Traffic Control using Distributed Reinforcement Learning

Vasanth Reddy Baddam
vasanth2608@vt.edu
Virginia Tech

ABSTRACT

Traffic congestion has been one of the major problems in urbanisation. The method to control traffic light signals has been outdated which is making drivers sit idle at intersections endlessly. To deal with this problem we need to implement the method intelligently. Many heuristics and Deep learning models have been applied to this problem. Though there has been an improvement over the traditional models, these models fail to work when deployed in a real-world scenario. Then researchers started to explore reinforcement learning methods and showed promising results. Still, the models haven't properly established a communication channel between the intersection to achieve the global optimum. In this paper, we set out to create a parameter sharing option between intersections and thus enabling communication between the junctions. This method showed promising results and overcame the baseline reinforcement, learning models.

KEYWORDS

Reinforcement learning, Decentralized architecture, Distributed reinforcement learning, Traffic light signal

.
, 9 .

1 INTRODUCTION

In recent years, traffic congestion has been increased exponentially. The main reason is the increase in vehicles. The number of vehicles per household increased. This increased in-vehicle travel resulting from the growing population alone. With the increased number of vehicle on the roads with the same old traffic rules is the main reason for the traffic congestion. The automobiles need to wait on the lanes waiting for their turn to go at the signals burns more fuel and releases more amount of gases into the atmosphere. The traffic congestion waste fuel and releases harmful gases which harm human health. As indicated by existing investigations, the vehicle area adds to 23% of all-out CO2 outflow from fuel ignition, and street traffic makes up around three-fourths of them[5]. Along with the release of harmful gases into the atmosphere, traffic congestion spoils the plans, and delay the meetings of the people for being stuck in the traffic. It is very crucial to improve traffic conditions to cut the fuel waste, increase the efficiency and ease the people's life.

There are many methods proposed to tackle traffic congestion such as freeway control, critical intersection control and ride-sharing but the main component in controlling the vehicles is the traffic light. The traffic light timing is set in such a way that it makes vehicles to wait at the intersections pointlessly. The traffic light control arrangement can be significantly improved by actualizing machine learning ideas. This paper centres around actualizing a learning calculation that will permit traffic control gadgets to consider traffic designs for a given convergence and improve traffic stream by changing stoplight timing. In this paper, we set out to build up a practical RL-based traffic signal control technique to empower traffic signal control for multi intersections. The technique would take the traffic condition as info and figure out how to choose for each crossing point about their next stage.

2 MOTIVATION

The traditional methods we currently deploy in the real-time world are:

2.1 Fixed time control

In this method[12][15], the fixed time is allotted for each signal according to the traffic history of that particular intersection. This method cannot be adaptive as the traffic flow changes from time to time. Likewise, the limit drop phenomenon brought about by the postponed vehicles in the arranging region isn't thought of, which subverts the functional execution of sign control. Under the fixed-time control technique, the signal controller can't get the real-time data of traffic stream and subsequently, in this manner can't progressively identify the traffic conditions inside the arranging territory just as the limit drop. Likewise, the limit drop marvel brought about by the postponed vehicles in the arranging region isn't thought of, which subverts the functional execution of sign control. Under the fixed-time control technique, the signal controller can't get the continuous data of traffic stream and subsequently, in this manner can't progressively identify the traffic conditions inside the arranging territory just as the limit drop.

2.2 Queuing process

The queuing process[2] inside the sorting zone is basic to the streamlining of signal timing at the convergence. As indicated by field perceptions, the path determination conduct of left-turn or through vehicles is influenced by the quantity of existing lining vehicles in every path and their separation as far as possible of each line. Therefore, it's problematic for the current examinations to expect that vehicles are equally conveyed over every path, which is conflicting with the handy experience and may prompt an imperfect signal plan.

2.3 Vehicle actuated control

This section of methods[4][13][9] uses real-time traffic scenario into calculation but can be only used on high traffic randomness

situations. Some of the algorithms involve control using Genetic Algorithm and efficient tree search algorithm which takes their objective as minimizing or maximizing the green signal phase. Be that as it may, these techniques generally rely upon the handcraft rules for current traffic condition, without considering the future circumstance. Along these lines, they won't reach the global optimum.

As the previously mentioned methods are incapable of dealing with multi and dynamic intersections, researchers sought to more intelligent methods. Work is carried in the areas of machine learning and deep learning methods such as CNN. More works try to use Reinforcement learning methods to control traffic congestion which gave better results compared to intelligent methods and outperformed the traditional methods. We set out to develop RL method in controlling the traffic congestion.

