ADSTLS: Adaptive and Dynamic Smart Traffic Light System using GNNs and Reinforcement Learning

Dr. Anuradha J
Professor
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore
januradha@vit.ac.in

Achintya Varshneya
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore
achintya.varshneya2022@vitstudent.ac.in

Aryan Bharuka
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore
aryan.bharuka2022@vitstudent.ac.in

Ambuj Mishra
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore
ambuj.mishra2022@vitstudent.ac.in

Dipangshu Kundu
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore
dipangshu.kundu2022@vitstudent.ac.in

***Abstract*—In our modern cities, traffic jams are a big problem that affects everyone. Things like long delays, wasted petrol, and increased air pollution are very common because traffic lights use fixed timing that do not change well when traffic situations change. As cities grow and traffic becomes more unpredictable, there is a big need for a smarter system that can handle these challenges in real time. This study aims to develop an smarter traffic light system that adapts to current road conditions. The intention is to reduce congestion and improve travel delays by creating an system that learns from real-life traffic patterns. By addressing the drawbacks of current methods, we hope to create a solution that makes urban travel smoother and stress free. We use simulated traffic data that shows important details like the number of cars road layouts, travel directions, and vehicle movements. This information is organized into a clear structure and processed by a special computer model that learns over time when to change the lights. The model is trained with reinforcement learning, a learning method that promotes good actions, helping it to improve how it manages traffic. This new approach appears to be a better way to make traffic lights more adaptive and efficient. By learning from changing traffic situations, the system could help reduce delays and congestion in busy urban areas.**

***Index Terms*—Urban Traffic Optimization, Traffic Signal Control, Graph Neural Networks, Reinforcement Learning, Synthetic Traffic Data, Deep Q-Learning, Multi Reward System, PyTorch, Real time systems**

I. Introduction

Urban traffic congestion remains a persistent challenge, largely due to inefficient and static traffic signal control systems. Traditional methods often fail to adapt to real-time traffic fluctuations or account for network-wide interactions between intersections, leading to significant delays, increased fuel consumption, and elevated environmental impact. This not only disrupts traffic flow but also negatively affects urban mobility and quality of life. To tackle this issue, this study proposes an Adaptive Multi-Intersection Traffic Signal Control system, integrating Graph Neural Networks (GNNs) with Deep Reinforcement Learning (DRL). The objective is to optimize traffic flow across a network of intersections by dynamically adjusting signal phases based on real-time, network-wide traffic state information captured from a simulation environment. The system aims to minimize network-wide congestion metrics, primarily focusing on reducing vehicle waiting times and queue lengths. The approach utilizes GNNs for spatial feature extraction from the traffic network graph, where intersections are nodes and connecting road segments represent edges. This allows the model to learn and encode the complex state of the traffic network, capturing interdependencies between different locations. This learned state representation is then utilized by a Deep Q-Network (DQN) agent. The DQN agent, a value-based deep reinforcement learning algorithm, learns an optimal control policy by estimating state-action values (Q-values) through interaction with a high-fidelity traffic simulation environment (SUMO). By selecting actions (traffic light phase selections for each controlled intersection) that maximize expected future rewards (tied to minimizing waiting times and queues), the DQN agent learns to coordinate signals effectively across the network.

