# Evaluation of Traffic Signal Control at Varying Demand Levels: A Comparative Study

Zhiyao Zhang[12], Marcos Quinones-Grueiro[1], William Barbour[1], Yuhang Zhang[12], Gautam Biswas[13], and Daniel Work[12]

***Abstract*— The topic of traffic signal control (TSC) for urban intersection networks has been a key research area for many years, but comprehensive comparisons among state-of-the-art methodologies remain limited. This calls for a more exhaustive approach to benchmarking TSC techniques. This study builds upon previous TSC benchmarking work by evaluating the performance of both reinforcement learning (RL) and model-based TSC algorithms across a spectrum of demand levels. We adapt new metrics for in-depth simulation analysis and ground our investigation in a real-world urban arterial scenario. Our results unveil a noteworthy observation: the performance of tested algorithms is inconsistent when evaluated under diverse demand levels and across different metrics. This highlights the importance of implementing multidimensional evaluations in future TSC studies for a more nuanced understanding of their performance.**

## I. Introduction

Traffic signal control (TSC) is an active traffic management (ATM) strategy that has been extensively studied. Traditional TSC approaches include fixed-time and actuated control. The former continuously implements a predefined signal timing plan. The latter modifies certain phases' lengths aiming to meet traffic demands estimated through traffic sensing technologies, such as inductive loops and cameras. On the other hand, adaptive traffic control approaches optimize signal timing parameters such as cycle length and split to accommodate changing traffic demand patterns like AM and PM peaks. Widely used systems include SCOOT [3] and SCATS [4]. Recent TSC advancements introduce the "pressure" concept for signalized intersection networks, a term initially used in wireless telecommunication network scheduling and routing problems. Prominent methods include back pressure (BP) [5], [6], [7] and max pressure (MaxP) [8], [9]. Pressure-based TSC methods are increasingly adopted due to a desirable attribute: the decentralized control approach can provably maximize the throughput.

[1]Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212, USA

[2]Department of Civil and Environmental Engineering, Vanderbilt University, Nashville, TN 37212, USA

[3]Department of Computer Science, Vanderbilt University, Nashville, TN 37212, USA

Emails: {zhiyao.zhang, marcos.quinones.grueiro, william.w.barbour, yuhang.zhang.1, gautam.biswas, dan.work}@vanderbilt.edu

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein. This work is supported by a grant from the U.S. Department of Transportation Grant Number 693JJ22140000Z44ATNREG3202. However, the U.S. Government assumes no liability for the contents or use thereof. The source code and original data are available in [1] and [2].

Learning-based TSC algorithms, particularly reinforcement learning (RL), have drawn substantial attention for their ability to learn how to adapt to environmental dynamics from data acquired through interactions with the network, reducing the reliance on expert knowledge. Initial RL-based TSC approaches employed Q-learning to approximate the value function or state-action function, which evaluates potential long-term gains when an action is selected in a state. Noteworthy contributions in this area include [10], [11]. The RL learning objective is to maximize long-term expected rewards, with common reward designs targeting queue length [12], [13], [14], travel delay [15], and wait time [12], [16], [13]. The advent of deep reinforcement learning (DRL) addresses the limitations posed by high state-action dimensionality. Recent years have seen a surge in publications on DRL-based TSC [17]. Chu et al. [13] proposed a decentralized solution for large-scale TSC, known as multi-agent advantage actor-critic (MA2C). IntelliLight [18] incorporated the image representation of car positions from surveillance cameras as part of the input. Wu et al. [19] implemented a multi-agent recurrent DDPG (MARDDPG) method that uses the LSTM network to extract temporal information, addressing high variance issues due to partial observability. A parameter-sharing PPO (PS-PPO) was proposed for distributed arterial corridor signal control in [12].

A fresh direction in state representation and reward design has been shaped by the recently introduced pressure concept. Conceiving traffic pressure as the queue length difference between an intersection and its downstream counterpart, it naturally serves as a locally-communicated state representation. PressLight [20] is recognized as the first approach to incorporate MaxP into DRL by formulating the reward as the intersection pressure's negative and defining the state space using the same information required by MaxP. MPLight [21], integrating a novel neural network structure, FRAP [22], learns the phase competition in diverse intersection settings, thus facilitating MaxP learning in a shared single agent for thousands of traffic lights. Heuristic pressure-based techniques, as exemplified in these works, could further mitigate the MaxP assumption concerning the nonexistence of queue expansion.

