CAPE: CLOSED-LOOP ADAPTIVE PLANNING AND EXECUTION FOR MULTI-NODE TRAFFIC FAULT RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Traffic fault recovery aims to restore network performance after a fault. Existing approaches follow a diagnose-then-control pipeline: localize a root cause and control there. This overlooks two properties: the root cause need not be the most effective control location, and a single intervention cannot restore a network whose congestion keeps evolving. We introduce CAPE(**C**losed **L**oop **A**daptive **P**lanning and **E**xecution), a closed-loop framework that adjusts interventions as traffic evolves. CAPE uses two-stage decision making: a Planner selects control nodes and measures, and an Executor generates safe actions, enabling effective search in a large hybrid action space. For rare control sets, CAPE calibrates a low-dimensional strategy code via paired simulations and distills it into a generator, improving execution without fine-tuning the full Executor. Across five networks and four fault types, CAPE achieves an average Recovery Gain of 89.70%, exceeding the strongest baseline by 9.15 percentage points. Code is available at `https://anonymous.4open.science/r/CPAE-2785`.

1 INTRODUCTION

Urban traffic faults are triggered locally, but their effects rarely stay local: queues spill back along upstream roads, and rerouting transfers pressure to neighboring intersections. Existing studies decompose recovery into fault understanding, which localizes anomalies and their sources (Chawla et al., 2012; Lan et al., 2014), and traffic control, which adjusts signal plans within a predefined set of intersections (Wei et al., 2019a; Zheng et al., 2019; Wei et al., 2019b). This pipeline leaves two properties unaddressed: the root cause need not be the most effective control location, and a single intervention cannot restore a network whose congestion keeps evolving.

Figure 1 illustrates both effects on the same incident. When only root-cause node B is controlled, B recovers but released and diverted flow create a downstream bottleneck at C. An adaptive controller instead limits inflow from A in the first period and shifts additional control to C in the second, leaving a shorter residual queue.

To address these challenges, we propose **CAPE** (**C**losed-loop **A**daptive **P**lanning and **E**xecution), a closed-loop framework for dynamic multi-node traffic fault recovery. Unlike root-cause or fixed-scope control, CAPE's Planner dynamically revises the control set from the current traffic state, shifting control to non-root nodes as congestion moves, while the Executor generates safe continuous actions for the selected set. This creates a planning–execution mismatch: the Executor cannot reliably implement rarely visited control sets, leading the Planner to undervalue them. CAPE resolves this mismatch by calibrating a strategy code with a few paired simulations and distilling it into a generator, correcting subsequent decisions without retraining the full Executor. Our contributions are:

- We formulate traffic fault recovery as a closed-loop multi-node control problem and show that dynamic scope revision outperforms fixed root-cause or neighborhood control.
- We introduce a state-conditioned Planner–Executor architecture that separates variable-size control-set selection from continuous action generation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

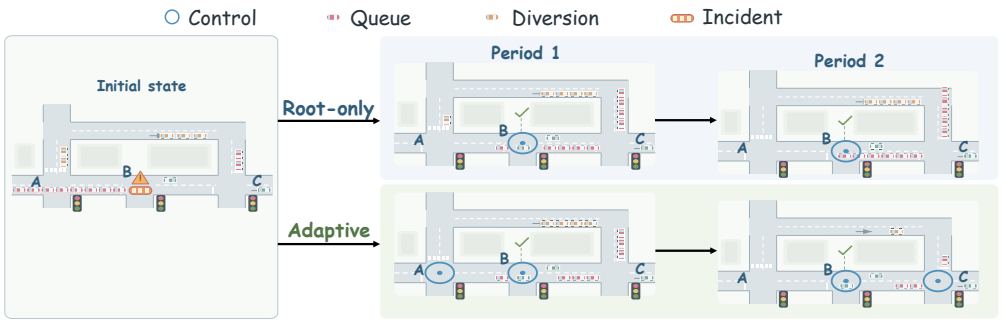

Figure 1: Motivation. Under the same incident at B, Root-only keeps control fixed at B, while Adaptive shifts control from A to C as congestion moves.

- We develop an advantage-disagreement calibration mechanism that corrects value estimates for rarely visited control sets via paired simulator probes and distills the calibrated strategy code for reuse.

2 Related Work

Traffic fault recovery has been studied through three closely related directions. The first identifies the source and propagation of a disruption. Lan et al. detect abnormal links and trace their propagation from historical traffic deviations (Lan et al., 2014), while Chawla et al. infer a sparse root-cause explanation from traffic anomalies and network relations (Chawla et al., 2012). Spatio-temporal models such as DCRNN and Graph WaveNet describe how traffic states evolve over connected roads (Li et al., 2018; Wu et al., 2019).

The second direction optimizes recovery actions at the network level. Resilience and vulnerability measures quantify performance loss and recovery time after disruptions (Berdica, 2002; Jenelius et al., 2006). Recovery-scheduling models determine the order and timing of restoration operations (Chang & Nojima, 2001), while incident-management models coordinate traffic redistribution during recovery (Liu et al., 2013; Lo & Szeto, 2002).

The third direction learns adaptive traffic-control policies from environment interaction. IntelliLight selects signal phases (Wei et al., 2018); FRAP models phase competition (Zheng et al., 2019); PressLight incorporates max-pressure principles (Wei et al., 2019a); and CoLight and MPLight address coordination and parameter sharing (Wei et al., 2019b; Chen et al., 2020). Advanced-MPLight and MetaLight improve representation and cross-scenario adaptation (Zhang et al., 2022; Zang et al., 2020), while recent work studies scenario-agnostic learning, graph meta-learning, and neighborhood-aware coordination (Jiang et al., 2024; Zhou et al., 2025; Zhu et al., 2025). Parameterized and hierarchical RL separate structural and continuous decisions (Nachum et al., 2018); CEM-MPC replans action sequences during execution (Bharadhwaj et al., 2020).

3 Method

3.1 Closed-Loop Recovery

We model a traffic network as a directed graph $G = (\mathcal{V}, \mathcal{E})$. At recovery period t, CAPE receives the latest 12 traffic snapshots, the network topology, the active control set, and recent control history. The active set is

$$\mathcal{S}_t = \{(v_i, m_i)\}_{i=1}^{K_t}, \qquad |\{v : (v, m) \in \mathcal{S}_t\}| \le 32, \tag{1}$$

where v_i is a controlled intersection and m_i is a legal control measure. The period reward is

$$r_t = -d_t - q_t - 2b_t - 0.10c_t - 0.05u_t, \tag{2}$$

where d_t, q_t, and b_t are normalized delay, queue intensity, and spillback, while c_t and u_t denote control and switching costs. Normalization and metric definitions are given in Appendix A.1.

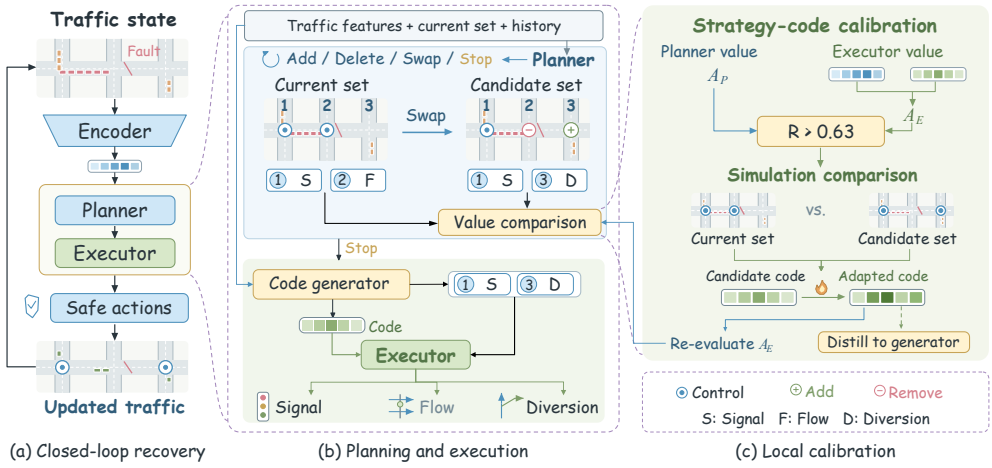

Figure 2: Overview of CAPE. The Planner revises a variable-size control set, the Executor generates continuous actions, and value disagreement triggers local strategy-code calibration.

Figure 2 summarizes the recovery loop. CAPE encodes the current traffic state, revises a control set, generates safe continuous actions, and executes them in SUMO. The resulting traffic state starts the next recovery period.

3.2 Planning and Execution

The shared encoder maps the observation window to node and network representations:

$$(H_t, g_t) = E_\phi(o_t, G), \qquad H_t = \{h_{t,v}\}_{v \in \mathcal{V}}. \tag{3}$$

A scoring head ranks legal node–measure pairs and retains at most 64 candidates in $\mathcal{C}_t$. The score only limits the search space: it is neither a root-cause probability nor a control value. Active pairs are always retained so that they can be removed. Architecture and candidate-scoring details are provided in Appendix A.2.