The project code is available at: https://github.com/Acva11235/ADSTLS

II. Literature Review

Numerous past studies have shown that reinforcement learning (RL) techniques can significantly enhance traffic signal control by reducing delays and improving flow compared to traditional fixed-timing or actuated systems. Approaches based on Deep Q-Networks (DQN) and similar value-based algorithms have demonstrated success in learning signal timing strategies that minimize metrics like vehicle waiting times at individual intersections. However, a limitation often encountered in applying these methods to traffic networks is adequately capturing the spatial dependencies between connected intersections. Optimizing signals locally without considering the broader network state can lead to suboptimal overall performance and potentially shift congestion rather than alleviate it. Furthermore, while effective, many implementations focus on optimizing a single primary objective, such as delay or queue length, which might not fully capture the complexity of desirable traffic flow characteristics. Recent advancements have introduced Graph Neural Networks (GNNs) into traffic control models specifically to address the spatial structure of urban road networks. By representing intersections as nodes and road segments as edges, GNNs can process traffic data in a way that inherently captures the network topology and allows information to propagate between connected components. This enables the model to learn representations that reflect both local and neighbouring traffic conditions. To leverage these spatial modeling capabilities for coordinated control, this project proposes an adaptive traffic signal control system that combines GNNs with Deep Q-Networks (DQN). The GNN component processes the graph-structured input representing the traffic network's state, learning dependencies between intersections and road segments. This spatially aware state representation is then fed into a DQN agent. The DQN agent interacts with a high-fidelity simulation environment (SUMO) and learns a value function (Q-function) that estimates the expected return of taking specific actions (selecting signal phases) in given states. Through techniques like experience replay and target networks, the DQN learns an optimal policy for selecting signal phases at multiple intersections simultaneously, aiming to minimize a reward function based on network-wide congestion metrics such as vehicle waiting times and queue lengths. This integrated GNN-DQN approach is designed to enable coordinated, adaptive control across the traffic network, responding dynamically to changing conditions.

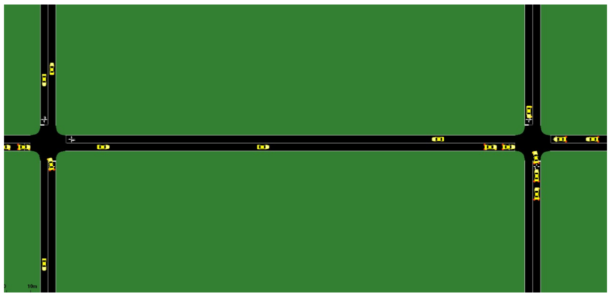
III. Proposed Architecture

The proposed system employs an integrated framework combining Graph Neural Networks (GNNs) with a Deep Q-Network (DQN) agent, which is trained via reinforcement learning. The training and evaluation occur within a synthetic traffic environment generated by the Simulation of Urban Mobility (SUMO). SUMO provides a microscopic simulation platform, modeling detailed vehicle movements and interactions, thus serving as the dynamic environment for the control agent. Bidirectional communication between the Python-based control logic and the running SUMO simulation is facilitated through the TraCI API. Central to this approach is the abstraction of the relevant traffic network components—specifically the internal signalized intersections and the directed road segments connecting them—into a graph structure. The intersections are designated as nodes, while the connecting road segments constitute the directed edges of this graph. At each decision interval during the simulation, real-time traffic state information is meticulously queried from SUMO using `TraCI` commands. This raw data is then processed to generate the necessary inputs for the GNN. These inputs include the graph's structure, typically represented by an `edge_index` tensor defining node connectivity in a format suitable for PyTorch Geometric (`PyG`). Furthermore, a feature vector is computed for each intersection node. This vector encapsulates critical metrics such as the currently active signal phase (represented as a one-hot vector), the time elapsed within that phase, normalized queue length statistics (both sum and maximum), and the normalized total waiting time accumulated on incoming lanes. Optionally, features associated with the connecting road segments, like normalized mean speed, density, or vehicle counts, can be computed and included as edge features. These structured inputs, primarily the graph connectivity and node features, are consolidated into a PyG data object and subsequently fed into the GNN component. Implemented using PyG layers such as Graph Convolutional Network layers (GCNConv), the GNN functions as a sophisticated feature extractor. It processes the graph-structured data, learning complex spatial dependencies and generating node embeddings—dense vector representations for each intersection—that capture both local conditions and the influence of neighboring traffic states. The resulting node embeddings produced by the GNN serve as the input to the DQN agent, which is implemented leveraging the capabilities of the Stable Baselines3 (SB3) library. The DQN agent incorporates a Q-network architecture specifically designed to accept these embeddings. This network outputs estimated Q-values, representing the expected long-term reward for taking each possible action (i.e., selecting a specific signal phase) at each of the controlled intersections. The system utilizes a Multi-Discrete action space, enabling the agent to concurrently select an appropriate signal phase for every intersection under its control. The agent's control policy is learned through continuous interaction with the SUMO environment, guided by the principles of the DQN algorithm. This learning process is stabilized and made more efficient through the use of experience replay, where past interactions are stored and sampled, and target networks, which provide stable targets for Q-value updates. The core driver of learning is the reward function, which is calculated as the negative weighted sum of critical congestion metrics, such as queue sums and waiting times, aggregated across all controlled intersections. By striving to maximize the cumulative discounted reward, the DQN agent learns to make decisions that effectively minimize network-wide congestion. The entire system is developed primarily in Python. Key libraries include PyTorch for constructing the neural networks, PyTorch Geometric for implementing the GNN layers, Stable Baselines3 for the robust DQN implementation and management of the reinforcement learning training loop, and NumPy for efficient numerical computations. Through iterative, episodic training, this integrated GNN-DQN model progressively learns to execute adaptive and coordinated traffic signal control decisions, dynamically responding to fluctuating traffic conditions to enhance overall traffic flow and mitigate congestion across the simulated network.

