CorridorLight: Cooperation as Task Negotiation via Counterfactual-Gain Gating for Traffic Signal Control

Pak Lon Ip[1]* paklonip@um.edu.mo

Pengfei Ren[1]* yc27491@um.edu.mo

Yuteng Lin[1] mc35365@um.edu.mo

Rongqin Chen[1] yc07917@um.edu.mo

Tsz Nam Chan[2] edisonchan@szu.edu.cn

Leong Hou U[1] ryanlhu@um.edu.mo

Abstract

Network-level traffic signal control (TSC) involves coordination under partial observability and strict real-time constraints. In real-world urban networks with dozens or hundreds of intersections, direct coordination incurs an intractably large joint action space, making centralized cooperation difficult to deploy in practice. We propose CorridorLight, which tackles this challenge by casting cooperation as task negotiation. A high-level Corridor Agent represents the network as a direction-level graph and periodically generates a small set of junction-disjoint corridor tasks via a seed-and-grow procedure. These tasks provide lane-level preference guidance, while Intersection Agents maintain fast decentralized control and engage in a continuous negotiation to selectively support the active corridor task based on real-time local conditions. By learning when to cooperate versus act selfishly, CorridorLight enables an adaptive trade-off between local delay minimization and corridor-level relief, leading to improved global performance. To this end, we derive a closed-form, Lipschitz-continuous gating function by maximizing a counterfactual net gain between cooperative and selfish behavior, yielding smooth, interpretable cooperation without brittle heuristics. Experiments on synthetic grids and real-world networks show consistent system-level improvements and outperform cooperative TSC baselines across diverse topologies and demand patterns. The code is under https://github.com/ippaklon/corridor-light.

1 Introduction

Urban Traffic Signal Control (TSC) is a challenging testbed for real-world reinforcement learning due to high-dimensional observations, severe partial observability, and strong spatiotemporal dependencies induced by vehicle platoons and spillback. While TSC is naturally modeled as a multi-agent reinforcement learning (MARL) problem, existing methods still struggle to reconcile *scalability* with *coordination*. Centralized approaches that treat an entire city as a single agent suffer from an exponential growth in the joint action space and quickly become computationally intractable on large networks [37, 10]. At the other extreme, decentralized independent learning scales well but often exhibits a "tragedy of the commons": locally greedy improvements at one intersection can worsen downstream congestion and harm global throughput [41, 53]. Value factorization and actor–critic methods such as QMIX [39] and MAPPO [58] partially alleviate this via centralized training, yet they can remain fragile under the highly non-stationary dynamics of city-scale traffic.

Cooperation under fixed interfaces. Existing cooperative MARL for TSC typically relies on one of three mechanisms: (i) explicit inter-agent communication, (ii) centralized critics/joint value functions

*Equal contribution.

[1]SKL of Internet of Things for Smart City, University of Macau.

[2]College of Computer Science and Software Engineering, Shenzhen University.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

under Centralized Training with Decentralized Execution (CTDE), or (iii) global reward shaping that forces all intersections to optimize a shared scalar signal. However, communication may be unreliable in realistic deployments, centralized critics become increasingly costly and brittle at city scale, and a single global scalar reward can conflict with local operational objectives [44, 43, 9] (e.g., emergency vehicles and intersection-specific efficiency trade-offs). This leaves a practical gap: we need a cooperation mechanism that scales to large networks, remains fully decentralized at execution time, and *does not expand the per-intersection observation/action interfaces*.

Traffic motivation: corridor (green-wave) structure. A key property of urban traffic is that congestion, spillback, and platoon progression predominantly propagate along directional arterials. Signal coordination along such arterials can create a green wave that reduces stops and improves corridor throughput [8, 66]. This motivates a *traffic-aligned* coordination primitive: rather than static global partitions, we form and prioritize a small set of *priority directional corridor tasks* that explicitly target flow along these propagation paths.

To this end, we propose **CorridorLight**, a bi-level cooperative learning framework that formulates network-level coordination as a dynamic task assignment problem (Figure 1). Our key insight is to view cooperation not as message sharing [55, 4], but as *dynamic task negotiation*: rather than dictating actions, a global planner proposes a small set of corridor tasks, and each local intersection negotiates its degree of participation using a counterfactual-gain gating function that weighs marginal global benefits against local friction.

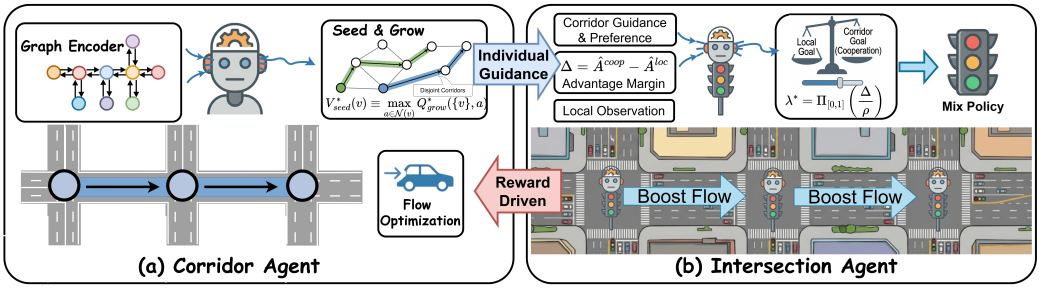


Figure 1: (a) The Corridor Agent encodes network states with a graph encoder and solves a seed-and-grow planning MDP to generate corridor guidance (paths and preference weights) that targets corridor-level flow. (b) Each Intersection Agent keeps its original local observation and signal-phase action set, receives the corridor guidance, estimates the marginal benefit of cooperation, and adaptively mixes a local policy with a corridor-aware policy to execute signal actions.

We instantiate this idea as a *task-based vertical cooperation* protocol that explicitly decouples the *formation* of global objectives from their local *execution*:

1. **Corridor Agent (global planner).** It observes graph-structured network states to identify bottlenecks and constructs a small set of junction-disjoint corridor tasks with internal worth estimates (for ranking). It then broadcasts only lightweight task descriptors and lane-level preferences that modulate local behavior.
2. **Intersection Agents (local controllers).** Each intersection retains fast decentralized actuation but selectively aligns with corridor tasks via a continuous cooperation gate, enabling smooth negotiation over hard switching.

Corridor grouping vs. global grouping. Prior work often groups intersections at a global level (e.g., through clustering or partitioning) and optimizes group-level objectives [35, 60, 40, 26]. Compared with such *global grouping*, our *corridor grouping* better matches traffic propagation: spillback and congestion mainly spread along directional arterials rather than arbitrary static partitions. More importantly, CorridorLight does not treat grouping as a one-shot combinatorial preprocessing step. Instead, we *formulate corridor construction as an episodic Markov Decision Process (MDP)* and learn a seed–grow planner that receives *direct reward-driven signals* for adding members, avoiding inefficient, hard-to-train grouping optimization via indirect long-horizon credit assignment [27, 19].

Interface-preserving and dependency-free. CorridorLight is interface-preserving and imposes no inter-agent information dependency: unlike information-sharing approaches [18, 65, 28], each

intersection agent keeps the same local observation and the same signal-phase action set. The corridor guidance provides only task descriptors and lane-level preferences, without exposing any other agent's private states or actions. As a result, an intersection does not need to explicitly model other agents' behaviors. Instead, it can focus on its own local context and adaptively determine its level of participation. This design also reduces decision complexity at both layers for scalability: at the upper layer, corridor formation becomes structured incremental planning on a sparse connectivity graph, avoiding network-wide combinatorial choices at each step. At the lower layer, each intersection processes guidance defined only on its own lanes and realizes cooperation via a lightweight mechanism while keeping the original action space unchanged.

Contribution Summary. *a) A new cooperation scheme without expanding interfaces.* We propose a bi-level cooperation framework that achieves network-level task optimization *without* enlarging per-intersection observation/action interfaces. Cooperation is realized via compact task descriptors (corridor masks and lane-level preferences), minimizing information dependency while preserving decentralized actuation.

b) Corridor planning with seed–grow task formation. We model corridor construction as an episodic MDP on a direction graph and propose a seed–grow planner that outputs a small set of junction-disjoint corridor tasks with worth estimates, enabling long-range coordination beyond local neighborhoods.

c) MGDA-based preference learning with automatic cooperation strength. We develop an end-to-end training recipe that stabilizes multi-objective learning via a Multiple Gradient Descent Algorithm (MGDA)-style shared-parameter update (replacing manual scalarization), together with a counterfactual-gain-based gating that determines the cooperation degree λ for each intersection.

2 Related Work

Traditional traffic signal control relies on engineered timing rules and heuristics, including fixed-time plans, traffic-responsive/coordinated systems, and feedback policies based on queue or pressure dynamics [51, 32, 20, 46, 48, 15]. These methods, ranging from analytical timing strategies and detector-based control to feedback policies such as MaxPressure [46], are interpretable and reproducible, but rely on simplified assumptions and hand-crafted objectives. Consequently, they struggle under high-dimensional, partially observed, and non-stationary city-scale dynamics. Learning-based approaches, particularly reinforcement learning, address these limitations by learning policies from data and capturing spatiotemporal dependencies end-to-end [57, 52, 54, 21, 63, 62, 56].

Explicit communication. To address the information-sharing challenge, many cooperative MARL methods enable coordination by exchanging representations at decision time, allowing each intersection to condition on non-local context [33, 14, 61]. A dominant paradigm aggregates multi-intersection features via attention or graph encoders to form coordination-aware inputs [53, 30], sometimes learning adaptive communication structures or global summaries [22, 40, 25]. While effective, such approaches incur communication overhead, introduce noisy or redundant signals, and complicate decision making as network scale grows.

Centralized training and global reward coordination. To improve training stability, a prevalent line of work adopts *centralized training with decentralized execution (CTDE)*, where agents leverage joint information during training while acting locally [31, 23, 50, 64]. This is commonly instantiated via value factorization (e.g., VDN, QMIX) or centralized critics (e.g., MAPPO) [45, 39, 38, 49, 58]. However, centralized critics and joint value functions can scale poorly and become brittle under highly non-stationary traffic dynamics.

A complementary line of work coordinates agents via *reward design*, using global or structured rewards to align local policies with system-level objectives [34, 6, 17]. However, such rewards can worsen *credit assignment* by entangling individual contributions; recent methods mitigate this with counterfactual baselines or collaborator selection [16, 40].

Overall, existing approaches face a fundamental trade-off: communication-heavy and centralized methods improve coordination but scale poorly and may blur each agent's marginal contribution to system-level performance, while reward-centric approaches better align global objectives yet suffer from diffuse and delayed feedback. These limitations suggest the need for structured, scalable coordination mechanisms that localize where cooperation matters and balance global objectives with local efficiency.

3 Method

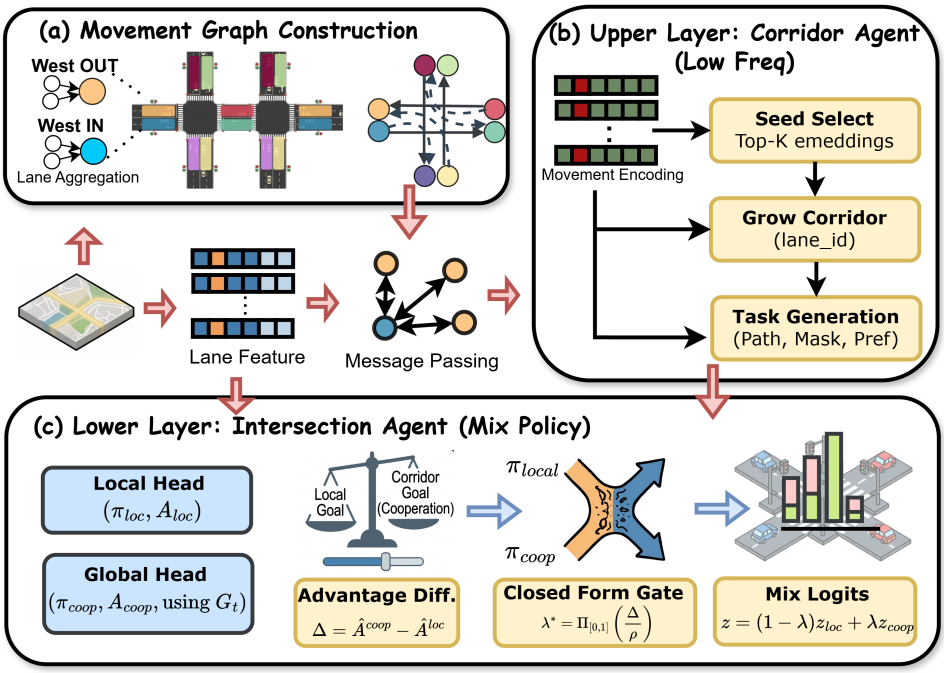


Figure 2: (a) Construct a movement graph by aggregating lanes with the same direction and linking them via junction topology. (b) At a lower planning frequency, the Corridor Agent selects seed nodes and grows junction-disjoint corridors, producing corridor paths, lane masks, and preference weights as tasks. (c) At a higher control frequency, each Intersection Agent adaptively mixes local and corridor-aware policies to generate the executed signal policy.