While many innovative methodologies have been introduced, open-source testbeds and benchmarks keep vital for comparisons of TSC techniques, especially those based on DRL. Yet, we notice significantly less effort in this area. Developed on the open-source simulation platform SUMO [23], Flow [24] was designed for testing mixed autonomy

traffic using DRL, serving as the basis for later released benchmarks [25]. In the TSC realm, SUMO-RL [26] is an interface designed for RL and TSC, upon which a benchmarking toolkit, RESCO, was constructed [27].

Influenced by the comprehensive experiments conducted by [27] on diverse scales of real-world scenarios - which unmasked a generalizability gap for realistic settings - our motivation is multifaceted: Firstly, in light of the dynamic, realistic traffic demand, we aim to explore the robustness and sensitivity of TSC methods. Secondly, we adapt benchmarking metrics that consider both completed and unfinished trips, thus offering a more comprehensive evaluation of simulation performance compared to measures that only consider completed trips. Thirdly, in addition to averages, we consider the distribution and variance of performance at the vehicle-level. With the aforementioned motivation, we conduct on a comparative study evaluating four representative TSC algorithms. The findings from our experiments indicate that each tested algorithm performs inconsistently across different demand levels and under various evaluation metrics.

The rest of this paper is arranged as follows: Section II elaborates the TSC problem, introduces the concept of pressure and the preliminaries of RL, and defines the metrics employed in this paper. Section III provides detailed descriptions of the experimental settings, including the chosen TSC algorithms and simulation specifications. In Section IV, we examine training results, network-level test results, and intersection-level test results. Lastly, Section V offers conclusions and future research directions.

## II. Benchmarking TSC Methods

### A. Problem Formulation

Controlling a signalized intersection involves alternating the passage of traffic movement groups through the intersection. An approach, comprising multiple lanes, is characterized as a section of road at an intersection's edge. A traffic movement crosses an intersection by exiting from an entry approach and entering an exit approach. A phase, defined as "the green, change, and clearance intervals in a cycle assigned to specified movement(s) of traffic" in [28], is considered served when cars of the associated movement(s) are permitted to move. [28] provides an example (shown in Fig 1) of phase assignment and sequence for a four-approach two-way intersection, where horizontal phases (e.g., $\phi 1$ and $\phi 5$) can be concurrently served as a phase group. Therefore, there are four phase groups in the provided example. Ultimately, a signal timing plan determines the sequence and duration to cycle through all signal phases.

### B. Definition of Pressure

The pressure of a turning movement is the difference between the queue length of the entry approach and that of its downstream counterpart. A simplified case considers a multi-lane entry approach at an intersection $D$ from a direction $d$ (e.g., Northbound) $L_d \in \mathcal{I}_D$ has one distinct lane for each direction of movement, $\{l_l, l_t, l_r\} = L_d$ respectively

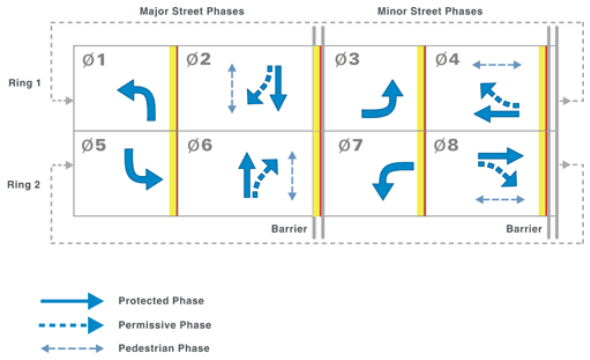


Fig. 1. An example of signal phase sequence (NEMA ring-and-barrier diagram [28]).

representing the left-turn, through, and right-turn movement, and the group of downstream lanes with respect to $L_d$ is $m \in \mathcal{O}_{L_d}$. Then, the pressure of the n-th phase $\phi_n$ at time $t$ is:

$$P_n(t) = \sum_{(l,m)} x_{(l,m)}(t); l \in L_d, m \in \mathcal{O}_{L_d}, \forall (l,m) \in \phi_n, \tag{1}$$

where $x_{(l,m)}$ is the difference of queue length between the downstream lane $m$ and the entry lane $l$. In this term, the phase is specified by a set of lane connections $(l,m)$ which may include multiple movements. Then, the pressure of intersection $i$ is the aggregation of the pressure of all predefined phases:

$$P_i(t) = \sum_{\phi_n} P_n(t), \forall \phi_n \in \oplus_D, \tag{2}$$

where $\oplus_D$ is the set of all phases in the intersection $D$.

