Discovering Programmatic Policies from Reinforcement Learning-Based Traffic Signal Controllers

Lindong Xie[1,2] **Yang Zhang**[1] **Beiyu Song**[1] **Zeren Xing**[1] **Flora D. Salim**[2] **Edward Chung**[1*]

[1]Department of Electrical and Electronic Engineering
The Hong Kong Polytechnic University, Hong Kong, China
[2]School of Computer Science and Engineering
University of New South Wales, Sydney, Australia
lindong.xie@connect.polyu.hk edward.cs.chung@polyu.edu.hk

Abstract

Traffic signal control (TSC) is critical for mitigating urban congestion. Deep reinforcement learning (DRL) has achieved strong performance by learning adaptive policies from traffic observations, but learned neural controllers are often opaque, structurally complex, and difficult to generalize across scenarios. We propose PPL-TSC, a programmatic policy learning framework that distills compact and human-readable TSC policies from DRL teachers. PPL-TSC formulates policy learning as an evolutionary search over phase utility functions (PUFs), which score candidate signal phases and select the highest-utility phase for execution. Each generation consists of four steps: policy generation, policy review, policy calibration, and policy evaluation. In the generation step, a large language model (LLM) acts as a semantics-aware evolutionary operator to propose and refine candidate PUFs. Generated PUFs are screened through explicit checks for syntactic validity, mathematical soundness, and consistency with TSC principles, with invalid ones repaired through iterative feedback. Valid PUFs are then calibrated to improve behavioral fidelity to the DRL teacher, and evaluated using a joint score combining teacher fidelity and traffic-control performance to select candidates for the next generation. Extensive experiments across diverse traffic scenarios and representative DRL-based controllers show that PPL-TSC produces compact and human-readable policies, achieves competitive performance for most teacher families, improves transfer generalization, and substantially reduces computational cost. Code is available at https://github.com/ForrestXie9/PPL-TSC.

1 Introduction

Traffic signal control (TSC) is a core problem in intelligent transportation systems, with direct implications for urban congestion, travel efficiency, and transportation-related emissions [1, 2]. Classical approaches to TSC include fixed-time control [3–5], which relies on predetermined timing plans derived from historical demand, and adaptive control [6–9], which adjusts signal decisions based on real-time traffic observations. While these paradigms have supported decades of deployment, they either lack responsiveness to non-stationary traffic patterns or depend heavily on manually designed control rules and simplified modeling assumptions (e.g., stationary demand) [10]. Designing effective TSC policies that are both adaptive and practically deployable remains a fundamental challenge.

*Corresponding author.

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

Table 1: Transfer performance of DRL-based TSC methods under Manhattan traffic-flow shifts and Manhattan-to-Los Angeles transfer.

Teacher	ΔATT (%) ↑		Win Rate (%) ↑	
	In-dom.	Cross-scen.	In-dom.	Cross-scen.
CoLight [14]	-0.24 ± 6.47	N/A	50.0%	N/A
EcoLight [18]	-0.37 ± 6.60	-51.79 ± 42.43	50.0%	8.0%
FRAP [15]	-0.07 ± 3.60	-20.73 ± 38.45	65.0%	24.0%
MPLight [19]	-0.14 ± 5.60	-2.34 ± 32.67	45.0%	28.0%
TinyLight [20]	-62.62 ± 172.58	N/A	50.0%	N/A

Note: ATT denotes average travel time. ΔATT denotes relative ATT improvement under transfer; negative values mean the transferred controller performs worse (i.e., ATT increases). ATT win rate is the fraction of transfer cases in which ATT does not increase. N/A indicates methods not applicable in cross-scenario transfer.

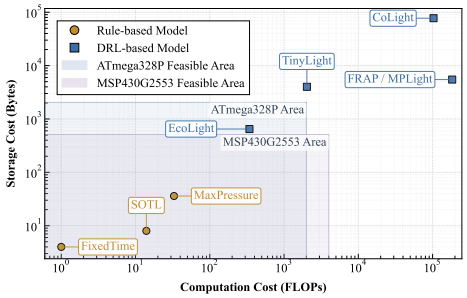


Figure 1: Storage and computational costs of DRL-based and rule-based TSC methods.

Deep reinforcement learning (DRL) has become a prominent paradigm for adaptive TSC [11–13]. Through simulator interaction, DRL agents learn neural policies that map traffic states to signal phase decisions, achieving strong empirical performance across diverse traffic networks [14, 15]. Despite this progress, deploying neural controllers in practice remains challenging. First, learned policies lack explicit traffic-control concepts (e.g., queue pressure, downstream congestion), making it difficult to inspect whether their decisions follow established traffic-control principles [16]. Second, policies trained for specific traffic patterns often degrade under distribution shifts [17]. As shown in Table 1, most representative DRL methods remain stable under in-domain traffic-flow variations in Manhattan but show performance degradation when transferred to Los Angeles. Under cross-scenario transfer, methods like EcoLight [18], FRAP [15], and MPLight [19] suffer performance drops, while others (e.g., CoLight [14], TinyLight [20]) are inapplicable due to action-space mismatches (e.g., differing phase definitions and dimensionality), preventing direct policy reuse without architectural adaptation. Addressing such shifts typically requires retraining with additional data and computation. Third, neural controllers impose nontrivial memory and computational costs during real-time inference [20]. Figure 1 compares storage and floating-point operations (FLOPs) per decision. DRL-based methods require substantially more resources than classical rule-based controllers (e.g., fixed-time [3], SOTL [6], and MaxPressure [7]). This resource gap is particularly consequential for deployment on low-resource edge platforms.

These limitations have motivated growing interest in explicit policy learning for TSC, where controllers are represented as rules or programs rather than black-box neural networks [17, 21, 22]. Existing methods construct such policies using Monte Carlo tree search [23] or genetic programming [24], but they typically search from scratch through extensive interaction with the traffic environment. This process is costly in large combinatorial policy spaces and fails to leverage the control knowledge already encoded in DRL teachers. Moreover, these methods provide limited guarantees that the discovered rules remain semantically consistent with established TSC principles.

To address these challenges, we propose PPL-TSC, a unified programmatic policy learning framework that learns explicit and human-readable TSC policies from diverse DRL teachers. Instead of discovering explicit policies through trial-and-error exploration from scratch, PPL-TSC leverages rollout data collected from a pretrained DRL teacher to guide policy learning. We represent the student policy as a phase utility function (PUF), an explicit phase-scoring function that scores candidate phases using phase-level traffic features and selects the highest-scoring phase. Its explicit expression makes the contribution of each term inspectable. Overall, our main contributions are summarized as follows:

- We propose PPL-TSC, a teacher-guided programmatic policy learning framework for TSC that distills explicit policies from pretrained DRL teachers, reducing reliance on costly policy search from scratch through extensive environment interaction.
- We formulate policy learning as an iterative generate–review–calibrate–evaluate process over phase utility functions. The process integrates LLM-based program search, explicit validity checks, intersection-specific calibration, and joint evaluation to learn human-readable policies aligned with teacher behavior and TSC principles.
- We conduct extensive experiments across representative DRL-based TSC methods and diverse traffic scenarios. PPL-TSC learns human-readable student policies that achieve competitive traffic

performance, improve generalization under transfer settings, and substantially reduce storage and computational costs relative to their DRL teachers.

2 Related Work

Early TSC systems relied mainly on fixed-time strategies, such as Webster's method [3], GreenWave [4], and MAXBAND [5], where signal timings are derived from historical demand and fixed during deployment. Adaptive methods later introduced online updates based on real-time traffic observations, including rule-based heuristics such as SOTL [6] and MaxPressure [7], and model-based systems such as SCOOT [8] and SCATS [9]. More recently, DRL has become a prominent paradigm for learning adaptive policies directly from traffic observations, reducing reliance on hand-designed rules and simplified traffic assumptions.

Existing DRL-based TSC methods can be broadly divided into value-based and policy-based approaches. Value-based methods, including FRAP [15], MPLight [19], CoLight [14], and TinyLight [20], estimate the long-term value of each candidate phase and select the phase with the highest value. Policy-based methods, such as A-CATs [25], policy-gradient methods [26], multi-agent actor–critic methods [27], and AttendLight [28], directly learn mappings from traffic states to control actions. Despite their empirical success, these neural controllers remain difficult to interpret, validate, and deploy in resource-constrained settings [17]. These limitations have motivated interest in explicit policy representations for TSC. Early work considered predefined polynomial control forms [29], whose parameters were fitted to mimic trained neural controllers. However, such fixed structures are often too restrictive to capture complex decision logic. More recent methods, such as π-Light [17], GPLight+ [21], and SymLight [22], use program search operators, including Monte Carlo tree search [23] and genetic programming [30], to construct explicit control policies through interaction with the traffic environment. While these methods improve transparency, they typically search from scratch in a large combinatorial policy space, require substantial environment interaction, and make limited use of the knowledge already encoded in strong DRL teachers. In addition, evaluating candidate policies mainly through traffic-control performance does not guarantee that the resulting expressions are mathematically valid or semantically meaningful.

PPL-TSC differs from these approaches by learning programmatic policies directly from pretrained DRL teachers rather than discovering them from scratch. It uses teacher rollout data to guide policy learning and integrates generation, review, calibration, and evaluation, with explicit checks of generated functions and joint evaluation of teacher fidelity and control performance.

3 Method

We first formalize teacher-guided policy learning for TSC and define the student policy representation. We then describe the four steps of PPL-TSC: generation, review, calibration, and evaluation.

3.1 Problem Formulation

We consider a traffic network with M signalized intersections. At decision step t, intersection i observes a local traffic state $\mathbf{s}_t^{(i)}$ and selects a signal phase $p_t^{(i)}$ from a feasible set $\mathcal{P}_t^{(i)}$. A signal phase is a predefined set of compatible, non-conflicting traffic movements that receive the right-of-way concurrently. This feasible set depends on the controller's action space: phase-selection controllers directly choose among all feasible phases, whereas keep/switch controllers choose between keeping the current phase and switching to the next one, i.e., $\mathcal{P}_t^{(i)} = \{p_{t,\mathrm{cur}}^{(i)}, \mathrm{next}(p_{t,\mathrm{cur}}^{(i)})\}$. Definitions of TSC terms are provided in Appendix A.1.1.

In DRL-based TSC, a neural policy π^{Tch}, referred to as the teacher policy, maps traffic states to signal phases and is learned through interaction with the environment. Executing the teacher over the network produces rollout data $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$, where $\mathcal{D}_i = \{(\mathbf{s}_t^{(i)}, p_t^{\mathrm{Tch},(i)})\}_{t=1}^{N_i}$ contains the observed state and teacher-selected phase for intersection i. Background on DRL-based TSC is provided in Appendix A.1.2.

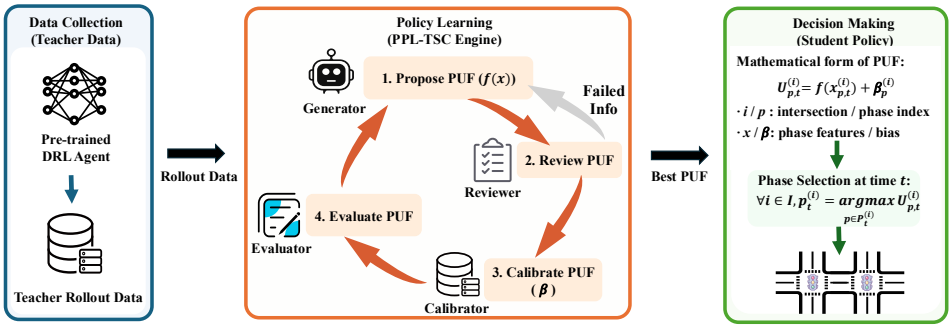


Figure 2: Overview of PPL-TSC. The framework consists of three modules: data collection, policy learning, and decision making. Data collection gathers rollout samples from a pretrained DRL-based TSC teacher. Policy learning obtains a phase utility function (PUF) through iterative generation, review, calibration, and evaluation. Decision making uses the learned PUF $f(\mathbf{x})$ with intersection-specific phase bias terms $\boldsymbol{\beta}$ for real-time signal phase selection.

Our objective is to learn a compact and human-readable student policy π^{Stu} that maps the same traffic states to phase decisions, approximates the teacher's behavior, and maintains competitive traffic-control performance.