Starting from the inherited set, the Planner iteratively samples ADD, DELETE, SWAP, or STOP:

$$e_{t,\ell} \sim \pi_P(\cdot \mid x_{t,\ell}), \qquad \mathcal{S}_{t,\ell+1} = \mathcal{T}(\mathcal{S}_{t,\ell}, e_{t,\ell}), \tag{4}$$

where $x_{t,\ell}$ contains the encoded traffic state, candidate pool, partial control set, and recent control history. The first three operations select a candidate pair, while STOP terminates planning. The Planner uses discrete SAC to estimate the long-term value of set edits.

After STOP, the strategy-code generator maps the selected set to a compact execution context,

$$z_t = G_\psi(H_t, g_t, \mathcal{S}_t), \qquad z_t \in \mathbb{R}^{128}, \tag{5}$$

and the Executor samples continuous actions as

$$a_t \sim \pi_E(\cdot \mid H_t, g_t, \mathcal{S}_t, z_t). \tag{6}$$

The action includes signal control, flow control, and path diversion. Before execution, it is projected to the state-dependent feasible action set $\mathcal{A}(o_t)$:

$$a_t^{\text{safe}} = \Pi_{\mathcal{A}(o_t)}(a_t). \tag{7}$$

Thus, the Planner determines where and what to control, whereas the Executor determines how to control it. The action space and safety constraints are specified in Appendix A.3.

3.3 Advantage-Disagreement Calibration

Calibration is evaluated before a proposed edit is committed. For readability, we omit period and internal-edit indices in this subsection and use x for the current Planner state. Given a candidate edit e, let

$$\mathcal{S}^+ = \mathcal{T}(\mathcal{S}, e). \tag{8}$$

Matched reference and edited SUMO branches use identical network, demand, fault configuration, simulator state, random seed, recovery horizon, and frozen policy parameters; they differ only in the compared control set. Let $\mathcal{J}(x, \mathcal{S})$ denote the corresponding discounted branch return. The paired intervention effect is

$$y(e) = \mathcal{J}(x, \mathcal{S}^+) - \mathcal{J}(x, \mathcal{S}). \tag{9}$$

The Planner advantage compares the candidate edit with preserving the current set:

$$A_P = Q_P(x, e) - Q_P(x, \textsc{Stop}). \tag{10}$$

Each compared set receives its own execution context,

$$z_{\mathcal{S}} = G_\psi(H, g, \mathcal{S}), \qquad z_{\mathcal{S}^+} = G_\psi(H, g, \mathcal{S}^+), \tag{11}$$

and the Executor advantage is

$$A_E = V_E(x, \mathcal{S}^+, z_{\mathcal{S}^+}) - V_E(x, \mathcal{S}, z_{\mathcal{S}}), \quad V_E(x, \mathcal{S}, z) = \mathbb{E}_{a \sim \pi_E(\cdot|x,\mathcal{S},z)} \left[Q_E(x, \mathcal{S}, z, a) \right], \tag{12}$$

where Q_E is the Executor action-value function. A_P estimates the structural value of an edit, whereas A_E estimates the value that the current Executor can realize for the corresponding set. Both are grounded in the paired effect:

$$\mathcal{L}_{\text{effect}} = \text{Huber}(A_P - y) + \text{Huber}(A_E - y) + 0.1(A_P - A_E)^2. \tag{13}$$

For a non-STOP edit, CAPE computes

$$R = \frac{|A_P - A_E|}{|A_P| + |A_E| + 10^{-6}}. \tag{14}$$

R is large when the two advantages differ markedly relative to their scale, including sign disagreement, and small when they have similar sign and magnitude. When $R > 0.63$ and no calibration has occurred in the current period, CAPE launches four paired SUMO probes. The encoder, Planner, Executor body, and value heads remain frozen; only the temporary code for $\mathcal{S}^+$ is updated. If $z^\star_{\mathcal{S}^+}$ is the adapted code, the generator is trained to reproduce it:

$$\mathcal{L}_{\text{distill}} = \left\| G_\psi(H, g, \mathcal{S}^+) - \text{sg}(z^\star_{\mathcal{S}^+}) \right\|_2^2, \tag{15}$$

where sg is stop-gradient. CAPE recomputes A_E using the adapted candidate code and resumes planning from the current partial set. Detailed calibration updates are given in Appendix B.

3.4 Training and Inference

We initialize the encoder with reconstruction and future-state prediction, cold-start the Planner and Executor with balanced paired edit trajectories, and then jointly optimize all modules with online discrete and continuous SAC. The joint objective combines Planner SAC, Executor SAC, encoder auxiliary learning, paired-effect alignment, and strategy-code distillation:

$$\mathcal{L}_{\text{CAPE}} = \mathcal{L}^P_{\text{SAC}} + \mathcal{L}^E_{\text{SAC}} + 0.05\mathcal{L}_{\text{enc}} + 0.20\mathcal{L}_{\text{effect}} + 0.10\mathcal{L}_{\text{distill}}. \tag{16}$$

At inference, CAPE constructs candidates, performs a bounded sequence of set edits, calibrates only when the disagreement condition is met, and executes the final safe action after STOP. Complete architectures, training budgets, action bounds, and inference limits are provided in Appendix A.

4 Experiments

4.1 Experimental Protocol

We evaluate CAPE in SUMO on five networks: a synthetic Grid-20x20 network, Cologne8 and Ingolstadt21 from RESCO (Ault & Sharon, 2021), LuST-Full (Codeca et al., 2017), and InTAS-Full (Lobo et al., 2020). Each network is tested under capacity-drop, demand-surge, signal-fault, and compound-fault scenarios, with 100 scenarios and five seeds per network–fault pair. Each 3600-second episode consists of normal traffic (0–900 s), an untreated fault interval (900–1800 s), and

Table 1: Recovery cost and Gain across five networks. Bold and underline mark the best and second-best deployable results per block; lower cost and higher Gain are better.

Method	Recovery Cost ↓		Recovery Gain (%) ↑					
	T_{80} (s)	Latency (ms)	Grid-20x20	Cologne8	Ingolstadt21	LuST-Full	InTAS-Full	Average
Static / Generic Planning								
FixedTime	1800.0	**0.12**	40.52	38.27	42.14	41.56	39.81	40.46
CEM-MPC	1765.2	764.29	63.84	62.15	64.91	64.28	63.55	63.75
MaxPressure	1758.8	<u>1.58</u>	64.67	63.12	65.73	65.06	64.38	64.59
MAPPO	1731.4	11.37	66.31	65.04	67.42	66.85	66.19	66.36
Traffic Signal RL								
PressLight	1652.1	4.61	70.18	69.45	71.34	70.92	69.87	70.35
CoLight	1589.6	17.43	72.35	71.21	73.46	72.84	72.03	72.38
Adv-MPLight	1554.3	15.82	73.11	72.08	74.32	73.75	72.94	73.24
GESA	1512.9	23.49	74.23	73.54	75.16	74.87	74.09	74.38
Recent MARL / Agent-based								
NB-MARL	1395.4	28.16	77.20	76.15	78.05	77.50	76.82	77.14
RMTC	1351.8	35.74	78.50	77.40	79.20	78.75	78.25	78.42
GraphMeta	1284.5	47.91	79.85	78.60	80.52	80.14	79.28	79.68
VLMLight	<u>1243.2</u>	342.18	<u>80.75</u>	<u>79.50</u>	<u>81.30</u>	<u>80.95</u>	<u>80.25</u>	<u>80.55</u>
Proposed								
CAPE (Ours)	**754.2**	94.63	**89.73**	**88.52**	**90.84**	**90.27**	**89.15**	**89.70**

a recovery interval (1800–3600 s) in which controllers operate. We report three metrics: Recovery Gain, T_{80}, and Latency; their definitions are given in Appendix C.1.

We compare against three groups of baselines. Static and generic planning methods include FixedTime, CEM-MPC (Bharadhwaj et al., 2020; Wada et al., 2015), MaxPressure (Varaiya, 2013), and MAPPO (Yu et al., 2022). Traffic-signal RL methods include PressLight (Wei et al., 2019a), CoLight (Wei et al., 2019b), Advanced-MPLight (Zhang et al., 2022), and GESA (Jiang et al., 2024). Recent MARL and agent-based controllers include NB-MARL (Ren et al., 2025), RMTC (Cheng et al., 2025), GraphMeta (Zhou et al., 2025), and VLMLight (Wang et al., 2025). All learning-based methods share the same observations and tuning budget; details are in Appendix C.2.

4.2 Main Recovery Results

Table 1 reports Recovery cost and Recovery Gain across five networks. CAPE achieves the best Gain on every network, averaging 89.70% and exceeding VLMLight by 9.15 pp, with margins from 8.90 to 9.54 pp. It also has the shortest T_{80}, 754.2 s, 489.0 s faster than VLMLight, showing faster convergence to its own recovery level. These gains support jointly revising the control set and generating actions conditioned on it as faults evolve.

CAPE's mean latency is 94.63 ms, below VLMLight (342.18 ms) and CEM-MPC (764.29 ms), although rule-based methods remain faster without learned inference. Full-actor adaptation reaches 91.26%, only 1.56 pp above CAPE, but requires 31.91 rather than 7.43 additional SUMO calls per episode. CAPE thus approaches the full-adaptation upper bound without updating the complete Executor. Per-fault results, confidence intervals, tests, tuning records, and adaptation-cost accounting appear in Appendix D.1 and Appendix D.1.1.

