Soft Computing Fusion with Applications

www.scfa.reapress.com

Soft. Comput. Fusion. Appl. Vol. 3, No. 4 (2026) 260–271.

Paper Type: Original Article

Q Learning Fixed Time and Actuated Signal Control under Simulated Demand Shifts

Zhang Hao[1], Zongke Bao[2,*], Lulu Cai[3]

[1] Institute of Education, Guizhou Normal University, Guizhou Province, China; haozhang_guizhou@sina.com.
[2] School of Accounting, Zhejiang University of Finance and Economics, 310018 Hangzhou, China; zongkebao@mail.cnas.ac.cn.
[3] Department of Electrical Automation, Quzhou University, Quzhou 324000, China; ysu-fbg@163.com.

Citation:

Received: 16 March 2026 *Revised: 05 June 2026* *Accepted: 10 July 2026*	Hao, Zh., Bao, Z., & Cai, L. (2026). Q learning fixed time and actuated signal control under simulated demand shifts. *Soft Computing Fusion with Applications, 3*(4), 260-271.

Abstract

Demand shifts can change the performance of a learned traffic-signal controller. This study reports a point-queue corridor simulation with six intersections, two-hour episodes, and Poisson arrivals. A single trained decentralized tabular Q-learning policy is compared with fixed-time and gap-out actuated controllers over nine demand regimes, three excluded from training, with twenty evaluation arrival realizations per regime. On training regimes the reported Q-learning delay reductions relative to fixed time range from 24.8% to 34.1%, versus 49.4% to 55.1% for actuated control. On held-out high arterial demand, Q-learning delay is 180.9 s per admitted vehicle versus 30.5 for fixed time, with throughput 94.9%. Actuated control has the lowest displayed delay in every regime. These results describe the reported simulator and one learned policy. Unbounded accumulation of unused discharge credit, service bypass for arrivals on empty green, unequal sensing and green constraints, and an unresolved green-share denominator prevent treating the comparison as validated physical capacity or field evidence. Independent simulator and policy replications are required before attributing the failure to a particular mechanism.

Keywords: Reinforcement learning, Traffic signal control, Urban corridor, Q-learning, Actuated control, Traffic simulation.

1 | Introduction

Urban road congestion concentrates at signalized intersections, and along urban arterials the coordination — or at least the consistency — of signal operation is one of the few levers that a road authority can actuate at low capital cost. Signal operation is therefore a classical instance of critical infrastructure control, in which locally rational decisions propagate through a spatially extended system and jointly determine network-wide performance [1]. Within the smart-cities research agenda, adaptive signal control occupies a privileged

Corresponding Author: zongkebao@mail.cnas.ac.cn
https://doi.org/10.22105/scfa.v3i4.115
Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

position because the underlying decision problem is well posed (minimize delay, queues, or emissions subject to safety constraints), yet its empirical solution on real networks remains difficult.

Conventional practice rests on two families of controllers. Fixed-time plans are computed offline from historical flow measurements using classical timing theory, most influentially the cycle-length and green-split formulae of Webster [2]. They require no detectors and no online computation, but they are blind to demand variation between plan updates. Actuated controllers, by contrast, extend or terminate green intervals in response to detector occupancy, which makes them responsive to short-term demand fluctuations without any explicit traffic model. Between these families lies a broad spectrum of adaptive schemes, and in recent years Reinforcement Learning (RL) has become the most actively explored candidate for populating that spectrum.

RL formulates signal control as a sequential decision problem in which an agent observes a state of the intersection, chooses an action such as extending or terminating the current green, and receives a reward such as the negative of the total queue length. Tabular Q-learning, introduced by Watkins and Dayan [3] and presented comprehensively by Sutton and Barto [4], is the simplest instance of this idea: it requires no traffic model, no forecasting module, and no neural network, which makes it attractive for constrained roadside hardware and for interpretability. Deep multi-agent extensions have since produced impressive in-simulation performance on both synthetic and real-world data.

A learned policy may encounter demand levels and directional compositions outside its training distribution. Such tests reveal behavior that in-distribution evaluation does not measure. The present study examines one simple tabular design, without establishing that all existing studies omit shifts or that the design is deployment-ready.

The reported experiment uses a point-queue corridor with six intersections, 7,200 one-second steps per episode, and a nominal saturation parameter 1,800 vehicles per hour per approach. Nine demand regimes cross levels and directional biases, with six used in training and three held out. Each controller is evaluated on twenty arrival realizations per regime. Conditional bootstrap and rank-test summaries quantify evaluation variability, but not variability across trained policies or simulator implementations.

The study combines fixed-time, actuated, and learned controls with an explicit held-out demand set. A reported conservation check finds zero vehicle-count discrepancy across 540 evaluation episodes, but conservation does not validate the discharge-capacity rule. The large delay under held-out arterial demand motivates inspecting state abstraction and policy timing; those diagnostics suggest hypotheses rather than isolate causes.