3 CHALLENGES

There are few challenges in deploying algorithms onto the traffic congestion problem. They are:

- Scalability: The proposed method should be able to handle a large scale traffic network scenario. For example, a city with thousands of intersection should be controlled by the algorithm and should reach a global optimisation goal.
- Co-ordination: The proposed model should be ready to accomplish coordination with the goal that the global traffic conditions can be optimised. In urban situations, optimising sign timings for traffic signals must be done mutually as signs are regularly right upfront closeness, which is generally known as coordinating signal timings. Inability to do so can prompt choices made at one signal falling apart traffic tasks at the other. In a city with thousands of intersections, each intersection needs to co-ordinate with its neighbour junction.
- Data: The data we input to the model should be available to us in the real-time scenario. The proposed model should cause no problem when we deploy in real-time. The data format we use at the time of simulation should not be hard to acquire at the time of executing in real-world

Our proposed method should be able to satisfy the above three challenges and achieve global optima.

4 RELATED WORK

With the superior performance of RL-based single intersection methods over conventional transportation methods efforts have been put into developing RLbased multi-intersection methods. Some proposed methods[8][11][16]following centralized optimisation where a single global agent controls all the junction can share the parameters between the junction but can scale up to more number of intersections. These kinds of methods fail in satisfying the first challenge. Secondly, the methods resort to decentralized optimisation where each junction behaves on its own can easily scale up but it is hard to create a path where intersection can communicate with each other. So it's hard for such methods to achieve global optima even when each intersection achieve their maximum reward. These methods have difficulty in satisfying the second challenge. Thirdly, some RL strategies accept the itemized traffic condition can be effectively gotten to and utilize entangled highlights to speak to the traffic condition, which is unreasonable for real-time deployment. For instance, aeronautical view about the crossing point is utilized in[6], while it is difficult to get elevated pictures for each convergence in real-time, which prevents them from being sent in reality application. These methods don't satisfy the third challenge.

Commonly, this methods[1][**?**][3] take the traffic on the road as a state and the procedure on light as an action. These techniques, for the most part, show better execution contrasted and fixed-time and traffic-responsive control strategies.

Techniques[10][14] planned the state as discrete qualities like the area of vehicles or number of held up vehicles. In any case, the discrete state-action pair value matrix requires a huge storage space, which shields these strategies from being utilized in huge state space issues. However, in our implementation, we use continuous input state for each time step as an encoder of information of vehicles present in lanes. This information is extracted from the simulation.

To accomplish coordination, an elective path is through brought together enhancement over multiple coordinated agents in a territory to guarantee the optimality. Be that as it may, as the system scale extends, the brought together improvement is infeasible because of the combinatorially huge joint action space, which has restrained boundless selection of this strategy to a city with thousands of intersections

To focus these problems, we set out to develop a centralized training and decentralized execution method taking out the advantage of both centralized and decentralised optimisation. In our method, we will enable the global agent to control junctions during training and cut itself from the network during execution. This method enables the parameter sharing between the junctions thus creating coordination between intersections. Additionally, we set out to use the data that is feasible to obtain in the real world scenario. With this implementation, we are satisfying the three challenges posed by the problem.

5 NOTATIONS

5.1 Traffic movements

Traffic movement is defined as the movement of traffic exiting from one lane and entering into another lane. For example, vehicles exiting from the left lane of S to the left lane of W. A simple intersection is given in fig. 1

5.2 Signal Phase

A signal phase is defined as the set of permissible traffic movements. For simplicity purpose, in our paper, we are considering only two phases. One phase is when the horizontal traffic movement is set and another phase when the vertical traffic movement is set. Signal phase set can be seen in fig. 2. There will be a little time gap when one phase transits into another phase. This signal phase is

