Adaptive Traffic Signal Control using Multi Agent Reinforcement Learning

Sonu Dixit (SR-14432)
M.Tech (Systems Engineering), Final Project
Advisor - Prof. Shalabh Bhatnagar
Department - Computer Science and Automation

Abstract— **Travel is an integral part of everyone's daily work. Travel time and experience is very much dependent on traffic conditions. Traffic junctions have a critical role to play in shaping a city's traffic conditions. Most of the traffic junctions use fixed time round robin signal timing. This kind of strategy is easy to implement but its not optimal. For optimal utilization of road infrastructure, traffic signal controllers are required to be adaptive according to current level of congestion. In this paper we have modelled the problem as multi agent reinforcement learning (MARL). Each junction is modelled as reinforcement learning (RL) agent. The objective of each agent is to reduce it's local (own and immediate neighbors) congestion. We have used advantage actor critic with proximal policy optimization (PPO) for training of RL agents. To the best of our knowledge, ours is the first work that applies PPO in context of Multi Agent Traffic problem. Advantage of multi agent PPO setting is that our solution is scalable, distributed. Through experiments, we demonstrate effectiveness of our proposed algorithm. We have used PTV Vissim software for simulation of traffic behaviour.**

I. INTRODUCTION

There is rapid increase in privately owned vehicles in developing cities. Traffic congestion becomes a serious problem and has spread from major metropolitan areas to numerous small to mid-size cities. It increases human travel time, fuel consumption and air pollution. Optimized traffic control systems directly contribute to travel time reduction, savings in fuel consumption, and vehicular emission reduction. Congestion can be caused by one of the following three reasons: **i)** A temporary obstruction within a traffic network such as an incident, **ii)** A permanent capacity constraint, and **iii)** A stochastic fluctuation in the traffic flow. We are focused on solving the third reason, because to solve the permanent capacity constraint hefty amount of capital and time is required. We want efficient utilization of existing infrastructure resources that we already have. RL based techniques are well suited for complex stochastic environment like the road traffic networks. Recent progress in the area of RL can lead us to Intelligent efficient Transportation system.

The objective is to maximize traffic flow, minimize delay experienced by commuters, minimize congestion in the network by performing adaptive control of traffic lights at intersections. The idea is to develop a function that takes in current traffic information and outputs the signal timings. This function is approximated via a Neural Network.

Traffic light strategies can be designed for single-intersection control and network wide control. For a single signalized intersection, an optimal traffic light switching scheme can be computed efficiently to minimize the queue lengths described by a model. But that will not help the entire network. So we need a coordinated signal control system for optimizing traffic flow in the whole network.

The traffic control problem may be considered as a Multi Agent System, where each agent is autonomous and responsible for controlling the traffic light sequences of a single junction. In the context of RL-TSC (Traffic Signal Control), this scenario is described by the term Multi Agent Reinforcement Learning (MARL) [1], which consists of multiple RL agents, learning and acting together in the same environment. The scenario is a Partially Observable Markov Decision Process (POMDP) [2], as it is impossible for agents to know every detail about the environment in large scale transportation networks in the real world. Each Agent has access to local status of the environment.

II. RELATED WORKS

Basic approach to control traffic signals is to train a single RL agent to control all junctions in the system. However, this approach is not feasible when considering large networks, due to a lack of scalability and the huge number of potential actions that may be selected. Thus, MARL is the standard approach in RL-TSC research, as it scales much better than a single agent controlling an entire network.
In the paper [3] authors have discretized the queue occupancy as low, medium, high. They have used Q-Learning [4] Algorithm for solving this MARL problem. And for exploration versus exploitation problem they have used Upper Confidence Bound (UCB) [3] strategy. This is a tabular Q Learning strategy.