3.2 Student Policy Representation

We represent the student policy as a phase utility function (PUF) that scores candidate phases. Let $\mathbf{x}_{p,t}^{(i)}$ denote the traffic features used to score candidate phase $p \in \mathcal{P}_t^{(i)}$, constructed from the local state $\mathbf{s}_t^{(i)}$ according to the teacher's input and action interfaces. The utility of phase p is defined as

$$U_{p,t}^{(i)} = f(\mathbf{x}_{p,t}^{(i)}) + \beta_p^{(i)}, \tag{1}$$

where $f(\cdot)$ is a programmatic expression shared across intersections to capture common decision logic, and $\beta_p^{(i)}$ is an intersection- and phase-specific bias term for local adaptation. For direct cross-scenario transfer, we retain $f(\cdot)$ and set $\beta_p^{(i)} = 0$ at each target intersection. For target-specific adaptation, the phase biases can be fine-tuned while keeping $f(\cdot)$ fixed.

The student policy then selects the phase with the highest utility:

$$p_t^{\mathrm{Stu},(i)} = \arg \max_{p \in \mathcal{P}_t^{(i)}} U_{p,t}^{(i)}. \tag{2}$$

3.3 PPL-TSC Framework

PPL-TSC searches for the PUF in Eq. (1) through an evolutionary learning process with four steps: generation, review, calibration, and evaluation, as illustrated in Fig. 2. The search proceeds for up to $G_{\max}$ generations with a population size N_{pop}.

3.3.1 Policy Generation

The generation module uses an LLM as a program synthesis operator to propose candidate phase-scoring functions $f(\cdot)$ from the teacher rollout data $\mathcal{D}$ [31, 32]. Its goal is to explore the space of readable phase-scoring rules while keeping generated candidates consistent with the available traffic features and basic TSC principles.

We use two prompt types. The initialization prompt $\mathcal{T}(\mathrm{Task}_{\mathrm{init}}, \mathrm{Ctx}_{\mathrm{init}}, \mathrm{Cons}, \mathrm{Out})$ creates the initial population from feature descriptions derived from teacher states. The evolution prompt $\mathcal{T}(\mathrm{Task}_{\mathrm{evol}}, \mathrm{Ctx}_{\mathrm{evol}}, \mathrm{Cons}, \mathrm{Out})$ generates offspring by conditioning on parent PUFs and their evaluation feedback. In both cases, Cons specifies syntactic, mathematical, and semantic requirements: generated code must be executable, numerical operations must remain robust under extreme inputs, and feature use should respect TSC principles such as monotonicity with respect to traffic load.

The output specification (i.e., Out) requires a candidate PUF together with a brief design rationale. Detailed example prompts and constraints are provided in Appendices A.2 and A.3.

Initialization prompts are used until the population reaches N_{pop}, after which evolution prompts generate offspring from selected parent PUFs.

3.3.2 Policy Review

After receiving candidate PUFs from the generation step, the policy review module checks whether the generated functions satisfy syntactic correctness, mathematical validity, and semantic consistency under the specified constraints. The review checks the generated function f before calibration; the calibrated biases then adjust the scores of individual phases. This explicit verification is needed because LLM-generated functions may still violate constraints despite prompt guidance [33]. Candidates that fail verification are marked as invalid and returned to the LLM with diagnostic feedback for revision. An example repair prompt is provided in Appendix A.2.

3.3.3 Policy Calibration

Policy calibration adjusts the bias terms in Eq. (1) after a candidate PUF passes review. The phase-scoring function $f(\cdot)$ is kept fixed and shared across intersections, while the phase biases are learned separately for each intersection from teacher rollouts. This design improves the student's agreement with local teacher decisions while preserving the generated rule and avoiding further interaction with the traffic environment.

For intersection i, let $\mathcal{P}^{(i)} = \bigcup_{t=1}^{N_i} \mathcal{P}_t^{(i)}$ denote the union of candidate phase sets over the rollouts. We assign one bias term $\beta_p^{(i)}$ to each phase $p \in \mathcal{P}^{(i)}$, collected in $\boldsymbol{\beta}^{(i)}$. We optimize $\boldsymbol{\beta}^{(i)}$ to maximize agreement with teacher decisions on $\mathcal{D}_i$:

$$\max_{\boldsymbol{\beta}^{(i)}} \frac{1}{N_i} \sum_{t=1}^{N_i} \mathbf{1}\left(p_t^{\text{Stu},(i)} = p_t^{\text{Tch},(i)}\right), \tag{3}$$

where $\mathbf{1}(\cdot)$ is the indicator function, which equals 1 when the student and teacher select the same phase and 0 otherwise. Since this objective function is non-differentiable, we use CMA-ES [34], a robust black-box optimization algorithm, to optimize $\boldsymbol{\beta}^{(i)}$ independently for each intersection.

3.3.4 Policy Evaluation

Policy evaluation ranks calibrated PUFs according to both teacher fidelity and closed-loop traffic performance. Fidelity measures whether the student reproduces the teacher's phase choices, while traffic performance assesses whether the student remains effective when deployed in the traffic environment. We combine the two criteria using the joint score

$$J(U) = w_1 F(U) + w_2 P(U), \tag{4}$$

where $F(U)$ denotes behavioral fidelity, $P(U)$ denotes traffic performance relative to the teacher, and $w_1, w_2 \geq 0$ balance the two terms. In our experiments, we use equal weights, $w_1 = w_2 = 1$, for teacher-policy fidelity and downstream control performance.

Behavioral Fidelity $F(U)$. We measure fidelity on the teacher rollout dataset $\mathcal{D}$ used in calibration, with student decisions given by Eq. (2). The fidelity score is the fraction of rollout samples on which the student and teacher select the same phase:

$$F(U) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{M} \sum_{t=1}^{N_i} \mathbf{1}\left(p_t^{\text{Stu},(i)} = p_t^{\text{Tch},(i)}\right), \tag{5}$$

where $\mathbf{1}(\cdot)$ is the indicator function and $|\mathcal{D}| = \sum_{i=1}^{M} N_i$. This metric lies in $[0, 1]$, with 1 indicating perfect agreement with the teacher on the rollout data.

Control Performance $P(U)$. We evaluate traffic control performance by deploying the student policy in the same TSC environment used to test the teacher. Let ATT^{Stu} and ATT^{Tch} denote the average travel times of the student and teacher controllers, respectively. The relative improvement over the teacher is

$$\Delta = \frac{ATT^{\text{Tch}} - ATT^{\text{Stu}}}{|ATT^{\text{Tch}}| + \epsilon}, \tag{6}$$

where $\epsilon = 10^{-4}$ is used for numerical stability. A positive Δ indicates that the student achieves a lower average travel time than the teacher, $\Delta = 0$ indicates equal performance, and $\Delta < 0$ indicates worse performance.

We convert this improvement to a bounded performance score in $(0, 1]$ as

$$P(U) = \exp\left(\min(\Delta, 0)\right). \tag{7}$$

This score equals 1 when the student matches or outperforms the teacher and decreases smoothly when the student performs worse.

After evaluation, each offspring is compared with its parent using the joint score in Eq. (4). The offspring replaces its parent only if it achieves a higher score; otherwise, the parent is retained. The search terminates when the maximum number of generations $G_{\max}$ is reached.

4 Experiments

We evaluate PPL-TSC through six research questions. **RQ1 (Performance):** Can PPL-TSC match the traffic-control performance of its DRL teacher under identical evaluation settings? **RQ2 (Fidelity):** How accurately does PPL-TSC reproduce the teacher's decision behavior? **RQ3 (Generalizability):** Does PPL-TSC generalize under same-scenario different-traffic-flow and cross-scenario settings? **RQ4 (Readability):** Can the learned student be expressed as a compact policy consistent with traffic control principles? **RQ5 (Resource Efficiency):** Is PPL-TSC more resource-efficient than neural teachers? **RQ6 (Ablation):** How do policy generation, calibration, and evaluation affect the effectiveness of PPL-TSC?

4.1 Experimental Setup

Simulator and traffic scenarios. We conduct experiments in CityFlow [35], an open-source microscopic traffic simulator. The benchmark includes single-intersection, corridor, and grid scenarios from seven road networks derived from five real-world urban areas [36]: three 1×1 Hangzhou networks (Hangzhou1/2/3), a 5×1 Atlanta corridor, a 3×4 Jinan grid, a 4×4 Los Angeles grid, and a 16×3 Manhattan grid. Jinan and Manhattan are structurally homogeneous, with all non-virtual signalized intersections sharing the same phase and link configurations, whereas Atlanta and Los Angeles are structurally heterogeneous due to multiple intersection templates. Each road network defines a scenario. For each scenario, we use one real-world traffic flow and generate four additional flows by perturbing each vehicle's departure time with uniform noise in $[-60\text{s}, 60\text{s}]$.

Teacher controllers and metrics. We use five DRL-based TSC methods as teacher controllers: CoLight [14], FRAP [15], MPLight [19], EcoLight [18], and TinyLight [20]. EcoLight natively uses a keep/switch action interface, so we report EcoLight (native) in this original two-action setting and additionally evaluate an EcoLight (phase-selection) variant in which the student directly scores all feasible phases, consistent with the other teacher families. We evaluate PPL-TSC using average travel time (ATT) for control performance and average behavioral fidelity (ABF) for teacher-policy fidelity. We also report relative ATT improvement (ΔATT), defined as the student's relative ATT reduction over the teacher, and ATT wins, where the student achieves ATT no worse than its teacher.

4.2 Performance and Fidelity (RQ1–RQ2)

We first evaluate whether PPL-TSC preserves the control performance and decision behavior of its DRL teachers in the original deployment setting. PPL-TSC (Tailored) learns a separate policy for each teacher–scenario–flow combination, whereas PPL-TSC (Unified) learns a shared programmatic policy for each teacher family across multiple scenarios. Table 2 reports ΔATT, ATT wins, and ABF over 35 scenario–flow combinations for each teacher family.

For RQ1, PPL-TSC achieves competitive closed-loop performance relative to its teachers for most teacher families. Tailored obtains a positive mean ΔATT for CoLight, FRAP, MPLight, and EcoLight (phase-selection), while Unified improves CoLight, FRAP, and EcoLight (phase-selection) and remains nearly neutral for MPLight. These results indicate that compact programmatic policies can preserve, and often improve, the teacher's control quality in the original deployment setting.

Table 2: Performance and fidelity of PPL-TSC over 35 scenario–flow combinations.

Teacher	PPL-TSC (Tailored)			PPL-TSC (Unified)		
	ΔATT (%) ↑	ATT wins ↑	ABF ↑	ΔATT (%) ↑	ATT wins ↑	ABF ↑
CoLight	$\mathbf{11.51 \pm 23.98}$	17/35	0.71 ± 0.17	$\mathbf{16.70 \pm 32.98}$	18/35	0.60 ± 0.22
FRAP	$\mathbf{11.79 \pm 27.12}$	24/35	0.63 ± 0.16	$\mathbf{12.07 \pm 32.99}$	22/35	0.58 ± 0.19
MPLight	$\mathbf{0.54 \pm 6.16}$	21/35	0.73 ± 0.12	-0.03 ± 10.96	17/35	0.69 ± 0.13
EcoLight (native)	-18.05 ± 41.66	15/35	0.84 ± 0.07	-23.27 ± 36.17	8/35	0.82 ± 0.07
EcoLight (phase-selection)	$\mathbf{22.70 \pm 27.23}$	30/35	0.48 ± 0.11	$\mathbf{20.99 \pm 30.63}$	30/35	0.40 ± 0.12
TinyLight	-8.43 ± 38.60	21/35	0.64 ± 0.16	-25.19 ± 34.56	8/35	0.61 ± 0.12

The main performance exceptions occur when PPL-TSC learns from TinyLight and EcoLight (native). The student policy distilled from TinyLight underperforms its teacher on average, especially under Unified. TinyLight's richer phase-level and incoming-lane inputs enlarge the search space, making the default search budget less effective. For EcoLight (native), the two-action keep/switch interface makes closed-loop performance sensitive to accumulated switching errors, leading to weaker ATT results especially in larger networks such as Jinan, Los Angeles, and Manhattan (Appendix A.8.2). Controlled diagnostic experiments for these two cases are provided in Appendix A.7.1.

