Reinforcement Learning Algorithms with Graph Convolution Networks for Traffic Signal Control

Shreya Salmalge and Shalabh Bhatnagar(✉)

Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India
shreyas@alum.iisc.ac.in, shalabh@iisc.ac.in

Abstract. Traffic congestion is the root cause of various social and economic problems like longer travel times, increased pollution, and fuel or energy consumption. Addressing the issue is becoming increasingly crucial with rising city traffic and limited road infrastructure. The way we change traffic signals has a significant impact on congestion in road networks. We implement reinforcement learning algorithms for controlling traffic signals adaptive to congestion in incoming roads at junctions. Road networks can be viewed as graphs with intersections as nodes and roads as edges. This motivates us to use graph convolutional networks (GCN) as function approximators in various RL algorithms applied to traffic signal control. We implement Deep Q-learning (DQN), Graph Convolutional Q-learning (GCQN), Graph Convolutional Actor-Critic (GCAC), and individual-DQN models to learn a deterministic policy for adaptive traffic signal control. We also present a comparison of the performances of these models and infer that GCQN models are better suited to work for large road networks. To the best of our knowledge, the Graph Convolutional Actor-Critic model is not used in any existing traffic signal control method. We also compare the GCQN and GCAC models against existing and state-of-the-art approaches. Experimental evaluation shows that our proposed method achieves performance levels comparable to the state-of-the-art techniques.

Keywords: Traffic signal control · reinforcement learning · graph convolution networks · actor-critic and deep Q-network algorithms

1 Introduction

The volume of traffic on the road has steadily increased as population and urbanization have grown over time. This has given rise to increased traffic congestion

This work was supported by a J. C. Bose Fellowship, Project No. DFTM/ 02/ 3125/M/04/AIR-04 from DRDO under DIA-RCOE, the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-23-0001), and the RBCCPS, IISc.

A. Kocian et al. (Eds.): INTSYS 2024, LNICST 608, pp. 191–208, 2025.
https://doi.org/10.1007/978-3-031-86370-7_12

across traffic networks. Designing traffic lights that change in response to different traffic conditions is an effective way to optimize traffic flow on a road network. Many traffic junctions worldwide currently use fixed signal timings, i.e., periodically cycle through the sign configurations in a round-robin manner. This method is incapable of adapting to changing traffic conditions. Any intelligent traffic signal control system needs data about traffic, such as waiting times of vehicles, queue lengths, the number of vehicles at the junction, etc., to make decisions adaptively. Some signal control systems use past data to determine signal control rules. These methods use the predetermined rules to select a signal configuration based on current traffic at the junction. While these methods perform better than fixed-timing signal control, they assume that traffic patterns will always resemble the past data used.

Traffic Light Control(TLC) has been a well-studied problem, and hence there have been substantial efforts to create models that control traffic lights to reduce congestion in the road network. Many offline traffic signal control techniques have been proposed for generating signal timings using a static optimizer. Several online TLC algorithms based on techniques, such as genetic algorithms [5], stochastic control [31], and dynamic optimization [23], Neural Networks [24] that adapt in real-time have also been proposed.

Unlike model-based methods, RL-based algorithms do not assume any model of the system and can effectively learn the optimal policy through interaction with the environment. Prashanth et al. [21] proposed a Q-learning-based algorithm with linear function approximation to control traffic lights at junctions. Often precise information about queue length and waiting times of vehicles is hard to obtain. Therefore, the method uses certain thresholds to classify queue lengths and waiting times. Another advantage of using thresholds is that it aggregates the state space, making learning easier for the model. Our methods also use these thresholds for queue lengths and maximum waiting times. Authors have also developed an algorithm to set these thresholds optimally [22]. Chenghao Li et al. [11] studied the fairness control of traffic light and propose a deep RL algorithm to optimize the fairness of all drivers' waiting time.

Dongfang Ma et al. [13], used a deep neural network to capture temporal dependencies followed by actor-critic model to control a single junction. Further, Afshin Oroojlooy et al. [18] proposed an attention based method to create universal model for intersections with any number of roads, lanes, phases (possible signals), and traffic flow. There have also been attempts [3,26] to integrate the 'Max pressure theory' into the reward design of RL methods.

Multi-agent RL has also been used as a potential approach to solve the TLC problem. Unlike previous RL approaches where a single agent controls all signals in a centralized road network, [12,20] present a multi-agent setting with an agent per traffic signal. All agents work cooperatively to reduce congestion in the network. Mohammad Aslani et al. [1] compare deep neural network actor-critic methods while also modeling vehicle and pedestrian traffic demands and driver behavior. Shantian Yang et al. [29] introduced the MOA3CG algorithm,

which uses Actor-Critic and Coordination Graph algorithms to select traffic light at junctions cooperatively.

There are numerous applications where one needs to work with graphs [6–8,28]. The use of graph convolution networks has become increasingly popular in traffic management applications such as traffic flow prediction [19], Autonomous mobility-on-demand systems [4], etc. The deep neural network, by its design, does not take into account the spatial features of the road network. In other words, there is no specific way for DNN to learn the relation between neighboring traffic signals. Also, for the single-agent DNN method, state space dimensionality increases linearly while action space increases exponentially. As a result, though DNN methods perform well on small networks, they can not perform well on larger road networks. The strength of GCN lies in its convolution mechanism that shares state features between neighboring nodes, which can be used to take cooperative actions at nodes of the graph. We represent the road network as a graph, with intersections in the road network as nodes and the roads as edges in the graph. We then leverage the capabilities of graph convolution networks for effective traffic light control.