4.3 Mechanism and Ablation Analysis

We ablate each CAPE component under the unified fault-recovery protocol. The flat joint policy replaces the Planner–Executor decomposition with a single policy that predicts a complete control decision in one step, while the remaining variants remove set editing, the strategy code, advantage-disagreement triggering, or paired effect alignment, respectively. All variants share the same networks, fault scenarios, training budget, and safety constraints.

Table 2: Mechanism ablation under the unified recovery protocol. The Δ column reports absolute degradation in percentage points (pp) relative to the full CAPE model.

Variant	Recovery Gain (%) ↑	Δ (pp)	p vs. CAPE
CAPE (Full Model)	**89.70 ± 1.15**	–	–
Architecture & Scope Variations			
Flat joint policy	86.45 ± 1.42	-3.25	< 0.01
w/o set editing	83.88 ± 1.65	-5.82	< 0.01
Calibration Mechanism Variations			
w/o disagreement trigger	85.32 ± 1.38	-4.38	< 0.01
w/o strategy code	81.24 ± 1.84	-8.46	< 0.01
w/o effect alignment	78.65 ± 2.12	-11.05	< 0.01

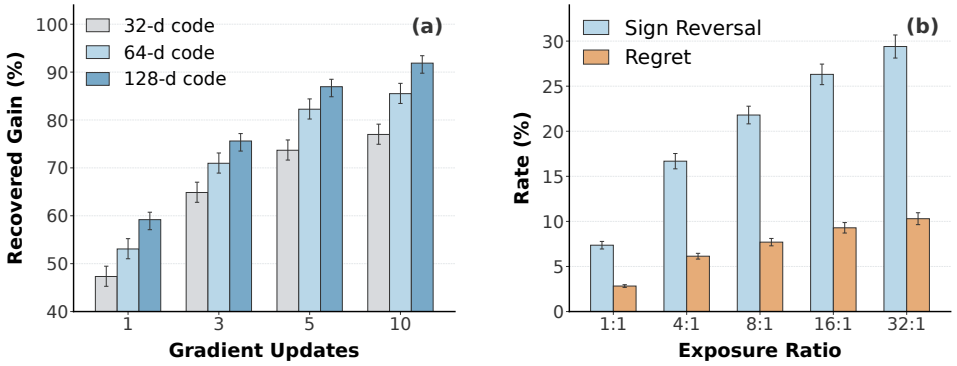

Figure 3: Strategy-code calibration. **(a)** Recovered matched-effect gain across code dimensions and gradient-update budgets. **(b)** Value-estimation error under controlled training exposure.

Table 2 reports the results in two blocks. The first block concerns the Planner–Executor decomposition and control-scope editing. Removing the decomposition drops Recovery Gain from 89.70% to 86.45%, showing that separating set construction from continuous execution already helps policy search. Removing set editing causes a larger drop to 83.88%, indicating that the gain comes mainly from the Planner's ability to incrementally add, delete, and replace controls as congestion evolves. The second block concerns advantage-disagreement calibration. Removing the disagreement trigger drops Recovery Gain to 85.32%, confirming that calibration is needed and that applying it selectively retains most of its benefit. Removing the strategy code drops it further to 81.24%, showing that the low-dimensional code is the carrier of calibrated execution capability. Removing paired effect alignment produces the largest degradation, 11.05 pp, because the Planner and Executor are no longer jointly grounded in matched intervention outcomes, and the Planner can misattribute insufficient execution capability to an ineffective control set. Per-network results are provided in Appendix D.2.

4.4 Strategy-Code Calibration

We first examine whether an uncalibrated Executor misestimates rarely visited control sets. Figure 3(b) reports sign reversal and regret under controlled exposure ratios. As the exposure ratio increases from 1:1 to 32:1, sign reversal rises from 7.36% to 29.40%, and regret rises from 2.83% to 10.30%. The growth is steep between 1:1 and 16:1, and then slows as the imbalance becomes extreme, indicating that value-estimation error saturates once a control set becomes very rare. An uncalibrated Executor can therefore provide misleading feedback for rarely visited but potentially useful control sets. Table 3 compares calibration schedules under the unified recovery protocol. Calibrating every period obtains the highest raw Recovery Gain, 90.18%, but requires 25.72 additional SUMO calls per episode. Advantage-disagreement triggering achieves 89.70%, only 0.48 pp lower, with 7.43 calls, or 28.9% of the exhaustive schedule's cost, while never calibrating reduces

Table 3: Calibration schedules under the unified recovery protocol. Bold denotes the best result.

Trigger Policy	Recovery Gain (%) ↑	Δ (pp)	Extra SUMO Calls ↓
Never (w/o trigger)	85.32 ± 1.38	-4.38	0.00
Every 4 periods	86.82 ± 1.45	-2.88	8.17
Uncertainty-only	87.45 ± 1.22	-2.25	6.50
Every 2 periods	88.54 ± 1.10	-1.16	14.35
Every period (Exhaustive)	**90.18** ± **0.95**	+0.48	25.72
Advantage disagreement (CAPE)	89.70 ± 1.15	–	**7.43**

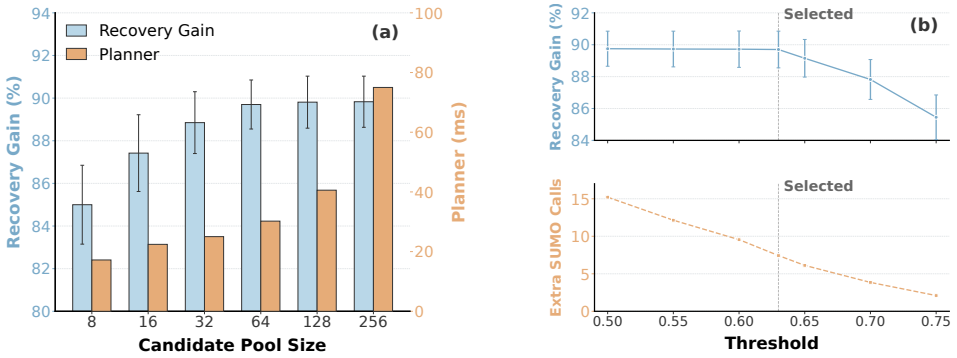

Figure 4: Control configuration. **(a)** Recovery Gain and Planner latency under different candidate-pool sizes. **(b)** Recovery Gain and extra SUMO calls under different disagreement thresholds.

Recovery Gain to 85.32%. Calibration need not be applied at every period; matched probes are most useful when planning value and current execution capability disagree on a candidate edit.

Figure 3(a) further examines the adaptation budget. Larger strategy codes recover more matched-effect gain at every update budget. For the final 128-dimensional code, five updates recover 86.95% of the matched-effect gain with 7.33 additional calls; increasing to ten updates recovers an additional 4.92 pp but increases calls to 13.07. At the same five-update budget, Full-actor adaptation is only 0.82 pp higher but requires 2.17× as many calls. After distillation, the generator recovers 80.54% matched-effect gain, retaining 92.63% of the temporary code's gain, whereas a random generator retains only 23.47%. Complete dimension sweeps and per-fault trigger results are provided in Appendix D.3.

4.5 Control Configuration and Sensitivity

We vary the candidate-pool size supplied to the Planner. Figure 4(a) shows that increasing it from 8 to 64 raises Recovery Gain from 85.00% to 89.70%, while Planner latency grows from 17.20 ms to 30.21 ms. With 8 candidates, recovery is 4.70 pp below the 64-candidate setting. Expanding to 128 or 256 yields at most 0.11 pp additional gain while raising latency to 40.58 ms and 75.00 ms. We therefore use 64 candidates as the operating point.

Figure 4(b) reports the sensitivity of the disagreement threshold. The upper panel shows that Recovery Gain remains above 89.70% for thresholds up to 0.63, then declines as the trigger becomes too conservative. The lower panel shows that extra SUMO calls fall monotonically as the threshold increases. The selected threshold $R > 0.63$ sits at the knee of both curves, retaining nearly all recovery gain at 7.43 extra calls per episode. We next evaluate the available control measures. Table 4 shows that signal control alone obtains 86.55% Recovery Gain. Adding flow control or path diversion raises the gain to 88.12% and 88.74%, respectively, while combining all three control families reaches 89.70%. Additional threshold, reward, safety, and resource analyses are given in Appendix D.4.

Table 4: Control-atom sensitivity under the unified protocol. Bold marks highest Recovery Gain.

Action Set	Recovery Gain (%) ↑	Control Cost ↓
Signal only	86.55	4.8
Signal + flow	88.12	6.5
Signal + diversion	88.74	7.2
Signal + flow + diversion	**89.70**	8.8

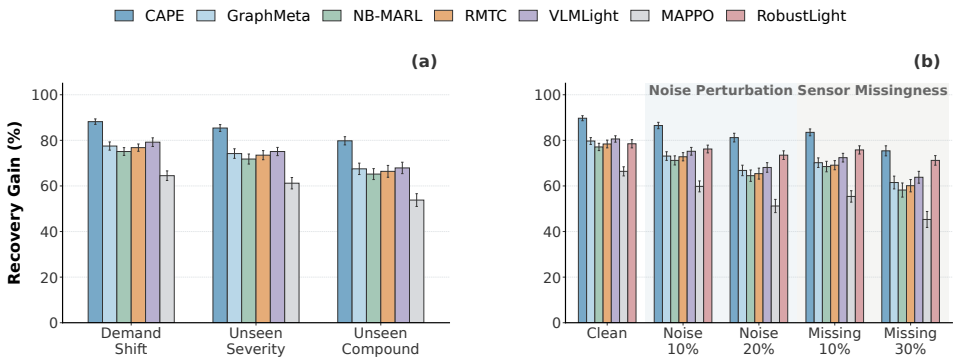

Figure 5: Generalization and observation robustness. **(a)** Under demand shifts, unseen severities, and unseen compound faults. **(b)** Under noisy and incomplete observations.