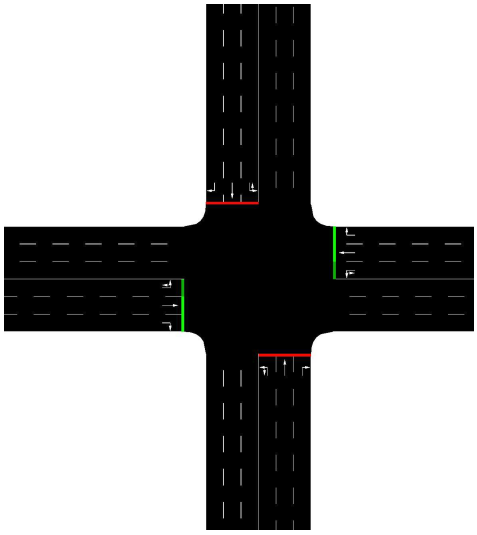

Figure 1: Intersection

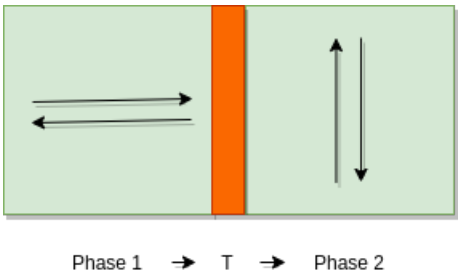

Figure 2: Signal Phases

represented by $L^t = \begin{bmatrix} 0 & 1 \end{bmatrix}$, When the green signal is given for the vertical phase and vice-versa for the horizontal phase.

6 STATE CONFIGURATION

The input state at a time t to the model is a concatenated vector of the position(P^t) of vehicles on the lane of all roads connected to the junction, normalized velocity(V^t) of all vehicles and the current phase(L^t) of the junction. In the given snap at a time t in fig 3, we will see the state of the observation. we will focus on the left road of the intersection.

6.1 Position

There are six lanes on each side of the road. Three lanes entering the intersection and other three exiting the intersection. we will encode each lane of the road. we will take the value 1 for each vehicle on the lane and 0 zero if there is no vehicle. we will take 12 cells from the junction. Thus we will obtain matrix of dimension 6×12. The position matrix for the fig 3 is given as in table 1: Position(P_{l_1})

0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	1	1
0	0	0	0	0	1	0	0	1	1	1	1
0	0	0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1

Table 1: Position matrix

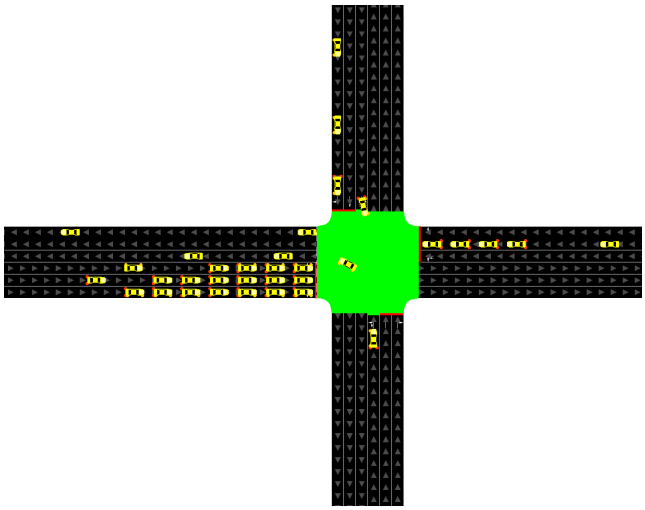

Figure 3: Simulation snapshot at a time t

0	0	0	0	0	0.1	0.1	0	0	0	0	0
0	0	0	0.8	0	0	0.1	0	0	0	0	0
0	0	0	0	0	0.85	0	0	0.1	0	0	0
0	0	0	0	0	0	0	0.95	0	0	0.9	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.98	0	0	0	0	0	0	0	0	0.8

Table 2: Velocity matrix

6.2 Velocity

Similarly, we will obtain normalised velocity of each vehicle present on the lane.

$$V_N = \frac{Velocity}{Maximumvelocityofthatlane}$$

The velocity matrix for the fig 3 is given as in table 2: Velocity(P_{l_1})

So for every edge we generate Position, Velocity and Signal phase(from section 5.2). These three together gives us the state/observation of each intersection. we need to concatenate there three matrices to get the state input to the model. The state at a time t is given by s^t