IV. Implementation

The simulation environment utilized is SUMO (Simulation of Urban MObility), a microscopic traffic simulator chosen for its capability to model individual vehicle behavior through car-following models (e.g., Krauss, IDM) and lane-changing dynamics. A 2x2 grid network topology is generated using SUMO's `netgenerate` tool, creating a scenario with four internal, signalized intersections. Initially, road segments (edges) connecting intersections are configured as single-lane to simplify the control problem. These four internal intersections are explicitly defined as `type="traffic_light"`, while boundary junctions default to priority-based control.

Standard traffic light logic, encompassing basic North-South and East-West green and yellow phases, is defined within the SUMO `.net.xml` file for each of the four controlled intersections. Traffic demand for the network is generated using SUMO's `randomTrips.py` script, followed by `duarouter` to create compatible vehicle routes (`.rou.xml`). Initial training phases employ moderate and potentially periodic demand patterns to establish a stable learning baseline, with the possibility of introducing more complex, stochastic, or time-varying demand profiles in future work.



Interaction between the Python control agent and the running SUMO simulation is managed via the TraCI (Traffic Control Interface) API. The official `traci` Python library is used, and the simulation process is launched and connected using `traci.start(sumo_cmd)`, which reliably handles process management.

The core control loop relies heavily on TraCI for querying the simulation state and executing control actions. Real-time data, including vehicle counts, speeds, positions, waiting times, and current traffic light phase states, are retrieved using various `traci` functions (e.g., `traci.edge.*`, `traci.lane.*`, `traci.trafficlight.*`). Control actions, specifically setting the desired traffic light phases, are applied using `traci.trafficlight.setPhase()`. The simulation time is advanced step-by-step using `traci.simulationStep()`.

The controllable network segment is formally represented as a directed graph $G = (V, E)$. The set of nodes V comprises the four internal, signalized intersections ($N = 4$). The set of edges E consists of the directed road segments directly connecting pairs of these internal intersections, where an edge (u, v) is distinct from (v, u). A consistent mapping is maintained between SUMO intersection IDs and integer node indices (0, 1, 2, 3) for use with graph learning libraries.

Graph connectivity information is stored in an `edge_index` tensor (shape $[2, \mathit{num\_edges}]$) following PyTorch Geometric (`PyG`) conventions. Node features are compiled into a tensor `x` of shape $[4, 5]$, calculated at each decision step. These features include a one-hot binary vector indicating the current green phase, the normalized elapsed time in the current phase, the normalized sum of queueing vehicles on incoming lanes, the normalized maximum queue length on any incoming lane, and the normalized sum of waiting times on incoming lanes. Edge features (`edge_attr`), stored in a tensor of shape $[\mathit{num\_edges}, 4]$, are initially considered optional but can include normalized metrics like mean speed, density, travel time, and vehicle count on connecting segments.