4.6 Generalization, Robustness, and Efficiency

We evaluate CAPE under traffic conditions outside the training distribution and under corrupted observations. Figure 5(a) reports Recovery Gain under demand shifts, unseen fault severities, and unseen compound faults. CAPE achieves 88.2%, 85.4%, and 79.8%, respectively, exceeding the strongest baseline in each setting by 9.0–11.9 pp. The gap is largest for unseen compound faults, the most difficult setting for all methods, yet CAPE still retains nearly 80% recovery, indicating that state-conditioned control-set planning transfers better to changed congestion patterns than policies tied to a fixed control structure.

Figure 5(b) evaluates additive Gaussian noise and sensor missingness. CAPE is strongest under clean observations, 10% and 20% noise, and 10% missingness. Under 30% missingness, RobustLight (Li et al., 2025) achieves 78.5%, compared with 75.4% for CAPE. This complementary result is consistent with RobustLight's explicit observation-reconstruction objective, while CAPE's main advantage concerns dynamic fault recovery and control-scope adaptation. Table 5 reports the runtime cost of the final 128-dimensional CAPE. On Grid-20x20, the mean and P95 decision latencies are 174.84 ms and 267.03 ms, both far below the 180-second control period. Across the five networks, mean latency ranges from 50.76 to 174.84 ms, peak memory from 3.96 to 9.84 GB, and additional SUMO calls from 7.21 to 7.78 per episode, showing that selective calibration adds limited decision overhead. Exact per-condition values are reported in Tables 18 and 19, and period-level traces are provided in Appendix D.5.

4.7 Case Study

Figure 6 presents a representative compound-fault episode on Grid-20x20, near the median final Recovery Gain. Panels (a) and (b) plot the queue-to-storage ratios of R1–R5 and R6–R10, respectively; the black Local curve aggregates the ten roads. Panel (c) maps control-set changes at the numbered 180-second planning periods: blue circles mark active controls, black crosses mark persistent capacity loss, and color encodes queue occupancy.

At $t = 1800$ s, persistent capacity loss leaves R1 near its storage limit and queues on adjacent and receiving roads. CAPE initially controls the bottleneck and nearby roads, then adds R3 and R4 at $t = 1980$ s. R2 and R3 improve, but recurring R4 and rising R7 show that local relief does not

Table 5: Inference overhead of final CAPE with a 128-dimensional strategy code. Latency includes encoding, planning, action generation, and safety projection.

Network	Mean (ms) ↓	P95 (ms) ↓	Memory (GB) ↓	Extra Calls ↓
Cologne8	50.76	77.22	3.96	7.21
Ingolstadt21	55.51	84.33	4.13	7.29
InTAS-Full	78.97	120.31	5.43	7.36
LuST-Full	113.05	172.43	6.92	7.51
Grid-20x20	174.84	267.03	9.84	7.78

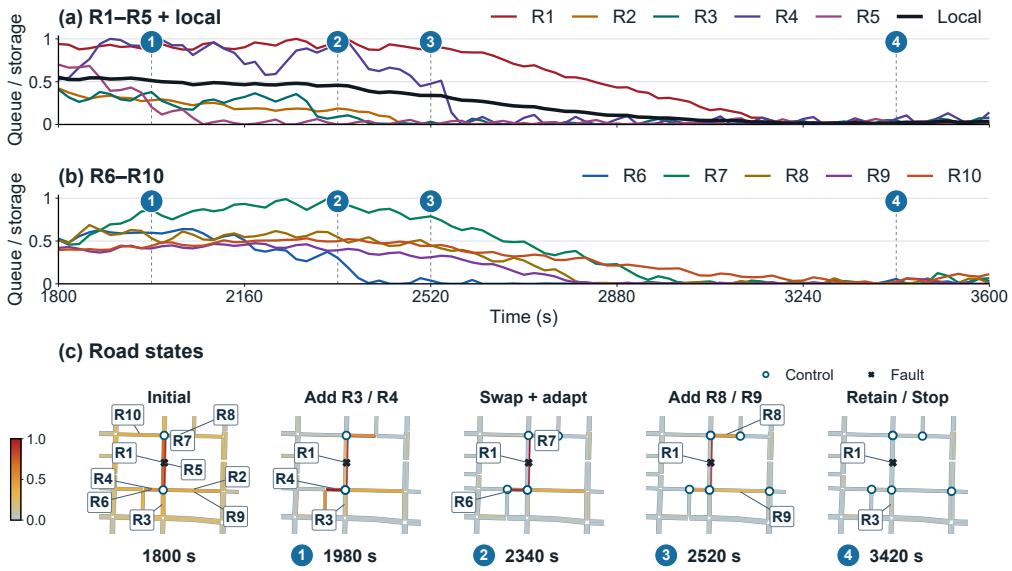

Figure 6: Representative compound-fault recovery on Grid-20x20. **(a)** Primary roads and local region. **(b)** Secondary roads. **(c)** Road-state snapshots at five key times.

eliminate transferred pressure. At $t = 2340$ s, the Planner swaps toward R6 and R7 and adapts the strategy code; R6 clears rapidly while R7 decays more slowly. It adds R8 and R9 at $t = 2520$ s as downstream queues propagate. R1 declines on a longer horizon because capacity loss persists. After R3 recovers, the Planner removes it at $t = 2880$ s; the $t = 3420$ s map retains necessary controls, and STOP ends editing for that period. The 20-second curves show delayed, fluctuating responses between 180-second planning updates.

5 CONCLUSION

We introduced CAPE, a closed-loop framework for multi-node traffic fault recovery. CAPE treats recovery as a dynamic decision process rather than a one-time repair at a diagnosed root-cause location. Its Planner continuously revises the control set from the evolving traffic state, while its Executor generates safe continuous actions for the selected set. To address the planning–execution mismatch caused by uneven exposure to control sets, CAPE uses advantage disagreement to trigger local strategy-code calibration and distills the calibrated code for subsequent reuse. Our experiments show that effective recovery often requires controlling nodes beyond the diagnosed root cause and revising the control scope after earlier interventions change the traffic state. Across five networks and four fault types, CAPE improves Recovery Gain by 9.15 percentage points over the strongest baseline. Future work will study calibration from field traffic data, recovery with partial controllability, and asynchronous intervention settings.

Ethics statement

All experiments in this work are conducted in traffic simulation and do not directly control real transportation infrastructure or use personally identifiable data. Although CAPE applies deterministic constraints to signal timing, release rates, and route diversion, simulation performance should not be interpreted as evidence of deployment safety. Any real-world use would require site-specific calibration, independent safety validation, regulatory approval, and human oversight.

AI use statement

In this work, we used generative AI tools to aid and polish writing and to support literature retrieval and discovery. The authors reviewed all AI-assisted text and literature-related outputs and take responsibility for the final content of this work.

Reproducibility statement

The problem definition, model components, objectives, trigger rule, and inference procedure are specified in Section 3. Experimental protocols, baseline adaptation, and principal evaluation conditions are described in Sections 4.1–4.2. Appendix A provides implementation details, Appendix B provides optimization and calibration details, Appendix C provides the experimental protocol, and Appendix D provides supplementary experimental results and statistical tests. Code and configurations will be available through the anonymous repository linked in the abstract.

References

James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Katja Berdica. An introduction to road vulnerability: What has been done, is done and should be done. *Transport Policy*, 9(2):117–127, 2002. doi: 10.1016/S0967-070X(02)00011-2.

Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *L4DC*, pp. 277–286, 2020.

Stephanie E. Chang and Nobuoto Nojima. Measuring post-disaster transportation system performance: The 1995 kobe earthquake in comparative perspective. *Transportation Research Part A: Policy and Practice*, 35(6):475–494, 2001. doi: 10.1016/S0965-8564(00)00003-3.

Sanjay Chawla, Yu Zheng, and Jiafeng Hu. Inferring the root cause in road traffic anomalies. In *2012 IEEE 12th International Conference on Data Mining*, pp. 141–150, 2012. doi: 10.1109/ICDM.2012.104.

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

Ming Cheng, Hao Chen, Zhiqing Li, Wang Jia, and Senzhang Wang. Role-aware multi-agent reinforcement learning for coordinated emergency traffic control. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4640.

Lara Codeca, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg SUMO traffic (LuST) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017. doi: 10.1109/MITS.2017.2666585.

Erik Jenelius, Tom Petersen, and Lars-Goran Mattsson. Importance and exposure in road network vulnerability analysis. *Transportation Research Part A*, 2006.