$$s^t = \begin{bmatrix} P^t & V^t & L^t \end{bmatrix}$$

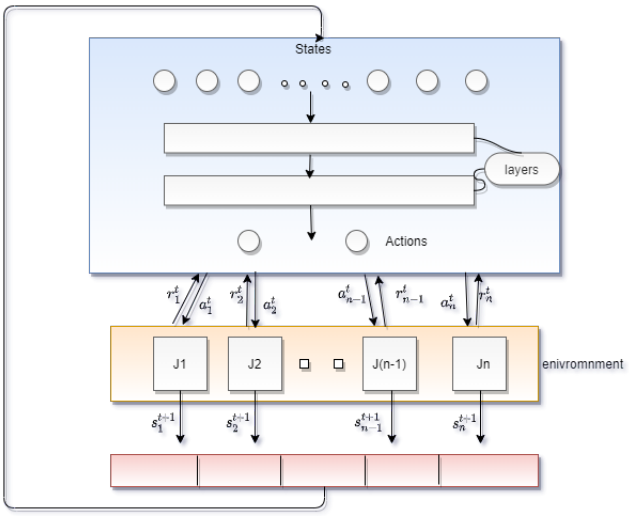

Figure 4: Model Framework

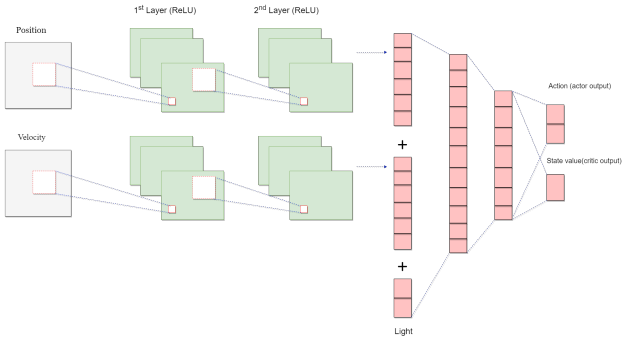

Figure 5: Network Architecture

7 PROBLEM

Each intersection point is constrained by a single agent. At time step t, each agent i sees the position, velocity and signal phase of its environment as its observation state s_i^t. Given the traffic circumstance and current traffic signal phase, the objective of the agent is to make an action a_t (i.e., which phase to set), with the goal that the total reward $r_t$can be maximized and move to the next observation state s_i^{t+1}. The agent tuple is given by $< s_t, a_t, r_t, s_{t+1} >$. Model framework is given in the fig. 4

8 MODEL NETWORK

This is the general network architecture of the algorithm. The state will be input to the neural network and outputs the state-action value for DQN, Policy distribution and State value for the actor-critic method. These algorithms are discussed in the next section. The network is given in the fig 5 The input for this network is the state of a junction $s^t = \begin{bmatrix} P^t & V^t & L^t \end{bmatrix}$ The input and layer dimensions is given as below:

- Input:
  - Position: (1×24×24)
  - Velocity: (1×24×24)
  - Light: (1×2)
- CNN Layers:
  - 1^{st} layer: 16 filters of 4×4
  - 2^{nd} layer: 32 filters of 2×2

  Flattening the CNN outputs and appending the three vectors into one and passing it into dense layers
- Dense layers
  - 3^{rd} layer: size 128
  - 4^{th} layer: size 64

  Output layers depends on the algorithm we are going to use. In this paper we set out to use both Deep Q Networks and Advantage actor-critic methods.
- Output layers
  - DQN: State-action value (1×1)
  - A2C: policy distribution/action(1×2) and State value (1×1)

9 AGENT

As each intersection is given operated by a single agent and it is given by the tuple $< s_t, a_t, r_t, s_{t+1} >$. we introduce state, action and reward representation below.

- **Observation/State:** Each agent observes the junction of the environment as its state. For a standard intersection with 4 traffic movements, its observation includes current position, the velocity of vehicles in all the lanes in a given junction and Signal phase of the junction. The observation is discussed in section 6
- **Action:** At time t, each agent chooses a phase p as its action a_t, indicating the traffic signal should be set to phase p. As there are two different signal phases, each agent has to choose one of the phases as its action a_t at each time step. Signal phase is clearly given in section 5.2
- **Reward:** Reward design[7] is one of the main factors in improving the efficiency of the problem. In this problem, we need to define reward function such that it enables the co-ordination between the junctions. A reward is defined in such a way that agent is penalised if the total waiting time of all the vehicles increases for each time step or each phase change and an agent is rewarded if waiting time decreases. we take two observation for each time step. When before the time step begins and another when time step ends. The reward is given by

$$R = r_1 - r_2$$

  where r_1 is the total waiting time of all vehicles at the beginning of the time step and r_2 is the total waiting time of all vehicles at the end of the time step. In such a way that an agent receives a negative reward if its action increases by waiting time and positive reward if waiting time decreases.

