Traffic flow optimization via reinforcement learning on signal timing on simulated road networks

CS 4100 Final Project, Fall 2024

Mauricio Tedeschi and Luisa Li

Northeastern University
tedeschi.m@northeastern.edu, li.tao@northeastern.edu

1 Introduction

Traffic congestion in dense urban networks is a growing daily challenge for commuters, leading to delays, increased fuel consumption, and environmental harm. In the United States, total traffic delays surged from 1.1 billion hours in 1982 to 5.5 billion hours in 2011, reflecting the mounting pressure on road infrastructure [1]. This issue not only disrupts commutes but also delays supply chains, creates economic inefficiencies, and raises fuel costs. Poor weather and underdeveloped infrastructure further exacerbate these issues, increasing the risk of accidents. Although efforts such as adding lanes or constructing ramps have been explored, these solutions are often impractical due to space and budget constraints [2].

To tackle these challenges, recent years have seen an increased interest in computational approaches, including predictive traffic flow models, speed limit control [2] and advanced traffic signal control systems [3]. Optimizing traffic signal timing, in particular, offers a cost-effective way to improve traffic flow in urban areas. A recent survey by the transportation data company Inrix found that poorly timed traffic signals account for at least 10% of total congestion and delays, highlighting the significant impact of addressing this inefficiency [4].

Building upon the state of the art in reinforcement learning (RL) approaches to optimize traffic signal timing by constructing models that emulate real-world traffic scenarios [2, 3, 5, 6, 7], we design a simulation model that leverages randomly generated, feasible road networks and utilizes RL to train traffic lights to minimize total congestion in the network.

In this paper, we apply Q-learning models to traffic flow optimization with the objective of minimizing traffic congestion. Our approach simulates traffic environments and trains traffic light control systems to predict efficient control patterns that can reduce waiting times, improve traffic flow, and mitigate environmental impacts. Specifically, we aim to demonstrate the feasibility and scalability of these models on urban traffic scenarios. The RL training space becomes much more complex as the model becomes more complex, thus we can only scale up to small subsets of urban networks, such as the Boston Back Bay area.

To achieve these goals, our approach integrates three core components: (1) graph-based road network modeling, where we construct directed graphs to represent road layouts and intersections using Python libraries such as OSMnx [8]; (2) traffic network simulation, where vehicles are modeled with assigned start and end nodes and follow routes between them, interacting with the network in discrete time steps; and (3) reinforcement learning-based control, where traffic light agents are trained using Q-learning to manage intersection signals and reduce congestion.

The remainder of the paper is organized as follows: In Section 2, we present the methods, detailing the graph representation of traffic networks, the simulation environment for vehicle movement, and the reinforcement learning framework. Section 3 describes the experiments and results, including training details, performance evaluation on synthetic and real-world road networks, and comparisons with baseline traffic signal control approaches. Finally, in Section 4, we conclude with a discussion of the implications of our findings and directions for future work.

2 Methods

2.1 Graph construction

A significant portion of this project was dedicated to applying graph theory principles to generate feasible road networks. The most critical part of our simulation for the road network is the graph and its interactions with the rest of the model. A directed graph is a common model for a road network for good reason. Intuitively, road segments can be modeled by edges and intersections as nodes.

To effectively model traffic flow, it is crucial to keep track of congestion information in road segments while simulating vehicle movement through the network. Representing intersections as nodes, however, introduces ambiguity. For example, determining whether a vehicle can move from road segment u to road segment v through a node becomes unclear, as intersections cannot simply be "turned on or off." Instead, we adopt a dual graph approach, where road segments correspond to nodes and *intersection paths* are represented as edges. An intersection path covers any maneuver a vehicle might make at an intersection, such as moving straight, turning right, turning left, or making a U-turn. This representation ensures that vehicles always move

between well-defined road segments, providing a clear and consistent simulation framework.