2 | Related Work

Webster timing relates an isolated-intersection cycle and splits to critical flow ratios under its approximate delay assumptions [2]. It is a standard engineering comparator, not universally optimal for a stochastic corridor with uncoordinated offsets. Directional demand changes and coordination can affect performance.

Point-queue models trade vehicle-level detail for speed. They differ from microscopic car-following simulation and from finite-storage cell-transmission models [5]. Queue integrals can summarize simulated waiting, but traffic-capacity fidelity depends on the discharge rules. Omitting spillback, turning, and platoon dynamics does not have a universally conservative effect.

Between fixed-time plans and learning-based controllers lies model-based adaptive control, of which constrained model predictive control is the most developed exemplar [6]. These methods optimize signal sequences over a rolling horizon from an explicit prediction model; their performance therefore depends on model quality, and their computational load scales with the prediction horizon and network size. Actuated control occupies the opposite, model-free corner: a gap-out rule terminates green when detectors report no waiting or arriving vehicles, subject to minimum and maximum green constraints. Actuated logic is local, cheap, and — as this study confirms — a demanding baseline for any learned controller.

RL-based signal control originates with tabular methods. Multi-agent deployments such as the system of El-Tantawy et al. [7] demonstrated intersection-level Q-learning agents that coordinate implicitly through the environment. The experimental review of Mannion et al. [8] surveyed RL algorithms for adaptive signal control and highlighted two recurrent weaknesses of the literature: heterogeneous simulation environments that impede comparison, and evaluation protocols dominated by in-distribution demand. The availability of large-scale traffic data and neural-network function approximation [9] subsequently enabled deep RL controllers such as IntelliLight [10], which learns expressive state-action mappings beyond the reach of lookup tables. Across this spectrum, tabular Q-learning retains two advantages that motivate revisiting it: its policy can be inspected cell by cell, and it can be executed on hardware with no floating-point accelerator.

The present gap is a scoped comparison of one learned tabular controller with two engineering references across an explicit demand split. It does not claim that asymmetric shift failure is absent from all prior work. The simulator and comparator information sets must be verified before conclusions are generalized.

3 | Research Gap and Questions

The question is whether this tabular implementation retains low delay when demand shifts outside its training set. The network and nominal demand streams are held fixed across reported controller comparisons, while controller sensing and phase guards differ. Thus a measured difference reflects the complete designs, not only their decision rules.

RQ1 asks how the learned controller performs on training demand regimes; RQ2 asks how it transfers to excluded regimes relative to fixed-time and actuated control. H1 predicts lower delay than fixed time on training regimes, and H2 predicts that the advantage persists under the held-out shifts. These are designated comparison questions; a dated pre-evaluation protocol is not established. Rank-test summaries are conditional on the reported simulator, arrival realizations, and one trained policy.

4 | Materials and Methods

4.1 | Corridor Point Queue Simulator

The simulator represents six intersections S0–S5 along an east–west corridor, with 200 m links, nominal speed 50 km/h, and rounded travel time 14 s. Fourteen external streams generate independent Poisson arrivals: two arterial boundary entries and two cross-street directions at each intersection. It omits turning, spillback, and offsets. These are simplifying assumptions, without a demonstrated direction of bias.

Each intersection has four point queues. The described implementation passes arrivals through an empty green queue without charging saturation service and otherwise adds them to a queue. Queued service uses credit incremented by 0.5 vehicles per green second and never reset. Accumulating unused service while a queue is empty can later permit discharge exceeding the intended per-second service capacity; empty-green pass-through can also bypass that capacity. The nominal saturation parameter therefore does not establish a capacity-limited simulator. Arterial departures are delayed downstream by 14 s, and cross-street departures exit immediately. The reported end-of-episode conservation identity is:

$$N_{\text{admitted}} = N_{\text{completed}} + N_{\text{queued}} + N_{\text{in transit}}. \quad (1)$$

