CS4246 - Smart Traffic Control using RL
Team - P05

Kumar Prabhat	e1070583@u.nus.edu	A0265901E
Sarji Elijah See Bona	e0960564@u.nus.edu	A0255894L
Sun Yu Ting	e0726163@u.nus.edu	A023163M
Tan Chee Heng	e0764286@u.nus.edu	A0237301U

GitHub Repository: https://github.com/dedsecrattle/Smart-Traffic-Management-AI

Abstract

In this project, we tackle the problem of urban traffic congestion by designing adaptive traffic light controllers using various reinforcement learning (RL) algorithms. Using SUMO-RL, we train agents to reduce overall vehicle waiting time and carbon emissions at intersections. We evaluate classical and advanced RL methods including Q-Learning, SARSA(λ), DDQN, PPO, TRPO, PPO-LSTM, and our own implementation of Proximal Conservative Q-Learning (PCQL). Our study finds that deep policy gradient methods, especially PPO-LSTM, offer the best trade-off between traffic flow, sustainability, and policy stability.

1 Introduction

Urban areas are experiencing a dramatic increase in vehicle density. According to a United Nations report, over 68% of the global population is projected to reside in urban regions by 2050. Traditional traffic control systems, typically fixed-timed or rule-based, cannot adapt in real-time to evolving traffic patterns. This results in increased congestion, longer waiting times, and higher CO_2 emissions.

Reinforcement Learning (RL), which learns optimal behavior through interactions with the environment, provides a promising alternative. Instead of hard-coding traffic light schedules, RL agents adaptively learn efficient signal control policies. Our project aims to evaluate multiple RL strategies within a simulated urban environment to identify the most effective approaches.

2 Motivation and Problem Statement

The inadequacy of fixed-time traffic control systems in dealing with varying traffic loads motivates the exploration of adaptive RL-based controllers. Our goal is to build agents that dynamically adjust signal timings based on traffic flow, while optimizing multiple objectives: **minimizing queue lengths**, **total waiting time**, and **CO_2 emissions.**

3 Simulation Setup and Traffic Configuration

We utilize the SUMO traffic simulator, integrated via SUMO-RL, to create two network configurations of increasing complexity: a single intersection and a 2×2 grid. Both setups reflect realistic directional traffic imbalances and allow us to evaluate RL agents under different coordination challenges.

3.1 Single Intersection

This scenario features a standard four-way junction with one RL agent controlling signal phases. Vehicles arrive asymmetrically—one every 2 seconds from the West–East direction (high load) and one every 5 seconds from the North–South direction (moderate load). The agent must adaptively balance phase priorities to minimize delay and prevent starvation.

3.2 2×2 Grid Network

The grid consists of four connected intersections, each controlled by an independent RL agent. Vehicles follow pre-defined routes across the network. Although spawn rates remain asymmetric, this setup introduces spatial dependencies: local actions affect neighboring intersections. The challenge lies in achieving decentralized yet coordinated traffic management across the grid.

4 Problem Formulation

We model the adaptive traffic signal control problem as a Markov Decision Process (MDP), where an agent (or multiple agents in the multi-intersection case) interacts with a dynamic traffic environment. The goal of the agent is to learn a policy that minimizes vehicular delay, queue length, and emissions by selecting optimal traffic signal phases in real-time.

Formally, the MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where:

4.1 State Space $\mathcal{S}$

Each agent observes a compact, real-valued representation of the traffic state at its intersection. The state vector typically includes:

- Queue length per incoming lane
- Accumulated waiting time per lane
- Current active phase index
- Elapsed time since the last phase change

This formulation ensures that the agent has partial observability of the surrounding traffic and must infer temporal patterns from recent trends.

4.2 Action Space $\mathcal{A}$

The action space is discrete and consists of permissible phase transitions. Each action corresponds to selecting the next traffic signal phase from a predefined set. For example, in a simple two-phase configuration:

- Action 0: Green for North–South direction
- Action 1: Green for West–East direction

Actions are executed with a mandatory yellow transition, and constrained by minimum and maximum green time bounds to ensure safe vehicle movement.

4.3 Transition Function $\mathcal{P}$

The environment transitions are governed by the SUMO traffic simulator, which deterministically simulates the flow of vehicles based on traffic signal phases, road geometry, and vehicle behavior models (e.g., car-following and lane-changing logic). While the simulator is deterministic, the arrival of vehicles is stochastic, making the environment effectively non-deterministic from the agent's perspective.

4.4 Reward Function $\mathcal{R}$

We employ both single-objective and composite reward functions. The reward is computed at each timestep t based on traffic conditions.

The design of the reward function is central to shaping the learning behavior of RL agents in traffic control. We consider two formulations: a delay-based reward focusing solely on time efficiency, and a composite reward that integrates multiple real-world objectives such as throughput, sustainability, and fairness.

4.4.1 Delay-Based Reward

The delay-based reward is a simple yet effective signal that captures the change in cumulative vehicle delay across the network:

$$R_t = D_{t-1} - D_t$$

Here, D_t denotes the total delay experienced by all vehicles at time step t. A positive reward is obtained when cumulative delay decreases, incentivizing the agent to select actions that reduce vehicle waiting time. However, this reward lacks granularity and may not explicitly account for emissions or traffic flow fairness.

4.4.2 Composite Weighted Reward

To address the limitations of delay-only rewards, we define a multi-objective reward that combines traffic efficiency, environmental impact, and fairness considerations:

$$R_t = w_v V_t - w_q Q_t - w_w W_t - w_f F_t - w_c C_t + P_f$$

where:

- V_t: Number of vehicles that passed through the intersection during time step t (positive contribution)
- Q_t: Total queue length across all incoming lanes (penalty)
- W_t: Total accumulated waiting time (penalty)
- F_t: Total fuel consumption in liters or mL/s (penalty)
- C_t: Total CO_2 emissions in milligrams per second (penalty)
- P_f: Fairness penalty term (described below)

The weights are empirically set as follows:

$$w_v = 2.0, \quad w_q = 0.05, \quad w_w = 0.01, \quad w_f = 0.1, \quad w_c = 0.05$$

These weights reflect the relative importance of promoting vehicle throughput while discouraging congestion and environmental harm.