Haoyuan Jiang, Ziyue Li, Zhishuai Li, Lei Bai, Hangyu Mao, Wolfgang Ketter, and Rui Zhao. A general scenario-agnostic reinforcement learning for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 25(9):11330–11344, 2024. doi: 10.1109/TITS.2024.3377106.

Jinsong Lan, Cheng Long, Raymond Chi-Wing Wong, Youyang Chen, Yanjie Fu, Danhuai Guo, Shuguang Liu, Yong Ge, Yuanchun Zhou, and Jianhui Li. A new framework for traffic anomaly detection. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 875–883, 2014. doi: 10.1137/1.9781611973440.100.

Mingyuan Li, Jiahao Wang, Guangsheng Yu, Xu Wang, Qianrun Chen, Wei Ni, Lixiang Li, and Haipeng Peng. Robustlight: Improving robustness via diffusion reinforcement learning for traffic signal control. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 36192–36214. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25cs.html.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Yue Liu, Peng Li, Kevin Wehner, and Jie Yu. A generalized integrated corridor diversion control model for freeway incident management. *Computer-Aided Civil and Infrastructure Engineering*, 28(8):604–620, 2013. doi: 10.1111/mice.12024.

Hong K. Lo and W. Y. Szeto. A cell-based variational inequality formulation of the dynamic user optimal assignment problem. *Transportation Research Part B: Methodological*, 36(5):421–443, 2002. doi: 10.1016/S0191-2615(01)00011-X.

Silas C. Lobo, Stefan Neumeier, Evelio M. G. Fernandez, and Christian Facchi. InTAS – the ingolstadt traffic scenario for SUMO. *arXiv preprint arXiv:2011.11995*, 2020.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.

Yilong Ren, Yizhuo Chang, Zhiyong Cui, Xiao Chang, Haiyang Yu, Xiaosong Li, and Yinhai Wang. Is cooperative always better? multi-agent reinforcement learning with explicit neighborhood backtracking for network-wide traffic signal control. *Transportation Research Part C: Emerging Technologies*, 2025. doi: 10.1016/j.trc.2025.105265.

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

Kentaro Wada, Kento Usui, and Hideki Yaginuma. The cross-entropy method for coordinated traffic signal control with physical queues. *Journal of Japan Society of Civil Engineers, Ser. D3 (Infrastructure Planning and Management)*, 71(5):I$_1$055 $-$ $-I_1064, 2015. doi : .$

Maonan Wang, Yirong Chen, Aoyu Pang, Yuxin Cai, Chung Shue Chen, Yuheng Kan, and Man On Pun. Vlmlight: Safety-critical traffic signal control via vision-language meta-control and dual-branch reasoning architecture. In *Advances in Neural Information Processing Systems*, volume 38, 2025. 10.52202/085713-1320.

Hua Wei, Guanjie Zheng, Hanning Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *KDD*, 2018.

Hua Wei, Chacha Chen, Guanjie Zheng, et al. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *KDD*, 2019a.

Hua Wei, Nan Xu, Huichu Zhang, et al. Colight: Learning network-level cooperation for traffic signal control. In *CIKM*, 2019b.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 2019.

Chao Yu, Akash Velu, Eugene Vinitsky, et al. The surprising effectiveness of ppo in cooperative, multi-agent games. In *NeurIPS*, volume 35, pp. 24611–24624, 2022.

Xinshi Zang, Huaxiu Yao, Guanjie Zheng, et al. Metalight: Value-based meta-reinforcement learning for traffic signal control. In *AAAI*, 2020.

Liang Zhang, Qiang Wu, Jun Shen, Linyuan Lü, Bo Du, and Jianqing Wu. Expression might be enough: Representing pressure and demand for reinforcement learning based traffic signal control. In *International Conference on Machine Learning*, volume 162, pp. 26645–26654, 2022.

Guanjie Zheng, Yuming Xiong, Xinshi Zang, et al. Learning phase competition for traffic signal control. In *CIKM*, 2019.

Zhicheng Zhou, Ya Zhang, and Xinde Li. A deep reinforcement learning model for large-scale traffic signal control based on graph meta-learning using local subgraphs. *Science China Information Sciences*, 68(7):172203, 2025. 10.1007/s11432-023-4280-6.

Yaqiao Zhu, Hongkai Wen, Geyong Min, and Man Luo. Hilight: A hierarchical reinforcement learning framework with global adversarial guidance for large-scale traffic signal control. *arXiv preprint arXiv:2506.14391*, 2025.

A Implementation Details

A.1 Notation and Metric Definitions

Table 6 lists the symbols used throughout the paper.

Table 6: Notation used in CAPE.

Symbol	Meaning
$G = (\mathcal{V}, \mathcal{E})$	Directed traffic network graph
o_t, K_t	Observation window and active node–measure-pair count
H_t, g_t	Node-level and global traffic representations
$\mathcal{C}_t, \mathcal{S}_t$	Candidate pool and active control set
$x_{t,\ell}, e_{t,\ell}, \mathcal{T}$	Planner state, operation, transition operator
z_t, a_t, π_P, π_E	Strategy code, action, Planner policy, Executor policy
Q_P, Q_E, V_E	Planner, Executor, and expected Executor values
A_P, A_E, R	Planning advantage, execution advantage, disagreement score
$\mathcal{J}$	Discounted return from a matched simulator branch
d_t, q_t, b_t, c_t, u_t	Delay, queue, spillback, control, switching terms
$D_{\text{normal}}, D_{\text{fault}}, D_m(t)$	Delays used for Recovery Gain
$\mathcal{L}, D_{\text{scale}}, \tau_b$	Lane set, delay normalization scale, and spillback threshold
$\Pi_{\mathcal{A}}, \text{sg}$	Safety projection and stop-gradient operator

For the training reward in Eq. 2, let $\mathcal{L}$ denote the set of network lanes and let D_{scale} be the 95th percentile of network-average delay over normal-traffic training trajectories. The normalized terms are

$$d_t = \frac{\overline{\text{delay}}_t}{D_{\text{scale}}}, \qquad q_t = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \frac{\text{queue}_{t,\ell}}{\text{storage}_\ell}, \tag{17}$$

$$b_t = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \mathbf{1}\left[\frac{\text{queue}_{t,\ell}}{\text{storage}_\ell} \geq \tau_b\right], \qquad \tau_b = 0.9, \tag{18}$$

where $\overline{\text{delay}}_t$ is network-average delay in period t. For each active node–measure pair (v, m), $a^{\text{nominal}}_{v,m}$ is the nominal action and $a^{\min}_{v,m}, a^{\max}_{v,m}$ are its legal bounds. Control and switching costs are

$$c_t = \frac{1}{\max(1, |\mathcal{S}_t|)} \sum_{(v,m) \in \mathcal{S}_t} \left| \frac{a_{t,v,m} - a^{\text{nominal}}_{v,m}}{a^{\max}_{v,m} - a^{\min}_{v,m}} \right|, \qquad u_t = \frac{|\mathcal{S}_t \triangle \mathcal{S}_{t-1}|}{\max(1, |\mathcal{S}_t \cup \mathcal{S}_{t-1}|)}. \tag{19}$$

Thus, q_t, b_t, c_t, and u_t are dimensionless, while the fixed D_{scale} keeps d_t comparable across training trajectories.

A.2 Network Architecture and Candidate Construction

The temporal encoder has two Transformer layers with hidden dimension 128, four attention heads, and GELU activations. The graph encoder has three sparse graph-attention layers with hidden dimension 128, four heads, and edge gating. The Planner is a two-layer Set Transformer with hidden dimension 256 and four heads. The Executor is a shared two-layer MLP with hidden dimension 256 and separate heads for the three action families. The strategy-code generator is a two-layer 256–256 MLP that outputs a 128-dimensional code. Twin-Q and V heads are two-layer MLPs with hidden dimension 256. The final model has 3,674,560 trainable parameters; safety projection has no trainable parameters.

For every legal node–measure pair, the candidate scorer receives the corresponding node representation, global representation, measure embedding, active-set indicator, and recent-control-history embedding. It outputs one score per pair. The 64 highest-scoring legal pairs form $\mathcal{C}_t$. Before truncation, currently active pairs are inserted into the candidate pool if necessary, ensuring that all active measures remain available for DELETE or SWAP. The scorer is trained jointly with CAPE and is used only for candidate coverage; it is not a diagnosis or root-cause classifier.

A.3 Action Space, Safety Projection, and Inference Limits

Signal actions adjust green ratios, cycle lengths, offsets, and phase durations. Flow actions adjust entry or ramp release rates in $[0.4, 1.0]$, and path-diversion actions adjust legal alternative-route ratios in $[0, 0.3]$. Multiple measures can be active at one node, but at most 32 distinct nodes are active in one period. The final action is divided into three waves spanning 0–60, 60–120, and 120–180 seconds.

The deterministic safety projection maps the raw action to the feasible set under the current traffic-signal configuration. It enforces minimum green, yellow, and all-red durations; cycle-length bounds; allowable phase transitions; release-rate bounds; and diversion-ratio bounds. The projection is applied after the Executor generates actions and before the actions are sent to SUMO.