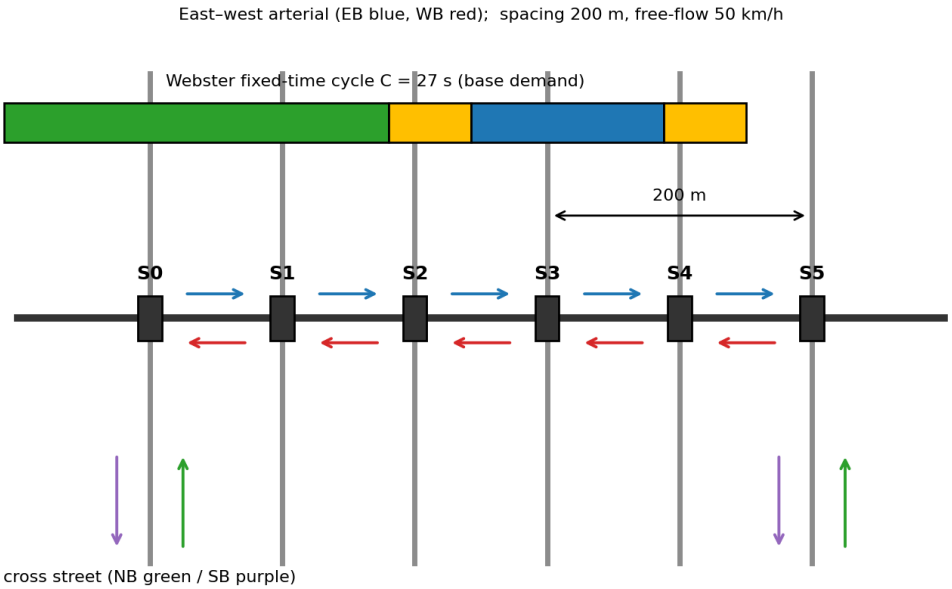


Fig. 1. Simulated corridor (S0–S5) and the Webster fixed-time cycle computed from the base demand.

Arterial and cross-street demands are independent Poisson streams; link travel time is 14 s (200 m at 50 km/h); the Webster cycle is C = 27 s with g_EW = 14 s and g_NS = 7 s (*Fig. 1*).

4.2 | Signal Phases and Baseline Controllers

Every intersection cycles through four phases: east–west green, amber, north–south green, amber, with an amber interval of 3 s. The fixed-time baseline applies the Webster design computed from the base balanced demand and holds it constant in all regimes. With critical flow ratios y_EW = 0.333 and y_NS = 0.156 (Y = 0.489) and lost time L = 6 s, the Webster cycle formula C = (1.5L + 5)/(1 − Y) yields 27.4 s, rounded to C = 27 s with green times g_EW = 14 s and g_NS = 7 s. The implied approach capacities are 933 veh/h (arterial) and 467 veh/h (cross street). All intersections run identical timings with zero offsets, so the fixed-time baseline is an uncoordinated Webster plan rather than a progression-optimized one; this is stated explicitly because it affects the interpretation of the baseline's absolute delay.

$$C = \frac{1.5L + 5}{1 - Y}, \quad s_{EW} = s\frac{g_{EW}}{C}, \quad s_{NS} = s\frac{g_{NS}}{C}. \tag{2}$$

The actuated controller uses current arrivals or standing queues as detector inputs, minimum green 8 s, gap-out 3 s, and maximum green 22/12 s for EW/NS. The Q-learning state described below includes binned queues, phase, and elapsed time, but not current arrival detections. Its 10/60 s guards also differ. These are comparisons of different sensing/action designs rather than matched-information decision rules.

4.3 | Q-learning Controller

The learned controller consists of 6 decentralized tabular Q-learning agents, one per intersection, that act simultaneously but share no parameters or communications; coordination can emerge only through the coupled environment. Each agent observes a discretized local state s = (phase direction, time-in-phase bin, arterial queue bin, cross-street queue bin): phase direction is binary (EW- or NS-oriented), time-in-phase is binned at 10 and 30 s, and each queue pair is binned with edges [1], [5], [11], giving 2 × 3 × 4 × 4 = 96 states. The action space is binary — 2 actions: keep the current phase or switch (after amber) — with a minimum green of 10 s and a forced switch at 60 s. The one-step reward is the negative total queue length at the intersection, r = −(q_EW + q_NS), which makes the learning objective a proxy for local delay.

The six agents use learning rate α=0.15 and discount γ=0.95. Behavior is ε-greedy, with ε=0.30×0.96^n after n episodes, giving approximately 0.0076 after ninety. Training comprises fifteen episodes per training regime,

shuffled across six regimes, with fresh seeded arrivals. The reported evaluation freezes the Q-tables and sets ε=0. The standard Q-update is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]. \quad (3)$$

4.4 | Evaluation Metrics

Delay is the sum of queued vehicle-seconds divided by admitted vehicles, excluding free-flow travel and residual future waiting after the two-hour horizon. A low-throughput episode may therefore leave additional delay uncounted. Mean/max queues and completion throughput supplement the metric. An EW share must specify whether its denominator is the full episode or green time only; the reported fixed-time value indicates a full-cycle fraction. Episode bootstrap intervals are conditional on this simulator and the fixed learned policy:

$$\bar{D} = \frac{\sum_t \sum_a q_a(t)\,\Delta t}{N_{\text{admitted}}}, \quad \Delta\% = 100\,\frac{D_{\text{fixed}} - D_{\text{controller}}}{D_{\text{fixed}}}. \quad (4)$$