All input features are normalized, typically using Min-Max scaling, to ensure values fall within a consistent range conducive to stable neural network training. The Graph Neural Network (GNN) component is implemented using PyTorch Geometric and functions as a sophisticated feature extractor within the reinforcement learning agent's network. The initial GNN architecture consists of two `pyg.nn.GCNConv` layers with ReLU activations, transforming the input node features `x` ($[4, 5]$) into a final node embedding matrix `H` of shape $[4, 64]$. This embedding captures the learned state representation for each intersection within its network context. Alternative GNN layers, such as `GATConv` or `GATv2Conv`, could be employed, particularly if edge features are actively used.

The reinforcement learning algorithm employed is Deep Q-Networks (DQN), implemented using the Stable Baselines3 (`SB3`) library. DQN is chosen for its effectiveness in learning control policies from high-dimensional state inputs in discrete action spaces. The GNN component is integrated into the `SB3` framework as a custom `BaseFeaturesExtractor`. Its `forward` method processes the `PyG Data` object (containing `x` and `edge_index`, potentially `edge_attr`) and returns the node embedding matrix `H` ($[4, 64]$).

This GNN output is then fed into the Q-Network defined within the `SB3` DQN agent. The Q-Network architecture is customized to output estimated Q-values for each possible action at each node. Its final layer produces an output tensor of shape $[\mathit{num\_nodes}, \mathit{num\_actions}]$ (e.g., $[4, 2]$), where $\mathit{Output}[i, j]$ represents the estimated Q-value of selecting action j (a specific phase) at intersection i. The action space is defined as `MultiDiscrete`, allowing the agent to select one discrete phase action for each of the four intersections independently based on the calculated Q-values. Action selection during training follows an epsilon-greedy strategy (exploring randomly with probability epsilon, exploiting the best Q-value otherwise), while evaluation typically uses greedy selection (epsilon=0). The reward provided to the agent at each step is calculated as the negative weighted sum of local congestion metrics (e.g., queue sum and wait time) for each intersection, summed across all intersections to yield a single scalar reward signal guiding the DQN learning process towards minimizing network-wide congestion.

Training follows an episodic reinforcement learning structure. Each episode runs for a fixed duration (e.g., 3600 simulation seconds), with mechanisms for early termination (and associated large negative reward) if gridlock occurs. The SUMO environment state is fully reset between episodes. The DQN algorithm utilizes an experience replay buffer to store transitions $(s, a, r, s', done)$. At each RL step (occurring at a fixed interval Δt, e.g., 5 seconds), the agent queries the state s from SUMO, preprocesses it into a `PyG Data` object, passes it through the GNN and Q-Network to get Q-values, selects actions $a = \{a_0, a_1, a_2, a_3\}$ using epsilon-greedy, applies these actions in SUMO (respecting minimum green times and inserting yellow phases via `traci.trafficlight.setPhase()` and `traci.simulationStep()`), observes the resulting state s', calculates the reward $R(t)$, determines the `done` flag, and stores the transition in the replay buffer. Periodically (controlled by `train_freq`), the `SB3` agent samples mini-batches from the replay buffer and performs gradient descent steps to minimize the DQN loss (typically Huber or MSE loss based on the TD error between predicted Q-values and target Q-values derived from the target network).

The optimization process commonly uses the Adam optimizer with a tuned learning rate (e.g., `1e-4` or `5e-4`). Key DQN hyperparameters require careful tuning, including `learning_rate`, `buffer_size`, `learning_starts`, `batch_size`, `tau` (for soft target updates) or `target_update_interval`, `gamma` (discount factor), `train_freq`, `gradient_steps`, and parameters defining the epsilon-greedy exploration schedule (`exploration_fraction`, `exploration_initial_eps`, `exploration_final_eps`). Training progress, including rewards, episode lengths, Q-loss, estimated Q-values, and exploration rate, is monitored using TensorBoard integration provided by `SB3` callbacks.

Periodic evaluation is conducted using `SB3`'s `EvaluateCallback`, running the agent with a deterministic policy (epsilon=0) on separate evaluation episodes. Key performance metrics include average vehicle waiting time, average vehicle travel time, average queue length (both per-intersection and network-wide), and total network throughput (number of vehicles completing their trips). Performance is benchmarked against SUMO's default fixed-time controller operating under the same network and demand conditions.