### C. Reinforcement Learning

RL algorithms aim to learn a data-driven policy from interactions between the agent and the environment. The RL problem is usually formulated as a Markov Decision Process (MDP) defined by the tuple $(S, A, P, R, \gamma)$, where $S$ denotes the state space, $A$ is the action space, $P(s, a, s') : S \times A \times S \in [0, 1]$ is a transition function describing the probability of state $s$ transitioning to the next state $s'$ upon executing the action $a$, $R(s, a, s') : S \times A \times S \to \mathbb{R}$ is the reward function, and $\gamma$ is a parameter that allows computing a long-term reward discount. In this format, the long-term reward can be written as:

$$G_T = \sum_{t=1}^{T} \gamma^{t-1} R_t, \tag{3}$$

where $T$ defines the duration of an episode comprising a set of interactions such that a terminal state is reached. For continuous learning systems, $T$ can be further truncated for learning purposes.

Q-learning is a renowned value-based RL method where the Q values (state-action values) represent the long-term reward of a state-action pair [29]. These values are updated

according to the Bellman Optimality Equation:

$$Q^{\pi}(s,a) = \sum_{s' \in S, r \in R} P(s', r|s, a)[r + \gamma \max_{a'} Q^{\pi}(s', a')] \quad (4)$$

### D. Metrics for Evaluation

Common metrics for TSC evaluation include *throughput*, *travel time*, and *travel delay*. The first one is defined by the number of vehicles that traverse a network or complete their journey during the simulation. The third one measures the time delay due to signal operations as the discrepancy between a vehicle's actual travel time and the free-flow speed travel time. However, within the realm of traffic simulations, these metrics might overlook unfinished trips and varying trip lengths. For instance, vehicles traveling longer distances on arterial roads are likely to encounter more stops and wait times than those covering shorter distances. Prompted by this practical issue, we adapt two metrics for evaluating TSC performance:

- **Pace**: Pace of a vehicle is the inverse of the average speed, indicating the average time a vehicle needs to travel a unit of distance. We use pace instead of speed as extreme low-speed outliers are less apparent due to the nature of rational functions. In this paper, the unit of pace is second/meter.

$$\textbf{pace} = \frac{\text{traveled duration}}{\text{traveled distance}}. \quad (5)$$

- **Time Loss Index (TLI)**: The Travel Time Index (TTI) is a critical metric for assessing travel time reliability. It is defined as the ratio of average travel time during a period of interest to the travel time at free-flow speed and has been utilized to evaluate arterial corridor traffic mobility [30]. We use a modified version of the TTI, known as the time loss index, to also account for departure delays:

$$\textbf{TLI} = \frac{\text{travel delay} + \text{departure delay}}{\text{traveled duration}}. \quad (6)$$

  TLI is designed with the following rationale: 1) most simulation platforms can measure all components of TLI; 2) vehicles queuing outside the simulation network (when network boundaries are oversaturated) are disregarded in TTI, but captured in TLI; Note that unlike TTI, which always exceeds a value of 1, TLI can range from a positive value below 1 to greater than 1 when the cumulative delay both inside and outside the network surpasses the travel duration. Note when accounting for vehicles that never start a trip (high departure delay, but travel delay and duration are both 0), we use a minimum travel duration of 1 second to avoid an infinite TLI.

## III. Experiment Settings

### A. Algorithm Selection

We conduct our experiment leveraging the RESCO library [27], with functional additions and modifications. We select four methods for training and testing: two RL-based methods (Independent DQN or IDQN, and MPLight) and two model-based control methods (MaxP and Longest Queue First or LQF). MPLight is chosen due to its successful testing at an extensive scale with more than 1000 traffic lights, and because it incorporates the concept of "pressure" into the DQN framework. MaxP is a renowned pressure-based control method, and LQF is a locally greedy method that does not require inter-signal communication. All methods have the same action space, consisting of all predefined phase groups to be served. All implementations have been reproduced from RESCO.