CorridorLight is a bi-level cooperative traffic signal control framework that separates *global task formation* from *local multi-objective execution* under end-to-end training (see Figure 2). The upper layer, a **Corridor Agent**, periodically constructs a small set of *junction-disjoint (non-overlapping)* corridor tasks, each defined by a sequence of intersections and per-lane preference weights. The lower layer consists of **Intersection Agents**, one per junction, which retain fast decentralized control while selectively cooperating with active corridor tasks via a continuous gating mechanism.

Unlike cooperation schemes based on explicit inter-agent messaging or large centralized critics, our approach uses a compact global task descriptor G_t (corridor masks and preference weights) and an endogenous participation gate $\lambda_{i,t}$, where i indexes intersections and t indexes control steps. This reframes cooperation as *structured task allocation with continuous local negotiation*: the global layer identifies which corridors to coordinate, while each local agent chooses its commitment by weighing the counterfactual marginal gain of cooperation against the cost of deviating from local optimality. This yields a scalable, interface-preserving mechanism for targeting corridor-scale congestion under real-time constraints.

3.1 Movement Graph and GNN Encoding

Corridor planning primarily reasons about *directional flows* (e.g., eastbound-through, northbound-left) rather than lane geometry. Raw lane graphs are often too fine-grained and layout-dependent: varying lane counts, turn pockets, and connectivity yield variable-dimensional states that generalize poorly. We therefore aggregate the lane network into a *movement (direction) graph* that preserves flow semantics while abstracting lane-count details (see Fig. 2).

At each planning step, we convert the directed lane graph into a movement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by merging parallel lanes that share the same approach and movement (e.g., eastbound through). Each node $v \in \mathcal{V}$ denotes a movement and each directed edge $(u, v) \in \mathcal{E}$ indicates feasible downstream connectivity across junctions. This yields a fixed semantic granularity that aligns state dimensions

across heterogeneous intersections, avoiding full lane-level graphs (too fine/layout-dependent) and junction-level graphs (too coarse for directional flow).

To make corridor formation depend on network structure rather than isolated local measurements, we encode $\mathcal{G}$ with a directed message-passing Graph Neural Network (GNN). Let $h_v^{(0)} = \phi(x_v, \text{lane features})$ be the initial node features, where $\phi(\cdot)$ denotes a linear projection layer that embeds raw lane features $x_v \in \mathbb{R}^d$ into a hidden space; a K-layer message-passing network produces embeddings $h_v^{(K)}$ that aggregate upstream/downstream context. These embeddings capture information beyond local neighborhoods and serve as structured observations for the Corridor Agent's seed-and-grow decisions.

3.2 Corridor Agent

3.2.1 Seed–Grow Decomposition

At planning cycle τ, the Corridor Agent uses the movement graph and its GNN embeddings to construct at most M ordered corridors. A lightweight head scores movement nodes as seeds. The planner examines the top-k candidates, skipping those whose junctions are already assigned to accepted corridors, as tracked by J_{used}.

From each remaining seed, a masked policy π_{grow} sequentially selects a feasible successor or `STOP`, subject to a limit of $L_{\max}$ nodes, including the seed. Only corridors with at least two nodes are accepted. Their junctions are added to J_{used} before the next seed is considered; singleton corridors are discarded. Planning ends when M corridors have been accepted or the candidate seeds are exhausted. The feasibility mask defined below ensures that accepted corridors are junction-disjoint.

Each intersection on an accepted corridor receives a task descriptor G_t containing only its relevant lane masks and preference weight. These descriptors constitute the agent's output and remain active until the next planning cycle. Algorithm 1 summarizes the procedure, and Figure 3 illustrates it.

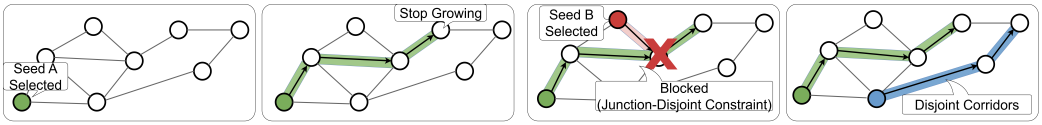


Figure 3: Seed-and-grow corridor construction with junction-disjoint constraints across accepted corridors.

3.2.2 Corridor Generation as an Episodic MDP

We model growth from a selected seed as a finite-horizon MDP under a fixed planning context ξ_τ, comprising the traffic observation, GNN embeddings, and J_{used}. Seed selection and updates to J_{used} occur between construction episodes. Construction itself neither advances environment step nor updates the traffic observation.

The state is the ordered prefix $P_q = (v_0, \ldots, v_q)$. At a nonterminal prefix, the feasible action set is

$$\mathcal{A}_{\xi_\tau}(P_q) = \left\{u \in \mathcal{N}^{\text{out}}(v_q) : u \notin \{v_0, \ldots, v_q\},\ J(u) \notin J_{\text{used}}\right\} \cup \{\texttt{STOP}\}, \tag{1}$$

where $J(u)$ is the junction associated with movement node u. Thus, growth follows directed movement edges without revisiting a node or entering a junction assigned to an accepted corridor.

The policy $\pi_{\text{grow}}(a_q \mid P_q, \xi_\tau)$ selects from this masked set. A successor is appended deterministically; `STOP` or the node limit terminates the episode. If no successor is feasible, `STOP` is the only action.

All corridors planned at cycle τ are evaluated over the same execution interval $t_\tau \le t < t_{\tau+1}$. For an accepted corridor c, extension $v_q \to v_{q+1}$ receives credit $r^c_{\tau,q}$ from the lanes represented by its selected successor v_{q+1}. At each execution step, the available rewards of these lanes are summed; these sums are then averaged over steps with valid observations. Lane rewards use the same definition and scaling as those of the local agents.

An idealized terminal-reward MDP is obtained by fixing the lower-level policy and the protocol for completing, executing, and evaluating the corridor tasks. Construction transitions then have zero

intermediate reward, $\gamma_{\mathrm{g}} = 1$, and expected terminal payoff

$$\bar{R}_{\xi_\tau}(c) = \mathbb{E}[R_\tau(c) \mid \xi_\tau, c]\,. \tag{2}$$

The implementation uses the collected feedback in two ways. The corridor-level feedback $R_\tau(c)$ supervises the seed score and the trajectory-averaged grow score. The growth actor instead uses a PPO-style objective weighted by each valid extension's own credit $r^c_{\tau,q}$, rather than by the corridor payoff or a return-to-go. This edge-local weighting is a credit-assignment surrogate, not the standard policy-gradient estimator for the terminal-reward MDP: an extension's feedback may depend on later growth decisions and other tasks executed in the same window. Appendix A.3 describes the credit calculation, the handling of missing observations, and other implementation details

3.2.3 Value Consistency between Seeding and Growing

Seeding and growing are designed as separate modules: seeding ranks eligible movement nodes globally, whereas growing selects feasible extensions locally. Rather than treating them as unrelated scoring problems, we connect them through the value functions of the same corridor-construction MDP. Under the fixed planning context ξ_τ and execution protocol of Section 3.2.2, let $V^*_{\mathrm{seed}}(v \mid \xi_\tau)$ denote the maximum expected terminal payoff obtainable from seed v, and let $Q^*_{\mathrm{grow}}(P, a \mid \xi_\tau)$ denote the expected terminal payoff of taking action a at prefix P and growing optimally thereafter.

Proposition 3.1 (Seeding–Growing Bellman Consistency)**.** *For any eligible seed v,*

$$V^*_{\mathrm{seed}}(v \mid \xi_\tau) = \max_{a \in \mathcal{A}_{\xi_\tau}((v))} Q^*_{\mathrm{grow}}((v), a \mid \xi_\tau). \tag{3}$$

The proof is given in Appendix D.2.

Selecting a seed initializes growth at the singleton prefix (v) without advancing the environment or adding reward. Both sides therefore optimize over the same feasible continuations. The identity links seed quality to achievable downstream value, rather than to local traffic conditions alone: seeding compares the best continuations available from different starting nodes, while growing chooses how to realize that value from the current prefix.

This functional relationship motivates our shared GNN encoder. Although their decision scopes differ, both value targets depend on the same movement connectivity, traffic context, and reward structure. The seed head uses the shared embeddings to approximate V^*_{seed} or its induced global ranking, while the grow scoring head additionally conditions on the prefix and candidate action to provide a proxy for Q^*_{grow}. The architecture thus shares environmental representations while retaining decision-specific heads.

3.3 Intersection Agent: Task Negotiation via Counterfactual-Gain Gating and Multi-Objective Control

When the Corridor Agent identifies high-value corridor tasks, execution is delegated to distributed Intersection Agents. To induce corridor-level coordination without expanding local interfaces, the Corridor Agent provides *corridor guidance* by assigning each corridor movement a lightweight preference signal, which biases participating intersections toward corridor-aligned actions while they continue optimizing their own local objectives. To balance corridor relief with local throughput, we introduce a *Counterfactual-Gain Gating Mechanism* that dynamically arbitrates between a local policy π^{loc} and a task-conditioned *cooperative* policy π^{coop} via a continuous coefficient $\lambda_t \in [0, 1]$, rather than using fixed-weight scalarization.

3.3.1 Dual-Policy Formulation

For each agent i, we define two distinct policy heads sharing a common encoder:

- **Local Policy $\pi^{loc}_i(a|o^i_t)$:** Optimized purely for local intersection metrics (e.g., minimizing queue length), representing the agent's self-interest.
- **Cooperative Policy $\pi^{coop}_i(a|o^i_t, G_t)$:** Conditioned on G_t (lane mask and preference weights), optimized to support corridor-level relief while maintaining local feasibility.

Logits Interpolation (Executed Policy). For discrete actions, we implement policy mixing by interpolating logits and then applying softmax:

$$\begin{aligned} z_i(o_t^i, G_t) &= (1-\lambda_{i,t})\, z_i^{loc}(o_t^i) + \lambda_{i,t}\, z_i^{coop}(o_t^i, G_t), \\ \pi_i(a \mid o_t^i, G_t) &= \mathrm{softmax}\big(z_i(o_t^i, G_t)\big). \end{aligned} \tag{4}$$

Here $\lambda_{i,t} \in [0,1]$ is the participation gate.

Task Descriptor G_t and Lane-Level Preferences. Each intersection receives a compact task descriptor G_t from the Corridor Agent, consisting of (i) a binary lane mask indicating which local lanes belong to an active corridor and (ii) per-lane preference weights p_ℓ encoding the priority. Importantly, G_t is *intersection-local*: it is defined only over lanes within intersection i (e.g., $\ell \in \mathcal{L}_i$), so its dimensionality is fixed by the local junction geometry and does not scale with network size. The cooperative policy head conditions on G_t by modulating lane features, e.g., $f'_\ell = (1+p_\ell) f_\ell$, while the local policy head remains unchanged. Crucially, the executed action space is *unchanged*: both π_i^{loc} and π_i^{coop} output logits over the same discrete signal-phase set $\mathcal{A}_i$, so introducing the cooperative head does not increase $|\mathcal{A}_i|$ and therefore does not expand the per-intersection decision space. **Implementation-aligned note:** if available, we optionally concatenate a scalar grow Q-value (the corridor planner's local segment worth for this junction) to the *cooperative critic* input only, to aid value estimation. This scalar is not fed into the policy head and does not affect lane feature weighting. This design provides task awareness without expanding the actuation interface.

3.3.2 Optimization Objective: Marginal Gain vs. Friction

To determine $\lambda_{i,t}$, each agent i solves a local optimization where cooperation ($\lambda_{i,t} > 0$) occurs only when the *marginal gain* $\Delta_{i,t}$ is positive. We formally define this *participation advantage* using standardized advantages to prevent scale mismatch between the two objectives from trivially saturating the gate:

$$\Delta_{i,t} = \hat{A}_{i,t}^{coop} - \hat{A}_{i,t}^{loc}, \tag{5}$$

where A^{loc}, A^{coop} are computed with separate critics (e.g., GAE), and $\hat{A}^{loc}, \hat{A}^{coop}$ denote per-objective normalized advantages (zero mean, unit variance). Intuitively, $\Delta_{i,t} > 0$ indicates that the cooperative objective is locally more favorable than the local objective on this on-policy sample, warranting a larger participation degree.