10 DQN ALGORITHM

In a single agent, RL setting, an agent fully observes the state at a time t as s_t and takes an action a_t according to the policy π results in obtaining a reward r_t and transits into the state s_{t+1}. Its objective is to maximize an expectation over its discounted reward, $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$, where r_t is the reward received at a

Parameter	Value
Replay memory size	50000
Batch size	32
Epsilon (ϵ)	0.1
Discount factor (γ)	0.99
Gradient momentum	0.95
Squared gradient momentum	0.95

Table 3: Parameter setting for DQN agent

time t and γ is the discount factor. The Q function for a policy π is given as

$$Q^{\pi} = \mathbb{E}[R_t | s_t = s, a_t = a]$$

The optimal action-value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ obeys the Bellman optimality equation and is given by

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a') | s,a]$$

Deep Q learning uses neural networks as function approximator and approximate state-action value and the network is parameterised by θ and the Q function is given as $Q(s,a;\theta)$. Loss function of DQN is given by

$$\mathcal{L}_{\rangle}(\theta_i) = \mathbb{E}_{s,a,r,s'}[r + \gamma \max_{a'} Q^*(s',a';\theta_i^-) - Q(s,a;\theta_i)]$$

where θ_i^- is the parameters of a target network which is frozen for the number of iterations before it gets updated so that network is stable. DQN also uses experience replay: during learning, the agent builds a dataset of episodic experiences and is then trained by sampling mini-batches of experiences and DQN uses ϵ-greedy policy in which it chooses actions randomly with probably of ϵ. The parameter setting for DQN is given in table 3

11 ADVANTAGE ACTOR-CRITIC ALGORITHM

In actor-critic methods where;

- Critic estimates the value function. It can be either state value (V) or state-action value (Q)
- Actor updated the policy distribution as suggested by the Critic.

Thus, actor and critic work together to maximize the reward. Both the actor and critic learners are parameterized by neural networks. Instead of taking two different neural networks for both actor and critic learners, it is advisable to take a single neural network. correct network is shown in figure **??**. The advantages of using a single network:

- Less number of parameters are involved and can be easy to train
- It is such way that parameter sharing is happening between both the learners as they have similar low-level weights and different high-level weights.

The update for the neural network is given by

$$\nabla_{\theta} J(\theta) \approx \nabla log \pi_{\theta}(a|s) * \hat{A}(s,a)$$

where the $\hat{A}(s,a) = r_{t+1} + \gamma \hat{V}_{\Phi}(s_{t+1}) - \hat{V}_{\Phi}(s_t)$update is given by:

$$d\theta = d\theta + \nabla_{\theta} J(\theta)$$

parameter updates is given by:

$$\theta = \theta + \alpha d\theta$$

where α is the step size parameter.

12 MODEL ARCHITECTURE

In this paper, we set out to develop centralised training and decentralised execution architecture. As a baseline, we implement Decentralised architecture too and compare this both architectures using DQN and A2C algorithms.

12.1 Decentralized Learning architecture

In Decentralized optimisation, each junction acts on its own without having any interaction or communication between its neighbouring junctions. Their main objective is to maximize the cumulative reward. The only such factor that can enable coordination is the reward design. we designed the reward in a way that it affects its neighbouring intersections.

- No parameter is shared between the junctions
- Every agent at each intersection has its network and have to train each network
- More computation power is needed to train each network as more number of parameters updates is needed here

As shown in the figure 6, there are three junctions take input as state s^i as part of the the environment and take an action a_t^i which gives it maximum reward r_t^i and transits into next next state s_{t+1}^i. For each agent, all the states, actions, reward and next states are stored in a memory buffer and then get trained after enough batch is reached. for each time step, the reward stored for each agent in memory buffer is the cumulative reward, i.e given by $R = r_1 + r_2 + r3$ for the above given three agents. Generally R is given by

$$R = \sum_{i=1}^{n} r_i$$

where n is the number of intersection in an road network. Thus by storing cumulative reward instead of individual reward, we enable the communication between each intersection. This modification is done on the thought that waiting time of traffic at one junction iterates at all junctions throughout the network like cascading effect.

12.2 Distribution Learning Architecture

There are a few drawbacks in Decentralized architecture, such as no parameter sharing, difficult in creating a global reward function and more number of networks to train. In Distributed RL we will focus on training a global agent and push all the updated parameters to all the intersections and while executing, remove the global agent and act DRL as a decentralised network.

- Every junction shares the same set of parameters for its network and they all act in the same way but the only difference is that they get a different observation. Thus, parameter sharing is enabled in this architecture
- Only a global agent has to be trained each time, thus saving the computational power and space.