We conduct our experiments leveraging the RESCO library [27], with functional additions and modifications. We select four methods for training and testing: two RL-based methods (Independent DQN or IDQN, and MPLight) and two model-based control methods (MaxP and Longest Queue First or LQF). LQF is a locally greedy strategy for TSC serving for the phase with the longest vehicle queue. MaxP represents the state-of-the-art pressure concept utilized in TSC, and is valued for its demonstrated ability to stabilize the network when the average demand is bounded [8]. Drawing inspiration from MaxP, MPLight learns MaxP's control policy heuristically using RL. MPLight was chosen for testing as it has been utilized in the largest scale of simulation network in all existing studies. Ultimately, we sought an RL-based comparison for MPLight. Given MPLight's use of DQN's learning paradigm, and considering the good performance of the IDQN method in the benchmark upon which this work is based, we believed IDQN would be a valuable algorithm for further testing. Table I provides a summary of key features of the tested methods, where "local" and "adjacent" are types of the state space, indicating if the algorithm observes only the local intersection or adjacent ones as well. The MDP and network specifications of both learning-based methods are presented as follows:

TABLE I
FEATURES OF THE TESTED METHODS

| | local state | adjacent state |
|---|---|---|
| model-based | LQF | MaxP |
| learning-based | IDQN | MPLight |

**IDQN** [31]. In IDQN, each intersection has an individual DQN agent, with each agent training and performing locally without parameter sharing or communication. The state space consists of the number of approaching and queuing vehicles, as well as the sum of the waiting time and vehicle speed for each lane. The reward function is the normalized total waiting time at the intersection.

**MPLight** [21]. MPLight is derived from DQN but uses FRAP as the neural network structure. FRAP models the phase competition and shows invariance to symmetrical cases, enabling it to act as a shared single agent for heterogeneous intersections [22]. Its state space includes the current phase in service and the pressure of each phase calculated

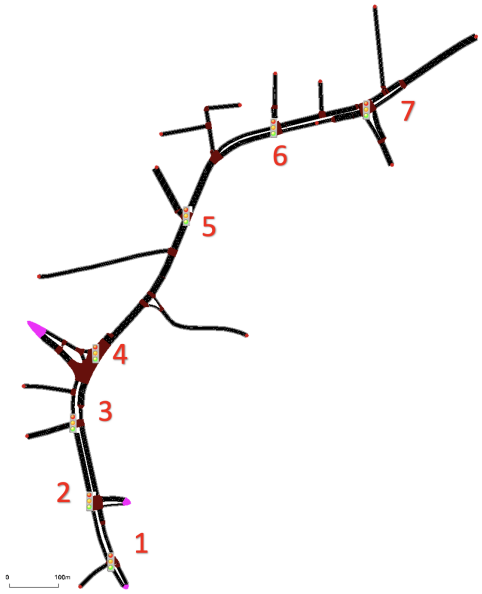


Fig. 2. Arterial with 7 signalized intersections in Ingolstadt, Germany. Numbers are intersection indexes.

from (1). The reward function is the negative intersection pressure derived from (2).

### B. Simulation Settings

Experiments are conducted on the SUMO platform. We select a major arterial segment with seven signalized intersections from the open-source SUMO scenario, InTAS [32], as shown in Fig. 2. Only network boundaries with edges directly connected to a signalized intersection serve as source and sink where traffic enters or exits the segment. We define low, medium, and high demand levels synthetically, corresponding to 2600, 3600, and 4400 trips per hour respectively. The low and high demand levels are manually tuned with the default signal timing plan with the former aiming to preserve the intersections to be undersaturated which aligns with common demand settings, and the latter on purpose to break the queue expansion assumption on boundary source edges.

For training the RL algorithms, we use a time-varying demand file that provides a low demand for the initial 15 minutes (warm-up period), a high demand for the next 45 minutes (pressure period), and a medium demand for the final 30 minutes (recovery period). The time-varying settings inspired by real-world detour cases enables the RL agent to learn to deal with the dynamics of traffic demand and avoid overfitting to specific demand levels. For the convenience of algorithm evaluation at fixed demands, we use three 1-hour demand files corresponding to the three demand levels. Each demand file undergoes 25 episodes of testing to account for simulation seed and parameter randomization. The time step length is set at 10 seconds, with a forced yellow transition at 5 seconds, and teleporting is disabled. All experiments are executed on a workstation equipped with an AMD 24-Core CPU, 256GB RAM, and four NVIDIA RTX A6000 GPUs.