Several existing methods have used graph convolution networks with reinforcement learning algorithms for traffic light control [17]. Tomoki Nishi et al. [17] have compared the performance of Q-learning with DNN and with GCNs on a regular grid road network. The state in this method comprises only of the queue length and vehicle speeds, whereas the reward structure consists of waiting times. Gyeongjun Kim et al. [9] proposed an asynchronous update method to train multiple actors with Deep graph Q-network. Hua Wei et al. [27] have used Graph Attention Network to facilitate communication between junctions and traffic lights cooperatively. To the best of our knowledge, no existing traffic light control method implements traffic light control with actor-critic using graph convolution networks using function approximation.

We formulate the traffic signal control problem in the Markov Decision Process (MDP) framework, where state quantifies the level of traffic congestion in the network, and the action of an agent is to select a traffic signal phase at every junction for the next signal cycle. We implement Q-learning and Actor-Critic algorithms with a Graph Convolution network as a function approximator. We call these models GCQN and GCAC, respectively. Using Graph convolution networks, we are able to utilize the graph structure of a road network and leverage neighbor nodes' information while making a decision. All our models select a single greedy action in the current state to maximize the total future reward. That is, these methods are trained to get a deterministic policy. We summarize our key contributions below. We perform implementations of all algorithms using the SUMO platform.

1.1 Our Contributions

1) We consider the problem of adaptive signal control of traffic lights at junctions. The objective is to select a traffic signal phase at each junction such that traffic flows through the network are maximized while congestion levels

are minimized. We propose Q-learning and Actor-Critic algorithms with a Graph Convolution network as function approximator.
2) We implement the GCQN and GCAC algorithms. For the purpose of comparison, we implement two variants of Deep Q-learning for traffic light control - a single DQN to control all traffic lights in the network (DQN) and individual DQNs to control every traffic signal (Ind-DQN).
3) We study the performance of the above models on two road networks: (i) a 2×2 grid network with four traffic signal junctions and (ii) Modified Sioux Falls network with 11 traffic signal junctions. Our experiments demonstrate that Graph Convolution models show promising results for problems of traffic signal control.
4) We compare our proposed models against existing approaches and observe that our proposed method achieves performance levels comparable to the state-of-the-art techniques.

The rest of this paper is organized as follows. In Sect. 2, we describe the traffic light control (TLC) problem as an MDP by identifying states, actions, rewards as well as state-action features. In Sect. 3, we describe the Graph Convolution Neural Networks as well as the Q-learning and Actor-Critic algorithms (GCQN and GCAC, respectively) based on these approximators. Section 4 describes implementation details on the SUMO platform and discusses the experimental results. We present concluding remarks in Sect. 5 and discuss some directions for future work.

2 Adaptive Traffic Light Control

In this section we present a formulation of the adaptive traffic signal control problem in the setting of MDPs with the discounted cost objective.

A Markov decision process comprises of a state space S, an action space A, a stationary transition dynamics with conditional probability distribution $p(s_{t+1}|s_t, a_t)$ satisfying the conditional Markov property, and a reward function $r : S \times A \to \mathcal{R}$. According to the conditional Markov property, for any states $s_n \in S$ and corresponding actions $a_n \in A$, $n \geq 0$ chosen in those states, we have $p(s_{m+1}|s_0, a_1, ..., s_m, a_m) = p(s_{m+1}|s_m, a_m)$, $m \geq 0$.

In this setting, the agent dynamically interacts with the environment that can be in one of the states at any instant. At time step t, the agent observes the state $s_t \in S$ and takes an action $a_t \in A$. The environment then transitions into a new state s_{t+1} at instant $t + 1$ according to the transition probability p. In addition, the agent gets a single-stage reward $r_t \equiv r(s_t, a_t)$. Suppose we consider episodes of a fixed or random length T. In other words, T can be a number or alternatively can be a random variable with a certain distribution. Then, we define the return from time t as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ where $0 < \gamma < 1$ is the discount factor. By a policy, we mean a decision rule for selecting actions. Policies can be deterministic or randomized. A deterministic policy prescribes the action to be chosen in any state of the environment while a randomized

policy prescribes a distribution depending on the state of the environment over the set of feasible actions. For simplicity, we consider all actions to be feasible in every state. In infinite horizon tasks as we consider, it makes sense to consider policies that are stationary or time-invariant.

The goal of the agent is to interact with the environment and learn a policy that maximizes the return. An agent uses a policy to select an action in the given state. We denote a stochastic policy by $\pi : S \rightarrow P(A)$ where $P(A)$ is the set of probability measures on A. Further, $\pi(a_t|s_t)$ is the probability of taking action a_t in state s_t. A deterministic policy is denoted by $\mu : S \rightarrow A$ where $\mu(s_t)$ corresponds to the action (a_t) chosen in state s_t under policy μ.

Under a given randomized policy π, Q-value function is defined as follows: $\forall s \in S$, $a \in A$,

$$Q^{\pi}(s,a) = E\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k)) \mid s_0 = s, a_0 = a\right]$$
$$= r(s,a) + E\left[\sum_{k=1}^{\infty} \gamma^k r(s_k, \pi(s_k)) \mid s_0 = s, a_0 = a\right]. \quad (1)$$

2.1 Elements of the Underlying MDP

Consider a road network with p junctions, of which m are traffic light junctions, $1 \leq m \leq p$. Each junction has multiple crossroads and a maximum of k incoming lanes. For adaptive traffic light control, we want to find a deterministic policy that takes in the state of congestion in the road network and decides which traffic light phase to turn on at every junction. The state s_t at time t is a vector of queue lengths and elapsed times in all the incoming lanes at the various junctions. The elapsed time on a lane is the maximum time a vehicle has waited in that lane since the signal turned red. This quantity is zero for lanes on which the signal is green, or where no vehicle is waiting in the lane.