Fairness Penalty P_f To discourage the agent from favoring high-traffic directions while starving lower-volume lanes, we incorporate a fairness penalty:

$$P_f = \begin{cases} -10 & \text{if } V_t = 0 \text{ and } Q_t > q_{\text{threshold}} \\ 0 & \text{otherwise} \end{cases}$$

This penalty activates when no vehicles pass despite the presence of a significant queue ($q_{\text{threshold}} = 10$), forcing the agent to eventually service all lanes, not just the dominant flow.

4.4.3 Design Trade-offs

While the delay-based reward is simpler and easier to optimize, it provides limited feedback. The composite reward enables fine-grained control but requires careful tuning of weights to prevent one objective (e.g., emissions) from overwhelming others (e.g., throughput). In our experiments, we evaluate both reward schemes across all algorithms to analyze their influence on learning stability and policy behavior.

4.5 Discount Factor γ

We use a discount factor $\gamma \in [0.95, 0.99]$ to prioritize long-term returns and allow the agent to anticipate delayed effects of current actions (e.g., downstream congestion reduction).

5 Reinforcement Learning Algorithms Evaluated

We evaluate a range of RL algorithms encompassing tabular methods, deep value-based methods, and policy gradient techniques. This section formally describes each method and provides the core mathematical equations that govern their learning updates.

5.1 Q-Learning

Q-Learning is an off-policy, model-free value iteration algorithm. It updates the Q-values using the Bellman optimality equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

where α is the learning rate and γ is the discount factor. It is simple but struggles with generalization in large or continuous state spaces.

5.2 SARSA(λ)

SARSA(λ) is an on-policy method that uses eligibility traces to bridge the gap between Monte Carlo and TD methods. The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t e(s_t, a_t)$$

with TD error:

$$\delta_t = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

and $e(s, a)$ is the eligibility trace, which decays over time. The algorithm improves credit assignment for temporally distant rewards.

5.3 Double DQN (DDQN)

DDQN mitigates the overestimation bias in standard DQN by decoupling the action selection and evaluation:

$$y_t = r_t + \gamma Q_{\text{target}}(s_{t+1}, \arg\max_a Q(s_{t+1}, a))$$

The Q-network is updated by minimizing the loss:

$$\mathcal{L}(\theta) = \left(Q(s_t, a_t; \theta) - y_t \right)^2$$

where Q_{target} is a separate target network periodically synced with Q.

5.4 Proximal Policy Optimization (PPO)

PPO is a policy gradient method that uses a clipped objective to avoid large policy updates. Its surrogate objective is:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance ratio and $\hat{A}_t$ is the estimated advantage.

5.5 Trust Region Policy Optimization (TRPO)

TRPO enforces a constraint on the step size in policy space using the KL divergence. It solves:

$$\max_\theta \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \quad \text{subject to } \mathbb{E}_t \left[D_{KL}(\pi_{\theta_{\text{old}}} \| \pi_\theta) \right] \leq \delta$$

This ensures stable and monotonic policy improvements at the cost of additional optimization complexity.

5.6 PPO with LSTM (PPO-LSTM)

PPO-LSTM extends PPO by replacing the MLP policy with a recurrent LSTM policy to incorporate temporal dependencies:

$$h_t = \text{LSTM}(x_t, h_{t-1}), \quad \pi(a_t|s_t) = \pi(h_t)$$

This allows the policy to retain memory of past states, which is useful in partially observable or delayed-reward environments.

6 Proximal Conservative Q-Learning (PCQL)

Proximal Conservative Q-Learning (PCQL) is a hybrid reinforcement learning algorithm that extends traditional value-based approaches by integrating conservative Q-learning with soft target updates. It is designed to address key challenges in safety-critical applications, such as traffic signal control, where overestimation of action values can lead to unstable or unsafe behaviors.

6.1 Motivation

Standard Q-learning algorithms often suffer from optimism bias, where the agent overestimates the value of unexplored or poorly understood actions. In traffic signal control, such behavior may result in inefficient phase switching, increased vehicle delay, or environmental penalties. PCQL introduces a conservative learning paradigm that explicitly penalizes overconfident value estimates, encouraging safer and more stable policies.

6.2 Loss Function

The PCQL objective is composed of two parts: a traditional Bellman error and a conservative regularization term that reduces overestimation in the learned Q-values.

$$\mathcal{L}_{\text{PCQL}} = \underbrace{(Q(s,a) - y)^2}_{\text{Bellman Error}} + \lambda \cdot \underbrace{\left(\log \sum_{a'} e^{Q(s,a')} - Q(s,a) \right)}_{\text{Conservative Penalty}}$$

where:

$$y = r + \gamma \max_{a'} Q'(s', a')$$

In this formulation:

- $Q(s,a)$ is the current Q-value for state-action pair (s,a).
- Q' is the slowly updated target Q-network.
- r is the immediate reward, and γ is the discount factor.
- λ is a hyperparameter controlling the strength of conservativeness.

The conservative penalty discourages large deviations between the selected action value and the average Q-values of all actions. This biases the agent toward safer, lower-variance decisions.

6.3 Proximal Target Updates

To enhance training stability, PCQL employs soft target updates inspired by Proximal Policy Optimization (PPO). Rather than periodically replacing the target network, PCQL performs incremental updates using an exponential moving average:

$$Q' \leftarrow \tau Q + (1 - \tau) Q'$$

where $\tau \in (0, 1)$ is a small smoothing constant (typically $\tau = 0.005$). This approach prevents drastic shifts in target values and helps ensure smoother convergence.

6.4 Benefits in Traffic Signal Control

PCQL introduces several practical advantages when applied to traffic signal control tasks:

- **Stability:** By regularizing overestimated Q-values and using soft updates, PCQL maintains more stable learning dynamics across episodes.
- **Safety-Awareness:** The algorithm discourages risky or rarely explored phase transitions unless they consistently lead to high rewards, reducing the chance of starvation in lower-traffic directions.
- **Conservative Exploration:** PCQL encourages the agent to be cautious early in training and only commit to high-value actions when sufficient evidence has been accumulated.

6.5 Empirical Behavior

In our experiments, PCQL demonstrated steady improvements in total waiting time and queue length. Although it converges slightly slower than aggressive learners like PPO, its robustness and low volatility make it suitable for deployment in dynamic and uncertain traffic environments. The learned policies exhibited minimal fluctuations and were more consistent across training runs, especially under variable traffic demand and stochastic vehicle arrivals.