## IV. Results and Analysis

In this section, we present and analyze the experimental results. The training results are intended to verify the reproducibility and consistency of our implementation with the RESCO testbed. For testing, we use three metrics - TLI, throughput, and pace - to compare network-level performance. Moreover, we take a closer look at the intersection level behavior and performance to understand how the tested methods operate and their impact on network performance by comparing the distribution of intersection waiting time.

### A. Training results

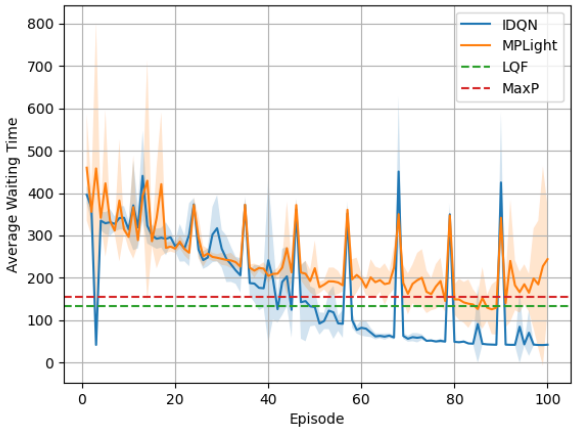


Fig. 3. The training curves of RL agents from 5 random seeds and a moving average window for 5 episodes. The error margins show one standard deviation from the mean values.

We employ the same metric as the RESCO paper, i.e., the average delay time per episode, to evaluate the RL methods' training process. Fig. 3 shows that both IDQN and MPLight converge as training progresses. Both methods show a similar rate of descent initially, but IDQN achieves faster convergence and minimum variance within 80 episodes, resulting in superior performance. On the other hand, MPLight only attains a slightly lower, albeit still with some variance, average waiting time compared to MaxP and LQF. Interestingly, MPLight tends to diverge towards the end of training, as also observed in [27], albeit with less deviation. We also notice regular "spikes" every ten episodes, a phenomenon seen in existing implementations as well. As these anomalies do not appear in testing when the neural networks aren't updated, we suspect they might be related to certain aspects of the training process. Nonetheless, it doesn't affect the evaluation of the policies, as we save the trained model at each episode and select the best-performing ones from the entire training phase for the subsequent tests.

### B. Network-level test results

Fig. 4 presents the network throughput during the simulation. Both model-based methods exhibit low-variance performance across low, medium, and high demands, although

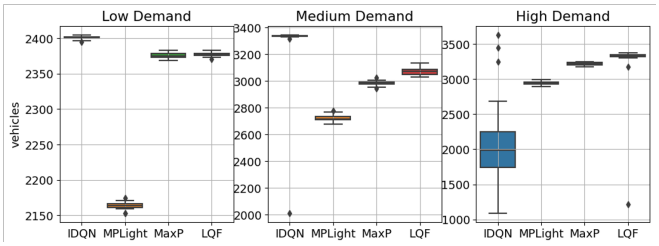


Fig. 4. The box plot of network throughput (higher is better).

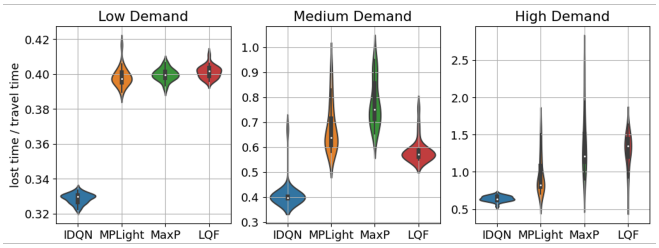


Fig. 5. The violin plot of episodic average TLI (lower is better).

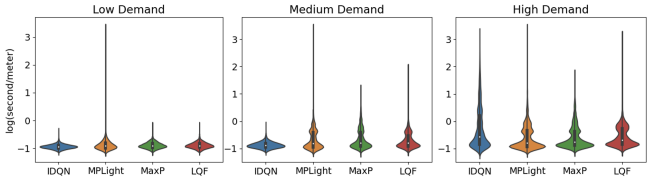


Fig. 6. The violin plot of the aggregation of paces over 25 rounds (lower is better). The y axis is a logarithmic scale for the convenience of comparison.