Control decisions are made by a controller that receives the state information from the various lanes and makes decisions on which green traffic light phase to turn on at each of the individual junctions. Action a_t comprises the vector of signal phases (each corresponding to the feasible combination of traffic lights to switch on) at each of the m signalized junctions of the road network. Thus, $a_t = (a_1(t), ..., a_m(t))$ where $a_i(t)$ is the signal phase at junction i during the time slot t. We consider the traffic signal cycle time T, which is divided into green phase time T_g followed by yellow phase time T_y. If, at a junction, the controller decides on the same green traffic light phase for two consecutive cycles, then the same green phase is kept on during the yellow phase following the first green phase at that junction. Note that, since the action is taken every T seconds (at the beginning of each traffic light cycle), actions are chosen when the states of the system are s_{nT}, $n \geq 0$.

The reward is the negative of the cost that in turn has two components. The first is the sum of the queue lengths at the individual lanes, while the second component is the sum of the elapsed times on all lanes. The idea here is to

regulate the flow of traffic in order to minimize the queue lengths while, at the same time, ensuring fairness so that a certain amount of green time does get allocated to each lane after some intervals of time have elapsed even when the queue lengths there are not high. This will ensure that no lane continues to receive the red light for an inordinately large time interval.

2.2 State Aggregation

As proposed in [21], we use state aggregation to decide state features. This is achieved as follows: Let L_1 and L_2 be two prescribed thresholds for the queue length at each lane with $0 < L_1 < L_2$ and let T_1 be the elapsed time threshold. For an incoming lane j at junction i, suppose the queue length at time t is $q_j^i(t)$ and elapsed time is $w_j^i(t)$. Then the aggregated state features $\sigma_{q_j^i(t)}$ and $\sigma_{w_j^i(t)}$ are given as:

$$\sigma_{q_j^i(t)} = \begin{cases} 0 & \text{if } q_j^i(t) < L_1, \\ 0.5 & \text{if } L_1 \leq q_j^i(t) \leq L_2, \\ 1 & \text{if } q_j^i(t) > L_2. \end{cases}$$

$$\sigma_{w_j^i(t)} = \begin{cases} 0 & \text{if } w_j^i(t) < T_1, \\ 1 & \text{otherwise.} \end{cases}$$

The queue length is thus clustered into three levels - less than L_1 (or low congestion region), more than L_1 but less than $L2$ (or medium congestion region), and more than L_2 (or high congestion region), respectively. Similarly, the elapsed time is clustered into two levels, say, low and high respectively. We follow the same notation for queue length and elapsed time throughout the section.

2.3 Traffic Light Control with DQN

For a given state s, the optimal policy is the one that selects the action a that gives the maximum $Q(s, a)$ value over all actions feasible in state s. While the Q-learning algorithm provides the optimal Q-values in the case of full-state representations making use of the iterative scheme [25], this is not practical to implement in these settings where the number of states and actions is excessive in general. The deep Q-network algorithm, DQN, uses a deep neural network to approximate the Q-function. The detailed characteristics of the algorithm are mentioned in [16].

We use Q_θ to approximate the Q-function where θ denotes the vector of network weights in the Q-learning scheme with neural network parameterization. The input state feature vector σ_{s_t} consists of state features for all incoming lanes at each of the junctions. That is, $\sigma_{s_t} = (\sigma_{q_1^1(t)}, ..., \sigma_{q_k^m(t)}, \sigma_{w_1^1(t)}, ..., \sigma_{w_k^m(t)})$. The output layer gives the Q-values of possible actions a_t for a given input state. Here, for a road network with m junctions and four traffic light phase choices at each junction, the size of the output layer would be 4^m. One output node

indicates the Q-value of one action (a combination of phases at all junctions). One typically selects an action with the highest Q-value for a given state.

As mentioned before, the reward $r(s_t, a_t)$ is a real number that is taken as the negative of cost. The cost in turn consists of the sum of the queue lengths and the sum of the elapsed times at the incoming lanes at all junctions.

$$r(s_t, a_t) = -\left(\alpha_i \sum_{i,j} q_j^i(t) + \beta_i \sum_{i,j} w_j^i(t)\right),$$

where we let $\beta_i = 1 - \alpha_i$. For our implementations, we select $\alpha_i = 0.5$ and $\beta_i = 0.5$, respectively. We also maintain target DQN $Q_{\bar{\theta}}$ to calculate the target values. The target network is periodically updated after a fixed number of iterations d. Other implementation details about collecting experience, the architecture of DQN, etc., are provided in Sect. 4.

3 TLC with Graph Convolution Networks

In this section, we discuss about the use of graph convolution networks (GCN) as the underlying neural network based Q-value function approximators. We also discuss two algorithms based on GCN, viz., GCQN and GCAC, respectively.