5 | Experimental Setup

Demand scenarios cross three demand levels — low (×0.65), base (×1.00), and high (×1.20) — with three directional biases: balanced (1.00, 1.00), EW-heavy (1.25, 0.80), and NS-heavy (0.80, 1.25), where the pairs are multipliers on (arterial, cross-street) base demands of 600 and 280 veh/h. The resulting nine regimes are split by a fixed protocol: Low / balanced, Low / EW-heavy, Base / balanced, Base / NS-heavy, High / balanced, High / NS-heavy are used for training, while Low / NS-heavy, Base / EW-heavy, High / EW-heavy are held out. The held-out set therefore contains one cross-street-heavy regime at low demand and the two arterial-heavy regimes at base and high demand, which lets us test generalization in both directional directions. All levels apply the same multipliers to both travel directions, so no level implies directional asymmetry by itself.

The described seed scheme uses base seed 42, training arrivals [42,2,k], evaluation batches [42,3,j], and bootstrap seeds [42,7000,n]. A common-random-number comparison requires exogenous arrival arrays to be identical across controllers, independently of their actions and other RNG calls. Twenty episodes per regime and 10,000 bootstrap resamples quantify conditional arrival variability. They do not estimate training-seed variability, because only one final learned policy is described. Holm adjustment covers 27 per-regime contrasts.

The source reports 168 s for training and 213 s for the complete pipeline. Reproduction requires actual code, environment, streams, learned Q-tables, and timing logs. Policies are frozen for evaluation. Matching arrival batches does not resolve the sensing, phase-guard, and service-capacity differences.

Table 1. Simulation, demand, and controller parameters (synthetic environment, fixed seeds).

Parameter	Value
Intersections on arterial	6 (S0–S5), 200 m spacing
Episode length / time step	7,200 s (2 h) / 1 s
Saturation flow per approach	1,800 veh/h (0.5 veh/s)
Link travel time / speed	14 s / 50 km/h
Amber interval	3 s (lost time 6 s)
Base arterial demand (per direction)	600 veh/h
Base cross-street demand (per node, per direction)	280 veh/h
Demand level multipliers (low/base/high)	×0.65 / ×1.00 / ×1.20
Directional bias multipliers (arterial, cross)	balanced (1.00, 1.00); EW-heavy (1.25, 0.80); NS-heavy (0.80, 1.25)
Webster fixed-time (from base balanced demand)	C = 27 s; g_EW = 14 s; g_NS = 7 s; zero offsets
Actuated control	min green 8 s; gap-out 3 s; max green 22/12 s (EW/NS)
Q-learning: agents / states / actions	6 / 96 / 2
Q-learning: state bins	queue edges [1, 5, 13] veh; time-in-phase edges 10/30 s
Q-learning: α / γ / ε schedule	0.15 / 0.95 / 0.30 × 0.96^ep (final 0.0076)
Q-learning: green constraints	min 10 s, max 60 s
Training / evaluation protocol	15 episodes per training regime (90 total); 20 evaluation episodes per regime × controller; seed 42

6 | Results

6.1 | Simulator Validation

The reported single-approach check measures 0.2667 vehicles per second under a fixed-time plan, versus nominal 0.2599, ratio 1.026. This one oversaturated test does not rule out unused-credit storage or empty-green bypass in other regimes. A conservation discrepancy of zero checks counting, not bounded service. The bias need not be equal across controllers because their green schedules and empty-queue durations differ. Capacity must be tested over empty-to-busy transitions, bursts, repeated cycles, and every control design before traffic-validity claims.

6.2 | Training Behaviour

Fig. 2 shows ninety training episodes and the ε schedule, decreasing from 0.30 to about 0.0076. The reported first/last nine-episode queue averages are 16.7 and 22.9, with mixed regime difficulty. This curve alone does not establish convergence. Evaluation comparisons describe the final policy conditional on the simulator.

The source reports high-balanced training spikes at episodes 42,82,89, with mean queues 30.4,44.2,53.7. These suggest unstable behavior for the sampled episodes, but identifying starvation or catastrophic forgetting requires phase/action traces and repeated training runs.