We introduce a quadratic regularization term to penalize drastic shifts and model the "friction" of deviating from local optimality, mathematically formalizing the negotiation process between self-interest and global task compliance. The objective for λ at step t is:

$$\max_{0 \le \lambda \le 1} J(\lambda_{i,t}) = \lambda_{i,t}\Delta_{i,t} - \frac{\rho}{2}\lambda^2, \tag{6}$$

where $\lambda\Delta_{i,t}$ represents the linearly scaled marginal return, $\frac{\rho}{2}\lambda^2$ represents the strictly convex friction cost, and $\rho > 0$ is a conservatism hyperparameter.

3.3.3 Closed-Form Solution and Theoretical Analysis

To ensure efficiency and interpretability, we derive a closed-form solution for the gating factor λ.

Proposition 3.2. *Given the advantage $\Delta_{i,t}$ and friction parameter $\rho > 0$, the optimal participation gate $\lambda_{i,t}^\star$ is unique and is given by the projection:*

$$\lambda_{i,t}^\star \;=\; \Pi_{[0,1]}\left(\frac{\Delta_{i,t}}{\rho}\right),$$

where $\Pi_{[0,1]}(x) := \min\{1, \max\{0, x\}\}$. Moreover, the mapping $\Delta_{i,t} \mapsto \lambda_{i,t}^\star$ is $\frac{1}{\rho}$-Lipschitz continuous.

Remark 3.3. The derived solution induces an interpretable three-region behavior: Positive advantage differences induce partial participation, while full participation occurs when the advantage difference reaches or exceeds ρ. Moreover, the Lipschitz continuity ensures that estimation errors in $\Delta_{i,t}$ result in bounded policy shifts, thereby bounding the sensitivity of the participation gate to advantage-estimation errors(Formal proof in Appendix D.3).

Mechanism Properties. The derived solution ensures critical properties for stable MARL training:

- **Three-region structure:** The optimal gate $\lambda^*_{i,t} = \Pi_{[0,1]}(\Delta_{i,t}/\rho)$ induces an interpretable piecewise behavior: if $\Delta_{i,t} \leq 0$ then $\lambda^*_{i,t} = 0$ (no participation); if $0 < \Delta_{i,t} < \rho$ then $\lambda^*_{i,t} = \Delta_{i,t}/\rho$ (partial participation proportional to marginal gain); and if $\Delta_{i,t} \geq \rho$ then $\lambda^*_{i,t} = 1$ (full participation). The parameter ρ thus acts as a "price of autonomy" that controls how easily an intersection commits to the global corridor task.
- **Sensitivity to estimation errors:** The gate $\lambda^\star_{i,t}(\Delta_{i,t})$ is $1/\rho$-Lipschitz continuous. Thus, an advantage-difference perturbation of magnitude ϵ changes the participation strength by at most $\min\{1, \epsilon/\rho\}$. Increasing ρ reduces this sensitivity, at the cost of more conservative participation for a fixed advantage difference.

3.4 Theoretical Analysis on Stability

Although bi-level optimization entails dynamic task generation, we analyze two properties relevant to training stability: the sensitivity of the executed policy to advantage-estimation errors and the first-order Pareto properties of the multi-objective update.

Theorem 3.4 (Bounded Policy Shift)**.** *Let $\lambda^*_{i,t}(\Delta_{i,t}) = \Pi_{[0,1]}(\Delta_{i,t}/\rho)$. Assuming the local and cooperative policy logits (z^{loc}, z^{coop}) are bounded, the divergence of the executed policy π_i induced by a perturbation in the advantage-difference estimation $\epsilon = |\widehat{\Delta}_{i,t} - \Delta_{i,t}|$ is bounded by:*

$$D_{TV}(\pi_i(\cdot|\widehat{\Delta}_{i,t})||\pi_i(\cdot|\Delta_{i,t})) \leq \frac{L_z}{\rho}\epsilon, \tag{7}$$

where D_{TV} is the Total Variation distance and L_z is a constant depending on the Lipschitz constant of the softmax function and the logit magnitudes.

Remark 3.5. This bound characterizes the propagation of advantage-estimation errors through the gate while holding the policy heads fixed. For a fixed estimation error and logit difference, increasing ρ reduces the upper bound on the induced policy shift. The bound complements advantage normalization, which addresses differences in the scales of the two objectives. A formal proof is provided in Appendix D.4.

Theorem 3.6 (Pareto Stationarity)**.** *Under standard smoothness assumptions for the objective functions J_{loc} and J_{coop}, the gradient update direction $g = w^*\nabla J_{loc} + (1 - w^*)\nabla J_{coop}$ computed via the MGDA rule ensures that the shared parameters converge to a Pareto stationary point.*

Remark 3.7. Following Sener & Koltun [42], the update direction g is a descent direction for both objectives, ensuring convergence to a stationary point where no common strictly improving first-order direction exists. (Proof in Appendix D.5).

Remark 3.8. (Controlling Deviation from the Local Policy). While Theorem 3.6 establishes convergence to a Pareto stationary point, the explicit trade-off between local efficiency and corridor-level cooperation is fundamentally governed by ρ. Because the participation gate λ^* is strictly bounded by $\Pi_{[0,1]}(\Delta_{i,t}/\rho)$ (Proposition 3.2), the maximum deviation of the executed policy from the purely local optimal policy is bottlenecked by $1/\rho$. Consequently, the framework provides a theoretical safeguard: local intersection throughput degrades by at most $\mathcal{O}(1/\rho)$ to accommodate corridor-level relief, ensuring that the system does not collapse into excessive altruism under extreme congestion.

4 Training

We jointly train both layers with PPO at asynchronous time scales, motivated by the principle of *time-scale separation* in bi-level optimization. The Corridor Agent replans every T_{plan} seconds (see Algorithm 1), while Intersection Agents act every environment step (or every fixed Δt).

Theoretical Justification for Bi-level Dynamics. Analyzing the fully coupled convergence of bi-level MARL is notoriously intractable due to non-stationary transition dynamics. However, our asynchronous design mitigates this: by enforcing $T_{plan} \gg \Delta t$, the lower-level Intersection Agents perceive a quasi-static task landscape (G_t), allowing their MGDA-based updates to approach the local Pareto stationary point (Theorem 3.6). Conversely, the upper-level Corridor Agent updates at a slower frequency, effectively observing the asymptotic execution behavior of the lower layer rather than its transient fluctuations. This separation limits the chaotic feedback loops typically seen in tightly coupled hierarchical MARL. We keep *separate on-policy rollout buffers* for corridor planning and intersection control (i.e., no shared replay buffer).

The Corridor Agent is trained on episodic corridor-construction trajectories. Intersection Agents are trained on environment rollouts under the mixed policy in Eq. (4), conditioned on the current task descriptor G_t and the gate from Proposition 3.2.

Two Returns, Two Critics. To reflect multi-objective execution, we maintain two reward streams per intersection: a *local* reward r^{loc} (e.g., waiting time/queue) and a *cooperative* reward r^{coop} that adds corridor-prioritized terms weighted by p_ℓ. We learn two value functions $V_i^{loc}(o)$ and $V_i^{coop}(o, G)$, compute separate GAE advantages A^{loc} and A^{coop} (with per-batch normalization to reduce scale mismatch), and form two PPO clipped objectives:

$$L_{loc}(\theta) = L^{clip}(\theta; A^{loc}),\ L_{coop}(\theta) = L^{clip}(\theta; A^{coop}). \tag{8}$$

MGDA-Style Update for Shared Parameters. Let $g_{loc} = \nabla_\theta L_{loc}$ and $g_{coop} = \nabla_\theta L_{coop}$ be policy gradients for the two objectives on the same on-policy batch. We compute a scalar weight $w^* \in [0, 1]$ that minimizes the norm of the convex combination (two-objective MGDA):

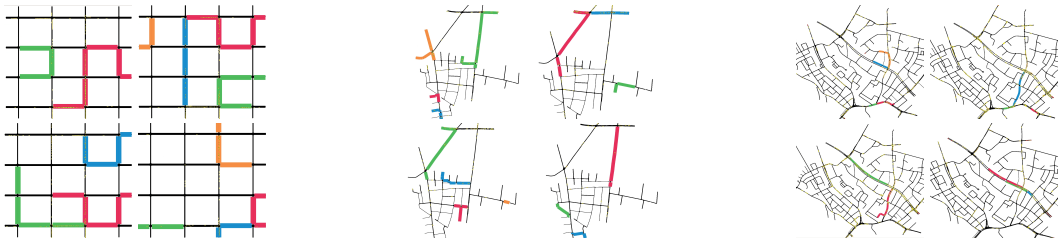
$$w^* = \text{clip}\left(\frac{(g_{coop} - g_{loc})^\top g_{coop}}{\|g_{loc} - g_{coop}\|^2 + \varepsilon},\ 0,\ 1 \right), \tag{9}$$

and update shared parameters (e.g., the shared encoder) using $g = w^* g_{loc} + (1 - w^*) g_{coop}$. To encourage specialization, we optionally update the local head with g_{loc} and the cooperative head with g_{coop} while keeping the shared representation Pareto-consistent under g.

5 Experiments

We evaluate CorridorLight on synthetic and real-world networks to assess traffic efficiency, convergence stability, and real-time coordination overhead.

Experimental Setup. We evaluate CorridorLight on SUMO (v1.21.0) [29] using one synthetic grid (*Arterial4x4*) and three real-world networks [2] (*Cologne8*, *Ingolstadt7*, *Ingolstadt21*). We compare our method against classical controls (FTC, MaxPressure [46], Gap-based Control [15]) and state-of-the-art MARL baselines, including MPLight [7], CoLight [53], MetaGAT [30], CoSLight [40], and MacLight [25]. We report **average trip time** (lower is better) and **throughput** (higher is better), which capture system-level travel efficiency and completion. Detailed descriptions of environments, baselines, hyperparameters, and the training protocol are provided in Appendix C.



(a) Arterial4x4 (b) Cologne8 (c) Ingolstadt21

Figure 4: Corridor snapshots illustrating diverse corridor structures generated by the trained corridor agent under varying real-time traffic flows. Yellow points indicate vehicles. Due to their small scale, they are more visible when the figure is viewed digitally with zoom-in.

Main Results. Table 1 summarizes performance across networks. CorridorLight achieves the best performance across all four benchmarks, obtaining the lowest average trip time under traffic demand at the preset flow limit. We report CorridorLight's best-performing configurations with $\rho = 0.5$ for *Arterial4x4* and $\rho = 1.0$ for all other benchmarks. Sensitivity to ρ is analyzed in Appendix C.9. The improvement is especially notable on the large and irregular *Ingolstadt21* network, where CorridorLight reduces average trip time from 290.73s to 248.24s, a 14.6% decrease over the best baseline, while also achieving the highest throughput. Fig. 4 provides representative corridor snapshots from three datasets, illustrating diverse corridor structures and congestion patterns under different traffic conditions. A live corridor demonstration is provided in the Git repository. The diversity of these snapshots suggests that CorridorLight can adapt its corridor-level coordination to both regular arterial layouts and irregular urban networks with long-range traffic dependencies. Consistent gains

Table 1: Results are reported as mean ± standard deviation. Best results are shown in bold, and second-best results are underlined.

	Avg. Trip Time (seconds) ↓				Avg. Throughput (vehicles / episode) ↑			
Methods	*Arterial4x4*	*Cologne8*	*Ingolstadt7*	*Ingolstadt21*	*Arterial4x4*	*Cologne8*	*Ingolstadt7*	*Ingolstadt21*
FTC	828.38±8.17	124.40±1.99	142.20±5.28	319.41±24.48	1234.30±6.50	2012.37±2.95	2616.42±22.47	3719.21±6.75
Gap	583.25±17.51	148.84±4.08	101.42±3.29	302.14±25.10	1721.10±41.06	1985.9±8.33	**2824.6±10.61**	3718.7±196.87
MaxPress.	497.15±6.91	97.29±1.54	146.83±1.77	375.25±2.40	1756.35±15.64	2012.83±1.07	2595.70±8.41	3481.21±14.38
MPLight	541.29±45.24	98.44±0.62	116.43±11.03	319.28±10.48	1743.64±37.46	2012.19±1.63	2746.46±36.88	3708.84±38.92
IPPO	401.15±12.08	90.87±0.40	104.14±6.25	295.07±13.78	1987.22±20.29	2014.58±1.75	2719.70±16.45	3759.30±45.79
MAPPO	565.67±44.80	97.68±2.03	113.59±14.54	453.61±29.66	1792.81±73.25	2012.66±0.94	2637.16±7.79	3453.18±74.25
CoLight	409.93±0.00	89.72±0.32	124.37±13.35	307.06±28.34	1971.60±126.06	2014.70±1.36	2668.47±9.52	3733.30±44.53
MetaGAT	374.80±0.87	90.74±0.76	98.21±5.47	290.73±0.45	2064.85±12.40	2013.85±0.55	2721.60±21.37	3761.18±40.90
CoSLight	364.21±4.78	90.46±0.55	117.61±5.28	309.64±4.63	2078.95±10.22	2014.40±1.13	2697.34±15.97	3727.65±72.41
MacLight	386.81±16.71	92.57±0.68	89.14±1.15	301.14±11.08	2051.94±20.94	2014.60±0.74	2813.80±20.26	3702.50±89.65
Ours	**344.22±12.79**	**86.77±0.76**	**78.13±1.98**	**248.24±4.61**	**2129.70±19.78**	**2016.43±1.22**	2690.62±3.54	**3856.64±39.77**

Table 2: Component ablation results across three traffic networks (mean ± standard deviation over three random seeds). TT denotes average travel time in seconds, and TP denotes average throughput.