3.1 Graph Convolution Network

A GCN [10,30] is a multi-layer neural network that can work with graphs and generates embedding vectors of nodes based on the features of their neighborhoods. Formally, consider a graph $G = (V, E)$, where $V(|V| = n)$ is set of nodes and E is sets of edges. Every node is assumed to be connected to itself, i.e., $(v, v) \in E$ for any $v \in V$. Let $X \in \mathcal{R}^{n \times m}$ be a matrix containing all n nodes with their features, where m is the dimension of each feature vector, each row $x_v \in \mathcal{R}^m$ is the feature vector for v. An adjacency matrix A of graph G has diagonal elements set to 1 because of self-loops. The degree matrix D is a diagonal matrix where $D_{ii} = \sum_j A_{ij}$. With one layer of convolution, GCN can capture information about immediate neighbors. With multiple GCN layers stacked, it can integrate information about larger neighborhoods. In an N-layer GCN network, every node can exchange information with up to N-hop neighbors. For a one-layer GCN, the new k-dimensional node feature matrix $L^{(1)} \in \mathcal{R}^{n \times m}$ is computed as $L^{(1)} = \rho(\tilde{A}XW_0)$ where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $W_0 \in \mathcal{R}^{m \times k}$ is a weight matrix. Further, ρ is an activation function such as ReLU, in which case, $\rho(x) = max(0, x)$. One can stack multiple GCN layers as follows: $L^{(j+1)} = \rho(\tilde{A}L^{(j)}W_j)$ where j denotes the layer number and $L^{(0)} = X$. The last layer's feature vectors can be passed to linear layers, i.e., layers with no convolution, in order to improve performance. These linear layers act as individual feed-forward networks for every node's last layer feature vectors.

3.2 TLC with GCQN

We use GCN to approximate Q-function that we now call Q_{gc} where gc are network weights. The GCN model consists of stacked graph convolution layers, followed by linear layers. It operates on a road network graph where n junctions are nodes and roads are the edges. Here, the input state feature σ_{s_t} to the GCN input layer is the matrix $X \in \mathcal{R}^{n \times 2k}$, where row $X_i \in \mathcal{R}^{2k}$ is a vector of $\sigma_{q_j^i(t)}$ and $\sigma_{w_j^i(t)}$. The output layer of GCN $Y \in \mathcal{R}^{n \times c}$ denotes Q-values where c is the maximum number of signal phase choices at any junction. Also, $Y_{ij} = Q_{gc}^{i,j}$ denotes the Q-value of turning the j^{th} signal phase on at junction i. Here, action can be taken at each junction by choosing the signal phase with the maximum Q-value for that junction.

Let $r(s_t, a_t) \in \mathcal{R}^n$ denote the reward vector with ith component $r^i(s_t, a_t)$ being the reward accumulated at junction i and given as follows:

$$r^i(t) \stackrel{\triangle}{=} r^i(s_t, a_t) = -\left(\alpha_1 \sum_j q_j^i(t) + \beta_1 \sum_j w_j^i(t)\right).$$

Similarly, the loss (L_{gc}) is also calculated individually at every junction. For traffic light junction i, we have $L_{gc}^i = \left(Q_{gc}^{i,a_t}(\sigma_{s_t}, a_t) - y^i(t)\right)^2$, where, $y^i(t) = r^i(t) + \gamma \max_j Q_{gc}^{i,j}(\sigma_{s_{t+T}}, j)$. For junctions with no traffic light, the loss is set to zero since the model does not have to learn to choose an action there. The total loss for the model is the sum of losses at all nodes. Thus, $L_{gc} = \sum_i L_{gc}^i$ The detailed training procedure is provided in Algorithm 1.

Algorithm 1. Training GCQN in batches with target GCQN

```
Initialize replay buffer B and GCQN Q_ω, target GCQN Q_ω', tuf = 0, d
For episode = 1 to max_episodes
    Initialize empty experience replay buffer B
    Reset the environment
    For step t = 1 to max_episode_length
        Select action a_t = (a_t^1, ..., a_t^n) according to ε-greedy strategy
        Execute action a_t and observe r_t, s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in B
        s_t = s_{t+1}
    For epoch = 1 to max_epochs
        Sample a mini-batch S of size s
        For (s_t, a_t, r_t, s_{t+1}) ∈ S
            Set y^i(t) = r^i(t) + γ max_j Q_ω'^{i,j}(σ_{s_{t+T}}, j) for each of the junctions i
            Set L_ω(t) = Σ_i (Q_ω^{i,a_t}(σ_{s_t}, a_t) − y^i(t))^2
        Update ω by minimizing loss L(ω) = (1/s) Σ_t L_ω(t)
        IF n%d = 0
            Update target network parameters as ω' = ω
        tuf = tuf + 1
```

A k-layered Graph Convolution leads to sharing of state features, i.e., queue length and waiting time information amongst k-hop neighbors of a traffic light junction. Thus, traffic light at a junction is chosen with some knowledge of congestion at k-hop neighbor junctions. The reward and corresponding loss are calculated separately for each junction. While training the models, the loss at junctions is back-propagated till its k-hop neighbors. GCN learns the right weights to combine state features of neighborhood nodes, that is, it learns the dependencies between neighboring nodes. A well trained graph convolution model can thus take an action at a traffic light junction in a cooperative manner in order to maximise the overall reward and thus minimise the overall congestion in the road network. This is in contrast to using individual DQNs at each junction that tries to maximize the rewards locally, at the junction. This is also confirmed by our experimental evaluations provided in Sect. 5.

3.3 TLC with GCAC

The actor-critic [2] is a widely used architecture that incorporates a combination of policy and value based methods to train the agent. A variant of this approach is the Advantage Actor-Critic (A2C) [15]. A2C consists of two components - an Actor that approximates the policy π_θ with parameter θ and a critic that approximates the value function $V_{\theta'}$ with parameter θ'. A2C uses the Advantage function $A(s, a)$ to calculate the Temporal Difference (TD) Error. The advantage function determines how much better a certain action is than the action chosen as per the given policy. Thus, if $Q^\pi(s_t, a_t) > V^\pi(s_t)$, it is advantageous to select action a_t in state s_t as opposed to selecting an action according to π in that state. Both actor and critic use the advantage function to calculate the loss.