For RQ2, PPL-TSC maintains moderate behavioral fidelity to the teacher policies. Tailored consistently achieves higher ABF than Unified across teacher families, indicating that fitting a policy to each teacher–scenario–flow setting tends to better match local phase decisions than using a single policy across multiple scenarios. EcoLight (native) achieves the highest ABF under both Tailored (0.84 ± 0.07) and Unified (0.82 ± 0.07) settings, although its ATT results remain weaker than those of the teacher. Among the phase-selection teachers, MPLight provides the strongest fidelity–performance balance: both variants stay close to the teacher in ATT while achieving high ABF values, suggesting that its decision logic is more amenable to approximation by a compact programmatic policy. Detailed scenario-level results are provided in Appendix A.8.2.

4.3 Generalizability (RQ3)

We evaluate generalizability under two transfer regimes: (i) same-scenario, different-traffic-flow, where a model trained on one traffic flow is tested on other traffic flows within the same scenario, and (ii) cross-scenario, where a model learned in one scenario is applied to different scenarios. For same-scenario, different-traffic-flow transfer, each scenario contains five traffic flows; each traffic flow is used once as the source flow and evaluated on the other four flows, yielding 20 transfer cases per scenario and 140 cases per teacher. For cross-scenario transfer, each source traffic flow is evaluated on five traffic flows from each of the other six scenarios, yielding 150 transfer cases per source scenario and 1050 cases per teacher. Table 3 reports relative ATT improvements and ATT win counts with respect to the transferred teacher policies.

Table 3: Generalization under two transfer settings. Higher ΔATT and ATT wins are better. CoLight and TinyLight are not applicable to cross-scenario transfer due to architectural limitations.

(a) Same-scenario, different-traffic-flow

Teacher	ΔATT (%) ↑	ATT wins ↑
CoLight	$\mathbf{15.66 \pm 38.49}$	61/140
EcoLight (native)	$\mathbf{24.51 \pm 28.37}$	122/140
EcoLight (phase-selection)	$\mathbf{25.19 \pm 31.00}$	120/140
FRAP	$\mathbf{11.84 \pm 36.96}$	89/140
MPLight	$\mathbf{0.57 \pm 11.79}$	55/140
TinyLight	-4.48 ± 50.05	98/140

(b) Cross-scenario

Teacher	ΔATT (%) ↑	ATT wins ↑
EcoLight (native)	$\mathbf{33.42 \pm 77.99}$	938/1050
EcoLight (phase-selection)	$\mathbf{44.85 \pm 34.32}$	954/1050
FRAP	$\mathbf{54.22 \pm 39.35}$	887/1050
MPLight	$\mathbf{46.04 \pm 36.10}$	946/1050

Table 3 shows different patterns across the two transfer regimes. Under same-scenario, different-flow transfer, PPL-TSC achieves the strongest gains for CoLight, EcoLight (native), EcoLight (phase-selection), and FRAP, indicating that the learned policies generalize beyond the source flow under moderate flow shifts. Cross-scenario transfer is more challenging due to changes in both network topology and traffic patterns. In this setting, programmatic students for EcoLight (native), EcoLight (phase-selection), FRAP, and MPLight achieve substantial improvements over the transferred teachers. These comparisons measure gains over transferred teachers, whose performance can itself deteriorate under distribution shift; they do not establish absolute control quality on the target networks. A

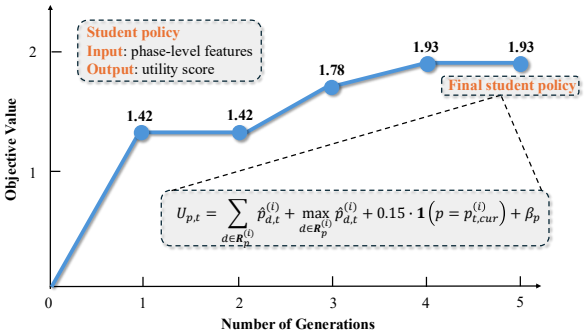


Figure 3: Evolution of student (i.e., PPL-MPLight) on Hangzhou3. $\hat{p}_{d,t}$ denotes normalized pressure, $\mathcal{R}_p$ the set of served directions, $p_{t,\text{cur}}$ the current phase, and β_p the phase bias. The objective value is the joint evaluation score in Eq. (4).

Table 4: Final phase biases and control performance comparison on Hangzhou3 (single intersection with four phases).

Quantity	Value
Phase-0 bias β_0	0.42
Phase-1 bias β_1	0.42
Phase-2 bias β_2	0.24
Phase-3 bias β_3	-0.34
Teacher ATT	86.01
Student ATT	84.04
MaxPressure ATT	103.98
ΔATT (%)	2.29
ABF	0.93
MaxPressure ABF	0.81

separate comparison using MPLight rollouts evaluates PPL-TSC against π-Light-Distill, GPLight+-Distill, and SymLight-Distill under the same teacher-guided protocol (Appendix A.8.5). PPL-TSC achieves the lowest mean and median cross-scenario ATT among these methods, providing additional evidence of transfer effectiveness beyond the teacher comparison.

4.4 Readability of the Learned Student Policy (RQ4)

We examine the MPLight-derived student, PPL-MPLight, on Hangzhou3. The student uses the teacher's current-phase and movement-pressure inputs. Fig. 3 traces the learned rule, and Table 4 reports its calibrated biases and performance. The learned policy is readable because each term has a clear traffic-control interpretation. Utility increases with both aggregate phase pressure and the maximum directional pressure within a phase, giving priority to phases serving congested movements. A small current-phase retention term discourages unnecessary switching, and the calibrated biases provide phase-specific offsets around the shared pressure-based rule.

This rule is closely related to MaxPressure [7], a classic pressure-based controller that selects phases using upstream–downstream queue differences. In this example, the student achieves both lower ATT (84.04 vs. 103.98) and higher behavioral fidelity to the teacher (ABF 0.93 vs. 0.81) than MaxPressure (Table 4). The explicit rule thus preserves recognizable pressure-based control logic while improving on MaxPressure in this case.

4.5 Resource Efficiency (RQ5)

We evaluate resource efficiency as an additional deployment benefit of the explicit programmatic representation, using two deployment-oriented metrics [22, 20]: mean parameter storage and computational cost. Mean parameter storage is measured in bytes from the number of stored scalar parameters or rule constants, assuming 32-bit values. Computational cost is measured by the estimated floating-point operations (FLOPs) required for one intersection-level signal decision, counting the forward pass of neural teachers or the evaluation of phase utility functions over candidate phases. These metrics quantify parameter storage and policy-scoring operations; executable code, runtime memory, and feature preprocessing incur additional costs. Fig. 4 compares DRL teachers and the programmatic students learned by PPL-TSC on a representative Hangzhou3 intersection. The shaded regions for ATmega328P and MSP430G2553 provide illustrative resource-budget references; actual deployment also depends on implementation overhead and the required decision interval.

As shown in Fig. 4, the programmatic students consistently lie in the low-resource regime. Compared with their DRL teachers, they typically reduce both storage and FLOPs by one to three orders of magnitude. For example, FRAP-derived students reduce mean parameter storage from 5,476 bytes to 47.2 bytes and require an average of 80.0 FLOPs, compared with 183,676 for the teacher. For MPLight, the derived students require only 28.0 bytes and 62.4 FLOPs on average. TinyLight-derived students are also highly compact, achieving storage and FLOP reductions of 98.16% and 95.14%,

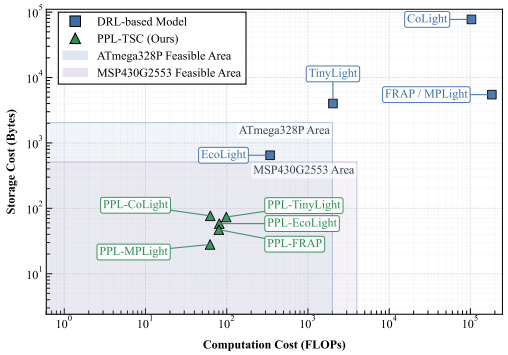


Figure 4: Parameter storage and policy-scoring FLOPs on a representative Hangzhou3 intersection. Shaded microcontroller regions serve as resource-budget references, not verified deployment limits.

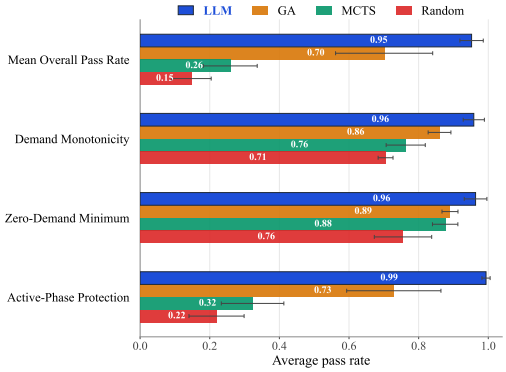


Figure 5: Average pass rate under three semantic constraints for functions generated by LLM, GA, MCTS, and random search.

respectively. Overall, these results establish lower storage and inference costs as an additional practical benefit of the programmatic representation, particularly for resource-constrained deployment.

4.6 Ablation Study (RQ6)

To answer RQ6, we perform targeted ablations on three components of PPL-TSC: policy generation, policy calibration, and policy evaluation. The calibration and evaluation ablations use the Tailored setting over 175 tasks: five teacher interfaces (CoLight, FRAP, MPLight, EcoLight (phase-selection), and TinyLight), seven scenarios, and five traffic flows.

We first examine policy generation. Fig. 5 compares LLM, GA [21], MCTS [17], and random search under a matched time budget in terms of their ability to produce expressions satisfying three traffic-related semantic constraints: demand monotonicity, zero-demand minimum, and active-phase protection. Definitions of these constraints are provided in Appendix A.3. We report the mean overall pass rate, defined as the fraction of expressions satisfying all three constraints simultaneously. LLM achieves the highest observed pass rate on each constraint and on all three jointly.

Table 5: Calibration and evaluation ablations in the Tailored setting over 175 tasks. The EcoLight results use the phase-selection interface, denoted EcoLight (phase-selection).

(a) Policy calibration

Variant	ΔATT (%) ↑	ATT wins ↑	ABF ↑
Full PPL-TSC	7.62 ± 28.80	113/175	0.64±0.14
w/o calibration	9.78 ± 34.96	116/175	0.55±0.18

(b) Policy evaluation

Variant	ΔATT (%) ↑	ATT wins ↑	ABF ↑
Full PPL-TSC	7.62 ± 28.80	113/175	0.64±0.14
Fidelity only	-14.07 ± 35.02	50/175	0.68±0.15
Performance only	12.82 ± 28.77	116/175	0.55±0.18

We then analyze policy calibration (Table 5(a)). Removing calibration slightly improves ATT-oriented metrics, but reduces ABF from 0.64 to 0.55. This pattern suggests that, in our setting, calibration mainly improves alignment with the teacher's decisions rather than traffic efficiency directly. Table 5(b) compares variants of the policy objective in Eq. (4). The fidelity-only objective yields the highest ABF but substantially degrades ATT. The performance-only objective improves ATT metrics but reduces ABF. The joint objective provides a compromise between the two single-objective variants: it achieves higher ABF than performance-only evaluation and better ATT outcomes than fidelity-only evaluation.

5 Conclusion

We present PPL-TSC, a teacher-guided framework for learning human-readable programmatic student policies from DRL-based traffic signal controllers. Across diverse scenarios and teacher families,

PPL-TSC achieves competitive closed-loop performance while maintaining moderate behavioral fidelity to its teachers. The learned policies show strong transfer performance under traffic-flow shifts and cross-scenario deployment, and substantially reduce storage and computational costs relative to neural teachers. Future work will extend PPL-TSC to broader TSC settings and evaluate its robustness in more realistic large-scale deployments.

Acknowledgments

This work was supported by the Mainland-Hong Kong Joint Funding Scheme of the Innovation and Technology Commission (Grant No. MHP/038/23) and the General Research Fund of the Research Grants Council of the Hong Kong Special Administrative Region, China (Grant No. 15224423). The authors declare no competing interests.

References

[1] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, 2019.

[2] Kan Wu, Jianrong Ding, Jingli Lin, Guanjie Zheng, Yi Sun, Jie Fang, Tu Xu, Yongdong Zhu, and Baojing Gu. Big-data empowered traffic signal control could reduce urban carbon emission. *Nature Communications*, 16(1):2013, 2025.