At inference, CAPE uses a 180-second outer period, at most 64 candidates, at most eight internal Planner edits, and at most one calibration event per period. STOP immediately terminates the current set-edit sequence. After calibration, planning resumes from the current partial set, allowing subsequent ADD, DELETE, SWAP, or STOP operations.

A.4 Training Configuration

Encoder initialization. The encoder is trained for 100,000 updates on normal, fault-event, nominal-control, bounded-random-control, and MaxPressure trajectories. Training uses batch size 256 and learning rate 3×10^{-4}. The objective combines masked Huber reconstruction with future-state prediction at one, three, and five control periods, weighted by 0.5, 0.3, and 0.2. The checkpoint with the lowest validation auxiliary loss is selected.

Planner–Executor cold start. The Planner, Executor, and value heads are initialized for 50,000 updates from paired edit trajectories with balanced support sampling. Discrete SAC is used for the Planner and continuous SAC for the Executor. The batch size is 256 and the learning rate is 10^{-4}. The checkpoint with the lowest validation total delay is selected.

Joint online training. All trainable modules are optimized for 200,000 online SUMO updates with batch size 512 and learning rate 3×10^{-4}. Fifteen percent of episodes use coverage sampling to increase exposure to infrequently visited control sets. The reward weights are 1.0 for delay, 1.0 for queue, 2.0 for spillback, 0.10 for control cost, and 0.05 for switching cost. Model selection uses the validation checkpoint with the lowest total delay subject to the safety constraints.

Triggered calibration. Each calibration uses matched SUMO probes, batch size 32, learning rate 0.05, and five strategy-code updates. Encoder, Planner, Executor body, and value heads are frozen. The calibrated candidate code is distilled into the generator, after which planning continues from the current partial set.

B Optimization and Calibration Details

B.1 SAC and Encoder Objectives

The Planner and Executor use standard discrete and continuous SAC, respectively, with twin Q-functions, target-value estimates, and entropy regularization. The Planner is trained on transitions over set operations, whereas the Executor is trained on transitions over safe continuous actions. Their objectives are denoted by $\mathcal{L}^{P}_{\mathrm{SAC}}$ and $\mathcal{L}^{E}_{\mathrm{SAC}}$ in Eq. 16.

The encoder auxiliary objective is

$$\mathcal{L}_{\mathrm{enc}} = \mathcal{L}_{\text{masked-recon}} + 0.5\mathcal{L}_{\mathrm{future},1} + 0.3\mathcal{L}_{\mathrm{future},3} + 0.2\mathcal{L}_{\mathrm{future},5}, \tag{20}$$

where masked reconstruction uses Huber loss and each future term predicts the traffic state at the indicated number of control periods. The paired effect loss in Eq. 13 jointly grounds the Planner and Executor advantages in the same matched simulator outcome.

B.2 Matched Probes and Local Code Adaptation

For a triggered candidate edit, the reference branch executes the current set $\mathcal{S}$ and the edited branch executes $\mathcal{S}^+$. The branches share the same network, demand, fault configuration, simulator state, random seed, recovery horizon, and frozen policy parameters. They differ only in the candidate set under comparison. The candidate-specific codes $z_{\mathcal{S}}$ and $z_{\mathcal{S}^+}$ are generated before computing A_E, ensuring that each execution value uses the context available for its own set. State cloning is available in our SUMO-based recovery setting and is not assumed to be directly available in live traffic.

Only the temporary candidate code $z_{\mathcal{S}^+}$ is trainable during local adaptation. It is initialized from the generator output,

$$z^{(0)} = G_\psi(H, g, \mathcal{S}^+). \tag{21}$$

For each of $B = 4$ matched probe pairs, let a_r^+ and J_r^+ be the action and return of the edited branch, let J_r^0 be the return of the reference branch, and define $y_r = J_r^+ - J_r^0$. Each branch starts from the cloned trigger state and runs through the remaining recovery horizon. The code objective uses the relative paired return as a policy-gradient weight:

$$\mathcal{L}_{\text{code}} = -\frac{1}{B}\sum_{r=1}^{B} y_r \log \pi_E(a_r^+ \mid x, \mathcal{S}^+, z). \tag{22}$$

Let $r^\star = \arg\max_r y_r$ identify the edited probe with the largest paired effect. The action and value objectives are

$$\mathcal{L}_{\text{action}} = \left\|\mu_E(x, \mathcal{S}^+, z) - a_{r^\star}^+\right\|_2^2, \qquad \mathcal{L}_{\text{value}} = \frac{1}{B}\sum_{r=1}^{B} \text{Huber}\left(V_E(x, \mathcal{S}^+, z) - V_E(x, \mathcal{S}, z_{\mathcal{S}}) - y_r\right), \tag{23}$$

where μ_E is the Executor mean action. The resulting local objective is

$$\mathcal{L}_{\text{cal}} = \mathcal{L}_{\text{code}} + 0.5\mathcal{L}_{\text{action}} + 0.2\mathcal{L}_{\text{value}}. \tag{24}$$

The four probe pairs form the calibration batch. Starting from $z^{(0)}$, CAPE performs five gradient updates with learning rate 0.05 while keeping the encoder, Planner, Executor body, and value heads frozen. The adapted candidate code is detached before generator distillation, so gradients do not flow through the local adaptation path. A calibration event is valid only for a non-STOP operation with $R > 0.63$ when no earlier calibration has occurred in the current period.

B.3 Calibration Procedure

For every proposed non-STOP edit, CAPE first evaluates the cached Planner and Executor advantages and computes R. If the trigger condition is not met, the edit is processed normally. Otherwise, CAPE runs four matched SUMO probe pairs, updates the temporary candidate code for five steps, distills the adapted code into the generator, recomputes the Executor advantage with the adapted code, and returns control to the Planner. No further calibration can be invoked during the same 180-second period. The trigger rate is reported at the recovery-period level: it is the fraction of recovery periods in which at least one calibration event is triggered. Each episode has ten 180-second recovery periods, and each period contributes at most one calibration event. The four paired probes reuse cached reference branches when available; after excluding the ordinary main trajectory and shared reference cache, one event contributes approximately six additional SUMO branch calls. Thus, for an episode i,

$$N_{\text{cal},i} = \sum_{p=1}^{10} \mathbf{1}[\text{calibration in period } p], \qquad C_i = \text{additional SUMO branch calls}, \tag{25}$$

and the reported quantities are the mean period-level trigger rate and mean C_i across the evaluation episodes. The call and rate values in the tables are computed from unrounded episode-level logs.

C Experimental Protocol

C.1 Scenario Construction and Evaluation Metrics

We use five SUMO networks: synthetic Grid-20x20, Cologne8 and Ingolstadt21 from RESCO (Ault & Sharon, 2021), LuST-Full (Codeca et al., 2017), and InTAS-Full (Lobo et al., 2020). For each network, we construct capacity drop, demand surge, signal fault, and compound fault scenarios. Each network–fault pair contains 100 pre-registered test scenarios. Five independently trained seeds are evaluated on the same scenario manifest, yielding 2,000 evaluation episodes per network and 10,000 episodes across the five networks for each method.

Each episode runs for 3600 seconds. The first 900 seconds establish the normal reference. A fault is injected at 900 seconds and remains untreated until 1800 seconds. Controllers operate only during the 1800–3600 second recovery interval. Capacity-drop faults reduce road capacity; demand-surge faults increase demand; signal faults invalidate the nominal signal program; and compound faults combine independently sampled component faults. Fault locations, onset times, severity ranges, demands, and random seeds are matched across methods.

For scenario i, let $D_{\mathrm{normal},i}$ and $D_{\mathrm{fault},i}$ denote the network-average delay in the normal reference window and at the end of the untreated-fault interval, respectively. For method m, let $D_{m,i}(t)$ denote the network-average delay in the 30-second evaluation window ending at recovery time t, where $t \in \{30, 60, \ldots, 1800\}$ seconds. The recovery gain at time t and the final Recovery Gain are defined as

$$\mathrm{RG}_{m,i}(t) = 100 \cdot \frac{D_{\mathrm{fault},i} - D_{m,i}(t)}{D_{\mathrm{fault},i} - D_{\mathrm{normal},i}}, \qquad \mathrm{RG}_{m,i} = \mathrm{RG}_{m,i}(1800). \tag{26}$$

The episode-level best recovery is

$$\mathrm{RG}^{\star}_{m,i} = \max_{t \in \{30,60,\ldots,1800\}} \mathrm{RG}_{m,i}(t), \tag{27}$$

and the time to 80% of that method-specific best recovery is

$$T_{80,m,i} = \min\left\{ t : \mathrm{RG}_{m,i}(t) \geq 0.8\, \mathrm{RG}^{\star}_{m,i} \right\}. \tag{28}$$

If the threshold is not reached during the recovery interval, $T_{80,m,i}$ is set to 1800 seconds. Recovery Gain and T_{80} are computed for each scenario and seed before aggregation across scenarios and seeds; the reported T_{80} is the resulting mean. Under this definition, the untreated-fault state has value 0%, while the normal reference has value 100%. Total delay, recovery time, throughput, control cost, and additional SUMO calls are computed under the same recovery window. The additional-call counter excludes the ordinary main trajectory and shared reference caching.

Latency is measured as the wall-clock time from receiving the current observation to emitting the legal action, after warm-up, on the same hardware and batch size for every method.