One way of representing the state[5] of an agent can be to discretize a length l of the lane segment, beginning at the stop line, into cells of length c. Each of the cell is a Boolean variable representing presence or absence of vehicle in that cell. If c is many times larger than the average vehicle length, the individual dynamics of each vehicle will be lost, however computational cost will be reduced. If c is much smaller than the average vehicle length, the individual vehicle dynamics will be retained, however the computational cost will increase unnecessarily. Finally the state is composed of three vectors, the first representing the presence of a vehicle or not in the cell, the second the speed of the vehicle and the third the current traffic signal phase (i.e., the most recent action selected). An example of this approach is shown in (Fig1).

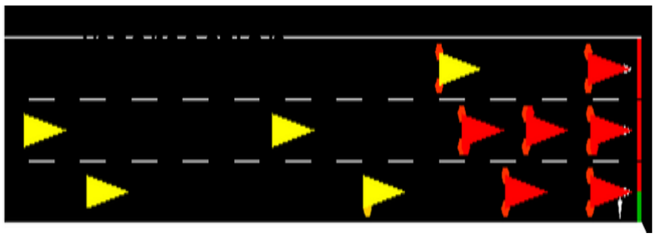
Fig. 1: an example for showing vehicles and lane in SUMO simulation environment

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

TABLE I: Boolean variable for vehicle presence Status Vector for the (Figure 1), 1 implies presence, 0 implies no Vehicle

0	0	0	0	0	0	0	0.2	0	0	0
0	0	0	0.5	0	0	0	0	0	0	0
0.6	0	0	0	0	0.4	0	0	0	0	0

TABLE II: Vehicle speed vector for (Figure 1)

Given two vectors (II,I) are combined with the recent traffic signal phase to form the state of system.Reward is the change in cumulative delay. They have used Deep Q-Network [6] for training. However this state information doesn't consider any information about the neighbors.
Authors in [7] have defined state as queue lengths at each incoming lane, number of vehicles, their current waiting time, current phase at that junction and an image representation of vehicles' position. Reward is a weighted combination of queue length, delay, waiting time, total travel time. This kind of state representation is too complex for implementation in real scenario.
Authors in [8] have used deep stacked Autoencoder [9] neural network to estimate the Q-function here. This neural network takes the state(queue length, speed) as input and outputs the Q-value for each actions(stay in same phase or change). Q-values are hidden representation of state in Autoencoder.

III. Problem Formulation

A. Markov Decision Process Framework

A Markov Decision Process (MDP) is a discrete time stochastic control process. It provides a mathematical modelling of sequential decision making situations where state of system is markovian. MDP is a tuple $< S, A, P, R, \gamma >$

- S is finite set of states
- A is finite set of actions
- P is state transition matrix, satisfying (1)
- R is scalar reward function.
- γ is discount factor $\gamma \in [0,1]$

$$P(s_{t+1}|s_t,a_t) = P(s_{t+1}|s_1,a_1,s_2,a_2..s_t,a_t) \quad (1)$$

Actions in a state are taken according to a policy, policy π is a distribution over actions given a state. $\pi(a|s) = P(A_t = a|S_t = s)$. Policy fully characterizes Agent's Behaviour. For a policy π:

$$Q^\pi(s_t,a_t) = E_{s_{t+1},a_{t+1},...}[\sum_{l=0}^{l=\infty} \gamma^l r(s_{t+l},a_{t+l})] \quad (2)$$

$$V^\pi(s_t) = E_{a_t,s_{t+1},a_{t+1},...}[\sum_{l=0}^{l=\infty} \gamma^l r(s_{t+l},a_{t+l})] \quad (3)$$

$$A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t) \quad (4)$$

Q is known as state-action value function. V is known as value function. A is known as Advantage function. Advantage estimates how good a particular action in a state is as compared to average performance in the state. Our objective is to find policy π^* that maximizes long term cumulative sum of rewards (5).

$$E[\sum_{t=0}^{t=\infty} \gamma^t r_t(s_t,a_t)] \quad (5)$$

B. Traffic Signal Control Problem as MDP

Problems involving sequential decision making can be modelled as MDP. We formulate the traffic signal control problem in MDP framework. We consider the problem of finding an optimal schedule for the sign configurations at traffic junctions with the aim of minimizing congestion in Network. The signals associated with a phase i.e., those that can be switched to green simultaneously form a sign configuration. Widely followed phasing systems are shown in (Fig 2, 3).