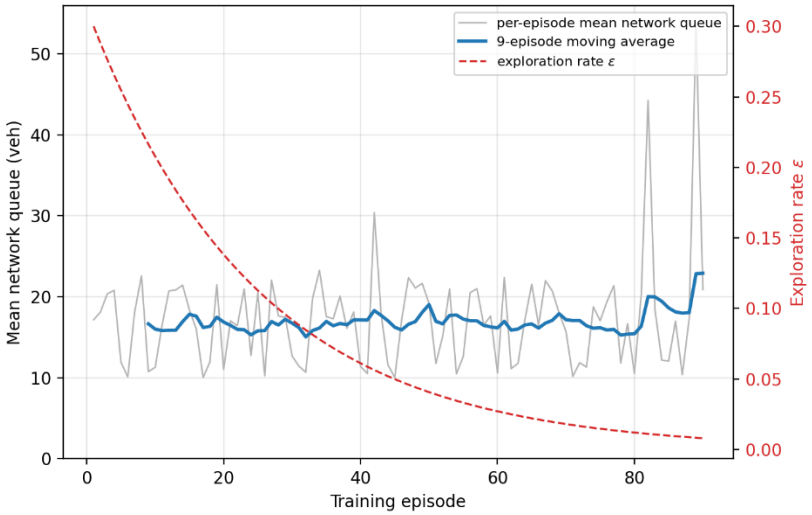


Fig. 2. Training curve of the decentralized Q-learning agents: per-episode mean network queue (grey), 9-episode moving average (blue), and exploration rate ε (red, dashed, right axis). Late-training spikes occur in the high balanced regime.

6.3 | Delay by Demand Regime

Table 2 preserves the reported delay and throughput summaries. Q-learning has lower mean delay than fixed time in seven of nine regimes, while actuated control has the lowest displayed mean in every regime. The two held-out EW-heavy regimes show the largest learned-policy delays. These rankings are conditional on the unresolved service-capacity rules and unequal comparator designs.

Table 2. Reported delay per admitted vehicle, bootstrap intervals, percentage improvement versus fixed time, and Q-learning throughput.

Regime	Fixed (s/veh)	Actuated (95% CI)	Q-learning (95% CI)	Δ Act vs Fixed	Δ QL vs Fixed	QL Throughput
Low/balanced	23.5	11.1 (11.0–11.2)	15.9 (15.5–16.3)	+52.8%	+32.2%	99.5%
Low/EW-heavy	29.0	13.0 (12.8–13.2)	19.9 (19.3–20.6)	+55.1%	+31.3%	99.3%
Low/NS-heavy †	19.1	9.4 (9.3–9.5)	12.9 (12.7–13.2)	+50.5%	+32.1%	99.6%
Base/balanced	23.8	11.2 (11.0–11.4)	16.1 (15.6–16.5)	+53.0%	+32.4%	99.5%
Base/EW-heavy †	29.4	12.0 (11.7–12.3)	40.4 (32.8–48.6)	+59.1%	−37.5%	98.9%
Base/NS-heavy	19.5	9.8 (9.7–9.9)	12.9 (12.7–13.2)	+49.4%	+33.6%	99.6%
High/balanced	24.1	10.9 (10.8–11.1)	18.1 (17.1–19.3)	+54.6%	+24.8%	99.5%
High/EW-heavy †	30.5	12.0 (11.7–12.2)	180.9 (162.8–198.8)	+60.8%	−493.2%	94.9%
High/NS-heavy	20.3	10.1 (10.0–10.2)	13.4 (13.1–13.6)	+50.3%	+34.1%	99.6%
All episodes (pooled)	24.3	11.1	36.7	+54.6%	−50.9%	—

Note: Positive Δ denotes lower delay; negative Δ denotes higher delay. † indicates a held-out regime. Twenty paired evaluation episodes per cell.

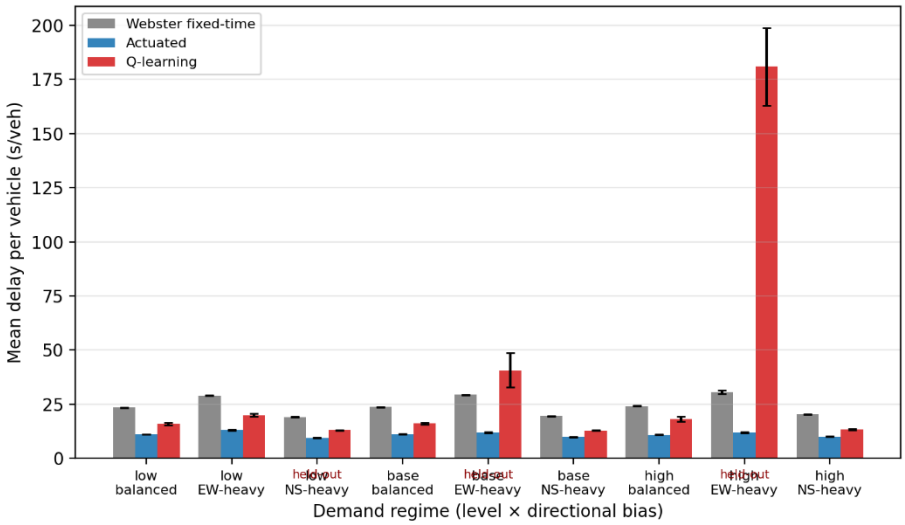


Fig. 3. Reported mean delay per admitted vehicle by demand regime for Webster fixed-time, actuated, and Q-learning control (bootstrap 95% confidence intervals; † marks held-out regimes).

