific queues, six local origin–destination demand rates, six destination-specific queues in the shared core, six current effective core-exit factors, normalized episode time, and a six-dimensional one-hot agent identity. Queue occupancies are divided by 1,800; demand rates and exit factors are used directly, and time is divided by the 240-step horizon. The core queues and exit factors are broadcast to all six agents. The observation excludes the latent variables driving future exit disturbances. The identity is already part of the environment observation, so the learner does not append another identity vector. There is no additional running observation normalization, and actor inputs use float32 arithmetic.

The actor has two 64-unit tanh hidden layers and a linear six-dimensional Gaussian-mean head, with a learned, observation-independent log-standard-deviation vector. A sigmoid transforms each latent Gaussian sample into an admission command in $[0, 1]$. The self-destination coordinate is masked to zero, leaving five effective controls per agent; deterministic evaluation applies the same sigmoid and mask to the mean. MAPPO, SafeDiSTL, and the uniform-attribution variants share

one actor across the six agents. MAPPO-L and MACPO instead maintain six independent actors with the same input features, hidden-layer sizes, and action transform.