We have used phasing system shown in fig(3) in our experiments. As this is easier to implement ,requires less infrastructure, and in cases where vehicle flow density is asymmetric, it performs better (analysis in appendix) than the one shown in fig(2).

An MDP formulation requires the characterization of states, actions, rewards. We consider a traffic network with N junctions, $N \geq 1$. Each junction i has a set $P(i)$ incoming lanes, and some(≥ 1) outgoing lanes. Every junction is a RL Agent. Formally let $J = \{1, 2, ..N\}$ be set of junctions in the network. Definitions:

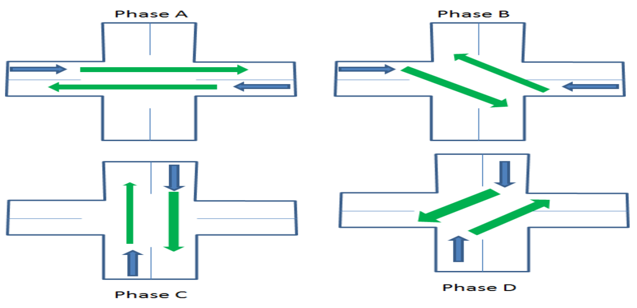

Fig. 2: Two lanes can be green at same time

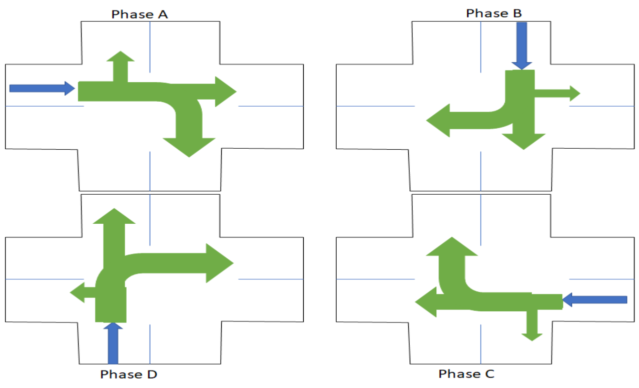

Fig. 3: Simple Phasing System, one lane allowed to go in all directions

$q^t_{i,j}$ =congestion(occupancy) at incoming lane$j$ of junction i at time t.
$N(i)$ =set of immediate Neighbors of junction i, including i.
$P(i)$ = set of incoming lanes at junction i.
Our algorithms requires a description of states, actions and rewards.
The **state** of junction i at time t is a $L+1$ dimensional vector of occupancy at its own incoming lanes and of its immediate neighbors and the current phase at this junction.
$state(i)_t = \{q^t_{j,k} : \forall k \in P(j), \forall j \in N(i)\}$
Current signal phase is the last dimension of state vector.
L can be different for different junction.
Action is to chose the time interval of green signal for the next phase. Action space is a discrete action in $\{20,25,30,....,70\}$ seconds.
Reward for every action by an agent i is the difference in congestion at its own and its neighbors lane before and after taking action. formally
$r^i(t+a) = \sum_{j=1}^{j=L} state(i)_{t,j} - \sum_{j=1}^{j=L} state(i)_{t+a,j}$
where a is action taken by agent i at time t. $state(i)_{t,j}$ is j^{th} component of state vector of agent i at time t.
Multi-Agent Structure comes from the fact that reward depends not only on action of the agent itself, but also on the joint actions of all neighboring agents. This is an infinite horizon MDP. We want to find policy(π^*) to maximize discounted sum of rewards.

$$\pi^* = \arg\max_{\pi_\theta} \sum_{t=0}^{t=\infty} E_{a_t \sim \pi_\theta(s_t), s_{t+1} \sim \rho} \gamma^t r(s_t, a_t, s_{t+1}) \tag{6}$$

ρ in eqn(6) is transition probability matrix, which is unknown. Since π in eqn(6) is parameterized by θ. In order to find π^*, we need to find corresponding θ^*.