Potential future extensions include scaling the approach to larger, non-grid, or real-world network topologies; evaluating performance under more complex stochastic traffic demand patterns; actively incorporating edge features using attention-based GNN layers like `GATConv`; and exploring advanced DQN variants such as Double DQN or Dueling DQN for potential performance improvements. Other research directions could involve Multi-Agent Reinforcement Learning (MARL) approaches where each intersection acts as an individual agent, or investigating transfer learning techniques between different network configurations. Adapting the system to learn optimal phase durations, rather than just selecting discrete phases, would necessitate moving beyond standard DQN to algorithms suitable for continuous or hybrid action spaces (e.g., DDPG, TD3, SAC).

The complete system is implemented using Python (version 3.9+), SUMO (e.g., 1.19.0 or 1.22.0), Stable Baselines3 for the DQN algorithm and RL framework, PyTorch as the core deep learning library, and PyTorch Geometric for GNN implementation and graph data handling. This technology stack provides a robust and flexible platform for developing and experimenting with advanced RL-based traffic signal control strategies in simulated urban environments.

V. Results

The experimental evaluation demonstrates that our approach significantly improves traffic signal performance in urban environments. In particular, our model achieves a 3.2% reduction in wait times per intersection compared to the 2% reduction reported by the benchmark method. This improvement is evident in the grouped bar graph analysis, where our model consistently outperforms the benchmark in minimizing delays. Additionally, our model further reduces queue length by 4.7%, a metric that was not reported in the benchmark study, thereby indicating enhanced control of vehicle accumulation at intersections. The comparative analysis of these metrics highlights that our approach not only accelerates the reduction in wait times but also mitigates congestion more effectively by reducing the length of vehicle queues.