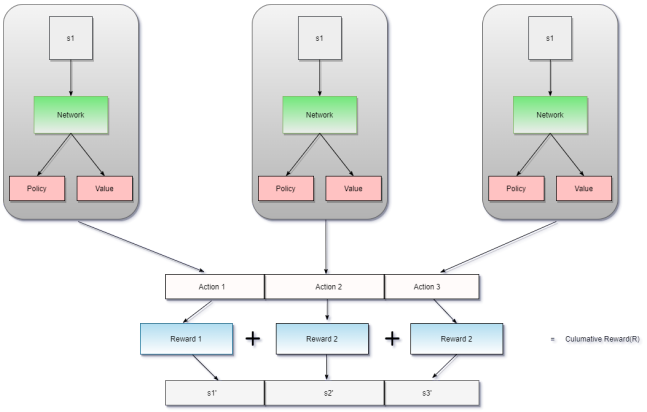

Figure 6: Decentralized architecture

As shown in the figure 7, there is a global agent controlling three junctions. At a time step t, each junction draws its set of observation from the shared environment and take an action a_t^i which gives its maximum reward r_t^i and transits into next state s_{t+1}^i. Then these junctions pushes all their individual gradients ∇_i into the global agent and global agent aggregates all the gradients. After a few iterations as soon as the memory buffer is full, global agent updated the parameters and each network pulls the updated parameters. The update is given as

$$\theta^{k+1} = \theta^k - \alpha \nabla_T^k$$

$$\nabla_T^k = \sum_{i=1}^{n} \delta \nabla_i$$

where, α is the step size parameter and ∇_T is the total aggregated gradient from all the junctions.

13 EXPERIMENT

We conduct our experiments on SUMO simulator. After the traffic information being taken care of into the test system, a vehicle moves towards its goal as indicated by the setting of the environment. we also implemented SUMO TraCi which allows the control of dynamic traffic at runtime. Running SUMO simulator as a server, it provides the position, velocity and signal phase of the intersection as we need for input to the algorithm as a state and also gives us an option to modify the traffic signal timings while in simulation.

This vehicle has its inward course which isn't imparted to different vehicles. It is additionally conceivable to characterize two vehicles utilizing a similar course. For this situation, the course should be "externalized" - characterized before being referenced by the vehicles. Likewise, the course should be named by giving it an ID. The vehicles utilizing the course allude it utilizing the "course"-characteristic. In traffic flow dataset, each vehicle has its ID, depart time and route edges it is travelling.

13.1 Datasets

Synthetic dataset, which focuses on bi-directional and dynamical flows with turning traffic, is used in our experiments. we use 2×2 in fig 8 and 4×4 intersection to carry out our experiment. Each intersection is a four-way connected road. We use a different set of traffic flows on our road network which co-relates to the real-time traffic flow. Different configurations is given in table 4

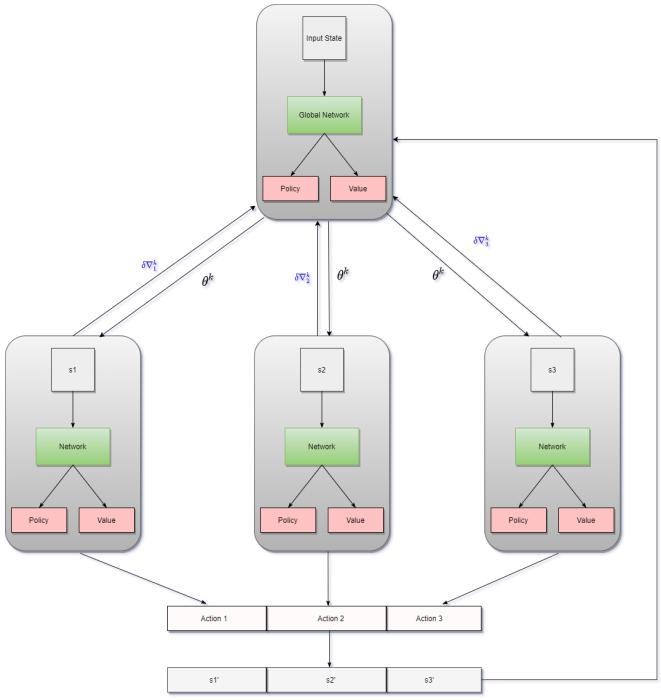

Figure 7: Distributed Architecture

Figure 8: 2×2 Network

Config	Start time	End time	Arrival rate(vehicles/sec)
1	0	50	0.38
2	0	50	0.45
3	51	110	0.38
4	51	110	0.45

Table 4: Network configuration

13.2 Metrics

we evaluate each method using the following metrics:

- **Average Reward**: Average reward over the number of episodes for each time the simulation runs. Average reward is given

by the equation in section 12.1. The method of having a high reward will be the most efficient.
- **Total waiting for time**: The total waiting time is the time waiting by all the vehicles at the intersections. It is the most used metric for traffic light control problem. The method with less travelling time would be best.