We use GCN to approximate the policy actor π_θ, where θ are the weights of the GCN. Likewise, we use GCN to approximate the value critic $V_{\theta'}$, where θ' are the weights of this GCN. The input to the actor GCN $X \in \mathcal{R}^{n \times 2k}$ is the same as the input to GCQN. The output layer of the actor gives probability distributions over possible actions a_t at every node for a given input state. Formally, for a network with n nodes and c maximum signal phase choices at any junction, the output layer of GCN is $Y \in \mathcal{R}^{n \times c}$, where $Y_{ij} = \pi^i_\theta(j|s_t)$ denotes the probability of turning the j^{th} signal phase on at junction i according to the actor's policy π_θ. Thus, at the ith junction, action is sampled from the distribution π^i_θ.

The input to the critic GCN $X \in \mathcal{R}^{n \times 2k}$ is the same as that to the actor GCN. The output layer of GCN is $Y \in \mathcal{R}^n$, where $Y_i = V^i_{\theta'}(s_t)$ denotes the component of node i in the value of state $V_{\theta'}(s_t)$. The reward $r(s_t, a_t) \in \mathcal{R}^n$ is calculated in the same way as for GCQN.

The actor and critic losses, viz., L^i_θ and $L^i_{\theta'}$, respectively, are also calculated for all nodes. For junctions with no traffic light, the loss is set to zero as before. The total losses of the actor and critic are the sum of respective losses at each node. Thus,

$$A^i(s_t, a_t) = r^i(s_t, a_t) + V^i_{\theta'}(s_{t+T}) - V^i_{\theta'}(s_t) L_\theta(s_t, a_t) = \sum_i L^i_\theta(s_t, a_t)$$

$$= -\sum_i \log(\pi^i_\theta(a_t|s_t) \times A^i(s_t, a_t) L_{\theta'}(s_t, a_t) = \sum_i L^i_{\theta'}(s_t, a_t) = \sum_i (A^i(s_t, a_t))^2.$$

The parameters of actor and critic are updated to reduce the respective losses. The training algorithm is given below, see Algorithm 2.

Algorithm 2. Training GCAC model with A2C algorithm

Initialize the actor and critic parameters θ and θ'
For *episode* = 1 to *max_episodes*
    Initialize empty experience replay buffer B
    Reset the environment
    For step $t = 1$ **to** *max_episode_length*
        Sample action a^i_t from $\pi^i_\theta(\cdot|s_t)$ for all junctions i
        Execute action $(a^1_t, ..., a^n_t)$ and observe r_t, s_{t+1}
        For junction i
            Set $return^i(t) = r^i(s_t, a_t) + V^i_{\theta'}(s_{t+1})$
            Set $A^i(s_t, a_t) = return^i(t) - V^i_{\theta'}(s_t)$
            Update θ' by minimising the critic loss $\sum_i (A^i(s_t, a_t))^2$
            Update θ by minimising the actor loss $\sum_i (-\log(\pi^i_\theta(a_t|s_t) \times A^i(s_t, a_t))$
        Set $s_t = s_{t+1}$

4 Experiments and Results

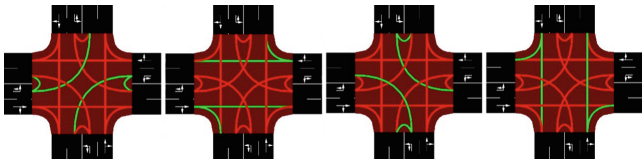

Fig. 1. Four possible green phases of a traffic signal at a junction: Green lines indicate turns that vehicles from corresponding incoming lanes are allowed to take. Each green phase has a corresponding yellow phase with green lines replaced by yellow lines (Color figure online)

We experiment with all methods described in earlier sections for adaptive traffic light control. We use "Simulation of Urban MObility" (SUMO), which is an open-source, highly portable, microscopic, and continuous traffic simulation package.

The GCN models are implemented with the PyTorch Geometric library. We implement TLC methods on two different road networks that we describe below. (i) a 2×2 grid road network with four traffic light junctions and (ii) Modified Sioux Falls road network with eleven traffic light junctions. Sioux Falls network is used in many transportation network studies. We modified it such that no junction has more than four incoming roads. Both networks are shown in Fig. 2. For state aggregation, we choose values of L_1, L_2, T_1 as 7 m, 15 m, and 13 s, respectively, for both these networks. Further, in both of these networks, at every traffic light junction, there are four possible green traffic light phases, as shown in Fig. 5. A traffic light cycle T lasts for 14 s. Green phase time T_g is 10 s, and yellow phase time T_y is 4 s. In each method we implement for TLC, the controller model selects the next green traffic light phase at each junction at the end of the green phase. If the next selected phase is different from the currently selected phase, then yellow phase corresponding to previous green phase is turned on otherwise green phase continues to be kept on. After T_y duration, the next green phase is turned on for T_g duration. In the 2×2 grid road network, each traffic light junction has eight incoming lanes - 4 incoming roads with two lanes on each road. In every episode, 1,000 vehicles are generated with random source and destination roads. The maximum length of an episode is 5,400 steps (seconds). However, an episode can end before the maximum steps if all vehicles have reached their destination. For every model, we perform training for 800 episodes. A batch of size 100 is sampled from experience collected in that episode. In each episode, the model is trained on 800 such batches. All models are trained with Adam optimizer with an initial learning rate set to 0.001. The discount factor γ is set to 0.75. Further, ϵ is set to 1 at the start of the training and is decreased with a factor of 0.997. The various architectures for all the 2×2 grid road network models used in the algorithms are given below. The input and output layer sizes of these models are calculated as described in (i)–(v) below.