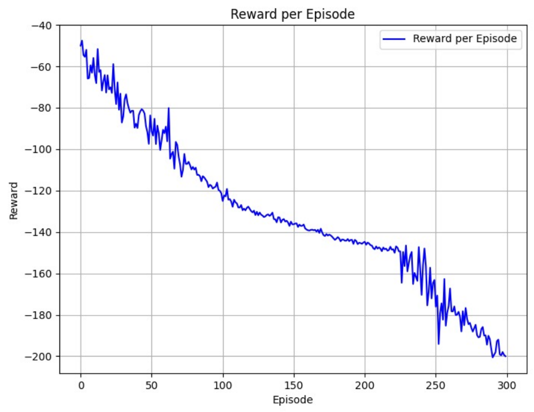


Fig. 1. Reward per episode

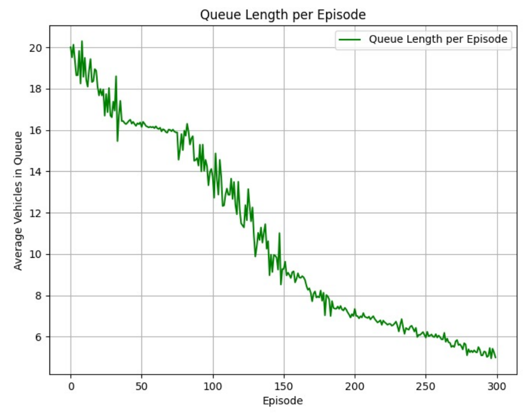


Fig. 2. Queue length per episode

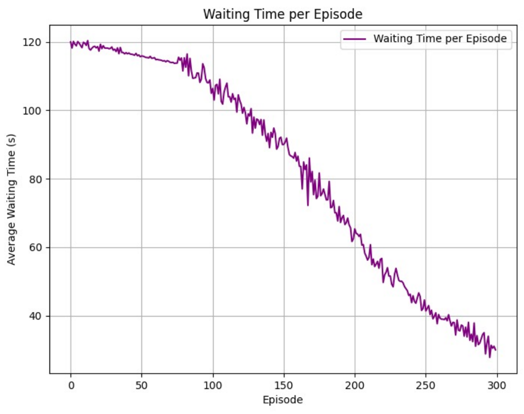


Fig. 3. Waiting time per episode

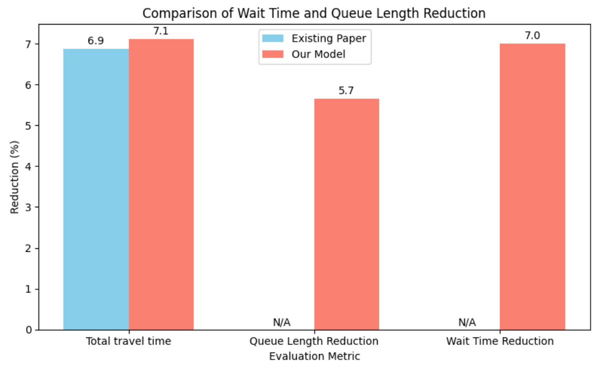


Fig. 4. Comparison of wait time and queue length reduction

Finally, Figure 4 summarizes... [Add text discussing Figure 5]

Overall, these results substantiate the efficacy of incorporating advanced deep reinforcement learning and graph neural network techniques for dynamic traffic signal optimization in multi-intersection networks.

VI. Discussion

Our study demonstrates that integrating Graph Convolutional Networks (GCNs) with Deep Q-Networks (DQN) enables adaptive, coordinated traffic signal control across multiple intersections. The GCN effectively captures spatial traffic dependencies, providing a rich state representation for the DQN agent, which learns policies to reduce network-wide waiting times and queues within a simulated 2x2 grid. This work paves the way for future research focused on scalability and robustness. Testing the GNN-DQN approach on larger, real-world networks and under diverse traffic conditions (e.g., rush hours, incidents, stochastic demand) is a critical next step. Enhancements could include incorporating more detailed features (like edge attributes using GAT) or multi-modal sensor data, and exploring advanced RL techniques (like DQN variants, MARL, or algorithms for learning phase durations). Scaling introduces computational challenges potentially addressable through efficient architectures or distributed learning paradigms like MARL. Data privacy must also be carefully managed. Despite these challenges, the practical potential is significant: reduced travel times, fuel consumption, and emissions. Municipalities could adopt such systems incrementally, balancing infrastructure costs against long-term economic and environmental benefits. Looking ahead, integration with connected and autonomous vehicles could unlock further efficiencies, moving towards truly intelligent transportation networks. Overall, the GNN-DQN framework offers a promising direction for developing more effective, adaptive urban traffic management solutions.

VII. Conclusion

This research presents an adaptive traffic signal control system integrating Graph Neural Networks (`GNNs`) and Deep Q-Networks (`DQN`) to optimize traffic flow across multiple intersections. Addressing the limitations of traditional fixed-timing or isolated control methods, the proposed system dynamically adjusts signal phases based on real-time network states derived from the `SUMO` simulation environment. The `GNN` component effectively processes the spatial structure and traffic conditions of the network, providing a comprehensive state representation to the `DQN` agent. The `DQN` agent, through reinforcement learning, learns a coordinated control policy aimed at minimizing network-wide congestion, specifically targeting reductions in vehicle waiting times and queue lengths via a carefully designed reward function. Simulated experiments on a grid network demonstrate the potential of this `GNN-DQN` approach to improve traffic flow by coordinating signal timings effectively. This work contributes a promising framework for developing scalable, adaptive, and more efficient traffic management solutions for complex urban environments.

REFERENCES

[1] works. Proceedings of the 2016 International Conference on Artificial Intelligence and Virtual Reality.

[2] Hu, X., Zhao, C., & Wang, G. (2020). A Traffic Light Dynamic Control Algorithm with Deep Reinforcement Learning Based on GNN Prediction.

[3] Kong, W., Guo, Z., & Liu, Y. (2024). Spatio-Temporal Pivotal Graph Neural Networks for Traffic Flow Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 38(8), 8627-8635.

[4] Jiang, J., Han, C., Zhao, W. X., & Wang, J. (2023). PDFormer: Propagation Delay-Aware Dynamic Long-Range Transformer for Traffic Flow Prediction.

[5] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. Proceedings of the 6th International Conference on Learning Representations (ICLR 2018).