We construct a graph $G = (V, E)$ as follows:

1. Begin with a desired number of nodes $V_{\mathcal{T}}$ in the original graph and construct a spanning tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$, where $V_{\mathcal{T}} \subseteq V$ and $E_{\mathcal{T}} \subseteq E$. Each edge in $E_{\mathcal{T}}$ is bidirectionally oriented at this stage.

2. Add a set of additional directed edges to $\mathcal{T}$ to increase the complexity of the edge set. Specifically, let E_{extra} be a set of distinct directed edges. Update the edge set:

$$E_{\mathcal{T}} \leftarrow E_{\mathcal{T}} \cup E_{\text{extra}}. \tag{1}$$

   This augmentation will transform $\mathcal{T}$ from a tree into a more general directed graph.

3. Initialize V and E as empty sets. Perform a *graph swap* on $\mathcal{T}$ to construct $G = (V, E)$ as follows:

   - For each edge $e = (u, v) \in E_{\mathcal{T}}$, create a new node in the dual graph:

$$V \leftarrow V \cup \{n_e\}. \tag{2}$$

   - For any sequence of connected edges $(u, v), (v, w) \in E_{\mathcal{T}}$, add a directed edge in the dual graph:

$$E \leftarrow E \cup \left\{\left(n_{(u,v)}, n_{(v,w)}\right)\right\}. \tag{3}$$

   In this mapping, the nodes of the original graph $V_{\mathcal{T}}$ become abstracted, and the edges $E_{\mathcal{T}}$ become the nodes V of the dual graph. We refer to G as the *dual* of $\mathcal{T}$.

2.2 Traffic network simulation model

In this section, we formalize the components of our traffic network simulation model, which will serve as the basis for the rest of the discussion and analysis.

A simulation S is a tuple (G, R, C, L) where $G = (V, E)$ is the graph representing the road network, R is the set of road segments in the network, C is the set of cars en route in the network, and L is the set of traffic light agents controlling intersections.

Once the road network graph G is constructed, the simulation begins by initializing a predefined number of cars. Each car is assigned a route in the network, which is generated using a depth-first search (DFS) algorithm between a randomized pair of start and end nodes. Traffic flow is simulated by vehicle objects moving between adjacent nodes in the graph, which corresponds to cars transitioning from one road segment to another through intersections.

2.2.1 Traffic flow constraints

To ensure the feasibility and realism of the simulation, we impose the following constraints on the traffic flow:

Road segment capacity Traffic congestion occurs on a road segment when the number of vehicles exceeds its maximum capacity [2, 9]. Each road segment $u \in V$ is assigned a maximum capacity $N(u)$ and maintains a set $C(u) \subseteq C$ of vehicles currently located on it. The simulation enforces the following constraint:

For any road segments $u, v \in V$, if a car $c \in C(u)$ is attempting to move to v and v has reached capacity, c must remain on u until v has available space.

Traffic signals Traffic signals control the movement of vehicles at intersections to prevent collisions and regulate flow. Each traffic light agent $\ell \in L$ is associated with:

- A domain of edges $E(\ell) \subseteq E$, where $E(\ell)$ represents the set of intersection paths controlled by ℓ,
- A schedule specifying the signal patterns over discrete simulation time steps.

The domains of all traffic lights form an exact cover of the edges in G, ensuring no overlap:

$$\bigcup_{\ell \in L} E(\ell) = E \tag{4}$$

$$\text{and} \quad E(\ell) \cap E(\ell') = \emptyset \text{ for } \ell \neq \ell'. \tag{5}$$

At each time step, traffic lights activate a subset of edges in their domain ($\subseteq E(\ell)$), allowing vehicles to move through the corresponding intersections. An edge $(u, v) \in E$ is considered *active* if a car is permitted to traverse the intersection from u to v based on the current signal state. The remaining edges remain inactive, restricting vehicle movement through those paths.