[3] Fo Vo Webster. Traffic signal settings. Technical report, 1958.

[4] John DC Little. The synchronization of traffic signals by mixed-integer linear programming. *Operations Research*, 14(4):568–594, 1966.

[5] John DC Little, Mark D Kelson, and Nathan H Gartner. MAXBAND: A versatile program for setting signals on arteries and triangular networks. 1981.

[6] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. Self-organizing traffic lights: A realistic simulation. In *Advances in Applied Self-Organizing Systems*, pages 41–50. Springer, 2007.

[7] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pages 27–66. Springer, 2013.

[8] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. SCOOT-a traffic responsive method of coordinating signals. Technical report, 1981.

[9] PR Lowrie. SCATS, Sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.

[10] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18, 2021.

[11] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[12] 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 Computing Surveys (CSUR)*, 50(3):1–38, 2017.

[13] Feng Xiao, Jiaming Lu, Lu Li, Wenwen Tu, and Chaojing Li. Advances in reinforcement learning for traffic signal control: a review of recent progress. *Intelligent Transportation Infrastructure*, 4:liaf009, 2025.

[14] 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. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, 2019.

[15] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972, 2019.

[16] Yuanyang Zhu, Xiao Yin, and Chunlin Chen. Extracting decision tree from trained deep reinforcement learning in traffic signal control. *IEEE Transactions on Computational Social Systems*, 10(4):1997–2007, 2022.

[17] Yin Gu, Kai Zhang, Qi Liu, Weibo Gao, Longfei Li, and Jun Zhou. π-Light: Programmatic interpretable reinforcement learning for resource-limited traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21107–21115, 2024.

[18] Sachin Chauhan, Kashish Bansal, and Rijurekha Sen. EcoLight: Intersection control in developing regions under extreme budget and network constraints. *Advances in Neural Information Processing Systems*, 33:13027–13037, 2020.

[19] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421, 2020.

[20] Dong Xing, Qian Zheng, Qianhui Liu, and Gang Pan. TinyLight: Adaptive traffic signal control on devices with extremely limited resources. *arXiv preprint arXiv:2205.00427*, 2022.

[21] Xiao-Cheng Liao, Yi Mei, and Mengjie Zhang. GPLight+: a genetic programming method for learning symmetric traffic signal control policy. *IEEE Transactions on Evolutionary Computation*, 2025.

[22] Xiao-Cheng Liao, Yi Mei, and Mengjie Zhang. SymLight: Exploring interpretable and deployable symbolic policies for traffic signal control. *arXiv preprint arXiv:2511.05790*, 2025.

[23] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[24] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, volume 34. Stanford University, Department of Computer Science Stanford, CA, 1990.

[25] Mohammad Aslani, Mohammad Saadi Mesgari, and Marco Wiering. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies*, 85:732–752, 2017.

[26] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Time critic policy gradient methods for traffic signal control in complex and congested scenarios. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1654–1664, 2019.

[27] 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.

[28] 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.

[29] James Ault, Josiah P Hanna, and Guni Sharon. Learning an interpretable traffic signal control policy. *arXiv preprint arXiv:1912.11023*, 2019.

[30] Milad Taleby Ahvanooey, Qianmu Li, Ming Wu, and Shuo Wang. A survey of genetic programming and its applications. *KSII Trans. Internet Inf. Syst.*, 13(4):1765–1794, 2019.

[31] Lindong Xie, Yang Zhang, Zhixian Tang, Edward Chung, Genghui Li, and Zhenkun Wang. Co-evolution of large language models and configuration strategies to enhance surrogate-assisted evolutionary algorithm. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 3321–3332, 2025.

[32] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[33] Sathvik Joel, Jie Wu, and Fatemeh Fard. A survey on LLM-based code generation for low-resource and domain-specific programming languages. *ACM Transactions on Software Engineering and Methodology*, 2024.

[34] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[35] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The World Wide Web Conference*, pages 3620–3624. Association for Computing Machinery, 2019. doi: 10.1145/3308558.3314139.

[36] Hao Mei, Xiaoliang Lei, Longchao Da, Bin Shi, and Hua Wei. LibSignal: an open library for traffic signal control. *Machine Learning*, 113(8):5235–5271, 2024.

A Appendix

A.1 Preliminaries

A.1.1 Term Definitions

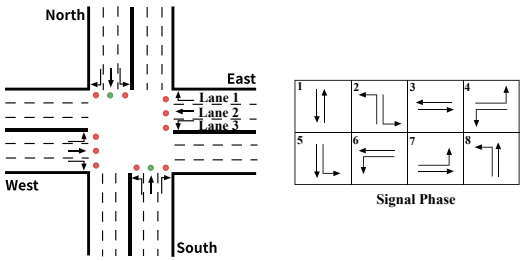


Figure A.1: Illustration of a four-approach signalized intersection with eight signal phases. Phase 1 is shown as the active phase, permitting one north-to-south through movement and one south-to-north through movement.

We use a *single signalized intersection* as an illustrative example to introduce several basic concepts in traffic signal control.

1) Intersection, Approaches, and Lanes. As shown in Fig. A.1, the intersection consists of four *approaches* corresponding to the cardinal directions: north, east, south, and west. Each approach contains multiple *incoming lanes*, where vehicles queue before crossing the stop line. In practice, lanes are assigned to different turning purposes, including *left-turn*, *through*, and *right-turn* lanes. Vehicles are discharged from these lanes when the active signal phase permits the corresponding movements.

2) Traffic Movements. A *traffic movement* represents the trajectory of vehicles from an incoming lane to an outgoing lane. As illustrated by the arrows in Fig. A.1, typical movements include *left-turn*, *through*, and *right-turn* maneuvers. Different movements may conflict if their trajectories intersect within the intersection area, and must therefore be coordinated to ensure safety and efficiency.

3) Phases. A *phase* is a predefined set of *non-conflicting traffic movements* that are allowed to proceed simultaneously. In the example in Fig. A.1, eight signal phases are defined to coordinate different

movements. During a given phase, vehicles associated with permitted movements receive a green signal, while conflicting movements remain stopped.

4) Phase Switching. *Phase switching* refers to the transition from one signal phase to another over time. For example, Fig. A.1 shows Phase 1 as the active phase, permitting the north-to-south and south-to-north through movements. At decision step t, intersection i observes a local traffic state $\mathbf{s}_t^{(i)}$ and chooses a phase $p_t^{(i)}$ from a feasible candidate set $\mathcal{P}_t^{(i)}$, which may contain all phases or only the current phase and its designated successor. The objective is to choose which phase to activate, or whether to keep or switch the current phase, to improve traffic efficiency.

5) Phase-Level Features. Each candidate phase p is associated with a set of *phase-level features* extracted from the current traffic state. Let $\mathcal{L}_p$ denote the set of incoming lanes associated with phase p, and let $\mathcal{M}_p$ denote the set of movements permitted by phase p. For each lane $\ell \in \mathcal{L}_p$ or movement $m \in \mathcal{M}_p$, base measurements such as queue length, waiting time, and vehicle count are available. We construct phase-level features by first mapping lane- or lane-link-level measurements to movement directions, such as north-to-south or east-to-west through movements, and then aggregating them over the movements served by phase p. This yields phase-level quantities such as pressure, vehicle counts, and waiting-related metrics. We further compute statistical descriptors, including $\{\min, \max, \mathrm{mean}, \mathrm{std}\}$, to capture distributional characteristics for normalization.

All aggregated and statistical features, together with the phase identifier, are concatenated to form the phase-level feature vector $\mathbf{x}_{p,t}^{(i)}$ for candidate phase p at intersection i and decision step t. This vector serves as the input to the utility function $f(\cdot)$.

A.1.2 DRL-based Traffic Signal Control Methods

Deep reinforcement learning (DRL) has been widely adopted for traffic signal control (TSC) due to its ability to learn adaptive control policies directly from traffic data. In DRL-based TSC, the signal control problem is typically formulated as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{K}, r, \gamma)$, where r denotes the reward function.

The *state* $s_t \in \mathcal{S}$ represents the current traffic condition at time step t, which may include features such as queue lengths, vehicle counts, or waiting times on incoming lanes. The *action* $a_t \in \mathcal{A}$ corresponds to the control decision taken by the agent, typically selecting the next signal phase or determining whether to switch the current phase.

After executing an action, the traffic environment transitions to a new state s_{t+1} according to the transition kernel $\mathcal{K}(s_{t+1} \mid s_t, a_t)$. The agent then receives a scalar *reward* r_t, obtained from r, which reflects the quality of the control decision, such as average travel time.

The objective of DRL-based traffic signal control is to learn a policy $\pi(a \mid s)$ that maximizes the expected discounted return over time:

$$\max_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right],$$

where $\gamma \in (0, 1)$ is the discount factor and T is the decision horizon. The learned policy enables the traffic signal controller to adaptively select signal phases based on real-time traffic conditions.

A.2 Prompt Templates

This subsection provides representative LLM prompt templates for policy generation and repair in PPL-TSC. Each template follows the structure $\mathcal{T}(\mathrm{Task}, \mathrm{Ctx}, \mathrm{Cons}, \mathrm{Out})$: task specifies the generation objective, context provides traffic features or parent PUFs, constraints encode the syntactic, mathematical, and semantic requirements used to guide generation, and output asks for an executable candidate PUF with a brief rationale. Fig. A.2 illustrates the initialization prompt, evolution prompt, and repair prompt.

A.3 Details of Syntactic, Mathematical, and Semantic Constraints

During policy review, each PUF produced by policy generation is explicitly checked against the same constraints encoded in the generation prompt before calibration. To ensure that generated utility

 # Prompt for Initialization

Task: Discover a student utility function that approximates the teacher model's decision behavior. The function ***Compute_utility(state, phase_id)*** assigns a utility score to each candidate phase; the phase with the highest score is activated. Prioritize diverse and expressive function structures during initialization.

Context: The available input features are ***{Teacher's state variables}***.

Constrains: The implementation must satisfy on ***{Syntactic correctness}***, ***{Mathematical validity}***, and ***{Semantic consistency}***.

Output: Provide the complete Python function ***Compute_utility(state, phase_id) that returns a float utility value***, followed by a brief explanation (2–3 sentences) describing the design rationale.

 # Prompt for Evolution

Task: Evolve an improved offspring utility function from the given parent. The function ***Compute_utility(state, phase_id)*** assigns a utility score to each candidate phase; the phase with the highest score is activated. The offspring must be structurally distinct from the parent and combine raw features meaningfully to avoid trivial heuristics.

Context: The parent function is ***{Parent}***. Evaluation information is provided in ***{Eval info}***. The available input features are ***{Teacher's state variables}***.

Requirements: Same as in the initialization prompt.

Output: Same as in the initialization prompt.

 # Prompt for Repair

Task: Repair an invalid utility function that violates specified constraints. The function ***Compute_utility(state, phase_id)*** assigns a utility score to each candidate phase; the phase with the highest score is activated. The repaired function must preserve the original behavioral intent as much as possible while strictly satisfying all constraints and eliminating any invalid operations.

Context: The function to be repaired is ***{Function}***. The identified constraint violations are detailed in ***{Cons info}***. The available input features are ***{Teacher's state variables}***.

Requirements: Same as in the initialization prompt.

Output: Same as in the initialization prompt.

Figure A.2: Illustration of the initialization, evolution, and repair prompt templates used for LLM-based policy generation and revision.

functions are executable, numerically stable, and consistent with traffic-control principles, we enforce three families of validation checks: syntactic, mathematical, and semantic. An overview is provided in Table A.1.

Table A.1: Constraint validation checks.

Family	Check	Description
Syntactic	Parseability	The generated function must have valid Python syntax and compile successfully.
	Executability	The generated function must be loadable and callable at runtime.
	Structural validity	The generated function must have a valid structure and return a value on every execution path.
Mathematical	Finite output	Output must be a finite real number (no `NaN` or infinite values).
	Domain safety	No invalid operations (e.g., division by zero, invalid logarithms, or square roots of negative inputs).
	Runtime robustness	No exceptions under real teacher states and extreme input scenarios (e.g., all-zero, all-max, mixed, single-feature extremes).
Semantic	Feature legality	Only allowed traffic features are accessed.
	Demand monotonicity	Phase utility or switching preference varies monotonically with demand according to the action interface.
	Zero-demand minimum	With other inputs fixed, setting non-negative demand features to zero must not increase phase utility.
	Active-phase protection	With demand fixed, the active-phase indicator must not reduce phase utility.