IV. Algorithm

We have used Advantage Actor Critic(A2C) [10] with Proximal Policy Optimization(PPO) [11] for training of each agent. In A2C method, actor maximizes expected total reward by estimating the gradient (7), and taking a step in the direction of gradient.

$$g = E[\sum_{t=0}^{t=\infty} A^\pi(s_t, a_t) \nabla_\theta log \pi_\theta(a_t|s_t)] \tag{7}$$

Critic estimates parameterized $V^\pi_\phi(s_t)$. A2C is an on-policy algorithm, it suffers from high sample complexity, and the gradient estimates suffer from high variance. We don't always get a improvement in policy after taking gradient steps because step size could be high. PPO is an optimization technique for policy gradient methods [12], that enforces a condition such that new policy after gradient update is not much far away from old policy.

Let $r_t(\theta)$ denote the probability ratio, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$. The modified objective in PPO is

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{8}$$

where *clip* is defined as:

$$clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon) = \begin{cases} 1+\varepsilon, & \text{if } r_t(\theta) \geq 1+\varepsilon \\ 1-\varepsilon, & \text{if } r_t(\theta) \leq 1-\varepsilon \\ r_t(\theta), & \text{otherwise} \end{cases} \tag{9}$$

The *min* in (8) in combination with clipping does not let the new policy move far from old policy. ε is a hyperparameter. Taking multiple gradient steps on samples from $\pi_{\theta old}$ helps in searching for a better policy in a restricted(because of min and clipping) region. Further we have added entropy (10) term to the objective to improve exploration by agent.

$$H(\theta) = -\sum_{a_t} \pi_\theta(a_t|s_t) log \pi_\theta(a_t|s_t) \tag{10}$$

$$max(L^{CLIP}(\theta) + \beta H(\theta)) \tag{11}$$

For actor our final objective is (11). β is a hyperparameter. For critic, objective is to minimize mean square loss (13) between estimated value(12) and output by neural network.

$$y_t = \sum_{l=0}^{l=T-1} \gamma^{t+l} r_{t+l} + V_\phi(s_{t+T}) \tag{12}$$

$$min(V_\phi(s_t) - y_t)^2 \tag{13}$$

$$\hat{A}(s_t, a_t) = y_t - V_\phi(s_t) \tag{14}$$

Our final algorithm is shown in Algorithm 1.

Algorithm 1 Multi Agent Training (PPO Actor Critic)

```
1: Load network file to VISSIM simulation software
2: Initialize every junction as agent($\pi_{old}, \pi_{new}, V_\phi$)
3: for time in range(totalTime) do
4:     run single step of VISSIM simulation
5:     for agents that need to take action do
6:         get $s_t, a_t, r_{t+a_t}, s_{t+a_t}$, put it into buffer
7:         if buffer has enough samples then
8:             Estimate Advantage using (14)
9:             for j in range(K) do
10:                sample minibatch from buffer
11:                train $\pi_{new}$ using (11)          ▷ actor training
12:            end for
13:            $\pi_{old} \leftarrow \pi_{new}$
14:            train $V_\phi$ using (13)                 ▷ critic training
15:        end if
16:        take action $a \sim \pi_{new}(s_{t+a_t})$     ▷ Sample action
17:    end for
18: end for
```