13.3 Notation

Results are given in table 5, table 6 and the plots are shown in fig 10, 11, 12 and 13. The notations for the models in the table is given as the below

- DQN - DeRL: Deep Q Network with Decentralized Architecture
- A2C - DeRL: Actor-Critic with Decentralized Architecture
- A2C - DRL: Actor-Critic with Distributed RL Architecture

14 PERFORMANCE

From the results, we see that Distribution RL is outperforming the Decentralized framework. This is mainly because of

- Enabling the parameter sharing in between the junctions/agents. As we can see that in decentralized architecture, no parameter is being shared so the performance is lower in A2C with and without the parameter sharing
- A2C-DRL is converging to a stable point where the normal decentralized RL is still oscillating to get converged.

15 CONCLUSION

In this paper, we propose a Distributed reinforcement learning which deals with Centralized training and Decentralised execution for traffic light control. Here, we achieved to improve performance when compared to the normal baseline methods. This model can be implemented to n number of traffic intersections.

We also acknowledge the limitations of our current approach. we didn't exploit more number of features that can feed into the model. We used only a limited number of features as an observation. Moreover, there is a room for defining the reward function which can improve the performance for decentralized architecture.

REFERENCES

[1] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. 2013. Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence* 26, 5-6 (2013), 1575–1587.

[2] Ghassan Abu-Lebdeh and Rahim F Benekohal. 1997. Development of traffic control and queue management procedures for oversaturated arterials. *Transportation Research Record* 1603, 1 (1997), 119–127.

[3] Bram Bakker, Shimon Whiteson, Leon Kester, and Frans CA Groen. 2010. Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information Systems*. Springer, 475–510.

[4] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. 2013. Self-organizing traffic lights: A realistic simulation. In *Advances in applied self-organizing systems*. Springer, 45–55.

[5] Anthony Downs. 2000. *Stuck in traffic: Coping with peak-hour traffic congestion*. Brookings Institution Press.

[6] Samah El-Tantawy and Baher Abdulhai. 2012. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC). In *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 319–326.

[7] Juntao Gao, Yulong Shen, Jia Liu, Minoru Ito, and Norio Shiratori. 2017. Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. *arXiv preprint arXiv:1705.02755* (2017).

[8] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 656–671.

[9] Jinwoo Lee, Baher Abdulhai, Amer Shalaby, and Eui-Hwan Chung. 2005. Real-time optimization for adaptive traffic signal control using genetic algorithms. *Journal of Intelligent Transportation Systems* 9, 3 (2005), 111–122.

[10] Li Li, Yisheng Lv, and Fei-Yue Wang. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 3, 3 (2016), 247–254.

[11] Patrick Mannion, Jim Duggan, and Enda Howley. 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic road transport support systems*. Springer, 47–66.

[12] Alan J Miller. 1963. Settings for fixed-cycle traffic signals. *Journal of the Operational Research Society* 14, 4 (1963), 373–386.

[13] Isaac Porche and Stéphane Lafortune. 1999. Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation System* 4, 3-4 (1999), 209–254.

[14] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)* (2016).

[15] Fo Vo Webster. 1958. *Traffic signal settings*. Technical Report.

[16] MA Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 1151–1158.

Table 5: Simulation results of 2×2 road network for travelling time

Model	Travelling time			
	Config 1	Config 2	Config 3	Config 4
DQN - DeRL	3077	2846	3985	3020
A2C - DeRL	2911	2365	3651	2915
A2C - DRL	2721	1948	3417	2825

Table 6: Simulation results of 2×2 road network for reward

Model	Reward			
	Config 1	Config 2	Config 3	Config 4
DQN - DeRL	10.76	19.25	16.75	13
A2C - DeRL	8.94	11.54	12.42	10.43
A2C - DRL	6.92	5.88	3.31	8.40

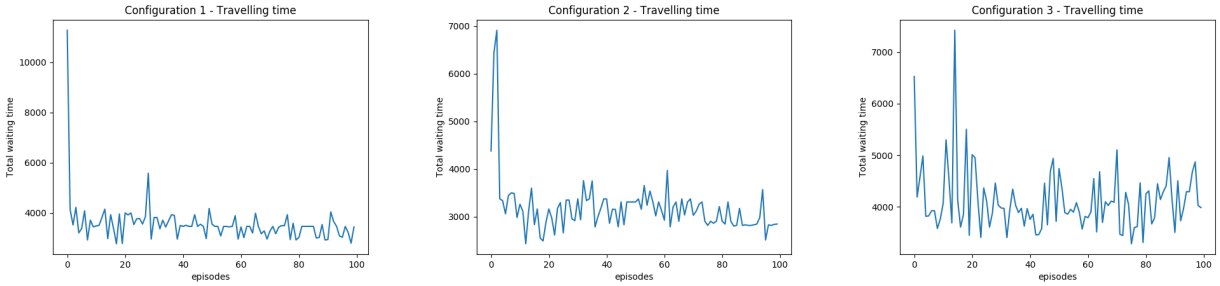

Figure 9: Total waiting of all the vehicles through out the simulation for 100 episodes for DQN-DeRL