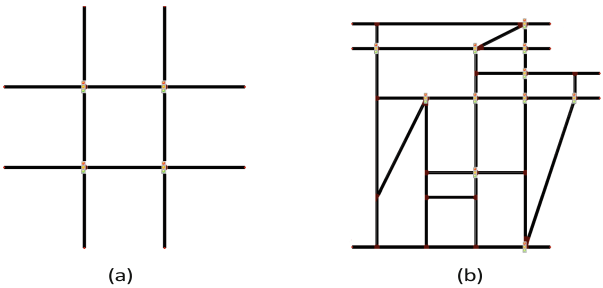

Fig. 2. (a) 2×2 Grid road network, (b) Modified Sioux Falls network

(i) Deep Q-learning network **(DQN)**: Input layer size of DQN is 64. The output layer size is 256. DQN has three hidden linear layers, each of size 128. The *Relu* activation function follows each hidden layer.

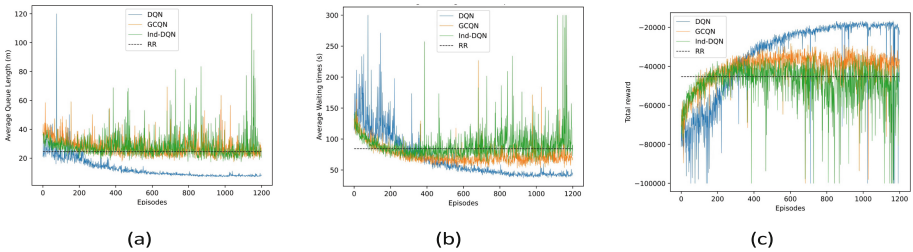

Fig. 3. Average queue length (a), Average waiting time (b) of vehicles and Total reward (c) over training episodes for modified 2×2 grid network

(ii) Graph Convolution Q-learning network (**GCQN**): There are 12 nodes in the network, out of which four are traffic light controlled. GCQN operates on a graph of 12 nodes. The input layer size here is 12×16. There are two stacked convolution hidden layers of size 12×64, each followed by tanh activation functions. These are followed by a linear layer of size 12×64 and tanh activation. This layer is then followed by an output layer of size 12×4.

(iii) Graph Convolution Actor-Critic Network (**GCAC**): The actor-network of GCAC is the same as GCQN, and the output layer is followed by the Categorical distribution layer that gives out a probability distribution over actions. The critic network is also the same as GCQN, except the output layer is of size 12×1.

(iv) Individual DQN networks (**Ind-DQN**): This architecture is similar to GCQN, where linear layers of the same size replace convolution layers. This is equivalent to having a separate DQN at each node of a network.

(v) Round Robin (**RR**): We compare the performance of each of the above networks with round-robin controlled traffic light junctions. In this method, traffic light phases move in cycles in a round robin manner and the same phase at any junction cannot occur consecutively. We keep traffic light cycle time to $T = 30$ seconds with $T_g = 25$ and $T_y = 5$. Since there is no learning involved in RR, it performs roughly the same in every episode. We take the average over a few episodes and use that as a baseline to compare against the other methods.

Figures 3 (a) and (b) show the comparisons of the average queue length at the junction and the average cumulative waiting time of vehicles, respectively. Since there is no training involved in the round-robin method, performance of this method is roughly the same in every episode. We take a single value averaged over 20 episodes for all the comparisons. The best performance for the 2×2 Grid network is achieved by the DQN model, followed by the GCQN model. The GCAC model, however, could not learn for this small road network, and hence result curves are excluded from the figures. The average queue length went down to about 8 m in the DQN model and 24 m in the GCQN model. The Average Cumulative waiting time of a vehicle went down to about 42 s in

the DQN model and 72 s in the GCQN model. Figure 3 (c) shows rewards over training episodes.

For a small graph such as a 2×2 network, DQN can perform well. Here, GCQN and GCAC models suffer from a well known over-smoothing problem in GCN. Using a Deeper Graph Convolutional Network (GCN) on a smaller graph leads to the model producing identical node features across the graph that hinders the performance of the model. But as the size of the graph increases, state space increases linearly w.r.t. the number of nodes, while the action space increases exponentially. Hence it becomes infeasible to use DQN for large networks. One way to avoid this could be to use separate DQNs at every junction. But this model cannot take into account congestion levels in neighborhood nodes. As a result, it performs poorly. The GCQN and GCAC models, on the other hand, perform well on larger graphs since these can share congestion information with neighborhood nodes with their convolution mechanism.

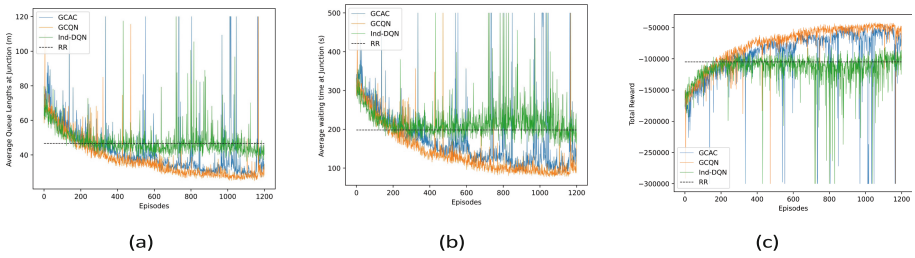

Fig. 4. Average queue length (a), Average waiting time (b) of vehicles and Total reward (c) over training episodes for modified Sioux Falls network

Keeping all the training configurations same, we experiment with different models on a larger road network - the modified Sioux Falls road network. Here one thousand five hundred vehicles are generated in each episode. There are 31 nodes in the network, out of which 11 are traffic light controlled. No junction has more than four incoming roads. Each road has two lanes. Architectures of models are similar to those for 2×2 grid network and are described below:

- Graph Convolution Q-learning network (**GCQN**): Input layer size is 31×16. The number of layers and activation functions is kept the same as for the 2×2 Grid network. Only sizes of layers now become 31×64, and the output layer is of size 31×4.
- Graph Convolution Actor-Critic Network (**GCAC**): The actor-network of GCAC is the same as GCQN, and the output layer is followed by the Categorical distribution layer that gives out a probability distribution over actions. The critic network is also the same as GCQN, except the output layer is of size 31×1.
- Individual DQN networks (**Ind-DQN**): This architecture is similar to GCQN, where linear layers of the same size replace convolution layers. This is equivalent to having separate DQN at every node of a network.