MaxP slightly underperforms LQF as the demand increases. Of particular interest are the results from the RL algorithms. IDQN shows very good performance at low and medium demands but deteriorates under high demand conditions, exhibiting considerable variance. In contrast, MPLight demonstrates good performance similar to the model-based methods across all demand levels. However, it performs worst for low and medium demands and cannot achieve performance comparable to MaxP. These results highlight a significant contrast in the average and variation of network throughput between the two RL methods.

We use violin plots for subsequent visualizations as they provide a more comprehensive depiction of data than box plots, showing not only specific feature values but also the distribution of data points based on kernel density estimation. This gives us a more intuitive understanding of variance.

Fig. 5 presents the TLI results. Surprisingly, despite its varying performance in maximizing throughput, IDQN consistently minimizes time loss at all demand levels, with a notably dense distribution around the average value. The other methods perform in a similar range, with MPLight slightly outperforming under low and high demand conditions and LQF yielding lower TLI than MPLight at medium demand.

Fig. 6 shows the results for vehicle pace, where each vehicle-based pace is shown as a separate data point. The violin plots show a similar distribution for MPLight, MaxP, and LQF. A noteworthy observation is that MPLight consistently shows a number of outliers. These data points originate from a specific edge at intersection 2, where vehicles are never served during the simulation. Despite having the same state space, MaxP does not exhibit the same issue, indicating a potential shortcoming of MPLight in fairly serving all traffic movements. Another observation, consistent with the network throughput metric, is that IDQN performs best under low and medium demand but worst under high demand compared to MaxP and LQF. While other methods still exhibit comparable TLI distribution, it appears that outliers might significantly contribute to the average and standard deviation values. In general, the RL methods tested either exhibit very good average performance while displaying high variance, or perform consistently across various demand levels without reaching optimal performance.

Numerical values for the mean and standard deviation of each algorithm at each demand level for each metric are summarized in Table II. The best-performing results — maximum mean values for throughput, minimum mean values for TLI and pace, and minimum standard deviation for all — are marked in bold. As indicated by the training results and [27], IDQN generally outperforms other methods under low and medium demand levels but is less robust under high demand. MPLight, although it doesn't often rank first, shows a consistently low model variation across all demand levels. However, this reveals that additional efforts are required to make the most of this feature while improving other metrics. Additionally, despite the absence of the queue expansion assumption, MaxP remains a desirable choice across all metrics.

TABLE II
SUMMARY OF NETWORK-LEVEL METRIC RESULTS
(MEAN/STANDARD DEVIATION)

| Throughput | IDQN | MPLight | MaxP | LQF |
|---|---|---|---|---|
| Low | **2400/2.240** | 2163/4.648 | 2375/3.406 | 2377/2.858 |
| Medium | **3284**/259.6 | 2724/**13.19** | 2986/16.42 | 3073/25.60 |
| High | 2094/621.1 | 2940/**21.63** | 3216/21.98 | **3246**/416.0 |
| TLI | IDQN | MPLight | MaxP | LQF |
| Low | **0.329/0.003** | 0.400/0.003 | 0.398/0.005 | 0.401/0.003 |
| Medium | **0.410**/0.054 | 0.775/0.091 | 0.676/0.094 | 0.584/**0.046** |
| High | **0.641**/0.036 | 1.385/0.401 | 0.955/0.245 | 1.291/**0.220** |
| Pace | IDQN | MPLight | MaxP | LQF |
| Low | **0.117/0.024** | 2.026/45.12 | 0.132/0.038 | 0.131/0.037 |
| Medium | **0.138/0.041** | 1.600/35.66 | 0.362/0.581 | 0.317/1.156 |
| High | 6.881/46.88 | 1.689/34.36 | **0.482/1.050** | 0.788/11.16 |

It is important to acknowledge that the existing tests in [27] encompass a variety of scenarios, including individual vs multiple intersections and synthetic vs realistic networks. Notably, the performance of each tested algorithm exhibits variance across these different scenarios, suggesting that performance outcomes are network-dependent. A limitation of our study is that our experiments are solely conducted within one realistic, multi-intersection scenario. Consequently, we are not able to explore how our findings differentiate across diverse scenarios. Nevertheless, our primary emphasis rests on the inconsistency of algorithmic performance across de-

mand levels and different metrics that our current experimental setup allows us to adequately examine.