2.2.2 Discrete time modeling

A simulation S updates its road network data over a finite number of time steps T. Figure 1a shows a snapshot of a simulated road network. At every time step $t = 1, 2, \ldots, T$, S performs a tick, in which the following occurs:

1. All traffic light agents in the network update their states based on predefined schedules, activating or deactivating edges in their respective domains.

2. Each vehicle in the simulation evaluates its current position in its assigned route and determines whether it can move forward to the next road segment. A vehicle moves forward if:

   (a) The edge connecting its current segment to the next is active.

   (b) The next segment is not at full capacity.

   If these conditions are met, the vehicle transitions to the next segment, updates its internal state, and increments its route position. Otherwise, the vehicle remains in place, contributing to network congestion.

3. If a vehicle reaches the final segment of its route, it is removed from the simulation.

4. The congestion level of the network is updated, defined as the total number of vehicles unable to move forward due to capacity constraints on their next road segment.

The tick function repeats this process for all traffic lights and vehicles in the network, producing an updated state of the road network at each time step.

2.3 Reinforcement learning

In this section, we discuss the integration of our simulation design into a reinforcement learning (RL) framework. We used OpenAI Gym to implement and run our Q-learning algorithm [10]. Because our simulation typically contains many traffic light agents, we adopted a multi-agent RL approach. This design involves maintaining the states and actions of multiple agents while performing sequential updates to their data, ensuring compatibility with the RL framework.

To optimize traffic light schedules in our simulation, we implemented Q-learning, a model-free reinforcement learning algorithm. This section describes the algorithm's setup and integration with our traffic simulation.

2.3.1 Q-learning

Q-learning aims to learn an optimal policy by iteratively updating a Q-value function $Q(s, a)$, where s represents the state of the environment and a denotes an action taken in that state. The update rule is defined as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \quad (6)$$

where α is the learning rate, determining the step size for updates, r is the reward received after taking action a in state s, γ is the discount factor, controlling the importance of future rewards, and $\max_{a'} Q(s', a')$ estimates the maximum future reward for the next state s' [2].

2.3.2 State space

The state (observation) space defines the information available to each traffic light agent for decision-making. For a traffic light agent $\ell \in L$, an observation is represented as a tuple (i_ℓ, o_ℓ), where i_ℓ is the number of incoming cars moving toward ℓ, and o_ℓ is the number of outgoing cars leaving ℓ. These values are computed at each simulation step by iterating over the vehicles in the network and counting the cars transitioning through intersection paths governed by the traffic light.

The complete observation for the system is a collection of tuples, one for each traffic light agent, expressed as:

$$\text{Observation space} = \{(\text{ID}(\ell), (i_\ell, o_\ell)) \mid \ell \in L\}, \quad (7)$$

where $\text{ID}(\ell)$ uniquely identifies the agent, i_ℓ represents the incoming car count, and o_ℓ represents the outgoing car count.

2.3.3 Action space

The action space defines the set of possible actions each traffic light agent can take during the simulation. For each traffic light agent, the action corresponds to selecting a schedule of signal durations from a predefined range of possible values. Specifically, the schedule is divided into k time intervals, where each interval can be assigned a duration up to a maximum value $t_{\max}$. The selected action determines the timing of the traffic light signals for the agent, and these durations are scaled by a factor of 10 to match the desired time granularity in the simulation.

The action space for a single traffic light agent is represented as:

$$\text{AS}(\ell) = \{(t_1, t_2, \ldots, t_k) \mid t_i \in \{0, 1, \ldots, t_{\max}\}\}, \quad (8)$$

where t_i is the duration of the i-th partition of the schedule, and k is the total number of partitions.

For the entire system, the action space is defined as a dictionary where each key corresponds to the unique ID of a traffic light agent, and the value is the agent's action space:

$$\text{AS}_{\text{sys}} = \{\text{ID}(\ell) : (t_i^\ell)_{i=1}^k \mid \ell \in L, t_i^\ell \in \{0, 1, \ldots, t_{\max}\}\}. \quad (9)$$

This structure enables quick lookup to determine the actions for a specific agent.