Syntactic checks. These checks ensure that the generated code is executable in the target pipeline. Each candidate must first pass Python compilation, including a valid function body and valid syntax. It is then loaded into the runtime environment via `exec()`, where the required entry point `compute_utility` must exist and be callable. Malformed structures, such as invalid function skeletons or missing return statements, are rejected before acceptance.

Mathematical checks. These checks ensure numerical validity and stability. All outputs must be finite; any `NaN`, `inf`, or `-inf` results in rejection. All operations must respect valid input domains, such as non-zero denominators, positive inputs to logarithms, and non-negative inputs to square roots. The function must also avoid extreme intermediate values that could cause overflow or numerical instability. To test robustness, we evaluate each candidate on phase-level inputs constructed under the following scenarios:

- **Real teacher states**: all samples $(\mathbf{s}_t^{(i)}, p_t^{\mathrm{Tch},(i)}) \in \mathcal{D}$, to ensure stability under the observed data distribution;
- **All-zero**: all traffic-measurement features set to zero, with phase indicators kept fixed;

- **All-max**: all traffic-measurement features set to their maximum observed values in $\mathcal{D}$, with phase indicators kept fixed;
- **Mixed-half**: half of the traffic-measurement features set to zero and the other half set to their maximum values;
- **Single-feature-max**: each traffic-measurement feature individually set to its maximum observed value, with all remaining features kept at their original sampled values.

Semantic checks. The feature whitelist covers the base traffic measurements `pressure`, `num_vehicle`, `num_waiting_vehicle`, `waiting_time`, and `sum_waiting_time`, together with `phase_id` and `current_phase`. Candidates may use the defined aggregation and normalization operations on these measurements to construct phase-level features. Access to undefined or disallowed inputs results in rejection.

Demand monotonicity follows the controller's action interface. For phase-local scoring, utility must be non-decreasing in the demand served by the candidate phase, with other inputs fixed. This applies to phase-selection policies and the original keep/switch student that scores the current and next phases separately. We evaluate this property on sampled real teacher states using multiple scaling factors. For each demand-related feature

$$x \in \{\texttt{pressure},\ \texttt{num_vehicle},\ \texttt{num_waiting_vehicle},\ \texttt{waiting_time},\ \texttt{sum_waiting_time}\},$$

we generate perturbed states with scaling factors $\alpha \in \{0.5, 0.8, 1.0, 1.2, 1.5\}$:

$$x_\alpha = \alpha \cdot x_{\text{base}},$$

while keeping all other features unchanged.

For a candidate utility function f, we evaluate the corresponding phase-level feature vector $\mathbf{x}^{(i)}_{p,t,\alpha}$ for each α. We sort the perturbed feature values in ascending numerical order and check that utility is non-decreasing in that order:

$$f(\mathbf{x}^{(i)}_{p,t,\alpha_k}) \geq f(\mathbf{x}^{(i)}_{p,t,\alpha_j}), \quad \forall x_{\alpha_k} > x_{\alpha_j}.$$

Ordering by feature value rather than scaling factor accommodates signed features such as movement pressure. This check assesses monotonicity at the tested inputs.

For the aligned keep/switch variant in Appendix A.7.1, we instead check the relative preference for switching. Let g and r denote green- and red-side demand, and define the switching margin as

$$m(g, r) = f_{\text{switch}}(g, r) - f_{\text{keep}}(g, r).$$

On sampled teacher states, we vary each demand input using the scaling factors above and hold the other inputs fixed. The margin must be non-increasing in g and non-decreasing in r: greater green-side demand favors keeping the current phase, and greater red-side demand favors switching. These checks assess monotonicity at the tested inputs.

Zero-demand minimum. With other inputs fixed, setting non-negative demand features to zero must not increase phase utility. For aligned keep/switch policies, we use the switching-margin checks defined above.

Active-phase protection compares otherwise identical candidate inputs that differ only in whether the phase is currently active. The check verifies that the active-phase indicator is not used to penalize the current phase under comparable demand conditions. This discourages excessive phase switching and reflects the practical need for short-term signal stability.

Decision rule. Candidates that fail any check are returned to the LLM with diagnostic feedback and reviewed again after repair. Only candidates that pass all checks proceed to calibration and evaluation.

A.4 Method Complexity Analysis

We analyze the complexity of PPL-TSC in terms of both search-time cost and deployment-time inference cost. Let $|\mathcal{D}| = \sum_{i=1}^{M} N_i$ denote the total number of rollout samples, $B_i = \max_t |\mathcal{P}_t^{(i)}|$ the

maximum feasible phase-set size at intersection i (all phases for phase-selection controllers and 2 for keep/switch controllers), C_f the cost of evaluating $f(\cdot)$ on one phase-level feature vector, and C_{LLM} the cost of one LLM prompt-response interaction. We further use $G_{\max}$ for the maximum number of generations, N_{pop} for the population size, R for the number of review inputs per candidate, T_i for the number of CMA-ES objective evaluations at intersection i, $\bar{B}$ for the average candidate phase-set size, T_{sim} for the number of simulator control steps, C_{env} for one simulator transition, and K_{rep} for the average number of LLM generations per accepted candidate, including repair calls.

Table A.2: Complexity of the main steps in PPL-TSC.

Component	Complexity
Policy generation	$O(G_{\max} N_{\mathrm{pop}} K_{\mathrm{rep}} C_{\mathrm{LLM}})$
Policy review	$O(RC_f)$ per reviewed candidate
Policy calibration	$O\Big(\sum_{i=1}^{M} T_i N_i B_i C_f\Big)$ per candidate
Policy evaluation	$O(1)$ for cached agreement; $O\big(T_{\mathrm{sim}}(M\bar{B}C_f + C_{\mathrm{env}})\big)$ for simulation
Deployment	$O(B_i C_f)$ per decision step

Table A.2 summarizes the costs. Across the search, generation requires $O(G_{\max} N_{\mathrm{pop}} K_{\mathrm{rep}} C_{\mathrm{LLM}})$, and review requires $O(G_{\max} N_{\mathrm{pop}} K_{\mathrm{rep}} R C_f)$, excluding lightweight parsing overhead. LLM latency depends on prompt and response lengths and repair frequency.

Calibration and evaluation. One CMA-ES objective evaluation scores up to B_i phases for each of N_i samples, costing $O(N_i B_i C_f)$. With T_i objective evaluations per intersection, full-search calibration costs $O\Big(G_{\max} N_{\mathrm{pop}} \sum_{i=1}^{M} T_i N_i B_i C_f\Big)$. The final fidelity value on $\mathcal{D}$ is cached, so its additional evaluation cost is $O(1)$ per candidate. Closed-loop simulation costs $O\big(G_{\max} N_{\mathrm{pop}} T_{\mathrm{sim}}(M\bar{B}C_f + C_{\mathrm{env}})\big)$ over the search. Simulation is typically the dominant cost in practice, followed by calibration.

Deployment. Scoring at most B_i phases costs $O(B_i C_f)$ per intersection-level decision. The phase-scoring function is shared; each intersection stores its own phase biases.

A.5 Limitations

First, PPL-TSC depends on teacher rollouts for behavioral supervision, so teacher quality and state coverage influence the learned policy. Students may inherit undesirable teacher behavior or perform poorly in states underrepresented in the rollouts. Although the joint objective includes closed-loop traffic performance and students can outperform their teachers, this feedback does not eliminate the limitations of teacher-derived data. Future work could expand rollout coverage through student-driven data collection and further refine programmatic policies through closed-loop optimization.

Second, PPL-TSC learns students within a fixed PUF-based policy class. While this class works well for several teacher families, it may be insufficient for teachers whose behavior depends on richer state representations or more complex feature interactions. Future work could explore richer programmatic policy classes that preserve compactness while covering a broader range of teacher behaviors. The verifier checks generated functions on sampled traffic states and constructed test inputs. A function that passes these checks may still behave undesirably in unseen states or different road networks. The predefined semantic checks may not capture all aspects of reasonable control behavior. Applying them to different teachers requires considering the input features, action meanings, and their relationships, and may require adapting the checks accordingly.

Third, the program structure is discovered through LLM-guided generation and refinement, making the final policy dependent on the model's ability to propose valid and useful candidate expressions. Although LLM-based generation achieves higher semantic-constraint pass rates than GA, MCTS, and random search under a matched time budget, the resulting policy structures may still be suboptimal. Future work could mitigate this dependence by integrating LLM-based synthesis with more systematic local search, stronger verification, and automated repair mechanisms.

Table A.3: Scenario-level results for CoLight. Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	96.00±5.20	**94.31±3.45**	98.63±4.76	**0.66±0.06**	0.60±0.06
Hangzhou2	**122.47±4.06**	123.52±2.71	127.24±3.66	**0.68±0.08**	0.62±0.09
Hangzhou3	**80.08±1.87**	82.39±1.15	84.65±2.37	**0.73±0.04**	0.71±0.04
Atlanta	**311.00±24.27**	311.36±58.31	325.75±65.71	**0.73±0.17**	0.65±0.14
Jinan	1109.23±311.54	937.99±433.65	**358.95±27.54**	**0.60±0.31**	0.15±0.04
Los Angeles	**846.30±16.21**	853.04±13.30	852.89±12.48	**0.95±0.05**	0.90±0.09
Manhattan	1668.92±90.51	607.53±104.25	**502.55±85.26**	**0.62±0.05**	0.56±0.05

A.6 Broader Impacts

PPL-TSC may improve the practical deployability of DRL-based traffic signal control by distilling neural controllers into compact programmatic policies. If used responsibly, such policies could help reduce traffic delay and improve control efficiency, with potential downstream benefits for fuel consumption and emissions. The compact programmatic representation may also make controller behavior easier to inspect, maintain, and adapt in resource-constrained deployments.

At the same time, traffic signal control is a safety-critical application, and results on standard simulators do not guarantee safe real-world deployment. Because PPL-TSC is teacher-guided, it may also inherit biases or failure modes from the teacher policy, which could affect different traffic participants unevenly. Real-world use therefore requires careful validation, human oversight, and additional safeguards before deployment.

A.7 Licenses for Existing Assets

This work uses existing simulators, traffic scenarios, controller designs, and LLM services only for research evaluation. We credit the corresponding original sources throughout the paper and summarize the asset usage and licensing information below.

- **Traffic simulator.** We conduct closed-loop evaluation in CityFlow [35]. CityFlow is publicly released under the Apache License 2.0 according to the CityFlow project repository and documentation.
- **Benchmark scenarios and traffic data.** The traffic scenarios used in our experiments are taken from the LibSignal benchmark [36] and related CityFlow-format traffic-signal-control benchmarks. LibSignal is released as an open-source research library.
- **Teacher controller designs.** We evaluate against CoLight [14], FRAP [15], MPLight [19], EcoLight [18], and TinyLight [20]. These methods are credited through their original publications. Any publicly available reference implementations or benchmark implementations used to reproduce teacher behavior are used only under their associated repository licenses and terms; this manuscript does not redistribute third-party controller code.
- **LLM services.** PPL-TSC uses LLM backbones such as Gemini, Qwen, Claude, GPT, Grok, and MiniMax for candidate policy generation and repair through their official access channels. These services are used according to their respective provider terms of service. We do not redistribute model weights or proprietary service outputs as standalone assets.
- **Generated artifacts.** The tables, plots, learned programmatic policies, and analysis outputs reported in this paper are generated by the authors from the experimental pipeline.

A.7.1 Diagnosis of EcoLight (native) and TinyLight

We further investigate the weaker in-domain results obtained when PPL-TSC is applied to EcoLight (native) and TinyLight. We conduct controlled follow-up experiments over 35 tasks (seven scenarios and five traffic flows), changing only the suspected factor while keeping all other training and evaluation settings fixed. Table A.9 reports average behavioral fidelity (ABF) and relative ATT improvement, where higher values are better.