C.2 Baseline Adaptation and Tuning

All learning-based baselines receive the same structured traffic observations and operate over the same 180-second recovery periods, with the same legal action bounds, safety constraints, and validation-based tuning budget as CAPE. External methods are adapted only at the observation and action interfaces required to run them in the common SUMO recovery environment; their core policy, communication, and learning mechanisms are retained.

FixedTime uses its native legal signal schedule and requires no tuning. MaxPressure tunes only its pressure variant. CEM-MPC tunes its planning horizon, sample count, and temperature. MAPPO tunes its policy and optimizer settings. PressLight, CoLight, Advanced-MPLight, and GESA use a registered search space for the remaining hyperparameters. GraphMeta, NB-MARL, RMTC, and VLMLight follow the same protocol. All tuning is validation-based and shares the same budget across methods. The registered search spaces and selected hyperparameters are reported in the supplementary configuration files.

D SUPPLEMENTARY EXPERIMENTAL RESULTS

D.1 PER-FAULT RESULTS AND ADAPTATION COST

Table 7 reports Recovery Gain aggregated by fault type across the five networks. Each value is the mean with a 95% hierarchical bootstrap confidence interval. We compare each baseline with CAPE using a paired hierarchical bootstrap test and adjust p values within the experiment using the Benjamini–Hochberg procedure. CAPE is the best deployable method for every fault type. Compound faults are more difficult for all methods, but CAPE retains 84.39% Recovery Gain.

Table 7: Per-fault Recovery Gain (%) under the unified protocol. Values are means ± 95% confidence intervals.

Method	Capacity Drop	Demand Surge	Signal Fault	Compound Fault	p vs. CAPE
FixedTime	42.10 ± 2.4	39.52 ± 2.7	41.80 ± 2.5	38.42 ± 2.8	< 0.01
MaxPressure	66.52 ± 2.1	64.91 ± 2.3	66.08 ± 2.2	60.85 ± 2.6	< 0.01
CEM-MPC	65.41 ± 2.2	64.22 ± 2.4	64.79 ± 2.1	60.58 ± 2.7	< 0.01
MAPPO	68.52 ± 2.0	66.84 ± 2.2	68.10 ± 2.1	62.18 ± 2.5	< 0.01
PressLight	72.15 ± 1.8	71.02 ± 1.9	72.45 ± 1.8	65.78 ± 2.2	< 0.01
CoLight	74.22 ± 1.7	72.88 ± 1.8	74.52 ± 1.7	67.90 ± 2.1	< 0.01
Advanced-MPLight	75.40 ± 1.6	73.75 ± 1.8	75.12 ± 1.7	68.49 ± 2.0	< 0.01
GESA	76.51 ± 1.5	75.12 ± 1.7	76.15 ± 1.6	69.74 ± 1.9	< 0.01
NB-MARL	79.22 ± 1.4	78.50 ± 1.6	78.34 ± 1.5	72.42 ± 1.8	< 0.01
RMTC	81.15 ± 1.4	79.24 ± 1.5	79.52 ± 1.5	73.77 ± 1.8	< 0.01
GraphMeta	82.50 ± 1.3	80.12 ± 1.5	81.15 ± 1.4	74.95 ± 1.7	< 0.01
VLMLight	83.18 ± 1.3	81.45 ± 1.4	82.12 ± 1.3	75.45 ± 1.6	< 0.01
CAPE	$\mathbf{92.14 \pm 1.0}$	$\mathbf{90.52 \pm 1.2}$	$\mathbf{91.75 \pm 1.1}$	$\mathbf{84.39 \pm 1.4}$	–

D.1.1 ADAPTATION COST

Full-actor adaptation is an upper bound rather than a deployable baseline. Table 8 shows that it improves Recovery Gain by only 1.56 pp over CAPE while using $4.29\times$ as many additional SUMO calls. Additional calls count calibration-related probe rollouts per episode and exclude the ordinary main trajectory and shared reference caching.

Table 8: Recovery quality and online adaptation cost. Full-actor adaptation is an upper bound, not a deployable baseline.

Method	Recovery Gain (%)	Extra SUMO Calls	Relative Cost
CAPE	89.70	7.43	$1.00\times$
Full-actor adaptation	91.26	31.91	$4.29\times$

Final baseline configurations, tuning budgets, registered search spaces, per-network results, and scenario-level outputs are provided in Appendix C.2 and the supplementary configuration files.

D.2 MECHANISM ABLATION

Table 9 reports the Recovery Gain of the full model and each ablation on all five networks. The main-text average results show that every component contributes to recovery; the per-network results test whether this conclusion is driven by a single network. All variants are evaluated with the same fault scenarios, training budgets, random seeds, and safety constraints as the full CAPE model.

The full CAPE model outperforms every ablation on each network. In particular, removing paired effect alignment produces the largest recovery loss and is the weakest configuration on all five networks. This result supports the role of matched intervention outcomes in grounding the planning and execution value estimates. Scenario-level paired tests and bootstrap distributions are included in the supplementary result files.

Table 9: Per-network Recovery Gain (%) for CAPE and its ablations. Each entry reports the mean ± 95% confidence interval.

Network	CAPE	Flat Joint	w/o Edit	w/o Trigger	w/o Code	w/o Alignment
Grid-20x20	**89.73 ± 1.15**	86.52 ± 1.45	83.95 ± 1.65	85.34 ± 1.40	81.35 ± 1.85	78.82 ± 2.15
Cologne8	**88.52 ± 1.25**	85.15 ± 1.50	82.55 ± 1.70	84.12 ± 1.45	80.05 ± 1.95	77.25 ± 2.25
Ingolstadt21	**90.84 ± 1.10**	87.65 ± 1.35	85.12 ± 1.55	86.55 ± 1.30	82.42 ± 1.75	79.95 ± 2.05
LuST-Full	**90.27 ± 1.12**	87.14 ± 1.40	84.65 ± 1.62	86.08 ± 1.35	81.85 ± 1.80	79.28 ± 2.10
InTAS-Full	**89.15 ± 1.18**	85.79 ± 1.42	83.13 ± 1.72	84.51 ± 1.42	80.53 ± 1.88	77.95 ± 2.08

D.3 Strategy-Code Calibration

This appendix provides the complete calibration diagnostics, adaptation budgets, code-reuse results, dimension sweeps, per-fault trigger results, and threshold sensitivity results summarized in Section 4.4.

D.3.1 Exposure Bias

Table 10 reports sign reversal and regret under controlled exposure ratios. This is an offline matched-edit diagnostic and does not use Recovery Gain.

Table 10: Value-estimation error under controlled training exposure. Lower values are better.

Exposure Ratio (Freq:Rare)	Sign Reversal (%) ↓	Regret (%) ↓
1:1 (Balanced)	7.36 ± 0.42	2.83 ± 0.15
4:1 (Uncommon)	16.68 ± 0.85	6.14 ± 0.32
8:1 (Rare)	21.80 ± 0.98	7.70 ± 0.41
16:1 (Very Rare)	26.31 ± 1.14	9.29 ± 0.58
32:1 (Extreme)	29.40 ± 1.28	10.30 ± 0.66

D.3.2 Adaptation Budget

Table 11 reports the recovered matched-effect gain and extra SUMO calls at different update budgets. Increasing the budget from five to ten updates recovers an additional 4.92 pp, but nearly doubles the extra SUMO calls from 7.33 to 13.07. At the same five-update budget, Full-actor adaptation reaches only 0.82 pp above the 128-dimensional strategy code while requiring $2.17\times$ as many calls. The strategy code therefore captures most of the achievable gain at a small fraction of the adaptation cost.

Table 11: Adaptation budget for the 128-dimensional strategy code. Higher recovered gain and fewer extra SUMO calls are better.

Setting	Recovered Gain (%) ↑	Extra SUMO Calls ↓
5 updates	86.95	7.33
10 updates	91.87	13.07
Full-actor (5 updates)	87.77	15.90

D.3.3 Code Reuse

Table 12 reports the retained matched-effect gain after distillation. The distilled generator recovers 80.54% of the matched effect, retaining 92.63% of the temporary code's gain, whereas a random generator retains only 23.47%. Distillation therefore transfers most of the adapted behaviour into the generator rather than collapsing to a random mapping. The Oracle row reports the upper bound obtained when the adapted code is used directly without distillation.

Table 12: Strategy-code reuse after distillation. Retention is measured relative to the temporary code.

Generator	Recovered Gain (%) ↑	Retention (%) ↑
Random	20.41	23.47
Distilled	80.54	92.63
Temporary	86.95	100.00
Oracle	98.31	–

D.3.4 Full Dimension Sweep

Table 13 reports the recovered matched-effect gain for all code dimensions and update budgets. Larger codes recover more gain at every budget, and the gap between the 64-d and 128-d codes widens from 2.34 pp at one update to 6.37 pp at ten updates. Even at a single update, the 128-d code already reaches 59.19%, showing that a higher-dimensional code can express useful adaptation before many gradient steps are taken.

Table 13: Recovered matched-effect gain (%) across code dimensions and update budgets.