Setting	arterial4x4		Cologne8		Ingolstadt21	
	TT ↓	TP ↑	TT ↓	TP ↑	TT ↓	TP ↑
Complete	344.22±12.79	2129.70±19.78	86.77±0.76	2016.43±1.22	248.24±4.61	3856.64±39.77
Greedy	355.91±13.43	2123.67±51.01	94.34±0.79	2014.33±2.52	286.95±14.06	3244.33±296.36
Disjoint-Relax	354.59±11.45	2144.00±31.18	88.51±0.30	2017.00±0.00	283.43±8.19	3440.33±196.18
FixWeight $(0.75, 0.25)$	353.75±15.94	2142.33±16.50	87.88±0.35	2016.67±0.58	275.90±24.49	3454.67±241.55
FixWeight $(0.50, 0.50)$	374.14±4.62	2135.33±15.18	88.90±0.78	2016.33±1.15	257.40±10.08	3609.00±145.91
FixWeight $(0.25, 0.75)$	363.51±6.49	2116.33±23.01	90.56±1.16	2016.00±1.00	269.88±20.44	3664.00±63.44

on *Ingolstadt7*, *Cologne8*, and *Arterial4x4* further support the robustness of CorridorLight across diverse traffic scenarios. In addition, we provide model convergence results and runtime analysis in the Appendix C, showing that CorridorLight trains stably and that the additional computational overhead does not affect real-time deployment.

5.1 Ablation Study

We ablate three key components: **Greedy** replaces the learned seed-and-grow planner with pressure-based greedy selection; **Disjoint-Relax** allows junctions to belong to multiple corridors; and **FixWeight** replaces MGDA with fixed-weight scalarization using $(0.75, 0.25)$, $(0.50, 0.50)$, or $(0.25, 0.75)$. All variants use the same experimental settings and are trained for 200 epochs with three random seeds. Detailed ablation settings and additional analysis are provided in Appendix C.9.

Table 2 shows that the complete model achieves the lowest mean travel time on all three networks. Greedy and Disjoint-Relax degrade travel time, particularly on `Ingolstadt21`, supporting the contributions of learned corridor planning and junction-disjoint assignments. Fixed-weight scalarization also yields higher travel times, with the preferred weights varying across networks and metrics, supporting MGDA's adaptive objective balancing. Although some variants slightly improve throughput on the smaller networks, these gains come with longer travel times.

6 Conclusion

We presented CorridorLight, a bi-level cooperation-as-task-negotiation framework that separates corridor task formation from local multi-objective execution. The corridor planner constructs structured tasks and broadcasts lightweight, intersection-local guidance, while each intersection negotiates its participation via a closed-form counterfactual-gain gate without expanding decentralized control interfaces. By localizing coordination and preserving per-intersection action spaces, CorridorLight reduces decision complexity while retaining real-time responsiveness. Experiments on synthetic and real-world networks show substantial gains over strong cooperative MARL baselines.

Acknowledgments This work was supported by the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0011/2025/RIC, 001/2024/SKL for SKL-IOTSC). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

References

[1] Lucas Nunes Alegre, Ana L. C. Bazzan, and Bruno C. da Silva. Quantifying the impact of non-stationarity in reinforcement learning-based traffic signal control. *PeerJ Comput. Sci.*, 7:e575, 2021.

[2] James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[3] Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[4] Nicolo' Brandizzi. Toward more human-like AI communication: A review of emergent communication research. *IEEE Access*, 11:142317–142340, 2023.

[5] Bogdan Butyrin, Artemy Rubtsov, Alexey Naumov, Vladimir V Ulyanov, and Sergey Samsonov. Gaussian approximation for two-timescale linear stochastic approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36627–36635, 2026.

[6] Jiajun Chai, Yuqian Fu, Dongbin Zhao, and Yuanheng Zhu. Aligning credit for multi-agent cooperation via model-based counterfactual imagination. pages 281–289. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024.

[7] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, et al. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *AAAI*, 2020.

[8] Lien-Wu Chen and Chia-Chen Chang. Cooperative traffic control with green wave coordination for multiple intersections based on the internet of vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1321–1335, 2016.

[9] Ming Cheng, Hao Chen, Zhiqing Li, Jia Wang, and Senzhang Wang. Role-aware multi-agent reinforcement learning for coordinated emergency traffic control. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[10] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE transactions on intelligent transportation systems*, 21(3):1086–1095, 2019.

[11] Gal Dalal, Gugan Thoppe, Balázs Szörényi, and Shie Mannor. Finite sample analysis of two-timescale stochastic approximation with applications to reinforcement learning. In *Conference On Learning Theory*, pages 1199–1233. PMLR, 2018.

[12] Christian Schröder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *CoRR*, abs/2011.09533, 2020.

[13] Thomas Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *JAIR*, 2000.

[14] Xinqi Du, Ziyue Li, Cheng Long, Yongheng Xing, Philip S. Yu, and Hechang Chen. Felight: Fairness-aware traffic signal control via sample-efficient reinforcement learning (extended abstract). In *ICDE*, pages 4726–4727. IEEE, 2025.

[15] Peter G Furth, Burak Cesme, and Theo HJ Muller. Lost time and cycle length for actuated traffic signal. *Transportation research record*, 2128(1):152–160, 2009.

[16] Harsh Goel, Yifeng Zhang, Mehul Damani, and Guillaume Sartoretti. Sociallight: Distributed cooperation learning towards network-wide traffic signal control. In *AAMAS*, pages 1551–1559. ACM, 2023.

[17] Everardo Gonzalez. Influence based reward shaping in multiagent systems. pages 2935–2937. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

[18] Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. In *NeurIPS*, 2022.

[19] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. Cr Royle. The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4), 1982.

[21] Haoyuan Jiang, Ziyue Li, Zhishuai Li, Lei Bai, Hangyu Mao, Wolfgang Ketter, and Rui Zhao. A general scenario-agnostic reinforcement learning for traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 25(9):11330–11344, 2024.

[22] Haoyuan Jiang, Ziyue Li, Hua Wei, Xuantang Xiong, Jingqing Ruan, Jiaming Lu, Hangyu Mao, and Rui Zhao. X-light: Cross-city traffic signal control using transformer on transformer as meta multi-agent reinforcement learner. In *IJCAI*, pages 94–102. ijcai.org, 2024.

[23] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[24] Siqi Lai, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. Llmlight: Large language models as traffic signal control agents. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 2335–2346, 2025.

[25] Sunbowen Lee, Hongqin Lyu, Yicheng Gong, Yingying Sun, and Chao Deng. Maclight: Multi-scene aggregation convolutional learning for traffic signal control. pages 1263–1271. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

[26] Mengxian Li, Qi Wang, and Yongjun Xu. GTDE: grouped training with decentralized execution for multi-agent actor-critic. In *AAAI*, pages 18368–18376. AAAI Press, 2025.

[27] Sheng Li, Jayesh K. Gupta, Peter Morales, Ross Allen, and Mykel J. Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. AAMAS '21, page 764–772. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[28] Xinran Li, Xiaolu Wang, Chenjia Bai, and Jun Zhang. Exponential topology-enabled scalable communication in multi-agent reinforcement learning. In *ICLR*. OpenReview.net, 2025.

[29] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018.

[30] Yican Lou, Jia Wu, and Yunchuan Ran. Meta-reinforcement learning for multiple traffic signals control. In Mohammad Al Hasan and Li Xiong, editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 4264–4268. ACM, 2022.

[31] 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.

[32] Peter Lowrie. *SCATS: Sydney Co-ordinated Adaptive Traffic System: A Traffic Responsive Method of Controlling Urban Traffic*. Roads and Traffic Authority of New South Wales, Sydney, NSW, Australia, 1990.

[33] Jiaming Lu, Jingqing Ruan, Haoyuan Jiang, Ziyue Li, Hangyu Mao, and Rui Zhao. Dualight: Enhancing traffic signal control by leveraging scenario-specific and scenario-shared knowledge. In *AAMAS*, pages 1283–1291. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024.

[34] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Credit assignment for collective multiagent RL with global rewards. In *NeurIPS*, pages 8113–8124, 2018.

[35] Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.*, 53(11):13677–13722, 2023.

[36] Afshin Oroojlooy, Mohammadreza Nazari, Davood Hajinezhad, and Jorge Silva. Attendlight: Universal attention-based reinforcement learning model for traffic signal control. *Advances in Neural Information Processing Systems*, 33:4079–4090, 2020.

[37] L.A. Prashanth and Shalabh Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421, 2011.

[38] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *NeurIPS*, 2020.

[39] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4292–4301. PMLR, 2018.

[40] Jingqing Ruan, Ziyue Li, Hua Wei, Haoyuan Jiang, Jiaming Lu, Xuantang Xiong, Hangyu Mao, and Rui Zhao. Coslight: Co-optimizing collaborator selection and decision-making to enhance traffic signal control. In *KDD*, pages 2500–2511. ACM, 2024.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[42] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[43] Amutheezan Sivagnanam, Ava Pettet, Hunter Lee, Ayan Mukhopadhyay, Abhishek Dubey, and Aron Laszka. Multi-agent reinforcement learning with hierarchical coordination for emergency responder stationing. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[44] Haoran Su, Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Emvlight: A decentralized reinforcement learning framework for efficient passage of emergency vehicles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4593–4601, 2022.

[45] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087, 2018.

[46] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[47] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3540–3549, 2017.

[48] Peter Wagner, Nathan Gartner, Robert Oertel, and Ting Lu. Webster's delay formula–revisited. 2014.

[49] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[50] Maonan Wang, Yirong Chen, Yuheng Kan, Chengcheng Xu, Michael D. Lepech, Man-On Pun, and Xi Xiong. Traffic signal cycle control with centralized critic and decentralized actors under varying intervention frequencies. *IEEE Trans. Intell. Transp. Syst.*, 25(12):20085–20104, 2024.

[51] F. V. Webster. Traffic signal settings. Technical Report 39, Road Research Laboratory, London, UK, 1958.

[52] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash V. Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *KDD*, pages 1290–1298. ACM, 2019.

[53] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. pages 1913–1922. ACM, 2019.

[54] Hua Wei, Guanjie Zheng, Vikash V. Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *SIGKDD Explor.*, 22(2):12–18, 2020.

[55] Annie Wong, Thomas Bäck, Anna V. Kononova, and Aske Plaat. Deep multiagent reinforcement learning: challenges and directions. *Artif. Intell. Rev.*, 56(6):5023–5056, 2023.

[56] Hang Xiao, Huale Li, Shuhan Qi, Jiajia Zhang, and Dingzhong Cai. Fglight: Learning neighbor-level information for traffic signal control. pages 2181–2189. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

[57] Kok-Lim Alvin Yau, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Comput. Surv.*, 50(3):34:1–34:38, 2017.

[58] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24611–24624, 2022.

[59] Zirui Yuan, Siqi Lai, and Hao Liu. Collmlight: Cooperative large language model agents for network-wide traffic signal control, 2025.

[60] Yifan Zang, Jinmin He, Kai Li, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. Automatic grouping for efficient cooperative multi-agent reinforcement learning. In *NeurIPS*, 2023.

[61] Haipeng Zhang, Zhiwen Wang, and Na Li. Matlight: Traffic signal coordinated control algorithm based on heterogeneous-agent mirror learning with transformer. In *AAMAS*, pages 2582–2584. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024.

[62] Yifeng Zhang, Harsh Goel, Peizhuo Li, Mehul Damani, Sandeep Chinchali, and Guillaume Sartoretti. Coordlight: Learning decentralized coordination for network-wide traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 26(6):8034–8049, 2025.

[63] Haiyan Zhao, Chengcheng Dong, Jian Cao, and Qingkui Chen. A survey on deep reinforcement learning approaches for traffic signal control. *Eng. Appl. Artif. Intell.*, 133:108100, 2024.

[64] Bin Zhou, Qishen Zhou, Simon Hu, Dongfang Ma, Sheng Jin, and Der-Horng Lee. Cooperative traffic signal control using a distributed agent-based deep reinforcement learning with incentive communication. *IEEE Trans. Intell. Transp. Syst.*, 25(8):10147–10160, 2024.