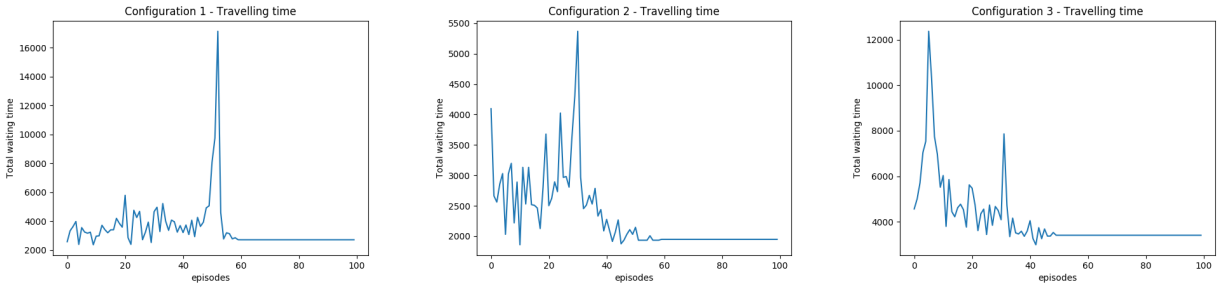

Figure 10: Total waiting of vehicles through out the simulation for 100 episodes for A2C-DRL

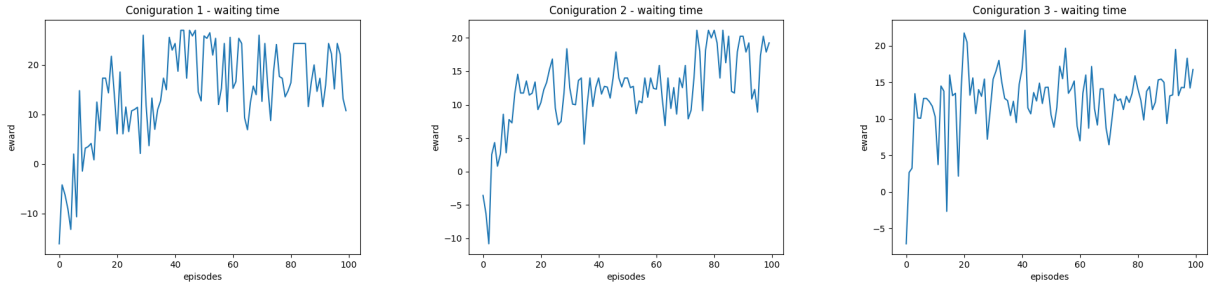

Figure 11: Average Rewards of intersections for each episode through out the simulation for 100 episodes for DQN-DeRL

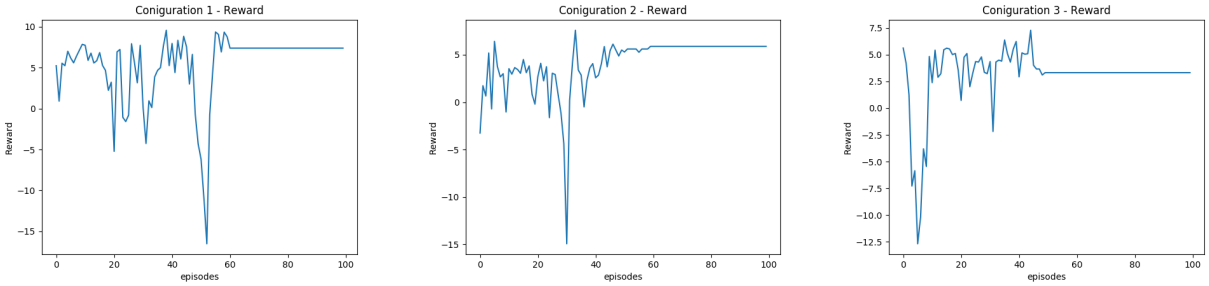

Figure 12: Average Rewards of intersections for each episode through out the simulation for 100 episodes for A2C-DRL