2.3.4 Reward function

The reward function is designed to minimize traffic congestion and maximize throughput. At each time step, the reward is calculated as:

$$r = -\sum_{c \in C} 1_{\text{isStuck}(c)} \quad (10)$$

where $\text{isStuck}(c)$ is the condition: c is attempting to move to its next node, $v \in V$, but v is at capacity. This formulation penalizes states with high congestion levels, encouraging actions that reduce traffic buildup.

2.3.5 Implementation details

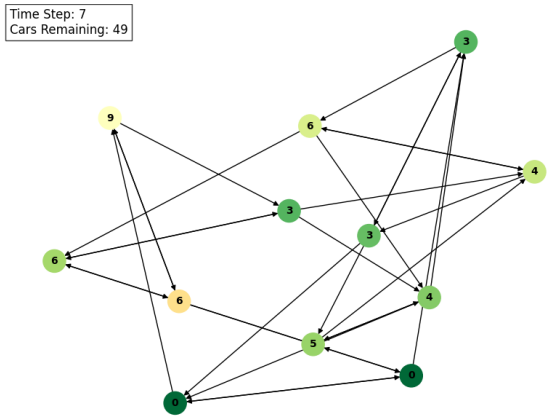
Some additional implementations include:

- **Action space constraints:** To limit the size of the action space, adjacent schedules for each traffic light are precomputed. These schedules restrict signal durations to discrete increments Δt, ensuring computational feasibility.

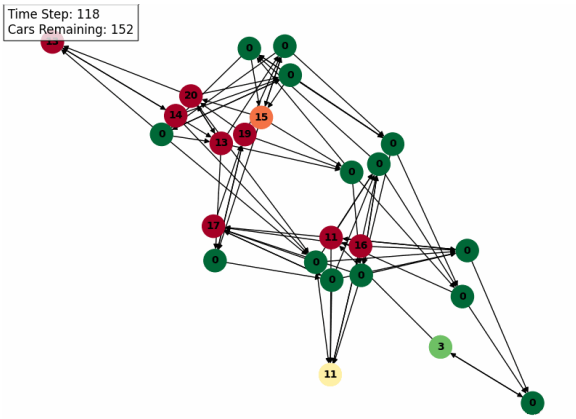
- **Greedy exploration tradeoff:** The exploration rate ϵ is decayed after each episode using an exponential schedule:

$$\epsilon \leftarrow \epsilon \cdot \lambda, \quad \text{where } \lambda < 1 \text{ (by default, } \lambda = 0.9999). \quad (11)$$

- **Environment hard reset:** After each episode, the simulation is reset to ensure the learning process generalizes across different initial conditions.



(a) Plot of a simulated road network. Nodes (road segments) are labeled with their current capacity and are color coded on a green-red map depending on their ratio of current to maximum capacity. Road segments were initialized to have capacities randomly between 10 and 20 vehicles. This network does not contain congestion as all segments contain only up to 9 vehicles.



(b) Plot of a simulated road network with traffic congestion, as indicated by the dark red nodes. Road segments were initialized to have capacities randomly between 10 and 20 vehicles. This creates blockages in the network and prevents cars from advancing to their next road segments.

Figure 1: Plots of different instances of simulated road networks.

3 Results

The following results demonstrate the effectiveness of reinforcement learning in optimizing traffic light scheduling within the simulation environment. The setup, as described in Section 2, involved training the Q-learning algorithm across various episodes to minimize congestion and increase traffic flow.

3.1 Simulation parameters

The key parameters for the trial runs are summarized as follows:

- **Total time:** 200 time steps.
- **Graph structure:** 10 nodes, 20 edges, 180 possible vehicle paths.
- **Traffic light configuration:** 4 time partitions per light, max light partition time of 30 time steps.
- **Vehicle parameters:** 180 vehicles traversing random paths.
- **Learning parameters:** 50 episodes, action space discretized in 4 partitions, delta time of 5 units.