Table A.4: Scenario-level results for FRAP. Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	128.94±31.88	109.79±19.76	**99.04±5.85**	**0.69±0.16**	0.61±0.23
Hangzhou2	150.24±56.75	128.03±14.62	**122.65±1.15**	**0.62±0.27**	0.59±0.31
Hangzhou3	132.47±110.34	134.01±115.03	**85.05±2.30**	**0.72±0.05**	0.58±0.30
Atlanta	**258.77±25.82**	333.29±31.27	364.11±13.07	**0.75±0.10**	0.66±0.11
Jinan	**307.61±15.17**	312.17±2.58	316.78±1.92	**0.62±0.02**	0.59±0.03
Los Angeles	714.95±74.80	**442.73±69.32**	466.25±74.95	**0.47±0.14**	0.45±0.10
Manhattan	639.19±423.21	**232.81±31.61**	235.63±23.04	**0.56±0.10**	0.56±0.11

Table A.5: Scenario-level results for MPLight. Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	100.83±3.36	**97.33±6.23**	100.04±1.62	**0.76±0.07**	0.72±0.07
Hangzhou2	**121.81±1.32**	121.97±1.33	122.89±2.44	**0.74±0.10**	0.71±0.10
Hangzhou3	82.72±2.73	**82.48±2.10**	83.51±2.42	**0.89±0.04**	0.88±0.04
Atlanta	249.45±15.03	**236.32±5.22**	241.55±4.61	**0.79±0.11**	0.71±0.15
Jinan	**297.07±7.04**	307.66±4.84	314.77±6.00	**0.72±0.10**	0.68±0.08
Los Angeles	588.16±99.49	579.86±61.50	**544.45±166.22**	**0.58±0.04**	0.56±0.15
Manhattan	**195.42±4.14**	197.74±6.87	198.95±3.51	**0.61±0.06**	0.60±0.06

Table A.6: Scenario-level results for EcoLight (native) under its keep/switch action setting. Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	205.88±37.67	**183.19±25.05**	194.82±15.38	**0.81±0.05**	0.77±0.05
Hangzhou2	135.03±0.54	**134.63±2.11**	135.92±2.79	**0.91±0.04**	0.90±0.04
Hangzhou3	98.05±1.77	**96.31±1.24**	97.23±1.52	**0.93±0.02**	0.92±0.03
Atlanta	**311.47±32.78**	329.75±20.98	380.94±38.96	**0.79±0.11**	0.77±0.05
Jinan	**378.74±3.97**	765.00±314.15	736.17±166.60	**0.84±0.01**	0.81±0.02
Los Angeles	**640.13±5.03**	674.28±108.16	785.57±116.09	0.80±0.03	**0.82±0.03**
Manhattan	**1066.33±66.77**	1325.00±16.36	1340.00±40.56	**0.80±0.01**	0.79±0.01

Table A.7: Scenario-level results for EcoLight (phase-selection). Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	205.88±42.11	126.10±31.32	**115.29±26.62**	**0.43±0.09**	0.29±0.06
Hangzhou2	135.03±0.60	127.90±3.37	**124.61±4.31**	**0.49±0.03**	0.38±0.08
Hangzhou3	98.05±1.98	**83.73±2.59**	86.99±1.66	**0.65±0.03**	0.62±0.02
Atlanta	**311.47±36.65**	358.34±55.18	381.22±11.42	**0.56±0.05**	0.48±0.07
Jinan	378.74±4.44	324.74±9.79	**320.34±3.63**	**0.39±0.02**	0.28±0.03
Los Angeles	640.13±5.62	**485.39±35.87**	530.55±98.75	**0.44±0.05**	0.41±0.06
Manhattan	1066.33±74.65	249.80±29.01	**241.49±33.82**	**0.42±0.03**	0.36±0.02

Table A.8: Scenario-level results for TinyLight. Results are averaged over the five traffic flows in each scenario.

Scenario	ATT ↓			ABF ↑	
	Teacher	Tailored	Unified	Tailored	Unified
Hangzhou1	98.76±5.81	**97.11±6.78**	105.11±15.31	**0.67±0.07**	0.57±0.06
Hangzhou2	121.84±1.20	**119.83±0.78**	123.16±5.12	**0.66±0.08**	0.57±0.09
Hangzhou3	**81.17±1.22**	81.53±1.11	83.86±3.85	**0.82±0.08**	0.70±0.09
Atlanta	**278.47±12.85**	359.23±54.13	375.47±30.47	**0.69±0.05**	0.65±0.07
Jinan	**299.54±2.83**	324.91±89.72	397.69±128.98	0.31±0.11	**0.42±0.10**
Los Angeles	**450.31±43.75**	517.40±147.30	619.48±164.61	**0.73±0.05**	0.72±0.06
Manhattan	762.03±532.47	**562.91±664.60**	1163.20±758.01	0.58±0.11	**0.63±0.03**

Table A.9: Controlled diagnosis of EcoLight (native) and TinyLight. For EcoLight (native), Original inputs score phases separately using phase-local demand, whereas Aligned inputs include joint green- and red-side demand. The search budget is reported as (population size, number of generations).

Teacher	Configuration	ABF ↑	ΔATT (%) ↑
EcoLight (native)	Original inputs	0.84	-18.05
	Aligned inputs	**0.94**	**2.71**
TinyLight	Search budget $(5, 5)$	0.64	-8.43
	Search budget $(10, 10)$	**0.71**	**1.51**

Table A.10: Supplementary generalization results for CoLight under same-scenario, different-flow transfer. Cross-scenario results are not available in the current benchmark. Rows correspond to the source scenario.

Source Scenario	Teacher ATT ↓	PPL-CoLight ATT ↓	ΔATT (%) ↑	ATT wins
Hangzhou1	**96.74±4.70**	99.36±2.41	-2.96±5.72	4/20
Hangzhou2	122.85±3.26	**122.36±1.07**	0.33±2.61	6/20
Hangzhou3	**81.04±1.28**	86.72±1.18	-7.04±2.17	0/20
Atlanta	**313.68±20.93**	402.94±28.94	-28.56±6.21	0/20
Jinan	1114.97±254.41	**751.00±474.67**	35.48±32.05	15/20
Los Angeles	846.71±16.73	**636.83±188.32**	24.57±23.06	16/20
Manhattan	1660.76±89.71	**202.42±8.44**	87.79±0.66	20/20

EcoLight (native). EcoLight (native) makes keep/switch decisions using demand from both green and red movements, but the original student scores the current and next phases separately using only the demand each phase serves. An incorrect switch changes the current phase and the decisions available at subsequent steps, allowing errors to accumulate during closed-loop control. Using aligned inputs that provide joint green- and red-side demand, with all other settings unchanged, increases ABF from 0.84 to 0.94 and improves mean ΔATT from -18.05% to $+2.71\%$. These results suggest that the input mismatch is a major contributor to the original performance degradation.

TinyLight. TinyLight uses both phase-level and incoming-lane features, giving the student a broader set of inputs to search over. Increasing the population size and number of generations from $(5, 5)$ to $(10, 10)$, with all other settings unchanged, raises ABF from 0.64 to 0.71 and improves mean ΔATT from -8.43% to $+1.51\%$. These results suggest that the default budget limits search quality for TinyLight and that a larger budget can improve both fidelity and control performance.

Table A.11: Supplementary generalization results for FRAP. Rows correspond to the source scenario.

Same-Scenario, Different-Flow

Source Scenario	Teacher ATT ↓	PPL-FRAP ATT ↓	ΔATT (%) ↑	ATT wins
Hangzhou1	126.50±27.48	**100.03±2.89**	18.10±13.19	20/20
Hangzhou2	142.02±40.37	**122.95±1.49**	8.99±15.76	18/20
Hangzhou3	134.77±101.22	**86.61±1.07**	12.54±30.97	7/20
Atlanta	**254.60±16.28**	389.20±25.79	-53.10±8.86	0/20
Jinan	**307.89±13.30**	319.69±1.61	-4.00±4.01	4/20
Los Angeles	703.20±77.96	**409.74±8.46**	40.92±7.48	20/20
Manhattan	675.21±396.23	**196.57±4.76**	59.44±20.08	20/20

Cross-Scenario

Source Scenario	Teacher ATT ↓	PPL-FRAP ATT ↓	ΔATT (%) ↑	ATT wins
Hangzhou1	764.29±635.23	**251.56±124.11**	37.06±37.37	114/150
Hangzhou2	803.49±584.30	**252.88±134.62**	50.01±33.28	129/150
Hangzhou3	914.93±500.84	**253.10±119.67**	62.16±30.44	136/150
Atlanta	1342.99±385.91	**222.45±160.33**	79.14±20.88	147/150
Jinan	717.59±589.02	**215.18±130.26**	40.03±44.67	104/150
Los Angeles	1128.05±548.72	**199.71±111.34**	66.97±36.88	138/150
Manhattan	688.96±366.22	**252.73±171.47**	44.18±46.45	119/150

Table A.12: Supplementary generalization results for MPLight. Rows correspond to the source scenario.

Same-Scenario, Different-Flow				
Source Scenario	**Teacher ATT ↓**	**PPL-MPLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	100.72±5.51	**98.96±7.21**	1.80±3.13	15/20
Hangzhou2	122.42±1.72	**122.19±1.44**	0.18±0.91	11/20
Hangzhou3	**82.86±1.23**	84.43±1.50	-1.89±1.48	2/20
Atlanta	251.50±17.14	**243.17±11.01**	2.87±7.81	10/20
Jinan	**297.09±6.17**	310.32±1.13	-4.50±2.12	1/20
Los Angeles	591.77±55.95	**537.51±156.13**	8.35±27.90	16/20
Manhattan	**195.72±2.98**	201.25±7.19	-2.80±2.41	0/20
Cross-Scenario				
Source Scenario	**Teacher ATT ↓**	**PPL-MPLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	792.01±647.69	**256.48±155.39**	40.67±34.82	115/150
Hangzhou2	798.34±661.12	**233.04±126.91**	44.34±33.91	124/150
Hangzhou3	843.40±531.29	**240.28±120.13**	58.53±27.53	150/150
Atlanta	952.78±522.35	**215.22±133.92**	60.49±35.90	144/150
Jinan	564.21±438.45	**204.58±129.43**	38.25±35.77	141/150
Los Angeles	981.66±481.97	**185.12±91.26**	68.68±29.09	150/150
Manhattan	288.98±240.39	**231.98±161.99**	11.29±20.26	122/150

Table A.13: Supplementary generalization results for EcoLight (native) under its keep/switch action setting. Rows correspond to the source scenario.

Same-Scenario, Different-Flow				
Source Scenario	**Teacher ATT ↓**	**PPL-EcoLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	209.20±35.75	**98.73±3.15**	51.52±7.63	20/20
Hangzhou2	134.42±1.02	**122.49±1.27**	8.87±1.15	20/20
Hangzhou3	97.86±3.08	**86.62±1.21**	11.40±2.95	20/20
Atlanta	**312.77±31.81**	373.94±5.14	-20.60±10.34	2/20
Jinan	379.52±6.98	**320.75±1.82**	15.46±1.39	20/20
Los Angeles	643.40±6.78	**408.17±9.61**	36.55±1.79	20/20
Manhattan	1033.27±46.12	**328.85±120.43**	68.38±11.03	20/20
Cross-Scenario				
Source Scenario	**Teacher ATT ↓**	**PPL-EcoLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	773.79±599.72	**251.53±124.29**	43.20±33.53	150/150
Hangzhou2	777.06±599.57	**247.81±127.96**	47.92±30.17	150/150
Hangzhou3	791.48±576.36	**253.73±122.18**	50.35±30.28	150/150
Atlanta	750.53±543.70	**205.91±121.93**	60.80±25.66	150/150
Jinan	537.76±534.79	**228.62±160.51**	35.39±30.98	139/150
Los Angeles	957.33±626.34	**200.58±111.70**	60.95±36.34	125/150
Manhattan	**386.32±174.02**	533.93±362.09	-64.69±157.99	74/150

Table A.14: Supplementary generalization results for EcoLight (phase-selection). Rows correspond to the source scenario.