7 Evaluation Results

We evaluate the performance of all algorithms on three key metrics: average waiting time (in seconds, all vehicles combined), total queue length (no. of vehicles per intersection combined), and CO_2 emissions (in grams total vehicles combined). Each experiment is conducted over multiple training episodes with stochastic vehicle spawn patterns and averaged across three random seeds to account for variance.

7.1 Performance Summary

Table 1 presents the final evaluation results for each agent after convergence. All RL agents outperform the fixed-time baseline across all metrics, with significant improvements in both efficiency and sustainability.

Table 1: Performance Comparison (Lower is Better)

Algorithm	**Waiting Time (s)**	**Queue Length**	**CO_2 Emission (g/s)**
Fixed-Time Baseline	148.3	31.2	912.4
Q-Learning	109.5	23.7	704.0
SARSA(λ)	99.1	21.5	681.8
DDQN	72.4	19.8	665.2
TRPO	63.1	15.9	603.5
PPO	58.7	14.6	584.3
PPO-LSTM	**52.9**	**13.2**	**550.8**
PCQL	61.0	17.1	612.7

7.2 Early vs. Late Training Dynamics

Each algorithm exhibited distinct learning behaviors:

- **Q-Learning** showed a steep initial drop in waiting time but plateaued after 40k steps. Its tabular structure lacks generalization capacity for unseen traffic patterns.
- **SARSA(λ)** converged more smoothly than Q-Learning due to its on-policy nature and eligibility traces. It maintained stable performance without major spikes.
- **DDQN** achieved rapid early gains in waiting time but was slightly less aggressive in queue reduction. Emission reduction remained modest, likely due to prioritizing flow over eco-efficiency.

- **TRPO** and **PPO** both converged reliably within 20k steps. PPO demonstrated better balance across all metrics, especially in queue regulation.
- **PPO-LSTM** consistently delivered the best overall results. Its recurrent policy enabled better anticipation of traffic bursts, leading to smooth traffic transitions and minimal idling.
- **PCQL** showed a slower but consistent decline in all metrics. Its conservative bias initially limited risk-taking, but it eventually achieved strong queue and delay control with minimal performance oscillations.

7.3 Environmental Impact

All RL agents led to significant reductions in emissions compared to the baseline. PPO-LSTM achieved a 39.6% reduction in CO_2 output, while PCQL achieved 32.9%. Agents trained with the composite reward function generally outperformed those using delay-only rewards, confirming that including emission terms in the reward design leads to more sustainable policies.

7.4 Reward Sensitivity Study

We also performed a reward ablation by varying the CO_2 emission weight w_c in the composite reward function. Increasing w_c led to greater emission reductions but at the cost of slightly increased waiting times, illustrating the inherent trade-off between efficiency and sustainability in multi-objective RL.

8 Graphical Analysis

To complement quantitative metrics, we analyze training behaviors over 100,000 simulation steps. Appendix A contains plots of key metrics including waiting time and CO_2 emissions.

8.1 PPO-LSTM Convergence

PPO-LSTM shows fast and stable learning, with a sharp drop in waiting time within 20k steps and low variance across runs. Its recurrent architecture enables effective temporal reasoning, helping anticipate traffic bursts.

8.2 PCQL Stability

PCQL exhibits slower but consistent learning. Its conservative penalty prevents early overestimation, resulting in smooth convergence with fewer spikes in emissions and queue length.

8.3 Reward Shaping Impact

Agents trained with composite rewards converge faster and exhibit more balanced behavior in all metrics. Including environmental and fairness components in the reward leads to stronger and more sustainable policies.

9 Discussion

Our results highlight several trade-offs and practical insights:

- **Efficiency vs. Safety:** PPO-based agents optimize throughput aggressively, while PCQL prioritizes stable and risk-averse policies.
- **Temporal Awareness:** PPO-LSTM outperforms others in dynamic scenarios due to its ability to capture sequential dependencies.
- **Reward Design:** Composite rewards that incorporate emissions, delay, and fairness result in better real-world alignment and policy generalization.
- **Scalability:** Independent learners suffice for small grids, but large-scale deployments would benefit from decentralized or coordinated MARL approaches.

- **Computational Trade-offs:** PPO variants require more resources, but produce superior policies. Lightweight methods such as SARSA may be preferable in constrained environments.

Table 2: Qualitative Comparison of RL Algorithms

Algorithm	**Learning Behavior**	**Performance Summary**	**Remarks**
Q-Learning	Gradual reduction, noisy learning	Moderate reduction in wait time and emissions	Requires careful tuning; simple but effective
SARSA(λ)	Smooth and stable from early training	Consistent across runs; modest emission gains	Conservative policy with less exploration noise
DDQN	Fast early drop, then plateaus	Strong waiting time reduction; moderate queue control	Prioritizes throughput; less tuned for sustainability
PPO	Rapid convergence; stable throughout	Best overall trade-off in all metrics	High-performing and robust
TRPO	Steady drop, slight noise	Comparable to PPO but slightly conservative	Enforces stability with constrained updates
PPO-LSTM	Smoothest and most stable learning	Lowest queue, waiting time, and CO_2	Leverages memory for temporal coherence
PCQL	Conservative early; robust late training	Strong queue and wait time reduction; minor emission fluctuation	Safety-aware, best for stability in unpredictable settings

10 Key Observations

- PPO-LSTM achieves the best overall balance between responsiveness and robustness.
- PCQL learns conservatively and ensures safety without a a significant loss in performance.
- SARSA(λ) offers stable early learning, but has limited final performance.
- DDQN is effective with proper tuning, but it can be unstable otherwise.

11 Limitations and Future Work

- Our evaluation is limited to small-scale topologies (single and 2×2 intersections).
- Future work may explore transfer learning or meta-RL to accelerate convergence across varying layouts.
- Integration with real-world data streams (e.g., GPS, traffic sensors) remains an open step.
- Reward signals can be extended to include noise, PM2.5 pollution, and emergency vehicle priority.

12 Division of Work

- **Kumar Prabhat:** All algorithm implementations, PCQL development, reward design, logging system.
- **Sarji Elijah:** Training analysis, visualization of learning curves, summary tables.
- **Sun Yu Ting:** Problem conceptualization, modeling of traffic imbalance.
- **Tan Chee Heng:** Review of the literature, discussion writing, editing, and consistency of reports.