3.2 Performance evaluation

To benchmark the performance of the proposed model, we track the number of vehicles that remain en route throughout the simulation. A lower number of remaining vehicles corresponds to improved traffic flow, as this suggests that vehicles are able to reach their destinations more quickly, thereby reducing overall congestion.

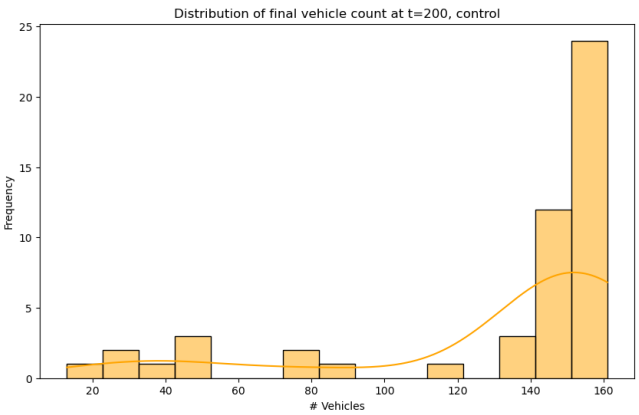
In order to evaluate the feasibility and effectiveness of Q-learning in optimizing traffic flow, we performed a baseline comparison using 50 simulation runs without learning. Each run was initialized with randomly generated vehicle paths and traffic light schedules. The distribution of vehicle counts at the final time step is shown in Figure 2a. On average, the control runs resulted in a mean final vehicle count of 42.89 with a standard deviation of 130.62, indicating highly variable outcomes depending on the random initialization. Nearly 50% of the simulations resulted in final vehicle counts between 150 and 160, as reflected in the pronounced peak in Figure 2a, though some runs achieved as few as 10 to 20 vehicles remaining.

The results obtained using the Q-learning approach show a marked improvement over the non-learning simulations. As depicted in Figure 2b, the frequency distribution of final vehicle counts reveals a peak between approximately 17 and 20 remaining vehicles. The learning-based simulations achieved a significantly reduced mean vehicle count of 5.57 with a standard deviation of 16.54, demonstrating a consistent reduction in congestion across episodes. Additionally, the maximum final vehicle count in the learning case was limited to 33, underscoring the efficacy of the Q-learning optimization in improving traffic flow.

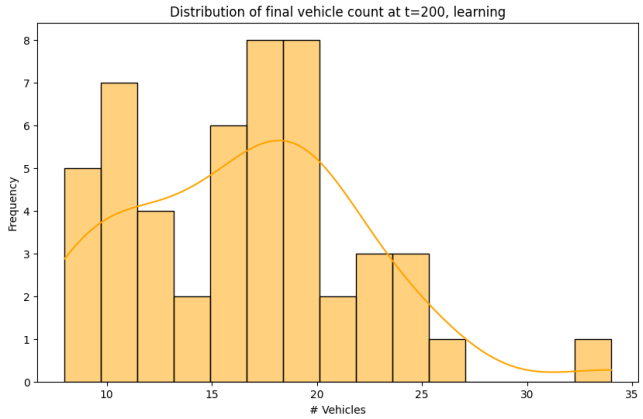
The stark contrast between the mean values and the reduced variability highlights the robustness of the Q-learning approach, which successfully adapts to dynamic traffic scenarios and outperforms the randomly initialized control case.

4 Conclusions

In this paper we explore the use of RL to optimize traffic light scheduling as a means of reducing congestion in urban road networks. Building on years of research in this niche, our approach applies Q-learning to dynamically adjust signal timings based on traffic flow conditions. The results demonstrate



(a) Distribution of final vehicle counts across 50 simulation runs with 200 time steps without Q-learning. The results highlight the variability of the control case, with a pronounced peak between 150 and 160 vehicles remaining. The high mean vehicle count of 42.89 and large standard deviation of 130.62 indicate inconsistent traffic flow performance due to random initialization of schedules and vehicle paths.