### C. Intersection-level test results

We also investigate the performance of the agents at the intersection level to understand how behaviors at this level might impact the overall network performance. In this subsection, we plot the distribution of vehicle waiting times required to pass through each individual intersection. The scales of the y-axes are zoomed in to focus on the majority of the probability distribution, although outliers exist beyond the displayed range. This visualization is presented in Fig. 7.

The results reveal that under low and medium demand levels, each IDQN agent efficiently manages traffic at every intersection; however, during high demand conditions, the distribution of vehicle waiting time becomes more dispersed than that for any other algorithm across all intersections. MPLight, in contrast, demonstrates acceptable performance only at intersections 1, 3, 4, 5, and 6, with noticeably higher variations at intersections 2 and 7. We speculate that the high variance of waiting time at intersection 2, where service failure occurs, significantly impacts network-wide performance. Interestingly, at the medium demand level at intersections 5, 6, and 7, MPLight, MaxP, and LQF all exhibit similarly high variation, while IDQN remains more consistent. This kind of comparison does not exist at other demand levels.

A key takeaway from the intersection-level analysis is that the performance of MPLight varies among intersections, which could lead to the formation of bottlenecks. It is possible that the nature of the shared agent is assumed to be insensitive to intersection heterogeneity, but the geometry is a hidden factor that needs to be learned, especially in the sense of pressure.

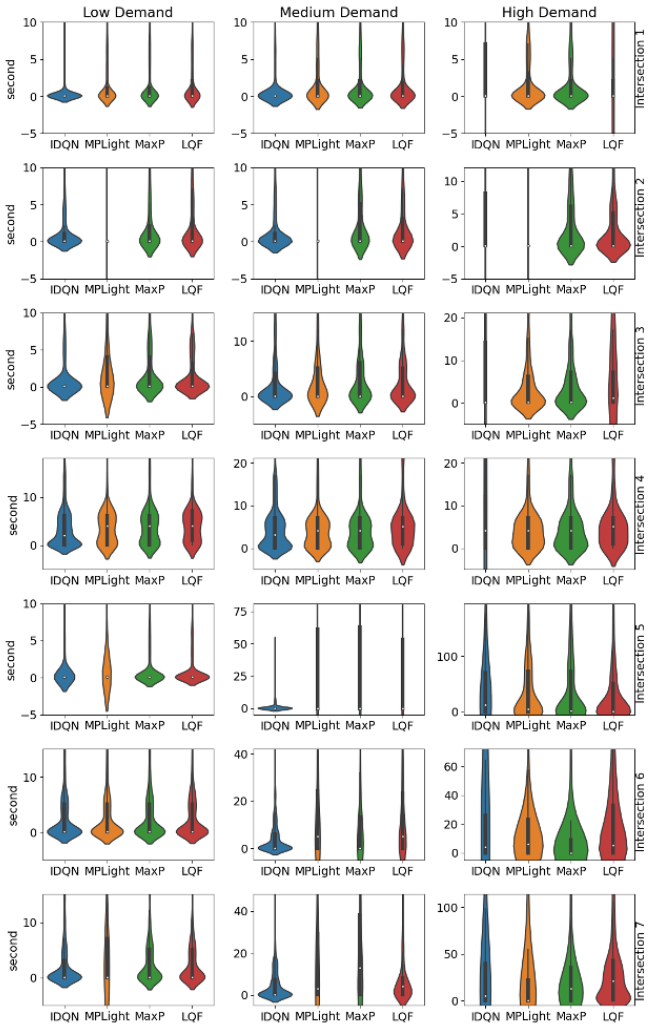


Fig. 7. The violin plot of the intersection-level vehicle waiting time when removing outliers. Horizontal: demand levels. Vertical: intersection 1 at the top and intersection 7 at the bottom.

## V. Conclusion

In this research, we carry out an evaluation of four traffic signal control (TSC) algorithms across varying demand levels. We assess the performance of two reinforcement learning methods, IDQN and MPLight, and two model-based approaches, Max Pressure and Longest-Queue-First. Alongside this, we utilize two additional metrics - time loss index and vehicle pace - for benchmarking the TSC performance. Post-processing of simulation results enables us to analyze performance both at the network and intersection levels. Notably, each algorithm demonstrates unique weaknesses in certain metrics at specific demand levels. This inconsistency in control performance, previously unaddressed in benchmark studies, underlines the importance of considering a variety of metrics and demand levels when evaluating TSC methods.