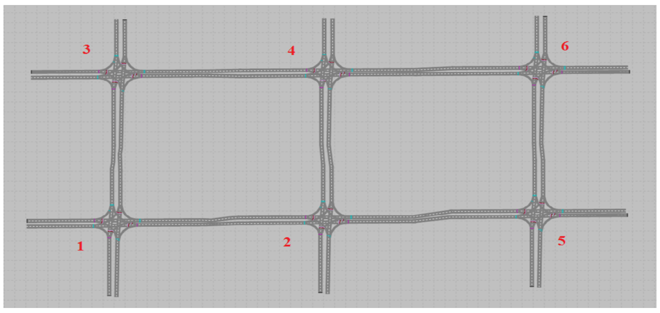

Fig. 4: Six junction network for experimentation purpose

V. Experimentation

We have used microscopic multi-modal traffic flow software PTV VISSIM for simulation of Traffic behaviour. We built a traffic network of six junctions for experimentation purpose. The network is shown in figure(4). Details of network shown in figure(4) are given:

- 6 junctions
- 10 vehicle input points, vehicle incoming follows poisson process, we set parameter(λ)
- 8 =(2roads x 4) incoming lanes at each junction
- 8 signals one on each incoming lane for each junction
- Queue Counter behind every signal to get congestion value on each lane

A. Network Architecture

Both the Actor Critic Networks are simple dense network with 2 hidden layers having 64 hidden nodes each. Both hidden layers have *tanh* activation. Every agent(junction) is represented by 3 networks(2 policy, and 1 critic networks). State dimensions at different junctions can be different(based on how many neighbors it has). Action space is same at each junction, discrete {20,25,...,65,70}. State space is continuous.

B. Hyperparameters

Initially we have set ε in (8) as 0.2 for all agents. During training we gradually reduce it, if we see reduction in performance of particular agent. β in (11) is initially set to 0.4 to support exploration. After sometimes ($\sim$ 6 lac) training seconds, we reduce it to 0.1, 0.05. Buffer size is 64. We sample a minibatch of 16 (4times),for training actor.

C. Training Strategy

These algorithms are very unstable. We have followed a few heuristics for getting better results. Initially all the Agents were trained together, i.e. all the RL agents in training mode. After some times($\sim$ 7 lac seconds), When all the networks have improved significantly from random actions, we put some agents in train mode and some in test mode, based on which one was performing better. Training only a few decreases the instability in agents training, because test mode agents are behaving in their optimal manner. We repeat this procedure of training only a few (permute every time based on performance) at a time for some iterations till we see significant improvement in performance.

D. Results

To get state at current time, we check how long the queue is filled and then we divide this length by some fixed number(200 in our case). Assuming we have a maximum view capacity of 200 meters on every lane of each junction. We have rounded the queue length to 2 decimal digits. We find this reasonable for following reasons:

- The trained model is supposed to be embedded in small devices (traffic signal) having limited computational precision of floating point.
- Queue Occupancy after 3rd decimal digits are irrelevant. As it doesn't affect the signal timings much since our action space is discrete. So we don't require so much of precision in terms of floating point computation.

We have tested the trained agent in fixed vehicle incoming rate, and randomly changing rate settings. Vehicle incoming follows Poisson process. Total run time in all the cases are 1 lac seconds. Parameters on which we have compared the results are following.

- Mean Occupancy : average of sum of occupancy at incoming lanes measured at the time of taking action, RL agent's primary objective is to decrease sum of mean occupancy at all the junctions.
- Average Speed : Total distance travelled by all vehicles divided by total time taken by all vehicles, calculated for every 1000 seconds interval
- Average Delay time : Total delay for all vehicles divided by total number of vehicles,calculated for every interval of 1000 secs.

1) Incoming Rate = 2000 vehicles per hour

Total Vehicle input is 20,000 vehicles per hour to the network(contains 10 vehicle incoming points). Figures(5,6) show Mean Occupancy comparison, We can see RL agents in combined affect has less occupancy(in whole network) than any fixed time algorithms. Figures(7,8) show average speed and delay comparison respectively. We can see our MARL algorithm has performed better than fixed time algorithms.