[65] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents Multi Agent Syst.*, 38(1):4, 2024.

[66] Ming Zhu, Xiao-Yang Liu, Sem Borst, and Anwar Walid. Deep reinforcement learning for traffic light control in intelligent transportation systems. *arXiv preprint arXiv:2302.03669*, 2023.

Appendix

A Notation and Algorithm

A.1 Summary of Notation

For clarity and ease of reference, we provide a comprehensive summary of the mathematical notations used throughout the main text in Table 3. The symbols are organized by their role in the hierarchical framework, covering network representation, global task formation, and local execution.

A.2 Algorithm

Algorithm 1 summarizes the proposed seed–grow corridor planning procedure. The input is a directed movement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ represents a traffic movement and each edge in $\mathcal{E}$ denotes a feasible downstream transition. Each movement is associated with an embedding h_v, which encodes its local traffic state and structural context. The algorithm aims to construct at most M corridors, where each corridor is an ordered sequence of movements with length bounded by $L_{\max}$. At each iteration, the candidate seed set $\mathcal{V}_{seed}$ is formed by excluding movements whose junctions have already been assigned to previously constructed corridors, as recorded in $\mathcal{J}_{used}$. A scoring function $s(\cdot)$ ranks these candidates, and the top-k seeds are selected into $\mathcal{S}$. Starting from each seed $v_0 \in \mathcal{S}$, the algorithm grows a path P by repeatedly selecting an admissible downstream movement from the masked action set $\mathcal{A}_{mask}$. This mask contains outgoing neighbors of the current tail node v_{tail} that have not appeared in P and whose junctions are not in $\mathcal{J}_{used}$, together with a special `STOP` action. The grow policy π_{grow} then samples an action a conditioned on the current path P, the movement embeddings $\{h_u\}$, and the feasible action mask $\mathcal{A}_{mask}$. If $a = \texttt{STOP}$, the current corridor terminates immediately; otherwise, the selected movement is appended to P. Thus, each corridor stops growing either when the policy selects `STOP` or when the maximum length $L_{\max}$ is reached. The overall planning process terminates when $|\mathcal{C}| = M$, meaning that the maximum number of corridors has been constructed. After each corridor is finalized, it is added to the corridor set $\mathcal{C}$, and its covered junctions are inserted into $\mathcal{J}_{used}$ to avoid overlap among corridors. Finally, the task descriptor G_t is constructed from the selected corridors using per-intersection lane masks and positional indicators p_ℓ, which encode the ordered role of each movement within its corridor.

Algorithm 1 Seed–Grow Corridor Planning

Require: Movement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; embeddings $\{h_v\}$; max corridors M; top-k k; max length $L_{\max}$
Ensure: Corridor set $\mathcal{C}$; task descriptor G_t

$\mathcal{C} \leftarrow \emptyset$; $\mathcal{J}_{used} \leftarrow \emptyset$
while $|\mathcal{C}| < M$ **do**
  $\mathcal{V}_{seed} \leftarrow \{v \in \mathcal{V} \mid \texttt{Junction}(v) \notin \mathcal{J}_{used}\}$
  $\mathcal{S} \leftarrow \texttt{TopK}(\mathcal{V}_{seed}, s(\cdot), k)$
  $P \leftarrow \emptyset$
  for each seed $v_0 \in \mathcal{S}$ **do**
    $P \leftarrow [v_0]$
    for $\ell = 1$ to $L_{\max}$ **do**
      $\mathcal{A}_{mask} \leftarrow \{u \in \mathcal{N}_{out}(v_{\text{tail}}) \mid u \notin P, \texttt{Junction}(u) \notin \mathcal{J}_{used}\} \cup \{\texttt{STOP}\}$
      $a \sim \pi_{\text{grow}}(\cdot \mid P, \{h_u\}, \mathcal{A}_{mask})$
      if $a = \texttt{STOP}$ **then**
        break
      end if
      Append a to P
    end for
  end for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{P\}$; $\mathcal{J}_{used} \leftarrow \mathcal{J}_{used} \cup \texttt{Junctions}(P)$
end while
Construct G_t from per-intersection lane masks and p_ℓ
return $\mathcal{C}, G_t$

Table 3: Comprehensive notation table used throughout the paper.

Symbol	Meaning
Movement Graph & Encoding	
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Movement graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$
$h_v^{(0)},\ h_v^{(K)}$	Initial and final embeddings of movement v in a K-layer GNN
$x_v \in \mathbb{R}^d$	Movement traffic feature vector; d is the feature dimension
$J(v),\ \mathcal{N}^{\text{out}}(v)$	Associated junction and outgoing-neighbor set of movement v
Corridor Agent (Global Task Formation)	
$\tau,\ q$	Planning-cycle and corridor-construction step indices
$T_{\text{plan}},\ \Delta t$	Corridor replanning and intersection-control intervals
$M,\ k,\ L_{\max}$	Maximum corridors, top-k seed count, and node limit per corridor (including the seed)
$P_q = (v_0, \ldots, v_q)$	Ordered corridor prefix at construction step q
J_{used}	Junctions assigned to previously accepted corridors
ξ_τ	Traffic and feasibility context fixed within a construction episode
$a_q,\ \mathcal{A}_{\xi_\tau}(P_q)$	Growth action and feasible action set (downstream successors or `STOP`)
π_{grow}	Masked policy for selecting corridor-growth actions
$V^*_{\text{seed}},\ Q^*_{\text{grow}}$	Optimal seed value and growth action value under fixed context ξ_τ
$r^c_{\tau,q}$	Execution credit for extension $v_q \to v_{q+1}$ of corridor c
$R_\tau(c),\ \bar{R}_{\xi_\tau}(c)$	Corridor execution payoff and its conditional expectation
G_t	Task descriptor over local lanes $\mathcal{L}_i$: corridor masks and preference weights
p_ℓ	Preference weight for lane ℓ in the active corridor task
Intersection Agent (Local Execution)	
$i,\ t$	Intersection-agent index and control step
o_t^i	Local observation of intersection agent i at step t
$\mathcal{A}_i$	Discrete signal-phase action set shared by both policy heads
$\pi_i^{\text{loc}},\ \pi_i^{\text{coop}}$	Local policy and corridor-conditioned cooperative policy
π_i	Executed policy obtained by gated logit interpolation and softmax
$z_i^{\text{loc}},\ z_i^{\text{coop}}$	Logits of the local and cooperative policy heads
$r^{\text{loc}},\ r^{\text{coop}}$	Local and corridor-prioritized cooperative reward streams
Counterfactual-Gain Gating & Optimization	
$\lambda_{i,t},\ \lambda^*_{i,t}$	Participation gate in $[0, 1]$ and its closed-form optimum
$A^{\text{loc}}_{i,t},\ A^{\text{coop}}_{i,t}$	Local/cooperative advantages; hats denote per-objective batch standardization
$\Delta_{i,t}$	Standardized advantage difference: $\hat{A}^{\text{coop}}_{i,t} - \hat{A}^{\text{loc}}_{i,t}$
ρ	Positive friction coefficient controlling participation
$V_i^{\text{loc}},\ V_i^{\text{coop}}$	Critics for local and cooperative returns
θ	Shared encoder parameters updated by MGDA
$L_{\text{loc}},\ L_{\text{coop}}$	Local and cooperative PPO clipped objectives
$g_{\text{loc}},\ g_{\text{coop}},\ g$	Local, cooperative, and MGDA-combined shared-parameter gradients
w^*	MGDA mixing weight; $\varepsilon > 0$ stabilizes its computation
$\epsilon,\ D_{\text{TV}},\ L_z$	Advantage-difference error, TV distance, and policy-shift bound constant

A.3 Corridor Learning and Broadcasting Details

Construction-process interpretation. Each corridor is constructed through a finite-horizon sequence of growth decisions conditioned on the planning context ξ_τ. The context remains fixed during construction, prefix transitions are deterministic, and construction decisions do not advance the traffic simulator. Traffic feedback is collected after the accepted corridors are broadcast to and executed by the Intersection Agents.

With the lower-level policy and the protocol for completing and executing the remaining corridor tasks fixed, the construction process admits a terminal-reward interpretation with zero intermediate rewards and construction discount $\gamma_{\mathrm{g}} = 1$. The terminal payoff is the corridor score collected during the subsequent execution interval.

This interpretation does not imply that the implemented edge-local policy weights are return-to-go advantages. The feedback assigned to an extension can depend on later construction decisions and on other corridor tasks executed during the same planning cycle.

Observation availability and extension credit. A corridor follows the physical traffic network and may include intersections that are not controlled by an Intersection Agent. Lane observations at such intersections may be unavailable. Consequently, an accepted corridor extension does not necessarily have measurable execution feedback.

Let $r_{\ell,t}^{\mathrm{lane}}$ denote the reward of lane ℓ at environment step t, using the same reward definition, sign, and scaling as the local agents. Let $\mathcal{L}(v)$ contain the lanes represented by movement v. Let $\mathcal{L}_t^{\mathrm{obs}}$ contain the lanes for which the observations required to compute the lane reward are available at step t.

Consider an accepted corridor $c = (v_0, \ldots, v_m)$ with m selected extensions. Extension $v_q \to v_{q+1}$ receives feedback from the available lanes represented by its selected successor v_{q+1}. Define

$$\begin{aligned} \mathcal{L}_t^{\mathrm{obs}}(v) &= \mathcal{L}(v) \cap \mathcal{L}_t^{\mathrm{obs}}, \\ \mathcal{T}_{\tau,q}^{c} &= \left\{ t \in \mathcal{I}_\tau : \mathcal{L}_t^{\mathrm{obs}}(v_{q+1}) \neq \emptyset \right\}, \\ \chi_{\tau,q}^{c} &= \mathbb{I}\left[|\mathcal{T}_{\tau,q}^{c}| > 0 \right]. \end{aligned} \tag{10}$$

Here, $\chi_{\tau,q}^{c}$ is a binary feedback-validity indicator. It equals one if at least one associated lane has available reward observations at some step within the planning-cycle execution interval $\mathcal{I}_\tau$, and zero if no such observation is available throughout that interval. It indicates observation availability, not whether the successor intersection is agent-controlled. An extension through a non-agent-controlled intersection can therefore have valid credit whenever the required observations are available.

The extension credit is

$$r_{\tau,q}^{c} = \begin{cases} \dfrac{1}{|\mathcal{T}_{\tau,q}^{c}|} \displaystyle\sum_{t \in \mathcal{T}_{\tau,q}^{c}} \sum_{\ell \in \mathcal{L}_t^{\mathrm{obs}}(v_{q+1})} r_{\ell,t}^{\mathrm{lane}}, & \chi_{\tau,q}^{c} = 1, \\ 0, & \chi_{\tau,q}^{c} = 0, \end{cases} \qquad q = 0, \ldots, m-1. \tag{11}$$

The credit first sums the available successor-lane rewards at each environment step and then averages these sums over steps with valid observations. The resulting credit is successor-specific, rather than a junction-wide reward. When $\chi_{\tau,q}^{c} = 0$, the stored zero credit is a bookkeeping placeholder for unavailable feedback, not an observed zero reward. In contrast, an observed reward equal to zero has $\chi_{\tau,q}^{c} = 1$ and remains valid feedback.

Fallback to local control. When intersection i is not assigned an active corridor task (i.e., its corridor lane mask is identically zero), we explicitly set $\lambda_{i,t} = 0$. Consequently, the executed policy reduces to $\pi_i(a \mid o_t^i, G_t) = \pi_i^{loc}(a \mid o_t^i)$. This recovers IPPO-style decentralized execution, where the intersection selects actions using only its local observation and the original signal-phase action set, without corridor-induced policy mixing.

B Extended Related Work

We provide a detailed comparison between our proposed framework and existing literature in three key dimensions.

B.1 Hierarchical MARL: Dictation vs. Negotiation

Traditional hierarchical approaches typically employ a strict "Manager-Worker" hierarchy[13]. For instance, Feudal Networks [47] and their traffic variants often dictate specific sub-goals (e.g., "reduce queue by 10%") that the worker *must* attempt to satisfy. This creates a "dictator problem" where the manager's global view may conflict with local safety constraints, leading to unsafe signal phases. **Our Approach:** We adopt a *negotiation-based* paradigm. The Corridor Agent does not enforce hard constraints but proposes *negotiable tasks* via preference weights (p_l). The Intersection Agent retains the ultimate decision-making power through the counterfactual-gain gating mechanism. This aligns with the concept of "Feudal Multi-Agent RL with Optional Subgoals", but we uniquely ground the option mechanism in a counterfactual gain rather than a learned latent variable.