Code Dimension	1 update	3 updates	5 updates	10 updates
32-d	47.32	64.86	73.68	76.98
64-d	53.07	70.96	82.25	85.50
128-d	59.19	75.61	86.95	91.87

D.3.5 Per-Fault Trigger Results

Table 14 reports Recovery Gain and trigger rate under each fault type. Compound faults are the most difficult case: they obtain the lowest Recovery Gain and the highest trigger rate, because their congestion patterns combine capacity loss, demand pressure, and signal disruption, which causes the Planner and Executor to disagree more often. The other three fault types are easier and trigger calibration less frequently, which is consistent with the overall average of 89.70% Gain and 7.43 extra calls.

Table 14: Advantage-disagreement calibration under each fault type.

Fault Type	Recovery Gain (%) ↑	Trigger Rate (%)	Extra SUMO Calls ↓
Capacity drop	92.14 ± 0.85	8.5	6.23
Demand surge	90.86 ± 0.92	9.2	6.74
Signal fault	91.35 ± 1.04	10.4	7.62
Compound fault	84.45 ± 1.79	12.5	9.13

D.4 Control, Trigger, and Reward Sensitivity

D.4.1 Candidate-Pool Sensitivity

Table 15 reports the complete candidate-pool sweep. Candidate recall increases with the pool size, whereas Recovery Gain saturates after 64 candidates. The final configuration uses 64 candidates.

The corresponding peak GPU memory values are 9.18, 9.30, 9.47, 9.82, 10.53, and 12.10 GB for 8, 16, 32, 64, 128, and 256 candidates, respectively. The complete resource records are retained in the supplementary configuration files.

Table 15: Candidate-pool sensitivity on Grid-20x20 under the unified recovery protocol.

Candidates	Recovery Gain (%)	95% CI	Recall (%)	Planner (ms)	Total (ms)
8	85.00	±1.85	83.50	17.20	168.50
16	87.42	±1.80	88.42	22.43	172.20
32	88.85	±1.45	89.46	25.02	173.80
64	89.70	±1.15	91.54	30.21	174.84
128	89.81	±1.22	95.70	40.58	189.30
256	89.83	±1.20	97.85	75.00	225.10

D.4.2 Trigger and Reward Sensitivity

Tables 16 and 17 report sensitivity to the trigger threshold and reward specification. The selected threshold is $R > 0.63$. Lower thresholds invoke more calibration, whereas higher thresholds reduce calls but can miss useful calibration events. The default reward gives the strongest recovery, while deterministic projection keeps safety violations below 0.01% in the registered runs.

Table 16: Sensitivity to the advantage-disagreement threshold.

Threshold	Recovery Gain (%)	Extra Calls	Calibration Rate (%)
0.50	89.75 ± 1.10	15.21	25.4
0.55	89.73 ± 1.12	12.14	20.2
0.60	89.72 ± 1.14	9.55	15.8
0.63	89.70 ± 1.15	7.43	12.4
0.65	89.15 ± 1.18	6.12	10.1
0.70	87.82 ± 1.25	3.85	6.3
0.75	85.45 ± 1.40	2.10	3.5

Table 17: Reward and safety sensitivity.

Reward Setting	Recovery Gain (%)	Control Cost	Safety Violations	Extra Calls
Default	89.70 ± 1.15	$1.00\times$	$< 0.01\%$	7.43
Delay half	86.25 ± 1.32	$0.90\times$	$< 0.01\%$	7.21
Queue half	87.14 ± 1.25	$0.95\times$	$< 0.01\%$	7.35
Spillback half	85.33 ± 1.45	$0.85\times$	$< 0.01\%$	7.12
Control cost doubled	84.12 ± 1.50	$0.60\times$	$< 0.01\%$	6.50

D.5 Generalization, Robustness, and Case Traces

This appendix reports out-of-distribution generalization, observation robustness, and period-level case traces. Runtime and resource measurements are reported in Table 5 in the main text.

D.5.1 Out-of-Distribution Generalization

Table 18 reports Recovery Gain under three out-of-distribution conditions: demand shift, unseen fault severity, and unseen compound faults. The three conditions stress different parts of the policy. Demand shift changes the volume of vehicles but keeps the fault structure familiar; unseen severity keeps the fault type but changes its intensity; unseen compound faults combine both, and therefore require the policy to generalize along two axes at once.

The ordering of conditions is consistent across all methods. Demand shift is the easiest condition, unseen severity is harder, and unseen compound faults are the hardest. This ordering holds for the weakest baseline MAPPO (64.5, 61.2, 53.8) as well as for the strongest baseline VLMLight (79.2, 75.1, 67.9). The gap between demand shift and unseen compound faults is about 10 pp for every method, which suggests that fault composition matters more than traffic volume when the policy is tested outside its training distribution.

CAPE retains 88.2%, 85.4%, and 79.8% under the three conditions, and exceeds the strongest baseline in each setting by 9.0–11.9 pp. The largest margin appears under unseen compound faults, where CAPE keeps nearly 80% of the fault-induced loss while VLMLight falls to 67.9%. This is the setting where the control scope matters most: compound faults create multiple interacting bottlenecks whose relative importance shifts over time, so a policy that selects control locations from the current state transfers better than one tied to a fixed scope. The margin under demand shift is smaller but still above 9 pp, showing that the gain is not limited to the most difficult condition.

Table 18: Out-of-distribution Recovery Gain (%). Values are means $\pm$ 95% confidence intervals.

Method	Demand Shift	Unseen Severity	Unseen Compound Fault
MAPPO	64.5 ± 2.1	61.2 ± 2.5	53.8 ± 2.8
NB-MARL	75.1 ± 1.7	71.8 ± 2.2	65.2 ± 2.4
RMTC	76.8 ± 1.6	73.5 ± 2.0	66.4 ± 2.6
GraphMeta	77.5 ± 1.8	74.2 ± 2.1	67.5 ± 2.5
VLMLight	79.2 ± 1.9	75.1 ± 1.8	67.9 ± 2.5
CAPE	$\mathbf{88.2 \pm 1.2}$	$\mathbf{85.4 \pm 1.5}$	$\mathbf{79.8 \pm 1.8}$

D.5.2 Observation Robustness

Table 19 reports Recovery Gain under clean, noisy, and incomplete observations. The perturbations simulate two common sensor-side failures: additive Gaussian noise that corrupts every observed value, and random missingness that removes a fraction of the observed entries. Both are applied to the same structured observation used by all methods, so the comparison reflects the robustness of the policy rather than the quality of the sensing pipeline.

Under clean observations, CAPE reaches 89.7%, which is the highest value in the table and matches the main-result average. As noise increases to 10% and 20%, CAPE decreases to 86.5% and 81.2%. The degradation is gradual, and CAPE keeps a margin of more than 11 pp over the strongest baseline at both noise levels. Under 10% missingness, CAPE reaches 83.5%, still above every baseline. These results indicate that the Planner's candidate scoring and the Executor's action generation are both reasonably tolerant of moderate observation corruption.

The single setting where CAPE is not the best is 30% missingness. Here RobustLight reaches 78.5% while CAPE obtains 75.4%. The result is consistent with RobustLight's explicit observation-reconstruction objective: when a large fraction of the input is missing, reconstructing the missing signal becomes more valuable than adapting the control scope, because the policy first needs a usable state estimate before it can decide where to intervene. CAPE's main advantage is largest under clean and moderately corrupted observations, where scope revision is the dominant factor. The two methods therefore address different failure modes, and their relative ordering depends on how severe the observation corruption is.

Table 19: Recovery Gain (%) under observation perturbations. Values are means $\pm$ 95% confidence intervals.

Method	Clean	Noise 10%	Noise 20%	Missing 10%	Missing 30%
MAPPO	66.4 ± 2.0	59.8 ± 2.4	51.2 ± 2.9	55.4 ± 2.5	45.3 ± 3.5
NB-MARL	77.1 ± 1.6	71.2 ± 2.0	64.5 ± 2.5	68.5 ± 2.3	58.2 ± 3.1
RMTC	78.4 ± 1.7	72.8 ± 1.8	65.4 ± 2.4	69.1 ± 2.0	60.1 ± 2.7
GraphMeta	79.7 ± 1.5	73.1 ± 1.9	66.8 ± 2.3	70.2 ± 2.1	61.5 ± 2.8
VLMLight	80.6 ± 1.4	75.2 ± 1.7	68.1 ± 2.1	72.4 ± 1.9	63.8 ± 2.6
RobustLight	78.5 ± 1.8	76.2 ± 1.7	73.5 ± 1.9	75.8 ± 1.8	$\mathbf{78.5 \pm 2.1}$
CAPE	$\mathbf{89.7 \pm 1.1}$	$\mathbf{86.5 \pm 1.4}$	$\mathbf{81.2 \pm 1.9}$	$\mathbf{83.5 \pm 1.5}$	75.4 ± 2.2

D.5.3 Representative Case Trace

The case trace reports period-level Recovery Gain, control-set operations, root and non-root node counts, selected measures, A_P, A_E, disagreement score R, and calibration events. The full quantitative record is retained in the supplementary result files; the main-text figure visualizes the same

episode at the primary-road, secondary-road, and snapshot level. The two tables above and the main-text case figure therefore serve different purposes: the tables quantify how the policy behaves across conditions and corruption levels, while the case trace shows what one closed-loop recovery actually looks like in terms of queue evolution, control-set revision, and calibration.