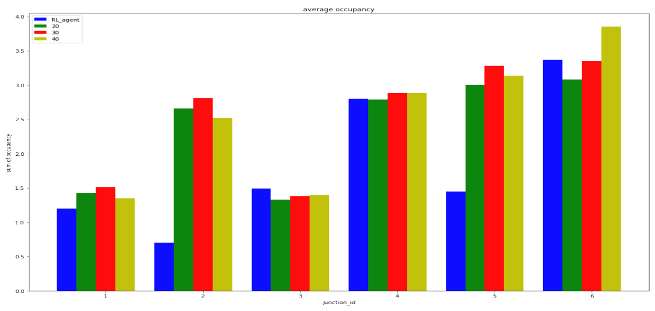

Fig. 5: Comparison of congestion at all junctions for RL, and fixed time(20,30,40) algorithms (Blue- RL, Green-20, Red - 30, Yellow-40). Summing up differences at all the junctions, our MARL algorithm is performing better than fixed time algorithms for fixed vehicle incoming rate 2000 per hour

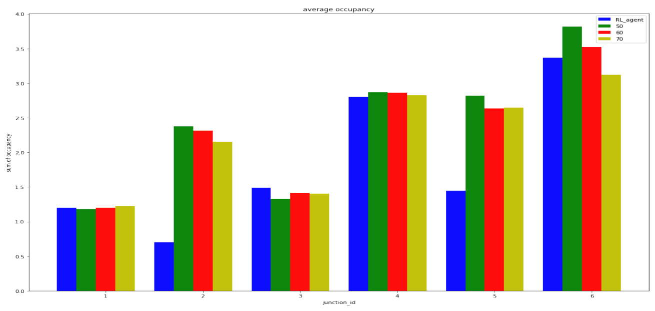

Fig. 6: Comparison of congestion at all junctions for RL, and fixed time(50,60,70) algorithms (Blue- RL, Green-50, Red - 60, Yellow-70). Summing up differences at all the junctions, our MARL algorithm is performing better than fixed time algorithms for fixed vehicle incoming rate 2000 per hour

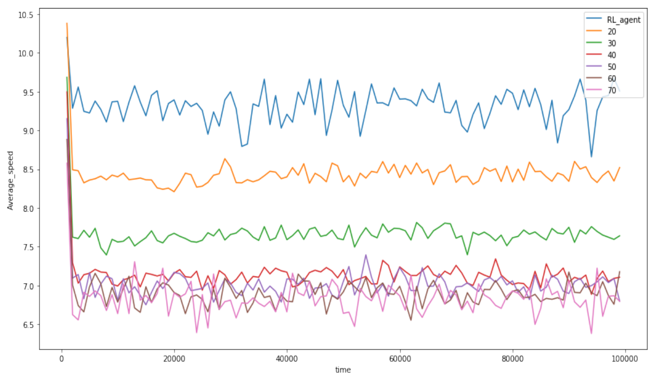

Fig. 7: Average speed comparison for 2000 vehicles per hour, calculated for interval of 1000 secs, we see that our MARL algorithm has higher average speed(Blue line) than any fixed time algorithms

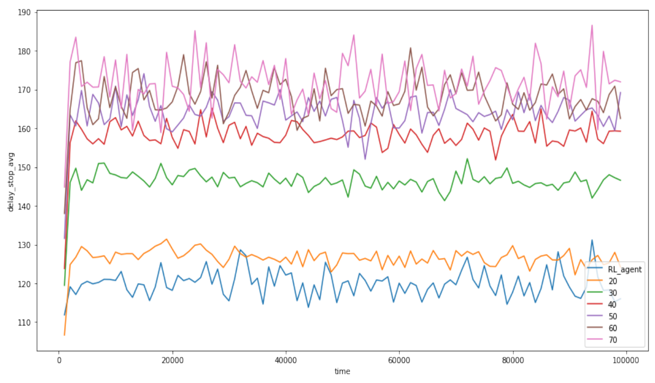

Fig. 8: Delay Stops time comparison for 2000 vehicles per hour calculated at interval of 1000 secs, we see that our MARL algorithm has lower average delay (Blue line) time than any fixed time algorithms

2) *Randomly Changing Incoming Rate*

The results Fig(9,10,11,12) shown are for randomly changing vehicle incoming rate $\in \{500, 1000, ...3000, 3500\}$. The change is done at all incoming points after every 50 minutes.