Same-Scenario, Different-Flow				
Source Scenario	**Teacher ATT ↓**	**PPL-EcoLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	209.20±35.75	**107.25±12.43**	47.05±11.14	20/20
Hangzhou2	134.42±1.02	**123.58±1.66**	8.06±1.43	20/20
Hangzhou3	97.86±3.08	**87.07±1.80**	10.92±3.59	20/20
Atlanta	**312.77±31.81**	376.35±1.39	-21.42±10.78	0/20
Jinan	379.52±6.98	**324.88±7.63**	14.34±3.33	20/20
Los Angeles	643.40±6.78	**405.74±9.03**	36.93±1.57	20/20
Manhattan	1033.27±46.12	**201.72±8.84**	80.44±1.20	20/20
Cross-Scenario				
Source Scenario	**Teacher ATT ↓**	**PPL-EcoLight ATT ↓**	**ΔATT (%) ↑**	**ATT wins**
Hangzhou1	773.79±599.72	**335.01±287.54**	33.77±37.04	123/150
Hangzhou2	777.06±599.57	**262.02±151.01**	45.88±31.28	145/150
Hangzhou3	791.48±576.36	**285.34±176.94**	46.14±33.08	140/150
Atlanta	750.53±543.70	**205.71±120.23**	60.67±25.67	150/150
Jinan	537.76±534.79	**220.34±140.70**	36.58±30.25	149/150
Los Angeles	957.33±626.34	**200.30±111.34**	61.00±36.27	125/150
Manhattan	386.32±174.02	**278.49±206.58**	29.90±31.34	122/150

Table A.15: Supplementary generalization results for TinyLight under the same-scenario, different-flow transfer. Cross-scenario results are not available in the current benchmark. Rows correspond to the source scenario.

Source Scenario	Teacher ATT ↓	PPL-TinyLight ATT ↓	△ATT (%) ↑	ATT wins
Hangzhou1	99.85±7.36	**95.99±3.09**	3.56±4.48	16/20
Hangzhou2	121.68±1.44	**119.59±1.27**	1.71±0.96	20/20
Hangzhou3	81.76±1.10	**81.19±1.30**	0.70±1.30	15/20
Atlanta	**276.85±16.05**	320.72±46.04	-16.94±22.24	5/20
Jinan	299.86±3.14	**284.36±1.94**	5.16±1.18	20/20
Los Angeles	**466.54±39.77**	490.47±98.35	-5.02±17.37	10/20
Manhattan	611.26±485.68	**373.68±330.27**	-20.54±126.79	12/20

A.8 Additional Experimental Details

A.8.1 Experimental Setup

Teacher Controllers. We use five representative DRL-based TSC methods as teachers. Most of them map intersection-level traffic observations to a phase-selection decision, while EcoLight uses a native keep/switch action interface. They differ substantially in how they represent traffic state and parameterize the control policy.

CoLight [14]. CoLight is a cooperative multi-intersection RL controller that uses graph attention to model interactions among neighboring intersections. For each intersection, the input includes the current phase and lane-level vehicle counts; these local features are further aggregated with features from adjacent intersections through an attention mechanism. The network outputs a value for each candidate phase, and the next phase is selected by taking the maximum-value action.

FRAP [15]. FRAP models TSC from the perspective of phase competition and explicitly exploits the rotational symmetry of intersections. Its input consists of the current phase together with lane-link-level traffic features, which are reorganized into a phase-competition representation describing the demand served by competing phases. The output is a value for each feasible phase.

MPLight [19]. MPLight extends FRAP to large-scale road networks through parameter sharing across intersections and a pressure-based learning objective. Its input includes the current phase and lane-link-level pressure features, where pressure measures the imbalance between upstream and downstream traffic. Similar to FRAP, the network outputs a value for each candidate phase.

EcoLight [18]. EcoLight is a lightweight RL controller designed for budget- and infrastructure-constrained settings. It uses a compact low-dimensional state derived from the current phase and incoming-lane vehicle counts, and formulates control as a binary decision: keep the current phase or switch to the next phase. Its output is therefore a two-dimensional action-value vector over these two choices. In PPL-TSC, these two actions correspond to $\mathcal{P}_t^{(i)} = \{p_{t,\mathrm{cur}}^{(i)}, \mathrm{next}(p_{t,\mathrm{cur}}^{(i)})\}$. We report this native keep/switch setting as EcoLight (native), and additionally evaluate an EcoLight (phase-selection) variant in which the student directly scores all feasible phases.

TinyLight [20]. TinyLight is a resource-efficient DRL controller tailored for deployment on devices with extremely limited memory and computation. Its input includes compact phase-level and incoming-lane traffic statistics, such as vehicle count, queue length, waiting time, delay, pressure, and the current phase. The network outputs a value for each feasible phase, and the phase with the highest value is selected for execution.

Implementation details. All experiments are conducted on a machine with an Intel Core i7-10700K CPU, 64 GB RAM, and an NVIDIA GeForce RTX 4080 SUPER GPU with 16 GB VRAM. Teacher controllers are implemented in PyTorch and trained for 128 episodes. For programmatic policy search, PPL-TSC uses Gemini 3.1 Flash with a population size $N_{\mathrm{pop}} = 5$ and a maximum of $G_{\max} = 5$ generations, unless otherwise stated. For bias calibration, CMA-ES uses a population size of 20 and a maximum of 20 iterations.

A.8.2 Scenario-Level Results

This subsection provides the detailed scenario-level results corresponding to Table 2. Tables A.3–A.8 report results for CoLight, FRAP, MPLight, EcoLight (native), EcoLight (phase-selection), and

TinyLight. For each teacher family, we report the mean and standard deviation over the five traffic flows in each scenario. Boldface indicates the best value within each scenario row, with lower values preferred for ATT and higher values preferred for ABF. These tables complement the aggregated summary in the main text by showing how the PPL-TSC (Tailored) and PPL-TSC (Unified) students behave across individual traffic scenarios.

A.8.3 Cross-scenario Generalization Details

This subsection provides supplementary transfer results corresponding to Table 3. Tables A.10–A.15 report transfer results for CoLight, FRAP, MPLight, EcoLight (native), EcoLight (phase-selection), and TinyLight. For each teacher family and each source scenario, we report average ATT under same-scenario, different-flow transfer; for teacher families with applicable settings, we additionally report cross-scenario transfer results. We focus on ATT here because the main goal of RQ3 is to assess whether the learned policy preserves traffic-control performance under distribution shift; compared with fidelity-oriented metrics, ATT provides a more direct measure of transfer effectiveness. Here, "ATT wins" counts the number of transfer test cases in which PPL-TSC improves upon or matches the teacher on ATT. For the same-scenario setting, each row aggregates over 20 transfer test cases; for the cross-scenario setting, each row aggregates over 150 transfer test cases when applicable. Cross-scenario transfer is not applicable to CoLight and TinyLight because their action dimensions vary across scenarios, whereas EcoLight, FRAP, and MPLight support both transfer settings.

A.8.4 Additional Out-of-Distribution Evaluation

We further evaluate PPL-TSC under two targeted out-of-distribution (OOD) disturbances: (i) *severe congestion*, where traffic demand increases to $1.5\times$ its original level during 40–60% of an episode, and (ii) *an accident*, simulated by closing all entry lanes of the highest-demand approach during the same period. All policies remain frozen, and each matched teacher–student pair uses the same network, traffic flow, random seed, disturbance, and simulation horizon. Each teacher interface is evaluated over seven scenarios, five traffic flows, and two disturbances, yielding 70 matched cases.

Table A.16: Performance under targeted OOD disturbances. ATT improvement is measured relative to the corresponding teacher; positive values indicate lower student ATT.

Teacher interface	ΔATT (%)	ATT wins
CoLight	0.66	35/70
EcoLight (phase-selection)	2.66	48/70
EcoLight (native)	1.07	30/70
FRAP	1.26	38/70
MPLight	1.19	40/70
TinyLight	-0.15	32/70
Overall	1.12	223/420

Under the two targeted disturbances, five of six teacher interfaces achieve positive mean improvements, with an overall improvement of 1.12% and 223/420 wins. These results show that PPL-TSC remains broadly competitive with its teachers under severe congestion and accident-induced lane closures.

A.8.5 Comparison with Distill-Adapted Programmatic Baselines

We compare PPL-TSC with three representative programmatic policy-search methods: π-Light [17], GPLight+ [21], and SymLight [22]. Because their original formulations learn through environment interaction rather than teacher rollouts, we adapt them into π-Light-Distill, GPLight+-Distill, and SymLight-Distill for a controlled teacher-guided comparison. These adapted baselines use the same MPLight teacher rollouts, phase interface, search budget, simulator settings, and evaluation protocol as PPL-TSC; SymLight-Distill is reimplemented from its paper because public code is unavailable. No method uses target-scenario search, calibration, or teacher data after transfer.

Each scenario contains five traffic flows, with one policy learned per source flow. For the 42 directed cross-scenario pairs, five source policies are evaluated on five target flows, yielding 25 evaluations per pair and 1,050 per method. For the seven same-scenario pairs, each policy is evaluated on the other four flows, yielding 20 evaluations per pair and 140 per method. This produces 1,190 evaluations per method and 4,760 evaluations in total.

Table A.17: Aggregate comparison with distill-adapted programmatic baselines. Cross-scenario ATT is reported as mean ± SD, with the median in parentheses, over the 1,050 off-diagonal evaluations. Best-pair counts and mean ranks use pair-level median ATT over all 49 source–target pairs.

Method	Cross-scenario ATT (s) ↓	Best pairs ↑	Mean rank ↓
PPL-TSC	**232.51 ± 154.20 (217.01)**	**24/49**	**1.94**
π-Light-Distill	305.99 ± 274.68 (230.98)	3/49	3.21
GPLight+-Distill	301.32 ± 286.78 (234.88)	11/49	2.54
SymLight-Distill	275.23 ± 249.27 (220.53)	11/49	2.31

PPL-TSC achieves the lowest cross-scenario mean and median ATT, obtains the best result on 24 of 49 source–target pairs, and has the lowest overall mean rank. Table A.18 provides the complete transfer matrix. Each cell reports median ATT in the order PPL-TSC / π-Light-Distill / GPLight+-Distill / SymLight-Distill; off-diagonal and diagonal cells aggregate 25 cross-scenario and 20 held-out same-scenario evaluations, respectively.

Table A.18: All-to-all transfer comparison with distill-adapted programmatic baselines. Each cell reports median ATT as PPL-TSC / π-Light-Distill / GPLight+-Distill / SymLight-Distill; boldface denotes the lowest value.

Source / Target	Atlanta	Hangzhou1	Hangzhou2	Hangzhou3	Jinan	Los Angeles	Manhattan
Atlanta	239.05 / 236.23 / 236.24 / **235.12**	98.57 / 100.18 / 100.94 / **98.20**	**121.98** / 122.96 / 122.89 / 122.32	**85.06** / 85.96 / 86.08 / 85.35	**310.48** / 972.14 / 451.72 / 327.69	**446.30** / 494.71 / 473.96 / 523.34	**216.61** / 551.18 / 284.36 / 217.70
Hangzhou1	**238.91** / 307.04 / 246.65 / 262.26	98.97 / **97.50** / 100.13 / 101.19	**121.98** / 122.89 / 122.34 / 122.11	85.09 / **84.01** / 85.24 / 84.90	**311.48** / 342.63 / 330.78 / 325.05	**458.47** / 763.85 / 604.76 / 700.44	**217.37** / 245.91 / 256.26 / 229.73
Hangzhou2	**239.32** / 245.45 / 243.67 / 248.47	99.13 / 96.18 / **95.20** / 96.18	122.21 / **122.07** / 122.25 / 122.19	85.33 / 84.66 / **84.35** / 84.66	316.33 / 331.47 / **308.03** / 313.59	454.33 / 487.48 / **453.35** / 490.67	220.93 / 230.42 / **216.49** / 222.03
Hangzhou3	246.89 / 245.41 / **244.26** / 308.25	95.70 / 100.12 / **93.81** / 95.97	122.14 / 122.49 / **121.99** / 122.03	84.43 / 85.07 / 83.86 / **83.56**	316.08 / 321.09 / 309.96 / **303.85**	**470.13** / 484.35 / 542.14 / 763.90	**210.03** / 221.39 / 224.13 / 218.08
Jinan	238.90 / 238.67 / **238.57** / 239.04	99.03 / 98.66 / 98.94 / **98.31**	**121.98** / 122.32 / 122.62 / 122.10	85.09 / 85.51 / 85.51 / **84.82**	310.53 / 310.86 / 309.31 / **309.12**	**445.59** / 460.05 / 483.16 / 472.98	216.53 / 218.87 / 214.51 / **213.95**
Los Angeles	**240.33** / 243.00 / 341.65 / 241.37	**96.87** / 102.88 / 106.42 / 100.10	**122.00** / 123.42 / 125.53 / 123.30	**85.32** / 86.08 / 91.05 / 86.08	**313.00** / 941.63 / 339.10 / 355.16	466.45 / 490.53 / **461.39** / 466.50	217.82 / 226.46 / **215.96** / 221.52
Manhattan	**238.69** / 262.08 / 244.86 / 241.96	98.85 / 98.66 / 97.90 / **96.87**	122.22 / 122.89 / 122.32 / **122.00**	**84.96** / 86.08 / 85.33 / 85.36	**310.95** / 340.81 / 328.71 / 325.27	**453.28** / 666.24 / 463.61 / 482.43	213.92 / 216.89 / 209.93 / **207.39**

A.8.6 Readability Examples Across Teacher Families

Table A.19 extends the MPLight case study in Section 4.4 to all five teacher families and both EcoLight interfaces. The table shows the phase-scoring component $f_p = f(\mathbf{x}_{p,t}^{(i)})$; ATT and ABF are measured using the complete policy $U_p = f_p + \beta_p^{(i)}$. The EcoLight (native) example uses the aligned inputs described in Appendix A.7.1.