References

Alegre, L. N. (2025). SUMO-RL: Reinforcement Learning Environments for Traffic Signal Control with SUMO. https://github.com/LucasAlegre/sumo-rl. Accessed: 2025-04-20.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095.

Eclipse SUMO (2025). Simulation of Urban MObility (SUMO). https://github.com/eclipse-sumo/sumo. Accessed: 2025-04-20.

Gao, J., Shen, Y., Liu, J., and Wang, Z. (2017). Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. *arXiv preprint arXiv:1705.02755*.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*.

Li, L., Lv, Y., and Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254.

OpenAI (2016). Gym: A toolkit for developing and comparing reinforcement learning algorithms. https://github.com/openai/gym. Accessed: 2025-04-20.

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

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. (2019). Stable Baselines3: Reliable Reinforcement Learning Implementations. https://github.com/DLR-RM/stable-baselines3. Accessed: 2025-04-20.

Stable-Baselines-Team (2021). Stable-Baselines3 Contrib: Experimental Reinforcement Learning Algorithms. https://github.com/Stable-Baselines-Team/stable-baselines3-contrib. Accessed: 2025-04-20.

United Nations, Department of Economic and Social Affairs, Population Division (2018). 2018 revision of world urbanization prospects. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html. Accessed: 2025-04-22.

Van der Pol, E. and Oliehoek, F. A. (2016). Deep reinforcement learning for traffic light control in vehicular networks. *arXiv preprint arXiv:1608.05768*.

Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. (2019). 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.

Wei, H., Zheng, G., Yao, H., and Li, Z. (2018). 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.

Zhang, Y., Zheng, G., Xu, Y., Wei, H., and Wu, Z. (2019). Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. *arXiv preprint arXiv:1905.05217*.

A Appendix

The following appendix presents the training performance of each RL algorithm in three key metrics: total waiting time, queue length, and CO_2 emissions. These plots are generated using a standardized script that processes per-step log files for each algorithm.

Important Note

Evaluation results are available on Table 1

Interpretation Guidelines

Each graph follows the same visual convention:

- **X-axis**: Time step (simulation steps = 1,00,000 steps).
- **Y-axis**: Metric value (e.g., waiting time in seconds).
- **Mean Curve**: Solid line showing the mean over multiple episodes.
- **Shaded Band**: One standard deviation range around the mean, capturing variability.
- **Smoothing**: A moving average (window size = 5) is applied to reduce noise.