B.2 Gating Mechanisms: Black-box vs.Counterfactual-Gain Interpretation

Dynamic weight adjustment is a core challenge in multi-objective RL. Standard Mixture-of-Experts (MoE) gates or scalarization networks (e.g., in PM-MORL) are typically implemented as a Softmax layer over a neural network's output, trained end-to-end to minimize prediction error. These are often "black-box" routers lacking physical meaning. **Our Approach:** Our gating mechanism is derived from a closed-form solution to a *counterfactual optimization problem* (Proposition 3.2). It represents an explicit economic trade-off between marginal global gain and local friction. Unlike heuristic thresholding, our gate λ^* is provably Lipschitz continuous with respect to the advantage estimates, ensuring stability (Theorem 3.4).

B.3 Comparison with Graph-based and Communication-Heavy Traffic Control

State-of-the-art methods like CoLight [53], MPLight [7], Attendlight [36], and recent transformer-based traffic agents rely on Graph Neural Networks (GNNs) or Attention mechanisms that require continuous, high-frequency communication ($O(E)$ per step) to share hidden states. This creates a single point of failure and high bandwidth demands. **Our Approach:** By decoupling global coordination from local control, we reduce the communication frequency to once every T_{plan} seconds (e.g., 60s). This makes the system significantly more robust to communication latency and packet loss, a critical advantage for real-world IoT deployment.

B.4 Emerging Trends: Large Language Models (LLMs) for Traffic

Recent works have explored using LLMs (e.g., CoLLMLight [59] and LLMLight [24]) as reasoning agents for traffic control due to their strong generalization capabilities. While promising, these methods suffer from (1) high inference latency (seconds per step), making them unsuitable for real-time control, and (2) hallucination risks. **Our Approach:** Our method remains a specialized, lightweight RL framework. However, the "Corridor Task" abstraction we propose is semantically meaningful and could potentially serve as an interface for LLM-based high-level planners in future work, bridging the gap between symbolic reasoning and numeric control.

C Experimental Details

C.1 Environment and Datasets

We conduct experiments using the SUMO traffic simulator (v1.21.0) [29]. To ensure a standardized evaluation and to quantify the impact of non-stationarity in RL-based traffic signal control (TSC), we implement our experiments using the SUMO-RL interface [1]. The evaluation covers one synthetic grid network and three real-world networks[2] with diverse topologies and demand patterns. For the real-world networks, the suffix in each benchmark name denotes the number of controllable signalized intersections rather than the total number of intersections in the extracted road network. The latter is

substantially larger because the networks also include uncontrolled intersections governed by priority rules. Since our corridor agent explicitly considers both controlled and uncontrolled intersections, we report both quantities below:

- ***Arterial4x4* (synthetic):** A 4×4 arterial/avenue-style synthetic network with 16 controllable signalized intersections.
- ***Cologne8* (real-world region):** A regional downtown sub-network from Cologne with 8 controllable signalized intersections and 173 total intersections.
- ***Ingolstadt7* (real-world corridor):** A corridor sub-network from Ingolstadt with 7 controllable signalized intersections and 66 total intersections.
- ***Ingolstadt21* (real-world region):** A larger downtown sub-network from Ingolstadt with 21 controllable signalized intersections and 517 total intersections.

C.2 Baselines

We compare CorridorLight against classical control approaches and representative MARL baselines that are compatible with the SUMO environment. All methods are evaluated under the same simulator configuration and timing settings. The baselines include:

- **FTC**: Fixed-time controller with pre-timed phase order and green splits.
- **Gap-based Control [15]**: Actuated controller that extends green based on vehicle detections and terminates on gap-out or max-green. We reproduce this baseline in our experimental framework by implementing the corresponding detection-based green-extension mechanism and gap-out/max-green termination criteria.
- **MaxPressure** [46]: Classical feedback policy that selects the phase maximizing queue-imbalance pressure between incoming and outgoing movements.
- **MPLight** [7]: Learning-based controller that incorporates the max-pressure principle as an inductive bias to scale to many intersections.
- **IPPO** [12]: Decentralized PPO agents (often with parameter sharing) trained from local observations and rewards without explicit coordination.
- **MAPPO** [58]: CTDE actor–critic with a centralized critic conditioned on joint information during training while keeping execution decentralized.
- **CoLight** [53]: Attention-based neighbor communication that adaptively aggregates neighboring intersections' features for coordination.
- **MetaGAT** [30]: Graph-attention communication baseline emphasizing generalizable neighbor aggregation across traffic patterns.
- **CoSLight** [40]: Jointly learns collaborator selection and signal control, enabling dynamic communication beyond fixed adjacent neighborhoods.
- **MacLight** [25]: Local–global communication that broadcasts a compact global context to complement local observations.

All methods are evaluated under the same simulator configuration and timing settings.

C.3 Observation and Action Space

Consistent with standard SUMO-RL settings, each Intersection Agent observes local per-junction features, including the current signal phase, per-lane queue lengths, and vehicle counts. The agent takes a discrete phase action at each step.

Uniquely, CorridorLight additionally conditions its cooperative policy head on a compact task descriptor G_t, which consists of per-lane corridor masks and preference weights. Crucially, this is achieved without expanding the physical action interface; the agent continues to output logits over the same discrete signal-phase set $\mathcal{A}_i$ as the baselines.

C.4 Hardware Setting

All experiments were conducted on a server equipped with an Intel(R) Xeon(R) Gold 6430 CPU, an NVIDIA L20 GPU, and 64 GB of RAM.

C.5 Reward Functions:

- **queue**: the number of *halting* vehicles on each lane.
- **waiting-time**: the accumulated waiting time on each lane.
- **diff-waiting-time**: the increment of the accumulated waiting time on each lane between two consecutive steps.

C.6 Running Time and Scalability Analysis

Each episode simulates 3600 seconds of real-world traffic operation. Table 4 compares the average episode runtime of CorridorLight and IPPO. Although CorridorLight is consistently slower than IPPO, it completes an episode on Ingolstadt-21, the largest original benchmark with 21 controlled intersections, in approximately 85 seconds. The additional overhead arises from corridor-level coordination and graph-based computation beyond the standard policy forward pass. Profiling on Ingolstadt-21 shows that corridor updates dominate runtime (57.63%), followed by environment stepping (22.59%) and action selection (17.51%).

Table 4: Average episode runtime and runtime breakdown on Ingolstadt-21.

Average episode runtime			**Breakdown on Ingolstadt-21**		
Map	Ours (s)	IPPO (s)	Component	Time (s)	Share (%)
Arterial4x4	53.78	18.83	Corridor update	49.17	57.63
Cologne8	38.33	13.51	Action selection	14.94	17.51
Ingolstadt21	85.32	51.20	Environment stepping	19.27	22.59
Ingolstadt7	44.70	12.90	Post-step and logging	1.94	2.27

To evaluate scalability beyond the original benchmarks, we conduct additional profiling experiments on synthetic 4×4, 8×8, and 16×16 grids based on Arterial4x4, containing 16, 64, and 256 controlled intersections, respectively. Traffic demand on the two larger networks is increased by factors of 4 and 8 relative to the 4×4 network.

As shown in Table 5, corridor update time increases from 18.3 to 52.0 and 172.1 seconds. Each fourfold increase in intersection count therefore incurs approximately $2.84\times$ and $3.31\times$ increases in corridor update time, indicating less-than-proportional growth over the evaluated range. Meanwhile, its share of total runtime decreases from 47.5% to 39.7% and 25.5%. On the 16×16 network, environment stepping becomes the dominant cost, accounting for 41.5% of runtime.

Table 5: Episode runtime breakdown in the synthetic-network scalability experiments. Values report seconds and percentages of total runtime.

Component	4×4 16 intersections	8×8 64 intersections	16×16 256 intersections
Corridor update	18.30 (47.5%)	52.00 (39.7%)	172.10 (25.5%)
Action selection	12.70 (33.0%)	47.30 (36.1%)	186.50 (27.6%)
Environment stepping	5.70 (14.8%)	24.60 (18.8%)	280.10 (41.5%)
Post-step processing	1.82 (4.7%)	7.23 (5.5%)	36.00 (5.3%)
Total	38.52	131.13	674.70

The largest network completes a 3600 seconds simulated episode in approximately 675 seconds, demonstrating faster-than-real-time execution with 256 controlled intersections under the evaluated

settings. These measurements support empirical scalability. Further runtime reductions can target efficient corridor construction and updates, batching, and parallelization, while simulator efficiency becomes increasingly important at larger scales.

C.7 Hyperparameters

Table 6 provides a comprehensive list of the hyperparameters used in our experiments. We follow standard PPO optimization settings to ensure a fair comparison. The configuration corresponding to the best-performing run on each map is available in our GitHub repository.

Table 6: Detailed hyperparameter configuration.

Hyperparameter	**Value**
Simulation horizon (`num_seconds`)	3600
Action step (`delta_time`)	10
Minimum green (`min_green`)	5
Reward function (`reward_fn`)	`[queue, diff-waiting-time, waiting-time]`
Corridor Agent (upper layer)	
Update interval (`corridor_delta_time`)	[10, 20, 40, 60]
Max corridors (`max_corridors`)	6
Max corridor length (`max_corridor_length`)	5
Top-k seeds (`seed_top_k`)	16
GNN encoder and grow module	
Architecture (`gnn_architecture`)	`[GAT, GCN]`
Hidden / output dim	128 / 64
#layers (`gnn_num_layers`)	3
Dropout (`gnn_dropout`)	0.1
Grow hidden dim (`grow_hidden_dim`)	128
Grow max neighbors (`grow_max_neighbors`)	4
Intersection Agent (lower layer)	
Policy hidden dim (`hidden_dim`)	128
Lane feature dim (`lane_feature_dim`)	10
Friction coefficient (`lambda_friction_coeff`, ρ)	[0.5, 1.0, 2.0]
PPO and MGDA	
Learning rate (`ppo_lr`)	3×10^{-4}
Discount (`ppo_gamma`)	0.99
GAE λ (`ppo_lam`)	0.95
Clip ratio (`ppo_clip_ratio`)	0.15
Value loss coef (`ppo_value_coef`)	0.5
Entropy coef (`ppo_entropy_coef`)	0.01
Max grad norm (`ppo_max_grad_norm`)	0.5
PPO epochs (`ppo_epochs`)	10
MGDA epsilon (`mgda_epsilon`)	10^{-8}

C.8 Training protocol

We jointly train the Corridor Agent and Intersection Agents with PPO at asynchronous time scales. The Corridor Agent replans every T_{plan} seconds (default 20s), while Intersection Agents act every environment step (10s). Although both layers interact with the same SUMO environment, we maintain separate on-policy rollout buffers for the corridor layer and the intersection layer (i.e., no shared replay buffer).

C.9 Ablation Details

C.9.1 Component Ablations

We evaluate three component ablations on `arterial4x4`, `Cologne8`, and `Ingolstadt21`:

- **Greedy.** We replace the learned seed-and-grow planner with a pressure-based greedy heuristic that selects the next intersection with the highest heuristic score.
- **Disjoint-Relax.** We relax the junction-disjoint constraint, allowing an intersection to belong to multiple corridors.
- **FixWeight.** We replace MGDA with fixed-weight scalarization of the two objectives, using weight pairs $(0.75, 0.25)$, $(0.50, 0.50)$, and $(0.25, 0.75)$.

All variants use the same training protocol, simulator settings, observation/action spaces, and evaluation procedure as the complete model. Each setting is trained for 200 epochs with three random seeds, and results are reported as mean $\pm$ standard deviation in Table 2.

Learned corridor planning. Replacing the learned planner with Greedy increases mean travel time and reduces mean throughput across all three networks. The largest degradation occurs on `Ingolstadt21`, where travel time increases from 248.24 s to 286.95 s and throughput decreases.

Junction-disjoint constraint. Disjoint-Relax increases mean travel time on all three networks. On `Ingolstadt21`, travel time rises to 283.43 s and throughput falls to 3440.33. Although relaxing the constraint slightly increases mean throughput on `arterial4x4` and `Cologne8`, these gains are accompanied by longer travel times. Thus, overlapping corridor assignments do not consistently improve performance.

Adaptive objective balancing. All fixed-weight variants yield higher mean travel times than the complete model. The preferred weights also depend on the network and metric: $(0.75, 0.25)$ achieves the lowest travel time among fixed-weight variants on `arterial4x4` and `Cologne8`, whereas $(0.50, 0.50)$ performs best on `Ingolstadt21`. On the latter network, $(0.25, 0.75)$ instead achieves the highest throughput among fixed-weight variants. This variation highlights the difficulty of selecting a single weight pair and supports MGDA's adaptive objective balancing.

C.9.2 Sensitivity to the Friction Coefficient

We vary the friction coefficient $\rho \in \{0.5, 1.0, 2.0\}$ on all three networks while retaining all components of the complete model. The coefficient controls the strength of the friction term in the objective. All variants use the same training protocol, simulator settings, observation/action spaces, and evaluation procedure.

Table 7: Ablation results on different traffic networks (mean $\pm$ standard deviation).