– Round Robin (**RR**): We compare the performance of each of the above networks with round-robin controlled traffic light junctions.

Figures 4 (a) and (b) show the comparisons of the average queue length at the junction and the average cumulative waiting time of vehicles, respectively[1]. Again all RL models are able to achieve better performance than the round-robin method. The best performance for this network is achieved by the GCQN model, followed by the GCAC model. The average queue length went down to about 30 m in the GCQN model and 38 m in the GCAC model. The Average Cumulative waiting time of a vehicle at the junction went down to about 102 s in the GCQN model and 172 s in the GCAC model. Figure 4(c) shows rewards over training episodes.

Table 1. Total Queue length and Total waiting times at the junction in road networks averaged over last 100 episodes of Training

	2×2 Grid network		Modified Sioux Falls network	
	Queue Length(m)	Waiting Time(s)	Queue Length(m)	Waiting Time(s)
Round-Robin	24.52	84.43	46.67	198.25
DQN	**7.87**	**42.53**	–	–
Individual-DQN	30.90	115.84	44.19	198.76
GCQN	24.27	72.03	**30.33**	**102.98**
GCAC	36.00	246.13	38.71	172.88

Similar to the 2×2 grid, the GCQN model performs better than individual DQN models. GCAC also performs better than individual DQN models. However, towards the end of the training, there are some episodes where the GCAC model shows a sudden increase in queue lengths and waiting times. GCQN model shows more stable behavior as compared to GCAC. The average queue length and waiting time for the various models is given in Table 1.

Additionally, we have benchmarked our GCQN and GCAC models against some state of the art methods. We perform this experiment in the Libsignal framework which provides OpenAI Gym-compatible environments for traffic light control scenarios and collection of baseline models [14]. In our experiments, we used the PressLight [26], MpLight [3] and CoLight [27] models from Libsignal directly, without altering any hyper-parameters. Experiments were conducted on a 4×4 grid road network with 16 intersections. We use here an open dataset - 'hangzhou_4×4_gudang_18010207_1h' traffic flow data, which is based on camera data in Hangzhou city. It consists of 4,000 vehicles departing in an hour from one of the nodes. We present below performance comparison in terms of Average

[1] Queue length values greater than 120 m are set to 120 m while plotting the curve to get proper visualization of performance comparison. Similarly waiting time and reward values are cropped as necessary in the respective graphs.

travel time of vehicles, Average queue length, Approximated delay. More details about experimental setup, traffic flow dataset and evaluation matrices are available in the github repository[2]. Due to lack of space, we do not show plots for the 4×4 grid setting. Due to the complexity of the model, GCAC needs around 200 episodes to converge. Table 2 shows the performance comparisons of all trained models. Here, all models are trained for 100 episodes, except GCAC that is trained for 200 episodes. Although CoLight gives the best performance among all models, GCQN does demonstrate noteworthy performance. In particular, GCQN converges faster than MPLight and PressLight and outperforms these, in terms of Queue length. Even though GCAC doesn't outperform any model, it shows relatively notable performance as well. One can experiment with various training techniques to achieve faster convergence in GCAC. Both GCQN and GCAC show comparable results to the SOTA methods and are worth exploring further for future research.

Table 2. Average Travel time, Queue Length, delay in 4×4 grid road networks with trained models (averaged over 5 episodes)

	Average Travel time	Queue Length	delay
CoLight	333.37	0.152	0.087
MPLight	373.71	0.287	0.096
PressLight	346.66	0.210	0.106
GCQN	347.00	0.195	0.100
GCAC	367.66	0.295	0.131

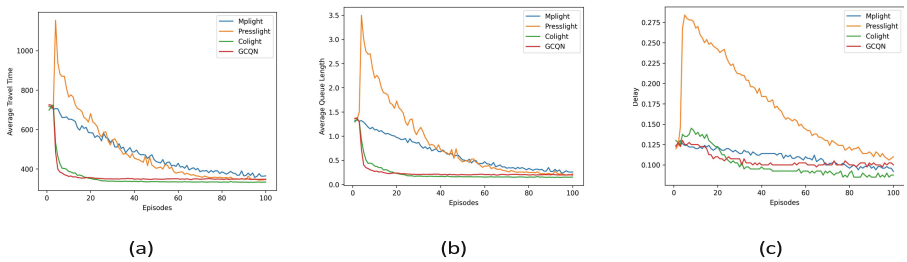

Fig. 5. Average travel time(a), Average queue length(b) and Approximate delay(c) of vehicles over training episodes for 4×4 grid network

5 Conclusion

Designing a traffic management system that adapts to the variation in traffic using traffic light control is a challenging task. Reinforcement learning presents

[2] https://github.com/traffic-signal-control/RL_signals.

an interesting paradigm for solving these problems. We have designed and evaluated different models for adaptive Traffic light control. The DQN model works better for small road networks but is unsuitable once the road networks become large. The GCQN model leverages the graph structure for road networks and shows promising results for use in larger road networks. We also proposed the GCAC model, whose results are comparable to GCQN in larger networks but are not as stable as GCQN. We also assess the performance of GCQN and GCAC models against state-of-the-art counterparts, and observe similar levels of performance. However, due to its moderately complex structure, the GCAC model requires more training iterations to converge. Exploring different techniques in training, GCAC may enhance convergence speed.