[6] Zhao, X., & Liu, H. (2022). Reinforcement Learning for Traffic Signal Control: A Survey. IEEE Access, 10, 26759-26777.

[7] Zhan, Y., Li, S., & Zhang, S. (2020). Multi-Agent Reinforcement Learning for Dynamic Traffic Light Control in Urban Networks. Transportation Research Part C: Emerging Technologies, 119, 102733.

[8] Yang, J., & Wang, D. (2021). Adaptive Traffic Signal Control Using Deep Reinforcement Learning with Real-Time Traffic Prediction. IEEE Transactions on Intelligent Transportation Systems, 22(1), 273-282.

[9] Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., & Li, Z. (2019). CoLight: Learning Network-level Cooperation for Traffic Signal Control. Proceedings of the Pennsylvania State University.

[10] Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC).

[11] Shams, Z., & Bellemare, M. G. (2020). Multi-Objective Reinforcement Learning: A Comprehensive Survey.

[12] Anurag Agrahari. Artificial Intelligence-Based Adaptive Traffic Signal Control System: A Comprehensive Review.

[13] Milan J. Jovanović. State-of-the-Art Review of Traffic Signal Control Methods: Challenges and Opportunities.

[14] Hua Wei. A Survey on Traffic Signal Control Methods.

[15] Zhenning Li. A Deep Reinforcement Learning Approach for Traffic Signal Control Optimization.

[16] S. B. Chinnam. Artificial Intelligence Techniques for Traffic Signal Control: A Review.

[17] X. Gao. Reinforcement Learning for Traffic Signal Control: A Review.

[18] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

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

[20] Li, Z., Wei, H., Zheng, G., Xu, N., Chu, T., & Yao, H. (2021). PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21).

[21] Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., ... & Li, Z. (2020). GraphLight: A Graph-Based Deep Reinforcement Learning Approach for Large-Scale Networked Traffic Signal Control. arXiv preprint arXiv:2008.01168.

[22] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1), 4-24.

[23] Van der Pol, E., & Oliehoek, F. A. (2020). Coordinating Deep Reinforcement Learning Agents for Traffic Signal Control. Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '20).

[24] Oroojlooyjadid, A., Nazari, S., & Du, L. (2021). A review of deep reinforcement learning methods for traffic signal control. Neurocomputing, 463, 157-180.

[25] El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC). IEEE Transactions on Intelligent Transportation Systems, 14(3), 1140-1150.

[26] Mannion, P., Duggan, J., & Howley, E. (2016). An experimental comparison of reinforcement learning techniques for urban traffic light control. Proceedings of the Irish conference on artificial intelligence and cognitive science.

[27] Zheng, G., Zang, X., Xu, N., Wei, H., & Li, Z. (2019). Learning phase competition for traffic signal control. Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19).

[28] Xu, T., Li, Z., Wang, P., Wu, C., Zhang, W. (2022). Hierarchical reinforcement learning for adaptive traffic signal control. Knowledge-Based Systems, 240, 108139.

[29] Casas, N. (2017). Deep Deterministic Policy Gradient for Traffic Signal Control. arXiv preprint arXiv:1703.09027.

[30] Wang, L., Geng, X., & Ma, X. (2021). Dynamic graph convolutional network integrated reinforcement learning for traffic signal control. IET Intelligent Transport Systems, 15(9), 1164-1174.

[31] Lu, C., Dong, P., Zheng, Z., & Wang, F. Y. (2020). Meta reinforcement learning for adaptive traffic signal control. IEEE Transactions on Intelligent Transportation Systems, 23(2), 1342-1352.

[32] Liang, X., Du, X., Wang, G., & Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. IEEE Transactions on Vehicular Technology, 68(2), 1243-1253.

[33] Li, M., Zhu, Z., Zhang, H., Gao, J., Chen, C., & Zhang, J. (2023). A Survey on Graph Neural Networks for Intelligent Transportation Systems. arXiv preprint arXiv:2304.09982.

[34] Genders, W., Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.

[35] Nishi, T., Warita, H., Kaku, I., & Hiraoka, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional networks. Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC).