Reinforcement learning-based multi-agent system for network traffic signal control

Multi Agent Systems

Motivation

Can we reduce the congestion, average delay and intersection cross blocking?

Problem Statement

Efficient traffic signal control policy using multi agent system and reinforcement learning.

Solution Techniques

How do we solve the above problem?

Introduction

- A critical traffic challenge is the scheduling and management of multi-intersection networks.
- Conventional traffic management systems fail to schedule large networks of signals in urban settings, due to the lack of a long-term reward policy.
- In a heavy-loaded multi-intersection traffic network, congestion occurring in a single lane impacts other intersections.
- Reinforcement learning efficient scheduling method that controls a dynamic and complex traffic environment is developed.
- We study of a five-intersection, centrally connected traffic network, which can be generalised to large real world networks.

Basic Principles of Reinforcement Learning

- RL is a field of study in machine learning where an agent, by interacting with and receiving feedback from its environment, attempts to learn an optimal action selection policy. RL algorithm used here is Q-Learning Algorithm.
- During each iteration, the agent observes its current environment, from which it infers the environment's state, then executes an action that leads the agent to the subsequent state. Thus it learns long term rewards.
- The goal of the agent is to maximise utility, by learning a good policy which is a mapping from perceived states to actions.

System Model

Basic Notation and Terminology

- **Traffic throughput** is defined as the average number of vehicles per unit of time that successfully traverse the intersection.
- **Traffic congestion** in multi-intersection settings, is a condition in which a link is increasingly occupied by queued vehicles.
- Low traffic throughput and high congestion both lead to an increase in vehicle delay, a fundamental metric in evaluating traffic signal controller performance.

Definition of RL elements

- The state is represented by an eight-dimensional feature vector with each element representing the relative traffic flow at one of the lanes.
- The **relative traffic flow** is defined as the total delay of vehicles in a lane divided by the average delay at all lanes in the intersection.
- For the outbound intersection agent, only local traffic statistics is considered . However, the central intersection agent is assumed to have access to all states of its neighbouring intersections.

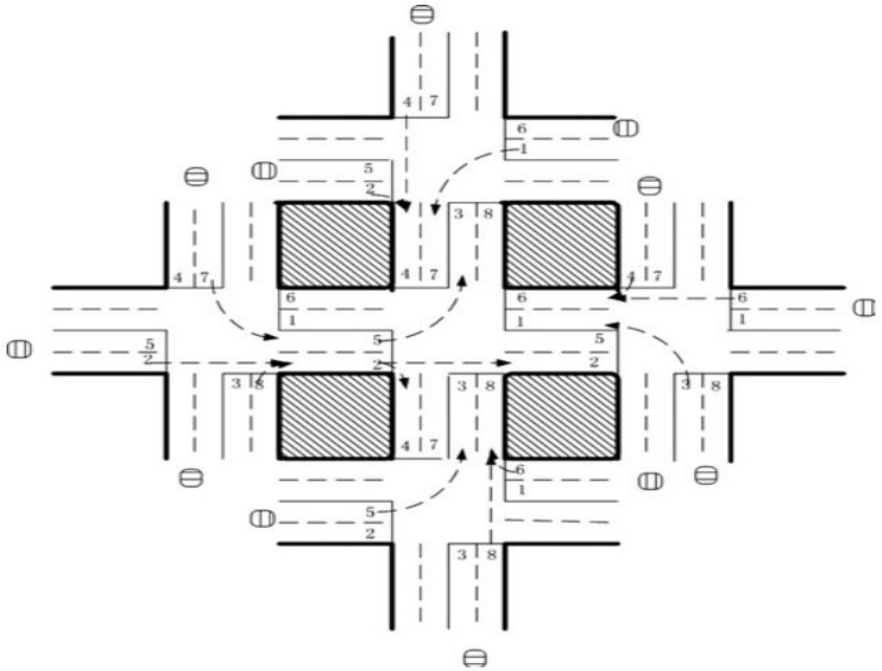


Figure 1 *Five-intersection, centrally connected vehicular traffic network studied*

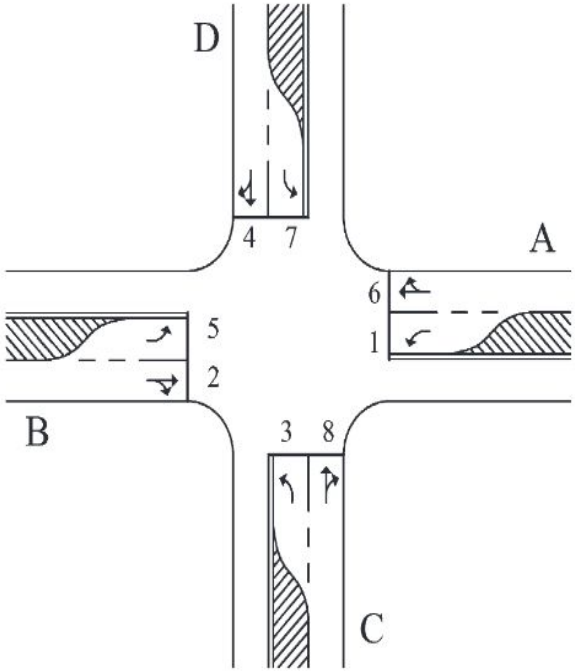


Fig. 1. Intersection model with standard phase numbering.

Intersection network configuration

- This is a five-intersection traffic network, in which the intersection at the middle is referred to as the central intersection. The other four intersections are labelled as outbound intersections.
- An intersection agent can only communicate with its immediate neighbours.
- During each simulation time step, new vehicles are generated, as governed by a Poisson process, outside each outbound intersection.
- Eight-phase combination schemes are available to each intersection.
- Vehicles in lanes which are notated by even numbers can either go straight or turn right. Vehicles at the odd lanes should turn left to their designated queues.

Action Set

- Non-conflicting phase combinations of the traffic network {(1,5), (1,6), (2,5), (2,6), (3,7), (3,8), (4,7), (4,8)}.
- In a multi-intersection network, each intersection agent individually selects an action among the available eight-phase combinations, according to its learned policy.

Reward Function

- The reward ranges from -1 to 1, where positive reward values are received if the current delay is lower than that of the previous time step.
- The agent is subject to a penalty if increased average delay is observed.
- Local Reward is calculated as

$$\mathbf{R = (D_{last} - D_{current})/(max(D_{last}, D_{current})}$$

where D_{last} and $D_{current}$ are the previous and current intersection total delay

Q Learning Algorithm

- The most important and popular model-free RL algorithm, Q Learning , is utilised to quantify the preference and effectiveness of selecting an action given a perceived state.
- However, our problem has infinite state spaces. This problem is addressed using a feed forward neural network, which is used for function approximation.
- Q Learning with function approximation can only be guaranteed to reach a suboptimal policy, and hence in order to improve it we use techniques of data standardization, momentum and learning rate adaptation.

$$\mathbf{Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \max_a(Q(s_{t+1}, a_t)) - Q(s_t, a_t))}$$

- a_t is the action executed while in state s_t leading to the subsequent state s_{t+1}, and yielding a reward r_{t+1}, a is the step-size parameter

Q Learning with function approximation

- Ideally, the estimation of the value function can be represented in a tabular form, for which an optimal policy can be obtained.
- Most real-world problems have large or infinite state spaces, with the problem at hand being no different. This can be addressed by utilising function approximation techniques. Commonly used function approximation methods included neural networks, cerebellar model articulator controllers
- The neural network used has 40 input nodes (representing the 5 state vectors), 25 hidden units and an output corresponding to each of the actions

Example

Algorithm Explanation

1. Here we have five intersection network and each intersection is represented by a 8*1 feature vector with each element representing the relative traffic flow at one of the lanes. Here we are getting 5x8 state matrix **s** from our simulation where **s[i,j]** denotes the relative traffic flow for a particular lane j of intersection i. Convert this 5x8 matrix into a 40x1 vector.

$$Relative\ traffic\ flow = \frac{Total\ delay\ of\ vehicles\ in\ a\ lane}{Average\ delay\ at\ all\ lanes}$$

Example

S = [-0.23225134689012306, -0.7857888341031736, 0.5704122025575415, -0.20170515401336386, -0.05309346581451146 5, -0.09513716966684016, 0.8517350972148592, -0.6529297648288199, -0.9213424919605675, 0.6836177234691716, -0.8186355484654066, -0.5836210319849646, -0.6378996516364446, -0.6288608525332069, 0.7840168437614357, 0.45449726091676856, -0.40751187825639446, 0.24783955654962053, 0.8332913423084365, 0.6593922290867686, 0.0036939679826024 64, -0.72978496102366, -0.888758611229286, -0.17200641237258418, 0.4570661847417754, 0.15658837743413323, -0.20890972145579867, -0.47244397870832766, -0.514271283769838, -0.46556131857807115, 0.7499217147039428, 0.16794223332040703, 0.2059927230884584, 0.8703837518153272, -0.5767750886959069, 0.2074951087206791, 0.6066602392560232, -0.45554491315677637, -0.8066466387695628, -0.7717258899969792]

Here we have 40x1 vector from our simulation.

2- Now we are applying feed forward operation on this state vector using our neural net.

Example Continuation(1)

3- Now we 8 non-conflicting phase combinations {(1,5), (1,6), (2,5), (2,6), (3,7), (3,8), (4,7),(4,8)}. We will get 8 output values, one for each action.

O = [0.5406596946626945, 0.45449041919773825, 0.4078099450429432, 0.2668070706989648, 0.463221008269231 74, 0.20372377003141684, 0.9439334305686512, 0.7474640395629771]

O is 8x1 vector.

4- Now we are selecting the action with the highest value. Here it is 7th action which corresponds to phase of (4,7). Now in our simulation, change the signal settings of our central intersection so that it now shows green lights for lanes 4 and 7. Store the reward ***r*** which we obtain. This reward is actually an indication of whether the delay at the central intersection has increased or not and ranges from -1 to +1. Also store the new state vector $\underline{s}^{1}$ which denotes the state of the world after taking action (4,7)

Example Continuation(2)

$\mathbf{S^1}$ = [-0.5236804666362231, -0.8197498730217307, 0.23660520269419338, -0.9866270078848498, -0.1391265713292522, 0.32134321610363314, 0.6289511230922669, -0.4754221613474525, -0.4710164751041226, -0.275737613075904l6, 0.15174848362526494, -0.8215486452471059, 0.4487589737586277, 0.3899301657839398, 0.8539895005941471, -0.8747798609952497, 0.9485896740060946, -0.4150050274016965 4, 0.42028279808858726, 0.37082149937539377, -0.8138663609848511, -0.6167897102548061, 0.3454255425582071, 0.7127944176953833, -0.8496320684609477, 0.948210855179116, -0.0942042744376097, 0.7695099788201598, -0.1396068535221664, 0.8401881894983154, -0.4750170739731552, 0.17540974711509394, -0.02109128619673384, -0.9601839569935531, 0.18846538267360047, -0.18080748971793992, -0.22952316463592126, 0.08221081893957805, -0.7066830448549624, -0.19949448006426 73]

S^1 is 40x1 vector

With the help of this reward function we can calculate our reward function

Dlast and Dcurrent are the previous and current intersection total delay

so in our case reward will be

$$r = \frac{D_{last} - D_{current}}{\max[D_{last}, D_{current}]}$$

r = 0.65

Example Continuation(3)

5- Apply a feed forward operation on this new state vector s^1 , here again we will get 8 output values. Store the greatest output value in maxQ.

$\mathbf{O^1}$ = [0.9189444606483333, 0.24772736877294332, 0.8520382387895318, 0.8792635516960547, 0.6611361496099144, 0.32275163130127793, 0.9442878211494131, 0.524553302622215]

maxQ = 0.9442878211494131

6- Our target value to train the network is (reward + gamma * maxQ)

Target = reward + maxQ = 0.65 + 0.5*0.9442878211494131 = 1.12214391057

Example Continuation(4)

7- Given that we have 8 outputs and we only want to update/train the output associated with the action we just took, our target output vector is the same as the output vector from the first run, except we change the one output associated with our action to: reward + (gamma * maxQ).

In our case, we replace the 7th value in our original O vector with target value.
Target = reward + maxQ = 0.65 + 0.5*0.94428782114941 31 = 1.12214391057
Originally
O = [0.5406596946626945, 0.45449041919773825, 0.4078099450429432, 0.2668070706989648, 0.463221008269231 74, 0.20372377003141684, 0.9439334305686512, 0.7474640395629771]

After replacing 7th value with target we get,

O^2 = [0.5406596946626945, 0.45449041919773825, 0.4078099450429432, 0.2668070706989648, 0.463221008269231 74, 0.20372377003141684, 1.12214391057, 0.7474640395629771]

Example Continuation(5)

8- Train the neural net by applying a backprop operation when the output is our modified O vector and state is the initial state.

Apply backprop when

S = [-0.23225134689012306, -0.7857888341031736, 0.5704122025575415, -0.20170515401336386, -0.05309346581451146 5, -0.09513716966684016, 0.8517350972148592, -0.6529297648288199, -0.9213424919605675, 0.6836177234691716, -0.8186355484654066, -0.5836210319849646, -0.6378996516364446, -0.6288608525332069, 0.7840168437614357, 0.45449726091676856, -0.40751187825639446, 0.24783955654962053, 0.8332913423084365, 0.6593922290867686, 0.0036939679826024 64, -0.72978496102366, -0.888758611229286, -0.17200641237258418, 0.4570661847417754, 0.15658837743413323, -0.20890972145579867, -0.47244397870832766, -0.514271283769838, -0.4655613185780711 5, 0.7499217147039428, 0.16794223332040703, 0.2059927230884584, 0.8703837518153272, -0.5767750886959069, 0.2074951087206791, 0.6066602392560232, -0.455544913156776 37, -0.8066466387695628, -0.77172588999697 92]

And

O^{2} = [0.5406596946626945, 0.45449041919773825, 0.4078099450429432, 0.2668070706989648, 0.46322100826923174, 0.20372377003141684, 1.12214391057, 0.7474640395629771]

Convergence for Q Learning by Function Approximation

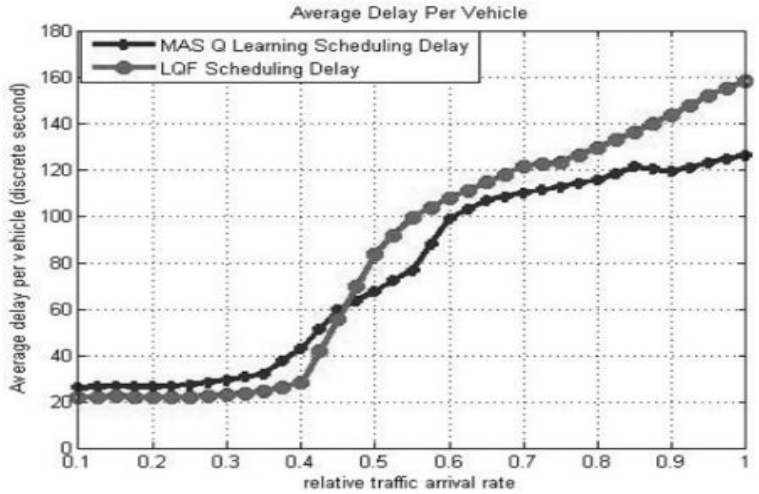
Although Q Learning in a tabular form has been proved to converge deterministically to an optimal policy, Q Learning with function approximation can only guaranteed to reach a suboptimal policy because of the limitations of the function approximation module.

We can improve the overall function of Neural networks in context of value function approximations, with the following techniques

- Data standardisation
- Activation function
- Learning rate adaptation
- Momentum
- Weight decay

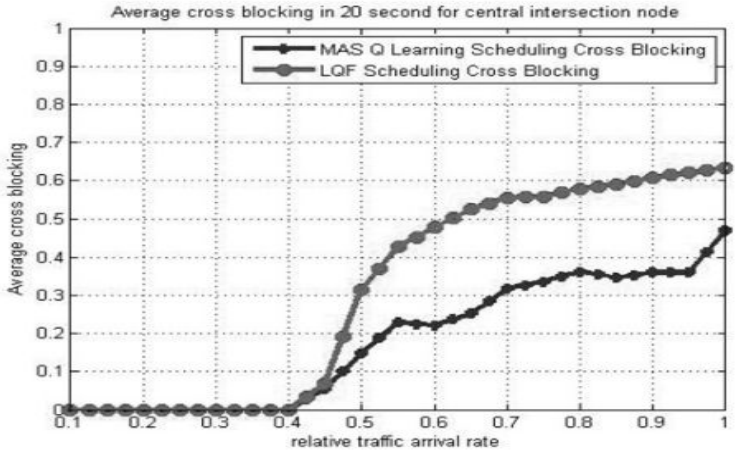
Simulation Results

Average delay for five intersection vehicles.



Average cross-blocking in 20

time units for the central intersection in the network



Project Plan

- The ultimate goal is to maximise traffic throughput, avoid traffic congestion and intersection cross-blocking and reduce overall vehicle delay via finding an optimum policy for scheduling the traffic lights in a dynamic manner.
- Implement Q-Learning with Neural Nets using Keras.
- Implementing a simulation for five intersection traffic network in python using Simpy.
- Plot graphs for average delay and cross blocking at five intersection traffic network using the simulation and compare the values for different situations.

References

- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.232.9789&rep=rep1&type=pdf
- http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4584232&tag=1
- http://outlace.com/Reinforcement-Learning-Part-3/

Questions?

Thank you

Akshat Tandon
Aman Varshney
Sreeja Kamishetty