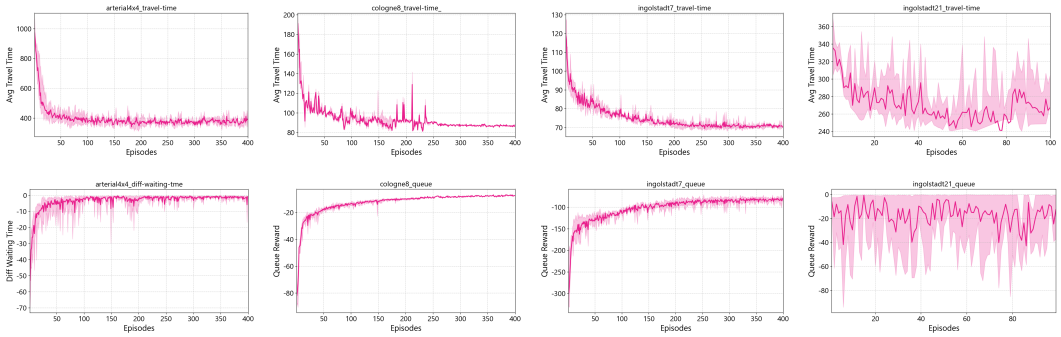
Setting	arterial4x4		Cologne8		Ingolstadt21	
	Travel Time (s)	Avg. Throughput	Travel Time (s)	Avg. Throughput	Travel Time (s)	Avg. Throughput
$\rho = 0.5$	344.22±12.79	2129.70±19.78	87.11±0.98	2016.50±1.21	253.50±6.93	3789.31±25.25
$\rho = 1.0$	365.73±29.54	2086.56±75.14	86.77±0.76	2016.43±1.22	248.24±4.61	3856.64±39.77
$\rho = 2.0$	351.86±19.36	2126.06±33.94	87.49±1.22	2015.70±1.39	260.77±7.43	3678.04±27.38

Table 7 shows that the effect of ρ is network-dependent and non-monotonic. On `arterial4x4`, $\rho = 0.5$ achieves the lowest mean travel time and highest mean throughput among the tested values. On `Cologne8`, performance varies only slightly, with $\rho = 1.0$ yielding the lowest mean travel time. On `Ingolstadt21`, $\rho = 1.0$ also achieves the lowest mean travel time, whereas $\rho = 2.0$ increases travel time and reduces throughput. These results indicate that increasing the friction coefficient is not uniformly beneficial and that its strength should be calibrated to the traffic network.

For reference, Table 8 reports corridor statistics (average corridor length and the average number of corridors selected per episode) across maps with the best setting.

Table 8: Corridor statistics across benchmarks.

Map	Avg. Corridor Length	Avg. # Corridors
Arterial4x4	3.59	0.57
Cologne8	3.68	2.81
Ingolstadt21	2.86	0.33
Ingolstadt7	3.31	0.538



(a) Arterial4x4 (b) Cologne8 (c) Ingolstadt7 (d) Ingolstadt21

Figure 5: Learning curves on different benchmarks

C.10 Learning Curve Assessment

To assess training stability and convergence, we provide learning-curve visualizations for all benchmarks (Fig. 5). Each plot reports the episode-wise performance on the corresponding evaluation metric (e.g., average travel time, queue reward, or differential waiting time) as training progresses. The solid curve shows the mean over multiple runs, while the shaded band indicates variability across runs, highlighting both sample efficiency and robustness of the learned policy. Across scenarios, the curves exhibit rapid early improvement followed by a plateau, suggesting consistent convergence behavior under different network topologies and objectives.

In addition, to examine convergence and the coordination between global and local objectives, we report per-intersection learning curves on Cologne8 in Fig. 6. Specifically, for each of the eight intersections, we visualize the convergence behavior with respect to both the global reward and the local reward. This provides a finer-grained view of how CorridorLight balances corridor-level coordination with intersection-level optimization during training.

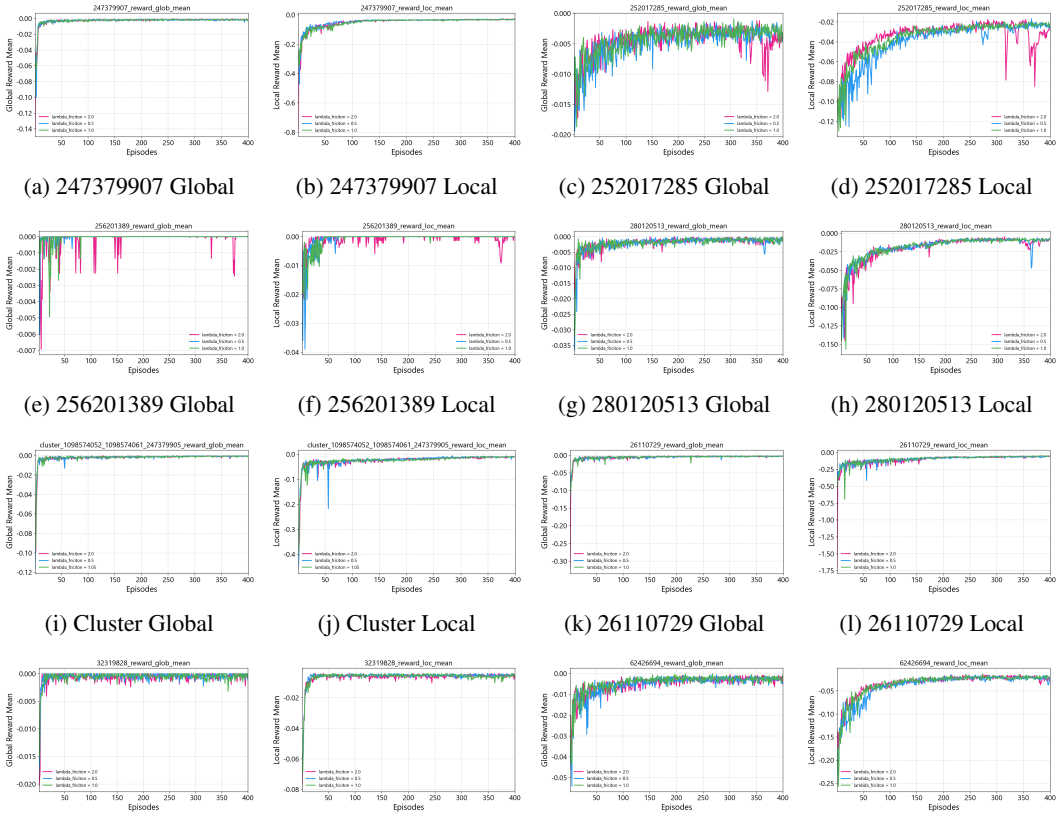
D Theoretical Analysis

D.1 Preliminaries and Proof Roadmap

Before presenting the detailed proofs, we outline the mathematical foundations used in our analysis and provide a roadmap connecting our theoretical results to the proposed framework's stability and optimality.

Mathematical Preliminaries. We rely on standard definitions from reinforcement learning, convex optimization, and measure theory.

- **Bellman Optimality Equation.** For an MDP with state space $\mathcal{S}$ and action space $\mathcal{A}$, the optimal value function $V^*(s)$ and the optimal action-value function $Q^*(s,a)$ satisfy the consistency condition:
$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a). \tag{12}$$
This fundamental relation underpins the architectural design of our Corridor Agent (Proposition 3.1).
- **Lipschitz Continuity and Projection.** A function $f : \mathcal{X} \to \mathcal{Y}$ is L-Lipschitz continuous if $\|f(x) - f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathcal{X}$. The projection operator onto a convex set $\mathcal{C}$,



(a) 247379907 Global (b) 247379907 Local (c) 252017285 Global (d) 252017285 Local

(e) 256201389 Global (f) 256201389 Local (g) 280120513 Global (h) 280120513 Local

(i) Cluster Global (j) Cluster Local (k) 26110729 Global (l) 26110729 Local

(m) 32319828 Global (n) 32319828 Local (o) 62426694 Global (p) 62426694 Local

Figure 6: Reward convergence curves of different intersections with global reward (corridor) and local reward. The number in each subcaption denotes the intersection name, and "Cluster" denotes a clustered intersection.

denoted as $\Pi_{\mathcal{C}}(x) = \arg\min_{y \in \mathcal{C}} \|x - y\|_2$, is non-expansive (i.e., 1-Lipschitz continuous):

$$\|\Pi_{\mathcal{C}}(x) - \Pi_{\mathcal{C}}(y)\|_2 \leq \|x - y\|_2. \tag{13}$$

This property is central to the stability of our Counterfactual-Gain Gating Mechanism (Proposition 3.2).

- **Total Variation Distance.** To measure the shift in the agent's policy distribution π, we use the Total Variation (TV) distance. For two discrete probability distributions P and Q on a finite space $\mathcal{A}$:

$$D_{TV}(P||Q) = \frac{1}{2} \sum_{a \in \mathcal{A}} |P(a) - Q(a)|. \tag{14}$$

We use this metric to bound the policy divergence caused by value estimation errors (Theorem 3.4).

- **Pareto Stationarity in Multi-Objective Optimization.** For a vector-valued loss $\mathcal{L}(\theta) = \{L_1(\theta), \ldots, L_m(\theta)\}$, a point θ is Pareto stationary if there exists no descent direction d that decreases all objectives simultaneously. This is satisfied when $0 \in \text{Conv}(\{\nabla L_i(\theta)\}_i)$, which justifies our use of the MGDA update rule (Theorem 3.6).

Proof Roadmap. Our theoretical analysis aims to demonstrate that the proposed bi-level framework is not only empirically effective but also theoretically grounded in three key aspects:

1. **Architectural Consistency (Proposition 3.1):** We first prove that the decomposition of the Corridor Agent into Seeding and Growing phases is mathematically consistent. By showing that the Seeding value is the optimal projection of the Growing Q-value, we justify sharing the GNN encoder between these modules.

2. **Execution Stability (Proposition 3.2 & Theorem 3.4):** We then focus on the Intersection Agent. We prove that our closed-form gating function λ^* is Lipschitz continuous (Proposition 3.2). Building on this, we establish that the final executed policy is robust: small errors in the advantage estimation result in bounded shifts in the policy distribution (Theorem 3.4), preventing the "bang-bang" control instability common in heuristic switching.
3. **Training Convergence (Theorem 3.6):** Finally, we address the optimization process. We show that the MGDA-based gradient update guarantees convergence to a Pareto stationary point, ensuring that the shared representation effectively balances local and global objectives without manual scalarization tuning.

Below, we provide the formal proofs for these statements.

D.2 Proof of Proposition 3.1 (Seeding-Growing Consistency)

Proof. The objective of the Seeding phase is to select a starting node v_0 that maximizes the expected return of the subsequent corridor generation episode:

$$v_0^* = \arg\max_{v \in \mathcal{V}} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = \{v\}\right]. \tag{15}$$

By definition of the state-value function in RL, the term inside the argmax is exactly $V^*(s_0)$ where $s_0 = \{v\}$.

According to the Bellman Optimality Equation, the value of a state is the maximum action-value available from that state:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} Q^*(s, a). \tag{16}$$

Therefore, the Seeding task is equivalent to finding the state $s = \{v\}$ that maximizes the value function $V^*(s)$, which is intrinsically derived from the Growing agent's Q-function.

□

D.3 Proof of Proposition 3.2 (Lipschitz Continuity of the Gate)

Proof. The objective function $J(\lambda) = \lambda\Delta_{i,t} - \frac{\rho}{2}\lambda^2$ is strictly concave with respect to λ, as the second derivative $\frac{\partial^2 J}{\partial \lambda^2} = -\rho < 0$. Thus, a unique global maximum exists. We first consider the unconstrained optimization problem. Setting the first-order derivative to zero yields:

$$\frac{dJ}{d\lambda} = \Delta_{i,t} - \rho\lambda = 0 \implies \lambda_{unc} = \frac{\Delta_{i,t}}{\rho}. \tag{17}$$

Incorporating the box constraint $0 \leq \lambda \leq 1$, the problem becomes a quadratic program over a convex set. The optimal solution is obtained by projecting the unconstrained stationary point λ_{unc} onto the feasible set $[0, 1]$. This yields $\lambda^* = \Pi_{[0,1]}(\lambda_{unc})$.

Recall the closed-form solution for the participation gate derived in Section 3.3.3:

$$\lambda^*(\Delta) = \min\left(1, \max\left(0, \frac{\Delta}{\rho}\right)\right). \tag{18}$$

This function is a composition of a linear scaling function $f(x) = x/\rho$ and the projection function $\Pi_{[0,1]}(x) = \min(1, \max(0, x))$. The projection function $\Pi_{[0,1]}(\cdot)$ is non-expansive, i.e., it is 1-Lipschitz continuous: $|\Pi(a) - \Pi(b)| \leq |a - b|$. The scaling function is obviously $1/\rho$-Lipschitz. Therefore, the composition is $1/\rho$-Lipschitz:

$$|\lambda^*(\Delta_1) - \lambda^*(\Delta_2)| \leq \left|\frac{\Delta_1}{\rho} - \frac{\Delta_2}{\rho}\right| = \frac{1}{\rho}|\Delta_1 - \Delta_2|. \tag{19}$$

This concludes the proof. □

D.4 Proof of Theorem 3.4 (Bounded Policy Shift)

Proof. We analyze the sensitivity of the output policy π_i with respect to the estimated advantage Δ. Let the mixed logits be $z(\lambda) = (1-\lambda)z^{loc} + \lambda z^{coop}$. The policy is given by $\pi = \text{softmax}(z(\lambda))$.

Consider two advantage estimates Δ and $\hat{\Delta}$ with error $\epsilon = |\Delta - \hat{\Delta}|$. Let λ and $\hat{\lambda}$ be the corresponding gates. From Proposition 3.2, we have $|\lambda - \hat{\lambda}| \leq \frac{\epsilon}{\rho}$.

The logits are linear in λ:

$$\|z(\lambda) - z(\hat{\lambda})\|_2 = \|(\lambda - \hat{\lambda})(z^{coop} - z^{loc})\|_2 = |\lambda - \hat{\lambda}| \cdot \|z^{coop} - z^{loc}\|_2. \tag{20}$$

Let $C_z = \sup \|z^{coop} - z^{loc}\|_2$ be the maximum bounded difference between the two policy heads (bounded by the neural network constraints). Then:

$$\|z(\lambda) - z(\hat{\lambda})\|_2 \leq \frac{C_z}{\rho}\epsilon. \tag{21}$$

The softmax function $\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$ is Lipschitz continuous with respect to the ℓ_2 norm, with Lipschitz constant $L_\sigma = 1$ (or smaller depending on definition, typically bounded by 1 for the Jacobian norm). Thus, the distance between the probability distributions in Total Variation (TV) or ℓ_1 norm is bounded:

$$\|\pi(\lambda) - \pi(\hat{\lambda})\|_1 \leq L_\sigma \|z(\lambda) - z(\hat{\lambda})\|_2 \leq \frac{L_\sigma C_z}{\rho}\epsilon. \tag{22}$$

Letting $L_z = L_\sigma C_z$, we obtain:

$$D_{\text{TV}}(\pi(\Delta)||\pi(\hat{\Delta})) \leq \frac{L_z}{\rho}\epsilon. \tag{23}$$

This implies that the policy shift is linearly bounded by the value estimation error, and the sensitivity is explicitly controlled by the friction parameter ρ. A larger friction ρ leads to a more stable policy. □

D.5 Proof of Theorem 3.6 (Convergence to Pareto Stationarity)

Proof. We consider the multi-objective optimization problem for the shared parameters θ:

$$\min_\theta \mathbf{L}(\theta) = \{L_{loc}(\theta), L_{coop}(\theta)\}. \tag{24}$$

A point θ is Pareto stationary if there exists no direction d such that $d^\top \nabla L_{loc}(\theta) < 0$ and $d^\top \nabla L_{coop}(\theta) < 0$. This is equivalent to saying that 0 lies in the convex hull of the gradients.

The MGDA update rule computes the direction g by solving:

$$\min_{w \in [0,1]} \|w\nabla L_{loc} + (1-w)\nabla L_{coop}\|_2^2. \tag{25}$$

Let the solution be g^*. According to Theorem 1 in [42] and standard convex analysis:

1. If $g^* = 0$, then the current parameters θ satisfy the Karush-Kuhn-Tucker (KKT) conditions for Pareto stationarity.
2. If $g^* \neq 0$, then $-g^*$ is a descent direction for both objectives. Specifically, it satisfies $g^* \cdot \nabla L_{loc} \geq \|g^*\|^2$ and $g^* \cdot \nabla L_{coop} \geq \|g^*\|^2$ (subject to the specific geometry of the convex hull).

In our RL setting, we apply this to the expected gradients. By using a stochastic gradient descent scheme with this common descent direction, and assuming standard conditions on the learning rate (Robbins-Monro) and bounded variance of the gradient estimators, the sequence of parameters converges to a set of points where the expected gradient norm approaches zero, i.e., a Pareto stationary point. □

D.6 Discussion on Theoretical Assumptions

Our derivation of the Lipschitz continuity for the gating function relies on the assumption that the advantage function is locally continuous. In traffic signal control, the underlying dynamics can be discontinuous due to discrete signal switching. However, the probabilistic nature of the policy (Softmax) and the value function approximation (Neural Networks) effectively smooth the expected returns over the optimization landscape. Thus, the Lipschitz assumption holds in expectation, ensuring the stability of the gating mechanism in practice.

Assumption on Bi-level Convergence. Our theoretical analysis in Section 3.4 primarily focuses on the lower-level multi-objective optimization (Theorem 3.6) and the stability of the gating mechanism. We assume that the macroscopic traffic topology and the corridor task generation evolve slowly enough to allow the lower-level policies to converge. A rigorous proof of the fully coupled global convergence would require strong assumptions about the ergodicity of the traffic MDP and the strict two-time-scale stochastic approximation bounds, which remains a challenging open direction for general hierarchical MARL frameworks.

Assumption on Trade-off Bounds. The trade-off bound $\mathcal{O}(1/\rho)$ discussed in Remark 3.8 assumes that the advantage difference $\Delta_{i,t}$ is accurately estimated by the critics. In highly stochastic environments, the variance in advantage estimation could temporarily loosen this bound. However, the Lipschitz continuous nature of our gate ensures that such estimation noises do not lead to catastrophic divergence in the local execution efficiency.

D.7 Conditional Global Convergence via TTSA

The Corridor Agent evaluates selected directional rewards using the same reward definition as the Intersection Agents. This structural connection supports coordination, but does not exclude objective conflicts or establish convergence by itself. We connect the two levels through two-timescale stochastic approximation (TTSA), following its classical convergence framework [3]. Related analyses establish finite-sample bounds for two-timescale reinforcement learning [11] and Gaussian approximation bounds for linear TTSA [5]. These results motivate our formulation, but do not directly establish convergence of the nonlinear PPO-based hierarchy.

Proposition D.1 (Conditional hierarchical convergence)**.** *Assume that: (i) the iterates are bounded, the expected update fields are Lipschitz continuous, and sampling errors are martingale differences with bounded conditional second moments; (ii) the lower- and upper-level learning rates satisfy*

$$\sum_n a_n = \sum_n b_n = \infty, \qquad \sum_n (a_n^2 + b_n^2) < \infty, \qquad \frac{b_n}{a_n} \to 0;$$

(iii) for each fixed ϕ, the lower-level limiting ODE has a unique globally asymptotically stable equilibrium $\theta^*(\phi)$, with uniform attraction and a Lipschitz equilibrium response; and (iv) the upper-level limiting ODE, evaluated at $\theta^*(\phi)$, has a unique globally asymptotically stable equilibrium ϕ^*. Then the standard TTSA convergence result [3] yields

$$(\theta_n, \phi_n) \longrightarrow (\theta^*(\phi^*), \phi^*) \quad \textit{almost surely}.$$

If the lower-level equilibrium satisfies the exact MGDA stationarity condition, it is Pareto stationary.

Proof. Since $b_n/a_n \to 0$, the lower level tracks $\theta^*(\phi_n)$ while the upper level evolves on the slower timescale. The upper-level updates therefore track their reduced limiting ODE. Its assumed stability gives $\phi_n \to \phi^*$; continuity of $\theta^*(\phi)$ then gives $\theta_n \to \theta^*(\phi^*)$ almost surely. □

Proposition 3.2 and Theorem 3.4 bound the sensitivity of the gate and executed policy to advantage-estimation errors. They support, but do not establish, the equilibrium-response and ODE stability assumptions above. Likewise, slower replanning alone does not imply $b_n/a_n \to 0$. This conditional result concerns stationary traffic dynamics and estimates satisfying the stated noise conditions; it is not an almost-sure convergence guarantee for our fixed-learning-rate PPO implementation.

E Additional Analysis

E.1 Smaller decision space at the upper level

Concretely, at a grow step with tail node v_{tail}, the admissible action set is $\mathcal{A}_{\text{mask}}(s) \subseteq \mathcal{N}_{\text{out}}(v_{\text{tail}}) \cup \{\texttt{STOP}\}$, hence $|\mathcal{A}_{\text{mask}}(s)| \leq |\mathcal{N}_{\text{out}}(v_{\text{tail}})| + 1 \ll |\mathcal{V}|$. In addition, the seeding stage restricts corridor initiation to a candidate set $\mathcal{S}$ of size k (top-k seeds), which bounds the number of corridor-generation episodes considered per planning cycle. Finally, the step-wise observation used for growth can be implemented with only the current tail embedding and the embeddings of its masked outgoing neighbors, avoiding full-graph conditioning at each decision step.

E.2 Interface-Preserving and Low-Overhead Cooperation at the Lower Level

At the lower level, CorridorLight keeps cooperation lightweight by preserving the original per-intersection control interface and localizing the task signal. Each Intersection Agent still selects from the same phase action set $\mathcal{A}_i$ as in standard decentralized TSC, cooperation does not introduce joint actions, message sharing. Instead, the Corridor Agent broadcasts a compact task descriptor G_t^i that is *intersection-local*, consisting only of (i) a binary mask over the lanes of intersection i and (ii) lane-level preference weights p_ℓ for those lanes. Therefore, the cooperation input size scales with the local lane count $|\mathcal{L}_i|$ rather than the network size $|\mathcal{V}|$ or $|\mathcal{E}|$, yielding $O(|\mathcal{L}_i|)$ task-conditioning cost per junction and avoiding $O(E)$ per-step neighbor communication used by graph-based coordination baselines.

Cooperation is implemented as a *single scalar* participation gate $\lambda_{i,t} \in [0, 1]$ that mixes a local head and a task-conditioned head through logits interpolation: $z_i = (1 - \lambda_{i,t}) z_i^{loc} + \lambda_{i,t} z_i^{coop}$. Crucially, computing $\lambda_{i,t}$ is $O(1)$ via a closed-form projection $\lambda_{i,t}^* = \Pi_{[0,1]}(\Delta_{i,t}/\rho)$, so the negotiation overhead is constant-time and does not require iterative optimization or additional learning modules. Moreover, because $\lambda_{i,t}^*$ is Lipschitz in the advantage difference, small estimation errors translate to bounded changes in the executed policy, preventing brittle on/off switching and reducing the effective coordination burden. Overall, the lower layer achieves scalable cooperation with minimal additional computation and bandwidth: each junction receives only a local mask and weights at planning frequency, and performs constant-time gating while retaining fast decentralized actuation.

F Limitation

Although the networks in our experiments are derived from real-world maps, all evaluations are conducted in simulation and may not fully capture the uncertainty and operational constraints of real deployments, including sensing noise, missing observations, pedestrian phases, and hardware-level signal-control constraints. CorridorLight is best suited to settings where congestion exhibits directional propagation, such as arterial traffic and green-wave coordination. In highly meshed urban networks, areas dominated by short turning movements, or regimes with diffuse area-wide congestion, the corridor abstraction may be less expressive. In addition, the junction-disjoint corridor constraint, introduced for scalability and conflict avoidance, can be conservative in networks where multiple high-value flows share the same bottleneck intersection. This may prevent important corridors from being selected simultaneously and rigidly assign priority at shared bottlenecks, limiting the planner's ability to represent overlapping traffic demands and potentially yielding suboptimal solutions. Finally, corridor prioritization may redistribute rather than eliminate delay: improvements in average travel time or throughput can mask increased waiting times for minor approaches.

G Outlook

Beyond traffic control, we view task negotiation with counterfactual-gain gating as a general recipe for scalable cooperation in multi-agent systems. A promising direction is to extend CorridorLight to more general MARL settings with heterogeneous agents, broader interaction graphs, and domain-specific task abstractions (e.g., dynamic coalition formation and hierarchical subtask discovery), while maintaining interface-preserving execution under partial observability and limited communication. We also plan to study how to learn transferable task descriptors and negotiation rules that generalize across environments, providing robust cooperation policies in diverse multi-agent scenarios.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Each aspect described in the abstract is supported by a corresponding section, theoretical analysis, result, or experiment in the article.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our work are discussed in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theorem statements and assumptions are presented in Section 3.4, while the corresponding proofs are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all code required to implement our method in the GitHub repository linked in the article abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide all code required to implement our method in the GitHub repository linked in the article abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all code required to implement our method in the GitHub repository linked in the article abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the detailed experiment results in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the hardware setting and running time in Appendix C

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite all environments and algorithms used in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The codebase is well documented and includes clear instructions

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.