The learned policy has higher delay than fixed time in both held-out EW-heavy regimes, especially the high-demand regime; these are outcomes of the stated simulator and controller settings, whose effective service capacity remains unverified (*Fig. 3*).

6.4 | Generalization to Held-Out Regimes

Under held-out low NS-heavy demand, Q-learning delay is 12.9 versus 19.1 s/vehicle for fixed time. Under base EW-heavy demand it is 40.4 versus 29.4, and under high EW-heavy demand 180.9 versus 30.5, with 94.9% throughput and mean queue 252.5. This rejects a uniform held-out advantage for this policy in the reported simulator. The displayed held-out +56.8% actuated and −166.2% Q-learning values are arithmetic means of the three regime-specific percentage improvements, not percentages computed from the held-out pooled means 11.1,78.1,26.3. Different aggregations answer different questions.

Table 3. Reported regime-group means.

Regime Group	Fixed (s/veh)	Actuated (s/veh, Δ vs Fixed)	Q-learning (s/veh, Δ vs Fixed)	QL Throughput	QL Mean Queue (veh)
Training regimes (mean of 6)	23.3	11.0 (+52.5%)	16.1 (+31.4%)	99.5%	19.6
Low/NS-heavy	19.1	9.4 (+50.5%)	12.9 (+32.1%)	99.6%	12.1
Base/EW-heavy	29.4	12.0 (+59.1%)	40.4 (−37.5%)	98.9%	47.0
High/EW-heavy	30.5	12.0 (+60.8%)	180.9 (−493.2%)	94.9%	252.5
Held-out regimes (mean of 3)	26.3	11.1 (+56.8%)	78.1 (−166.2%)	97.8%	103.9

Note: Percentage improvements average the regime-specific percentages; they are not ratios of the group mean delays. Positive denotes improvement.

6.5 | Queue Evolution in an Example Episode

Fig. 4 shows queue traces for a base-balanced example. Their periodic patterns are consistent with different phase schedules, but this single episode does not identify a unique mechanism. The source high-EW-heavy summaries include mean queue 252.5 and unfinished share 5.1%; queue-area delay omits the unfinished vehicles' future waiting.

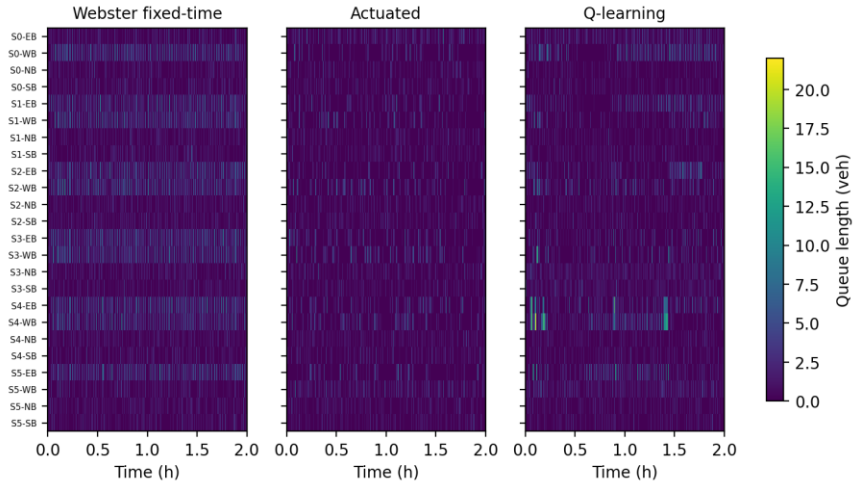


Fig. 4. Approach queues (vehicles) over time in the base/balanced example episode for the three controllers (rows S0–S5; identical stochastic arrivals). Darker cells indicate longer queues.

6.6 | Structure of the Learned Policy

Fig. 5 opens the learned controller for inspection. The left panels map the mean difference Q(s, switch) − Q(s, stay) over the queue-bin grid during each green orientation. The policy is qualitatively sensible: during east–west green the switch advantage falls to -70.6 when the EW queue bin is highest and the NS bin is low (i.e., extend arterial green under arterial pressure), and during north–south green the switch advantage rises to +13.3 when the EW queue bin is highest (i.e., return green to the arterial). Globally, 24.7% of the Q-table entries prefer switching, and 41.8% of the 576 agent-state entries carry any experience — the tabular policy is sparse and heavily shaped by the training regimes.

Fig. 5 reports EW shares 51.9% for fixed time, approximately 52.5–61.6% for actuated control, and 52.9–55.4% for Q-learning. Fixed time has 14 s EW green in a 27 s cycle, so 51.9% is a full-cycle fraction; its green-only share would be 14/21=66.7%. The same denominator must be established for every controller. If 55.4% is a full-cycle share, nominal mean arterial capacity is 1,800×0.554=997.2 vehicles/hour, above demand 900; the share alone therefore does not prove underservice. Actual service, idle periods, and stored-credit behavior require traces.