Network Configurations

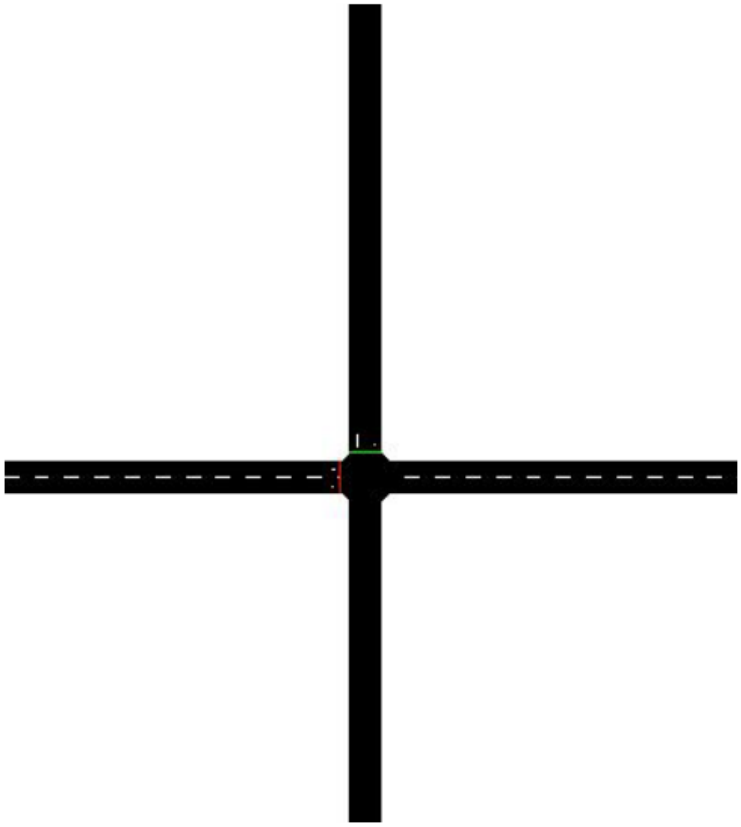

Figure 1: Single Intersection Network Layout

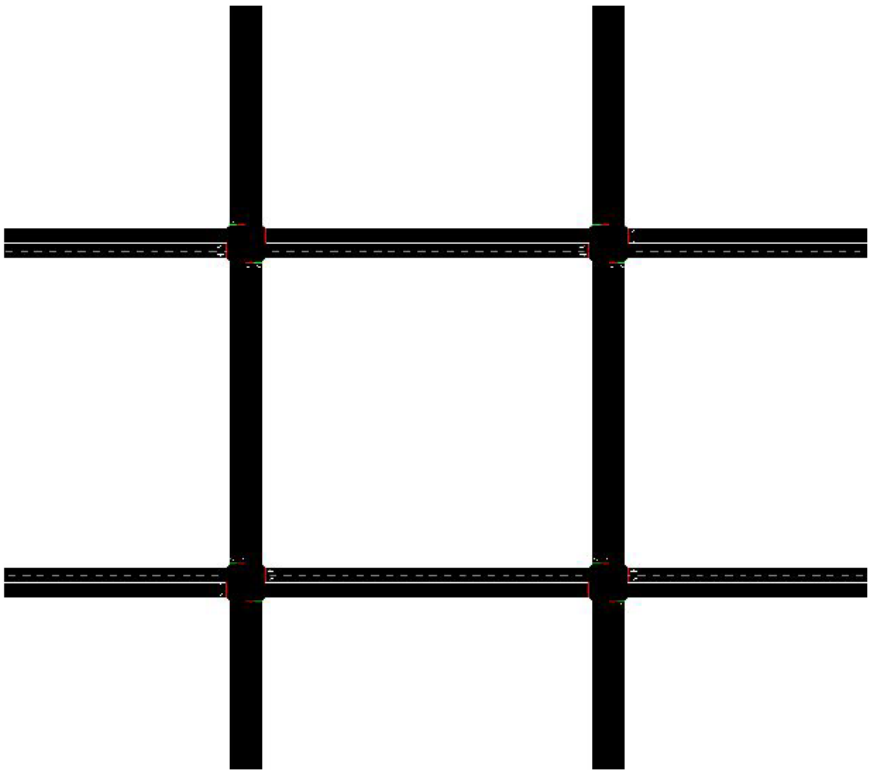

Figure 2: 2×2 Grid Network Layout

B Training Curves for Reinforcement Learning Algorithms

B.1 Q-Learning

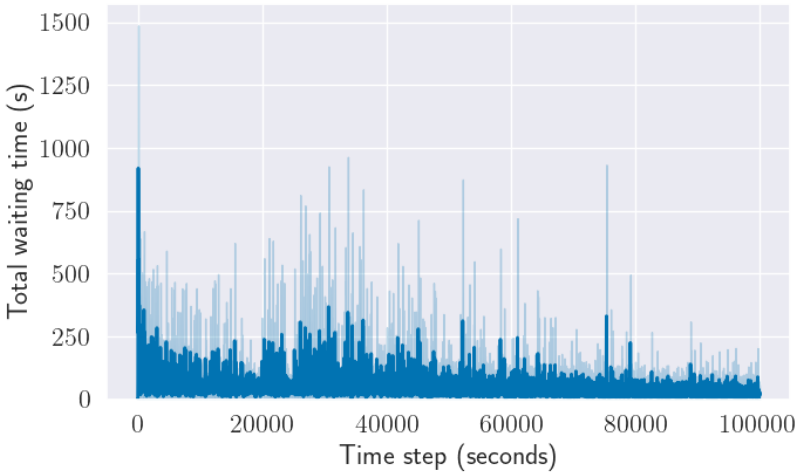

Figure 3: Total Waiting Time

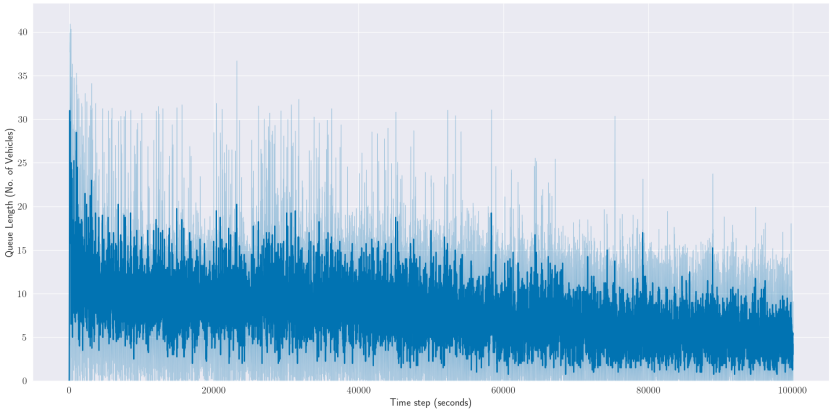

Figure 4: Queue Length

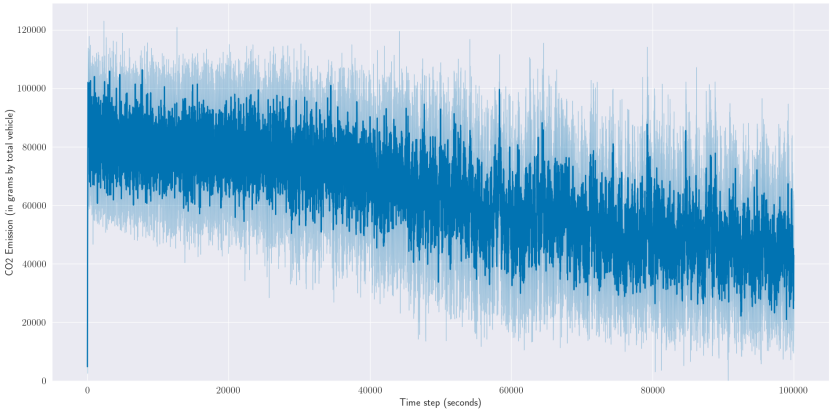

Figure 5: CO_2 Emissions

B.2 SARSA(λ)