Our analysis leads us to propose future research aimed increasing adaptability to demand pattern variety. We intentionally violate the queue expansion assumption in our design of the high-demand scenario, aiming to assess how agents respond to unexpected surges in arterial demands. Such scenarios, often associated with non-recurring congestion, frequently occur in practice and present significant challenges due to their unpredictable nature and the measurement uncertainties they introduce. We aim to draw attention to the necessity of designing and testing TSC algorithms for robustness against extreme traffic patterns.

## References


[1] Zhiyao Zhang, Marcos Quinones-Grueiro, William Barbour, Yuhang Zhang, Gautam Biswas, and Daniel Work. Source code. `https://github.com/Lab-Work/ITSC23_signal`, 2023.

[2] Zhiyao Zhang, Marcos Quinones-Grueiro, William Barbour, Yuhang Zhang, Gautam Biswas, and Daniel Work. Source data. `http://hdl.handle.net/1803/18314`, 2023.

[3] PB Hunt, DI Robertson, RD Bretherton, and M Cr Royle. The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4), 1982.

[4] P Lowrie. Scats-a traffic responsive method of controlling urban traffic. *Sales information brochure published by Roads & Traffic Authority, Sydney, Australia*, 1990.

[5] Jean Gregoire, Emilio Frazzoli, Arnaud de La Fortelle, and Tichakorn Wongpiromsarn. Back-pressure traffic signal control with unknown routing rates. *IFAC Proceedings Volumes*, 47(3):11332–11337, 2014.

[6] Jean Gregoire, Xiangjun Qian, Emilio Frazzoli, Arnaud De La Fortelle, and Tichakorn Wongpiromsarn. Capacity-aware backpressure traffic signal control. *IEEE Transactions on Control of Network Systems*, 2(2):164–173, 2014.

[7] Ali A Zaidi, Balázs Kulcsár, and Henk Wymeersch. Back-pressure traffic signal control with fixed and adaptive routing for urban vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2134–2143, 2016.

[8] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. *Advances in dynamic network modeling in complex transportation systems*, pages 27–66, 2013.

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

[10] Marco A Wiering et al. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000.

[11] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, 2003.

[12] Weibin Zhang, Chen Yan, Xiaofeng Li, Liangliang Fang, Yao-Jan Wu, and Jun Li. Distributed signal control of arterial corridors using multi-agent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[13] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

[14] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, 2019.

[15] Faizan Rasheed, Kok-Lim Alvin Yau, and Yeh-Ching Low. Deep reinforcement learning for traffic signal control under disturbances: A case study on sunway city, malaysia. *Future Generation Computer Systems*, 109:431–445, 2020.

[16] Qiang Wu, Jianqing Wu, Jun Shen, Bo Du, Akbar Telikani, Mahdi Fahmideh, and Chao Liang. Distributed agent-based deep reinforcement learning for large scale traffic signal control. *Knowledge-Based Systems*, 241:108304, 2022.

[17] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Zahra Shakeri Hossein Abad, Ana LC Bazzan, and Behrouz Far. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications*, page 116830, 2022.

[18] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2496–2505, 2018.

[19] Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(8):8243–8256, 2020.

[20] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1290–1298, 2019.

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

[22] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1963–1972, 2019.

[23] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

[24] Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2021.

[25] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Conference on robot learning*, pages 399–409. PMLR, 2018.

[26] Lucas N. Alegre. SUMO-RL. `https://github.com/LucasAlegre/sumo-rl`, 2019.

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

[28] National Academies of Sciences, Engineering, and Medicine. *Signal Timing Manual - Second Edition*. The National Academies Press, Washington, DC, 2015.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[30] William L Eisele, Yunlong Zhang, Eun Sug Park, Yanru Zhang, and Rachael Stensrud. Developing and applying models for estimating arterial corridor travel time index for transportation planning in small to medium-sized communities. *Transportation research record*, 2244(1):81–90, 2011.

[31] James Ault, Josiah P Hanna, and Guni Sharon. Learning an interpretable traffic signal control policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 88–96, 2020.

[32] Silas C Lobo, Stefan Neumeier, Evelio MG Fernandez, and Christian Facchi. Intas–the ingolstadt traffic scenario for sumo. *arXiv preprint arXiv:2011.11995*, 2020.