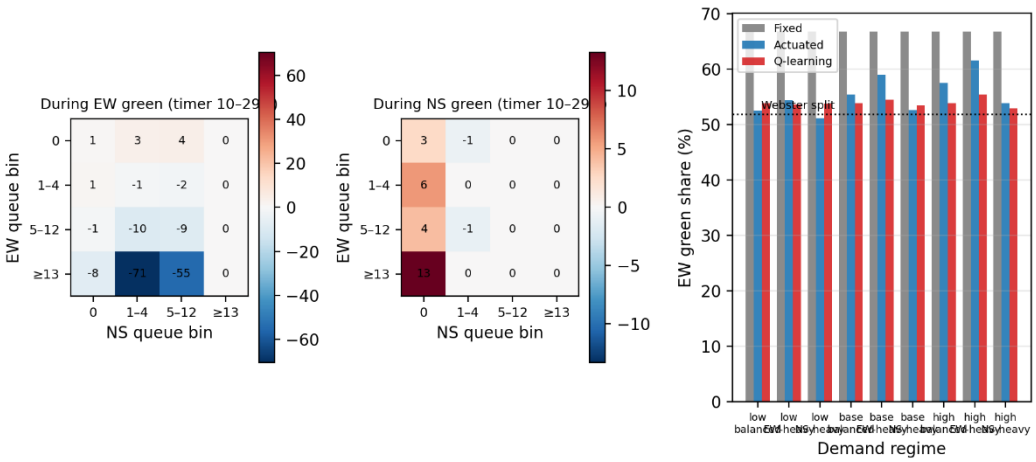


Fig. 5. Reported Q-value and EW timing diagnostics.

The source green-share label requires denominator verification; fixed-time 51.9% equals EW green divided by total cycle time, not all green time (*Fig. 5*).

7 | Statistical Analysis

Table 4 retains the source conditional rank-test summaries. Q-learning reportedly has lower delay than fixed time in 146 of 180 paired episodes, yet higher pooled mean delay (36.7 versus 24.3), because severe failures outweigh smaller wins. Wilcoxon signed-rank direction and arithmetic mean direction are different summaries. Actuated control has lower reported delay across the paired comparisons. Pooled tests across heterogeneous regimes and reused policy parameters require an explicit target and dependence treatment; they do not establish field reliability or simulator correctness.

The source reports Holm-adjusted per-regime p-values and large rank-biserial effects. Bootstrap intervals with twenty episodes reflect arrival variability conditional on one learned policy. Narrow intervals cannot account for untested training seeds, simulator defects, changed sensing/action guards, or external traffic dynamics. Original vectors, test settings, and common arrival arrays are needed to verify the reported statistics.

Table 4. Statistical tests on mean delay per vehicle (paired across common-random-number episodes; Holm-corrected per-regime tests reported in the text).

Test	n	Statistic	p-value	Effect size	Direction
QL vs fixed (pooled Wilcoxon)	180	$W = 5127$	1.62×10^{-5}	$r = -0.37$	mixed (see text)
Actuated vs fixed (pooled Wilcoxon)	180	$W = 0$	2.73×10^{-31}	$r = -1.00$	actuated better
QL vs actuated (pooled Wilcoxon)	180	$W = 0$	2.73×10^{-31}	$r = 1.00$	actuated better
Friedman across controllers	180 blocks	$\chi^2(2) = 304.8$	6.37×10^{-67}	$W = 0.85$	controllers differ

Note: Negative rank-biserial r indicates lower delay for the first-named controller.

8 | Discussion

The training-regime means support an advantage of the tested learned policy over the fixed-time plan within the reported simulation. The actuated implementation has lower displayed delays throughout. The comparison does not establish that reward design or decentralization alone caused the gain, because state abstraction, phase guards, sensing, and simulator interactions are bundled.

Held-out behavior is asymmetric in the reported results: low NS-heavy demand retains an advantage, while base/high EW-heavy demand loses it. The training curriculum covers high NS-heavy demand but only low EW-heavy demand. Sparse visitation and demand coverage are plausible contributors, not isolated causal findings from one trained policy.

Queue bins above thirteen alias moderate and large queues, so the state cannot distinguish their magnitude. The green-share summaries require a common denominator and do not demonstrate capacity below demand. The 60 s maximum green also does not make arterial-heavy allocation impossible: with EW green 60 s, NS minimum 10 s, and two 3 s amber periods, a feasible full-cycle EW fraction is $60/76 \approx 78.9\%$, nominally 1,421 vehicles/hour. This does not prove an optimal or safe policy, but refutes an impossibility argument based on the action guards alone. Simulator service rules and phase traces must be examined before explaining the collapse.