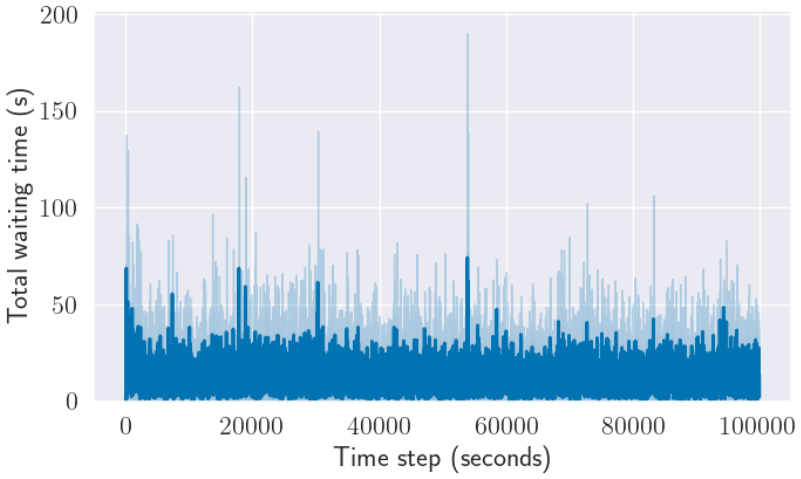

Figure 6: Total Waiting Time

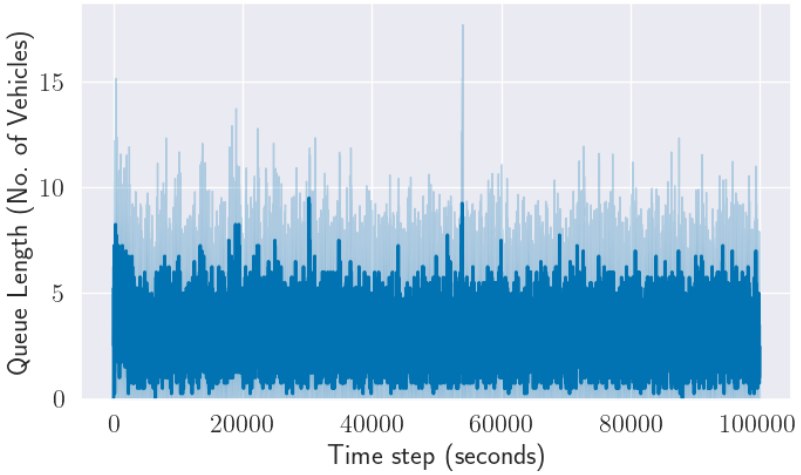

Figure 7: Queue Length

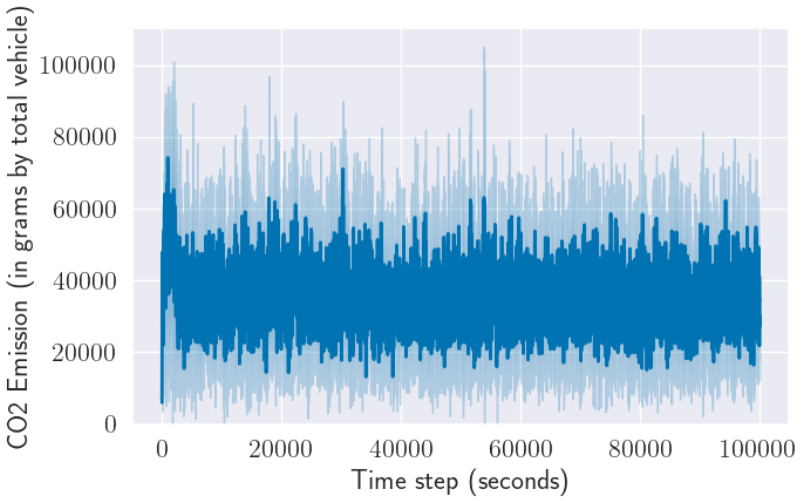

Figure 8: CO_2 Emissions

B.3 DDQN

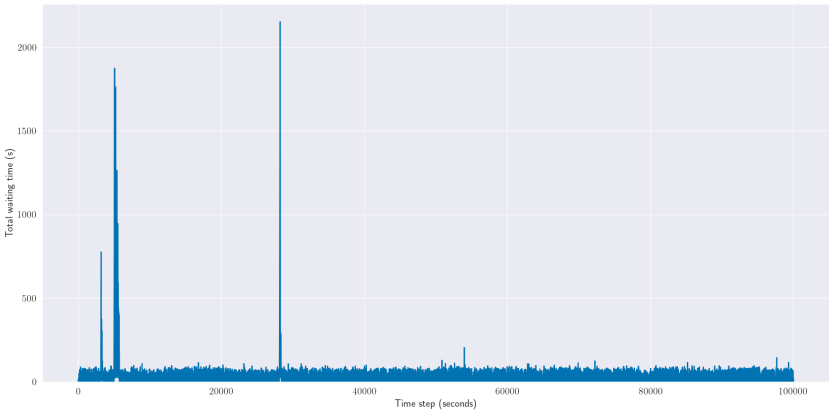

Figure 9: Total Waiting Time

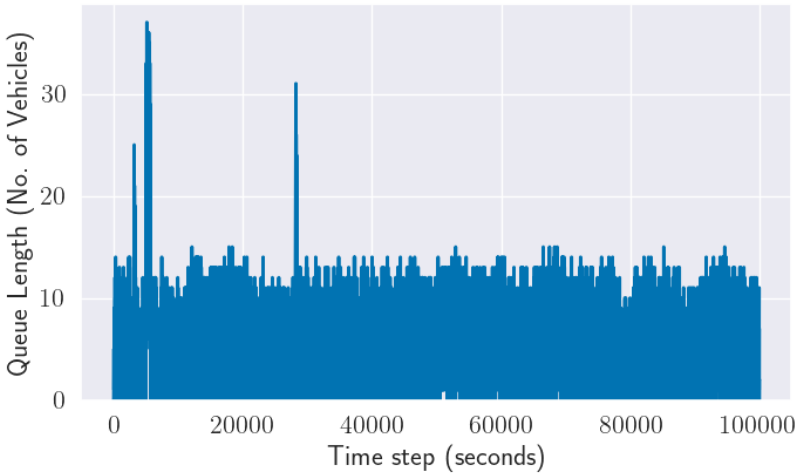

Figure 10: Queue Length

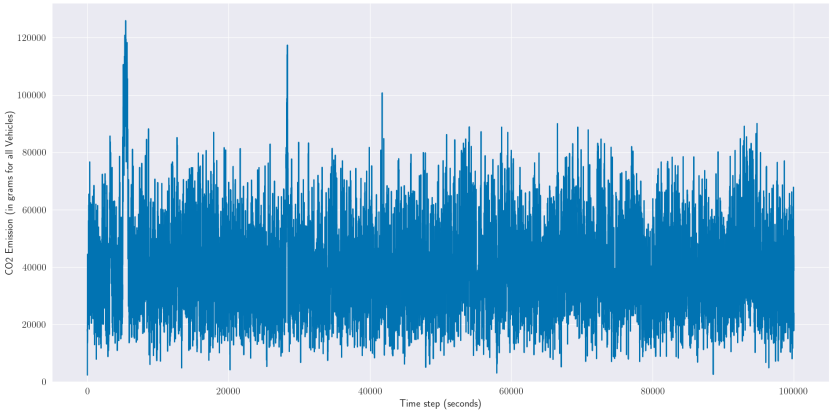

Figure 11: CO_2 Emissions

B.4 TRPO

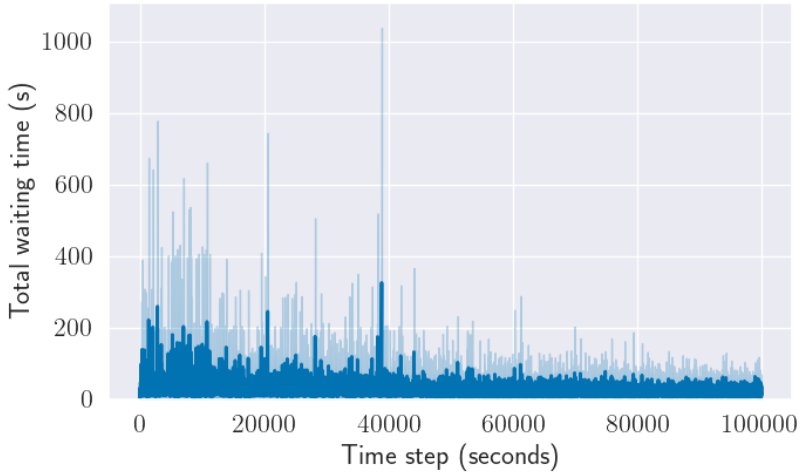

Figure 12: Total Waiting Time

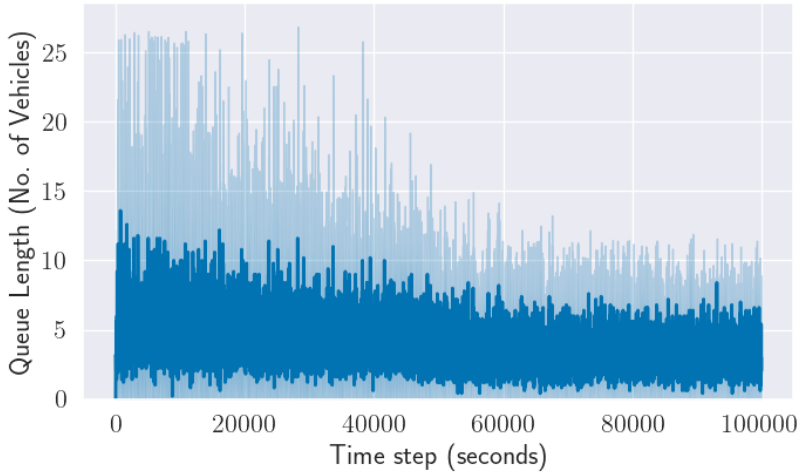

Figure 13: Queue Length

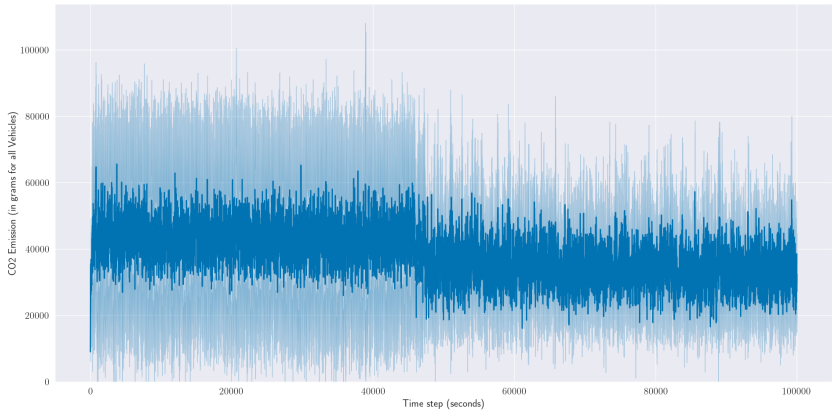

Figure 14: CO_2 Emissions

B.5 PPO