Here, I_p equals 1 when phase p is active and 0 otherwise. V_p and $V_p^{\max}$ denote total and peak vehicle demand, respectively; N_p and W_p denote queued vehicle count and accumulated waiting time. For brevity, the decision-step index t and intersection index i are omitted from the table expressions. The

Table A.19: Representative learned rules and traffic-control interpretations across teacher families. ATT is measured in seconds. Each row corresponds to one scenario–flow task.

Teacher / task	Phase-scoring component	Traffic-control interpretation	ABF ↑	ATT (s) ↓	
				Teacher	Student
CoLight Atlanta / flow 3	$f_p = V_p + 0.2 I_p V_p^2$	Prioritizes demand; a quadratic bonus retains a busy active phase.	0.94	339.20	317.90
EcoLight (phase-selection) Hangzhou2 / flow 2	$f_p = 0.7V_p + 0.3V_p^{\max} + 0.15 I_p$	Combines total and peak demand with an active-phase retention bonus.	0.94	135.93	134.13
EcoLight (native, aligned) Hangzhou2 / flow 2	$f_{\text{keep}} = \frac{g}{\sqrt{g+1}}$ $f_{\text{switch}} = \frac{r\log(1+r)}{g+1}$	Keeps green for served demand; switches as red-side demand grows relative to green-side demand.	0.99	135.93	135.05
FRAP Atlanta / flow 1	$f_p = \frac{\sum_{d\in\mathcal{R}_p} n_d}{17} + 0.2 I_p$	Scores aggregate vehicle demand linearly and adds a fixed bonus for the active phase.	0.89	255.03	245.01
MPLight Hangzhou3 / flow 4	$f_p = P_p^{\Sigma} + P_p^{\max} + 0.15 I_p$	Combines aggregate and bottleneck pressure with a retention bonus, following MaxPressure [7].	0.93	86.01	84.04
TinyLight Hangzhou3 / flow 1	$f_p = P_p^{\Sigma} + 0.5\log(1+W_p) + 0.2N_p + 0.5 I_p$	Prioritizes pressure, queues, and waiting time, with a retention bonus. The logarithm gives additional waiting time progressively less weight.	0.86	80.77	80.63

pressure terms follow Fig. 3. For the set of directions $\mathcal{R}_p$ served by phase p, $P_p^{\Sigma} = \sum_{d \in \mathcal{R}_p} \hat{p}_{d,t}$ and $P_p^{\max} = \max_{d \in \mathcal{R}_p} \hat{p}_{d,t}$, where $\hat{p}_{d,t}$ is normalized directional pressure. Before normalization, directional pressure is the upstream vehicle count minus the downstream vehicle count. For EcoLight (native, aligned), g and r denote green- and red-side demand, respectively. For FRAP, n_d denotes directional vehicle demand, and the constant 17 is the 95th percentile of `vehicles_raw` in the source data.

A.8.7 LLM Configuration, Cost, and Randomness

The main experiments use Gemini 3.1 Flash with temperature 0.8; top-p and maximum output length use the API defaults, and individual responses contain 407–779 tokens. To evaluate random-seed sensitivity, we conduct five searches for each of five teachers and seven scenarios, giving 175 searches. Within each of the 35 teacher–scenario combinations, we compute the coefficient of variation (CV; standard deviation divided by mean) of the final joint score across the five seeds and then average the CV across combinations. The remaining quantities in Table A.20 are averaged over all 175 searches.

Table A.20: LLM random-seed sensitivity and discovery cost.

Metric	Mean
Final joint-score CV	2.08%
Token usage per search	67.1K
Generation cost per search	USD 0.0353
LLM API latency per search	104.2 s

The low average final-score CV indicates limited sensitivity to sampling randomness under this protocol. LLM use is confined to policy discovery: reproducing a search requires the same model and API configuration and may still vary because of provider-side sampling, whereas exported policies execute deterministically as standalone Python code and require no LLM during inference or deployment. Token cost and API latency are provider- and time-dependent and are reported only for the evaluated configuration.

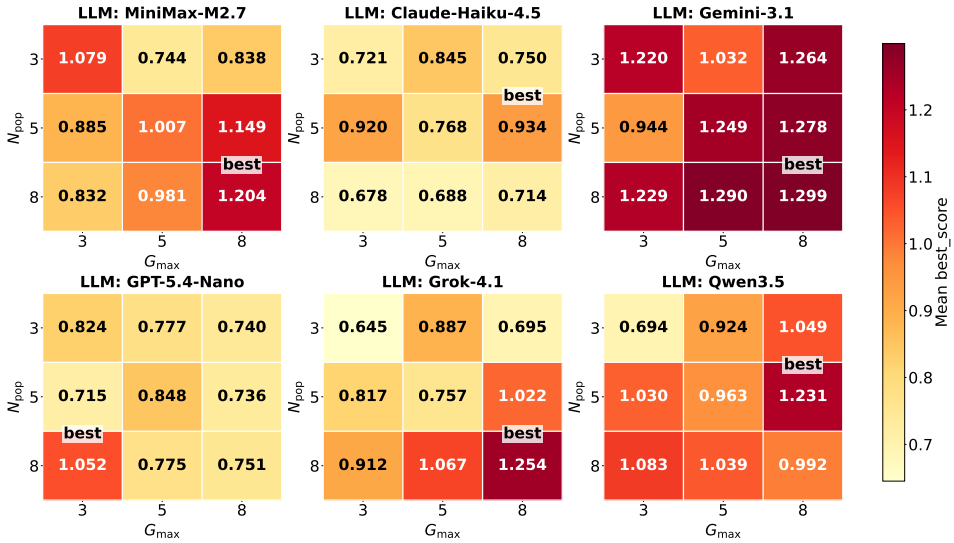


Figure A.3: Parameter sensitivity on MPLight/Los Angeles/flow0. Each heatmap shows the final best score under one LLM backbone, where the score is computed by the policy evaluation objective in Eq. (4). Rows correspond to N_{pop} and columns to $G_{\max}$.

A.8.8 Parameter Analysis

We analyze the sensitivity of PPL-TSC to three configuration factors: the LLM backbone, the population size N_{pop}, and the number of generations $G_{\max}$. Fig. A.3 visualizes the 54 runs on MPLight/Los Angeles/flow0 as per-backbone heatmaps of the final best score, where the score is the policy evaluation objective defined in Eq. (4).

Two observations are clear from the figure. First, the LLM backbone substantially affects the absolute search quality. Gemini 3.1 Flash is the strongest overall, with all nine settings above 0.94 and a peak score of 1.299 at $(8, 8)$. Grok-4.1 and Qwen3.5 form a second tier, reaching 1.254 and 1.231, respectively. By contrast, Claude-Haiku-4.5 and GPT-5.4-Nano are weaker, with best scores of only 0.934 and 1.052.

Second, larger generation budgets are usually more helpful than simply enlarging the population. For five of the six backbones, the best cell lies at $G_{\max} = 8$; the only exception is GPT-5.4-Nano, whose best score appears at $(8, 3)$. The preferred population size is less stable: the best setting uses $N_{\text{pop}} = 8$ for MiniMax-M2.7, Gemini 3.1 Flash, GPT-5.4-Nano, and Grok-4.1, but $N_{\text{pop}} = 5$ for Claude-Haiku-4.5 and Qwen3.5. Overall, $(5, 5)$ provides a conservative default, while larger-budget settings such as $(5, 8)$ and $(8, 8)$ offer stronger search quality when additional runtime is available.

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: The abstract and introduction accurately summarize the paper's core contributions: PPL-TSC, the phase utility function representation, and the generate–review–calibrate–evaluate framework. Their claims on transparency, teacher fidelity, transfer generalization, and computational efficiency are consistent with the experimental results reported in the main paper and appendix.

   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 paper includes a dedicated Limitations subsection in the appendix that discusses key constraints of the current approach, including teacher dependence and LLM-guided search dependence. It also outlines practical implications and future directions to mitigate these limitations.

   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: [N/A]

   Justification: The paper does not present formal theorems, lemmas, or proof-based theoretical claims. The main contributions are methodological and empirical.

   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: The paper and appendix describe the method pipeline, evaluation protocol, teacher settings, key hyperparameters, and implementation choices. The public repository at `https://github.com/ForrestXie9/PPL-TSC` provides implementation code, traffic networks and flows, configuration templates, and commands for teacher training, trajectory collection, policy learning, and evaluation. Teacher weights, trajectories, and learned policies can be generated using these commands; policy discovery requires user-configured LLM API access.

   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: The code and traffic networks and flows are publicly available at `https://github.com/ForrestXie9/PPL-TSC`. The README specifies the required environment and commands for installation, teacher training, trajectory collection, policy learning, and same-scenario and cross-scenario evaluation. Pretrained weights, collected trajectories, and generated policies are not bundled and must be produced using the provided pipeline. LLM-based policy discovery requires API access and may vary with provider-side sampling.

   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: The manuscript specifies the teacher families, scenario setup, optimization choices, and key hyperparameters for policy search and calibration, with additional details in the appendix. The test settings and evaluation metrics used in the main comparisons are also explicitly described.

   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: The paper reports variability using mean and standard deviation (shown as $\pm$ values in tables and summarized in appendix tables). The aggregation protocol for these statistics is described alongside the reported results.

   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: [No]

   Justification: Appendix A.8 specifies the CPU, GPU, memory, training episodes, and search settings. Appendix A.8.7 reports LLM API latency, but per-experiment execution times and the total compute budget are not fully reported.

   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: We have reviewed the NeurIPS Code of Ethics and are not aware of any aspect of this work that violates it. The study uses simulated traffic environments and does not involve human-subject experimentation.

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: [Yes]

    Justification: The manuscript includes a dedicated Broader Impacts subsection in the appendix. That discussion covers both potential benefits (e.g., improved traffic efficiency and reduced congestion-related emissions) and potential risks, together with cautionary deployment considerations.

    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: This work does not release high-risk generative models or scraped datasets that would require specialized misuse safeguards. The method targets traffic-signal control in simulation settings.

    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: The original simulator, benchmark, and teacher methods are credited through their original publications, with asset usage described in Appendix A.7. The public code repository provides third-party attribution in `THIRD_PARTY_NOTICES.md` and includes license files for the bundled simulator and dependencies. We respect the applicable licenses and terms of use for third-party code, data, and LLM services.

    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 released implementation is documented in the repository README at `https://github.com/ForrestXie9/PPL-TSC`, including installation requirements, supported teachers and student variants, configuration templates, execution commands, and the scope of included assets. Third-party attribution and licensing notes are provided in `THIRD_PARTY_NOTICES.md`.

    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 study does not involve crowdsourcing or human-subject data collection. All experiments are conducted in simulation environments.

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: No human participants are involved in this work, so IRB approval and participant-risk disclosure are not applicable.

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: [Yes]

Justification: LLM usage is a core part of the method: the model is used for candidate policy generation and iterative repair in the generate–review–calibrate–evaluate pipeline. The paper describes this role in both the abstract/method section and appendix prompt details.

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.