Late-training spikes and weak variation in reported EW shares motivate further state and timing audits. They do not establish catastrophic forgetting in tabular Q-learning; the neural forgetting source [12] concerns a different model class. Controlled comparisons with finer bins, arrival features, curriculum variants, and multiple training seeds are needed.

Gap-out control responds directly to arrivals and queues, but its information and guard limits differ from the learned policy. Its strong results show the value of including an engineering baseline in this simulator. They do not imply universally demand-proportional green, a generalization guarantee, or a field default.

9 | Implications

Evaluation should include excluded demand regimes, actuated references, and transparent timing diagnostics. Simulator conservation and service-capacity tests are separate requirements. The current capacity defects must be resolved and the comparisons rerun before using the displayed delays as physically validated outcomes.

A practical learned controller would need an adequate state, feasible phase guards, representative demand coverage, and robust validation against comparable engineering controls. This synthetic experiment does not support a field deployment recommendation. Arrival features and finer queues are testable improvements, rather than established remedies.

10 | Limitations

The point-queue model omits turning, spillback, platoon dispersion, pedestrians, and coordination. The unbounded unused-credit rule and empty-green pass-through require correction/validation; their effect is not known to be 2.6% everywhere or equal across controllers. The uncoordinated fixed-time baseline is not the best possible corridor plan. One curriculum and one trained policy limit inference, while Poisson stationary demand omits incidents and within-episode transitions. No real traffic data validate transfer.

11 | Reproducibility and Data Availability

Reproducibility requires the simulator, controller implementations, immutable arrival batches, all per-episode metrics, Q-tables, statistical code, and software versions. The described environment uses Python 3.12.14, NumPy 2.1.3, SciPy 1.14.1, scikit-learn 1.5.2, and matplotlib 3.9.2 [11], [13–15]. A programmatic builder can reduce transcription errors but does not certify service capacity or inferential validity. The supplied manuscript does not establish an archived code/data release or bit-for-bit reproduction.

12 | Conclusion

The reported six-intersection simulation shows lower learned-policy delay than fixed time on training regimes and a large failure under held-out high arterial demand. Actuated control has the lowest displayed mean delays. These are conditional observations of one policy in a point-queue implementation. Unbounded service credit, empty-green bypass, unequal sensing/phase guards, and timing-denominator ambiguity must be resolved before attributing the failure or claiming physically valid controller performance. Subsequent tests should rerun a capacity-validated simulator, match comparator information and actions, replicate training seeds, and retain held-out demand evaluation.

Funding

No external funding was received for this research.

Data Availability

All data are synthetic/simulated with documented generative models and fixed random seeds (Poisson arrival processes on a queue-based corridor simulator; no real-world data were collected or used). No human subjects, animals, or personal data were involved in this research.

Competing Interests

The authors declare no competing interests.

AI-Assisted Generation Disclosure

The source manuscript reports that this manuscript and the associated code and analysis were produced with AI assistance under the Interdisciplinary Research Portfolio Program. Its claim that all numerical results

originate from archived, seeded experiment code requires author confirmation and access to the versioned code and raw outputs; no accessible archive identifier is supplied here.

Code Availability

A public repository and archived version of the study-specific code and raw outputs are not specified.

References

[1] Rinaldi, S. M., Peerenboom, J. P., & Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE control systems magazine*, *21*(6), 11–25. https://doi.org/10.1109/37.969131

[2] Webster, F. V. (1958). *Traffic signal settings*. https://www.sinaldetransito.com.br/artigos/traffic_signals_webster.pdf

[3] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3), 279–292. https://doi.org/10.1007/BF00992698%0A%0A

[4] Sutton, R. S., Barto, A. G., & Barto, A. (1998). *Reinforcement learning: An introduction* (Vol. 1). MIT Press, Cambridge. https://doi.org/10.1017/S0263574799271172

[5] Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation research part b: Methodological*, *28*(4), 269–287. https://doi.org/10.1016/0191-2615(94)90002-7

[6] Mayne, D. Q., Rawlings, J. B., Rao, C. V, & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789–814. https://doi.org/10.1016/S0005-1098(99)00214-9

[7] El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto. *IEEE transactions on intelligent transportation systems*, *14*(3), 1140–1150. https://doi.org/10.1109/TITS.2013.2255286

[8] Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. *Autonomic road transport support systems*, 47–66. https://doi.org/10.1007/978-3-319-25808-9_4

[9] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0%0A%0A

[10] Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control. *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2496-2505). Association for Computing Machinery. https://doi.org/10.1145/3219819.3220096

[11] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., & Cournapeau, D. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[12] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., & Rusu, A. A. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

[13] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., & Cournapeau, D. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[14] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., & Grisel, O. (2011). Scikit-learn: Machine learning in Python. *The journal of machine learning research*, *12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html