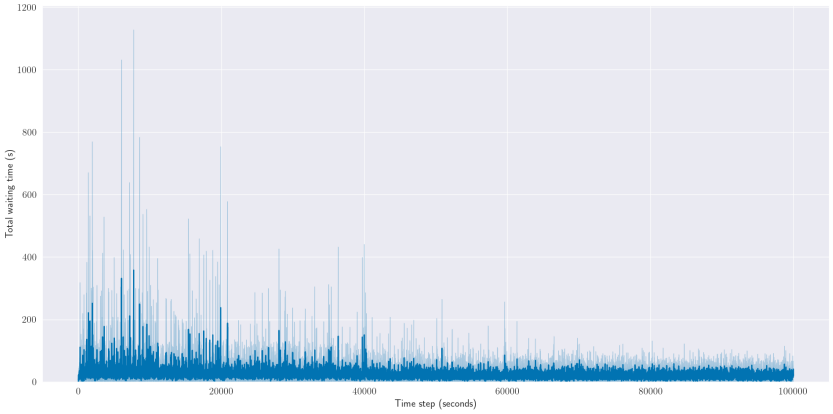

Figure 15: Total Waiting Time

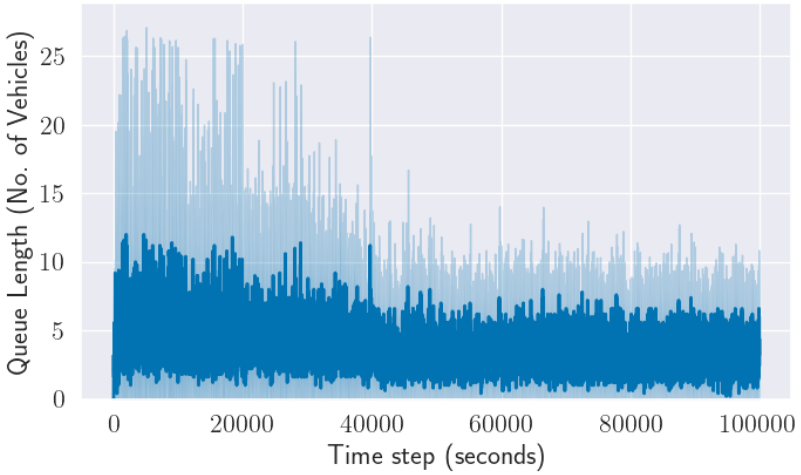

Figure 16: Queue Length

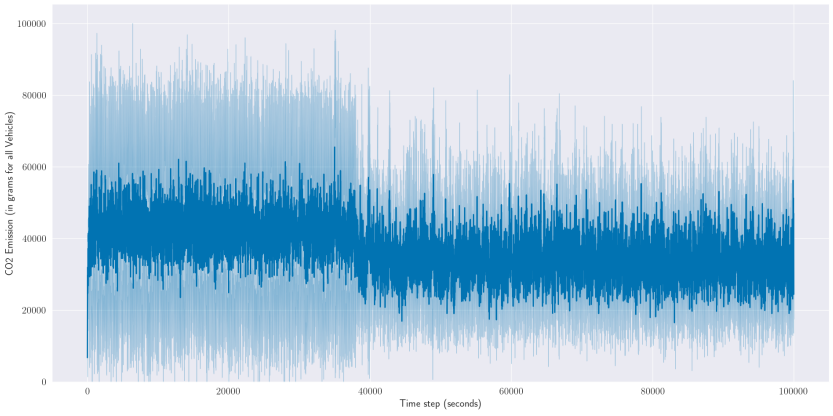

Figure 17: CO_2 Emissions

B.6 PPO-LSTM

Figure 18: Total Waiting Time

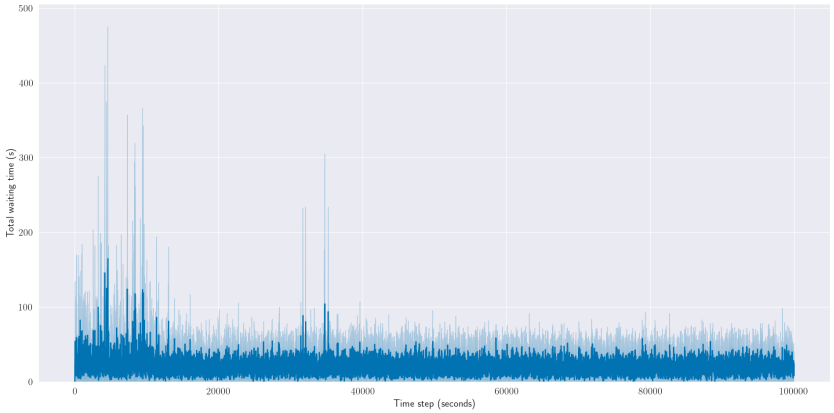

Figure 19: Queue Length

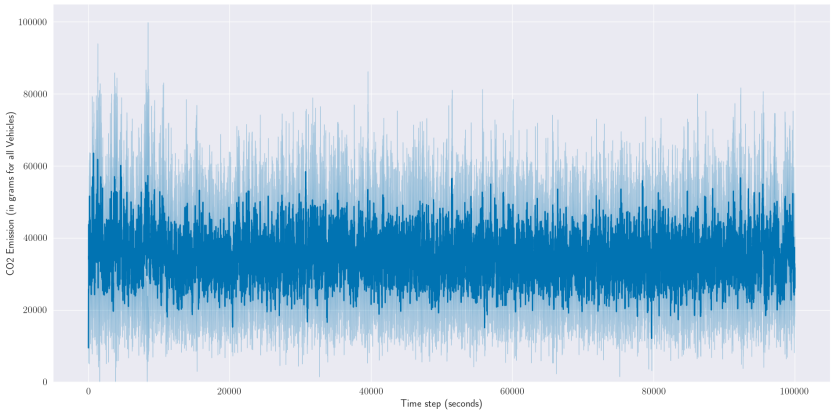

Figure 20: CO_2 Emissions

B.7 PCQL

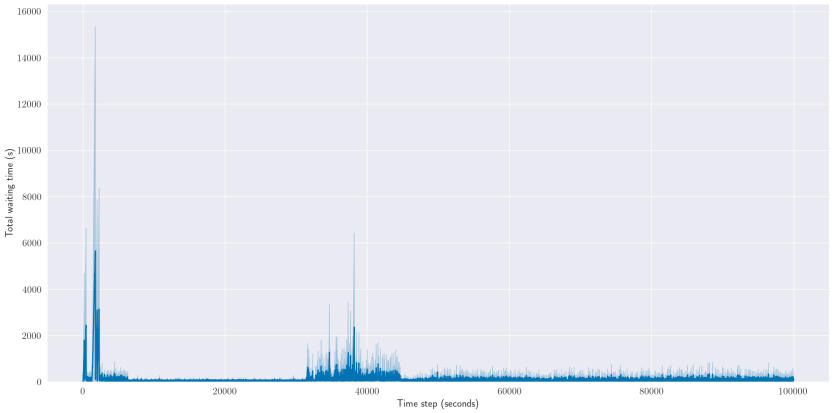

Figure 21: Total Waiting Time

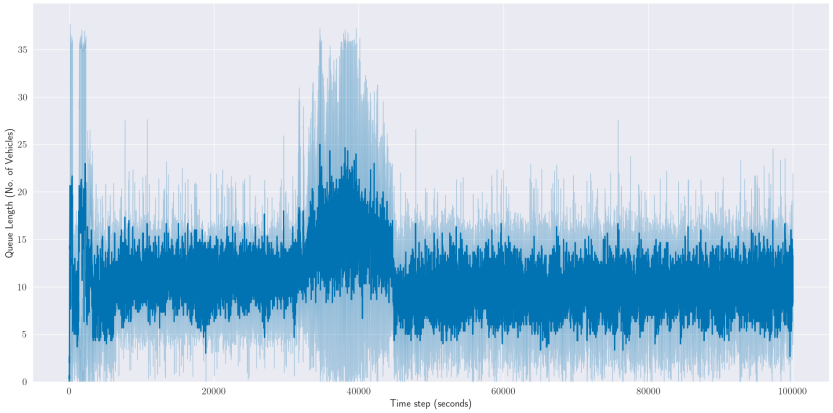

Figure 22: Queue Length

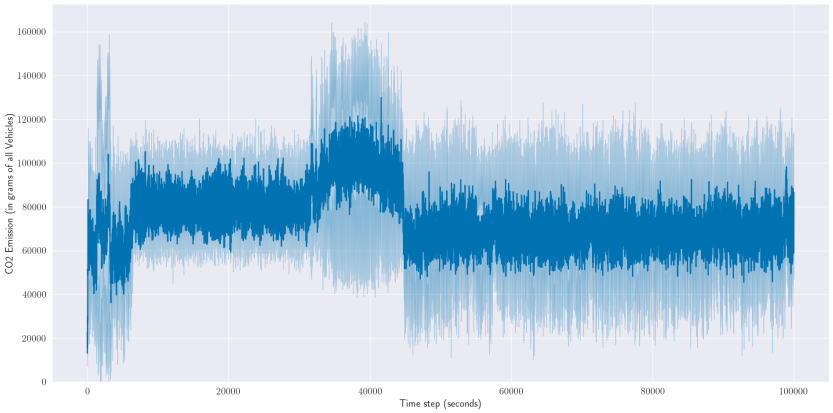

Figure 23: CO_2 Emissions

C AI Acknowledgement

This report was partially supported by the use of generative AI tools, including ChatGPT (OpenAI, GPT-4), for non-substantive assistance. These tools were used strictly for the following purposes:

- Draft refinement and language clarity improvements
- Latex formatting support and structure suggestions
- Explanation and summarization of mathematical equations
- Visualization section formatting and figure layout guidance

No AI tool was used to generate original technical content, research insights, or experimental results. All experiments, algorithm implementations, evaluations, and core report content were independently conducted and validated by the authors.