(b) Distribution of final vehicle counts across 50 episodes of Q-learning with 200 time steps. The results demonstrate the effectiveness of the optimized traffic light schedules, with a sharp peak between 17 and 20 vehicles remaining. The learning-based simulations achieved a significantly reduced mean vehicle count of 5.57 and standard deviation of 16.54, illustrating a substantial and consistent reduction in congestion compared to the baseline.

Figure 2: Comparison of vehicle count distributions without and with Q-learning, demonstrating the improvement in traffic flow optimization through reinforcement learning.

a significant reduction in congestion compared to a baseline of randomly initialized traffic light schedules, with our model achieving substantial improvements in traffic flow for small to medium-sized simulated urban networks. These findings align with prior work that highlights the potential of RL methods for adaptive traffic control, particularly in localized environments.

One of the strengths of our approach lies in its scalability and adaptability. By integrating graph-based road network modeling with multi-agent Q-learning, the framework is well-suited for dynamic traffic scenarios and provides a foundation for scaling to more complex systems. This adaptability has been a consistent theme in RL-based traffic management research, as such methods can accommodate the inherent variability in traffic conditions. However, similar to other RL-based approaches, our model has several limitations. The reliance on discrete action spaces and simplified simulation parameters constrains its ability to approximate real-world traffic dynamics, where patterns are continuous and often stochastic. Additionally, the choice to update only one traffic light per time step, while reducing action-space complexity, limits RL performance in larger networks. Furthermore, the computational cost associated with both training and real-time decision-making in larger networks poses a significant barrier for scalability.

To address these challenges, future work could explore the use of alternative RL algorithms, such as deep-Q networks (DQN) or policy-gradient methods, to handle larger state and action spaces more effectively. Incorporating real-world traffic flow data, geographical road network information, and external factors like weather conditions could also enhance the robustness of the model. Hybrid approaches that combine RL with heuristic optimization methods may further reduce computational overhead while preserving adaptability.

In summary, this project reinforces the potential of reinforcement learning for improving urban traffic light scheduling, while also emphasizing the challenges that remain in bridging the gap between simulated environments and real-world systems. While Q-learning demonstrates promise for small-scale applications, further advancements are needed to generalize this approach to the complexities of larger, more variable traffic networks. With continued refinement, RL-based frameworks could contribute to the long-standing goal of intelligent, adaptive traffic control systems.

References

[1] D. Schrank, B. Eisele, and T. Lomax, "TTI's 2012 URBAN MOBILITY REPORT,"

[2] E. Walraven, M. T. Spaan, and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 203–212, June 2016.

[3] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, pp. 1575–1587, May 2013.

[4] R. Johnston, "Poorly timed traffic signals are an even bigger time waster than previously thought, report finds," Feb. 2021.

[5] M. Wiering, *Multi-Agent Reinforcement Learning for Traffic Light Control.* Jan. 2000.

[6] M. A. Khamis and W. Gomaa, "Enhanced multiagent multi-objective reinforcement learning for urban traffic light control," in *2012 11th International Conference on Machine Learning and Applications*, (Boca Raton, FL), pp. 586–591, IEEE, Dec. 2012.

[7] J. Spall and D. Chin, "A model-free approach to optimal signal light timing for system-wide traffic control," in *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 2, pp. 1868–1875 vol.2, Dec. 1994.

[8] "OSMnx 1.9.4 documentation." https://osmnx.readthedocs.io/en/stable/.

[9] D. Zhao, Y. Dai, and Z. Zhang, "Computational Intelligence in Urban Traffic Signal Control: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 485–494, July 2012.

[10] "Gymnasium Documentation." https://gymnasium.farama.org/index.html.