We analyzed the performance of traffic light control algorithms on waiting times and queue lengths. There is still a need to come up with strategies to handle scenarios (such as accidents) where a certain road is temporarily not in use. Also, it may not be feasible to keep training RL agents on new data too often. Thus, one needs an optimal way to train and update the model.

We used Graph convolution networks in our traffic signal control solutions to capture the spacial relations between nodes of the road network. One can also try GCN layers followed by LSTM layers, which can be better at capturing temporal relations between subsequent states and actions.

References

1. Aslani, M., Mesgari, M.S., Wiering, M.: Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. Transp. Res. Part C: Emerg. Technol. **85**, 732–752 (2017)
2. Bhatnagar, S., Sutton, R.S., Ghavamzadeh, M., Lee, M.: Natural actor-critic algorithms. Automatica **45**(11), 2471–2482 (2009)
3. Chen, C., et al.: Toward a thousand lights: decentralized deep reinforcement learning for large-scale traffic signal control. In: AAAI Conference on Artificial Intelligence (2020)
4. Gammelli, D., Yang, K., Harrison, J., Rodrigues, F., Pereira, F.C., Pavone, M.: Graph neural network reinforcement learning for autonomous mobility-on-demand systems. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 2996–3003 (2021)
5. Girianna, M., Benekohal, R.F.: Using genetic algorithms to design signal coordination for oversaturated networks. J. Intell. Transp. Syst. - J INTELL TRANSPORT SYST **8**, 117–129 (2004)
6. Guo, Y., Wu, Q., She, H.: A routing optimization policy using graph convolution deep reinforcement learning. In: 2023 IEEE/CIC International Conference on Communications in China (ICCC), pp. 1–6 (2023)
7. Hegeman, T., Iosup, A.: Survey of graph analysis applications. CoRR abs/1807.00382 (2018)
8. Houidi, O., Bakri, S., Zeghlache, D.: Multi-agent graph convolutional reinforcement learning for intelligent load balancing. In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, pp. 1–6 (2022)

9. Kim, G., Sohn, K.: Area-wide traffic signal control based on a deep graph Q-network (DGQN) trained in an asynchronous manner. Appl. Soft Comput. **119**, 108497 (2022)
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR, abs/1609.02907 (2016)
11. Li, C., Ma, X., Xia, L., Zhao, Q., Yang, J.: Fairness control of traffic light via deep reinforcement learning. In: 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), pp. 652–658 (2020)
12. Li, S.: Multi-agent deep deterministic policy gradient for traffic signal control on urban road network. In: 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), pp. 896–900 (2020)
13. Ma, D., Zhou, B., Song, X., Dai, H.: A deep reinforcement learning approach to traffic signal control with temporal traffic pattern mining. IEEE Trans. Intell. Transp. Syst. **23**(8), 11789–11800 (2022)
14. Mei, H., Lei, X., Da, L., Shi, B., Wei, H.: LibSignal: an open library for traffic signal control. Mach. Learn., 1–37 (2023)
15. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning (2016)
16. Mnih, V., et al.: Playing Atari with deep reinforcement learning. CoRR, abs/1312.5602 (2013)
17. Nishi, T., Otaki, K., Hayakawa, K., Yoshimura, T.: Traffic signal control based on reinforcement learning with graph convolutional neural nets. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 877–883 (2018)
18. Oroojlooy, A., Nazari, M., Hajinezhad, D., Silva, J.: AttendLight: universal attention-based reinforcement learning model for traffic signal control (2020)
19. Peng, H., et al.: Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. Inf. Sci. **578**, 401–416 (2021)
20. Prabuchandran, K.J., AN, H.K., Bhatnagar, S.: Multi-agent reinforcement learning for traffic signal control. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 2529–2534 (2014)
21. Prashanth, L.A., Bhatnagar, S.: Reinforcement learning with function approximation for traffic signal control. IEEE Trans. Intell. Transp. Syst. **12**(2), 412–421 (2011)
22. Prashanth, L.A., Bhatnagar, S.: Threshold tuning using stochastic optimization for graded signal control. IEEE Trans. Veh. Technol. **61**(9), 3865–3880 (2012)
23. Sen, S., Head, K.L.: Controlled optimization of phases at an intersection. Transp. Sci. **31**, 5–17 (1997)
24. ul Asar, A., Ullah, M.S., Ahmed, J., ul Hasnain, R.: Traffic responsive signal timing plan generation based on neural network. In: 2008 IEEE International Conference on Automation Science and Engineering, pp. 833–838 (2008)
25. Watkins, C., Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**, 279–292 (1992)
26. Wei, H., et al.: 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, KDD 2019, pp. 1290-1298, New York. Association for Computing Machinery (2019)
27. Wei, H., et al.: CoLight: learning network-level cooperation for traffic signal control. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019. ACM (2019)

28. Xiangyun, Z., Lijun, W., Zhiyuan, L., Yulin, J.: Deep reinforcement learning with graph convolutional networks for load balancing in SDN-based data center networks. In: 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 344–352 (2021)
29. Yang, S., Yang, B., Wong, H.-S., Kang, Z.: Cooperative traffic signal control using multi-step return and off-policy asynchronous advantage actor-critic graph algorithm. Knowl.-Based Syst. **183**, 104855 (2019)
30. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. CoRR, abs/1809.05679 (2018)
31. Yu, X.-H., Recker, W.W.: Stochastic adaptive control model for traffic signal systems. Transp. Res. Part C: Emerg. Technol. **14**(4), 263–282 (2006)