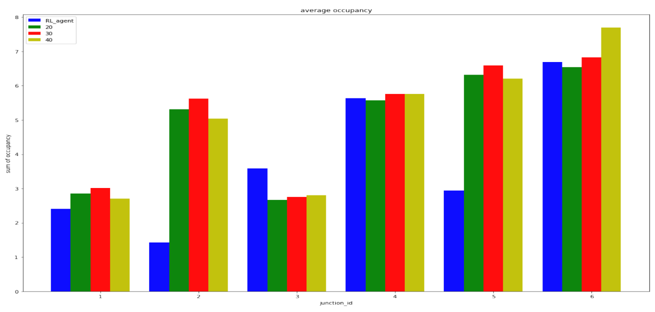

Fig. 9: Comparison of congestion at all junctions for RL, and fixed time(20,30,40) algorithms (Blue- RL, Green-20, Red - 30, Yellow-40). Summing up differences at all the junctions, our MARL algorithm is performing better than fixed time algorithms for randomly changing vehicle incoming rates

Figure(9, 10) shows the average congestion comparison between fixed time algorithms and RL agents algorithm for every junction in network. At junctions 2,5 RL agents are performing a lot better, where as at junction 3 RL agent is performing a bit worse. But if we combine their performance for all the junctions, RL agents are performing better than all fixed time algorithms. These are an average for 1lac seconds.

Figure(11) shows comparison in average speed of all vehicles in network for different algorithms. Figure(12) shows comparison in average delay of all vehicles in network for different vehicles. All the values have been calculated at interval of 1000secs. We can see our MARL algorithm has performed better(high average speed, low delay) than any fixed time algorithms.

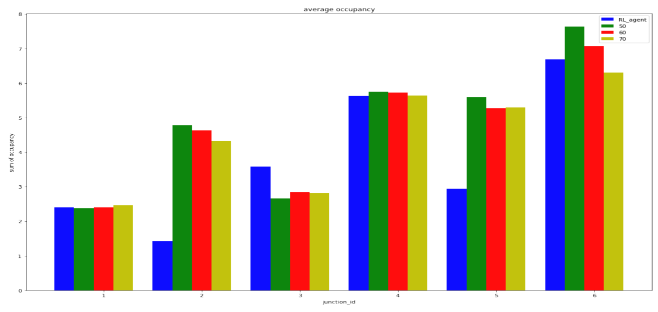

Fig. 10: Comparison of congestion at all junctions for RL, and fixed time(50,60,70) algorithms (Blue- RL, Green-50, Red - 60, Yellow-70). Summing up differences at all the junctions, our MARL algorithm is performing better than fixed time algorithms for randomly changing vehicle incoming rates

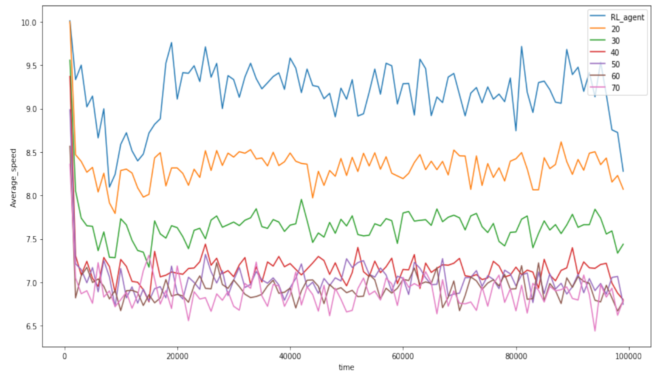

Fig. 11: Average speed comparison (for randomly changing incoming rate) calculated at interval of 1000 secs, we see that our MARL algorithm has higher average speed(Blue line) than any fixed time algorithms

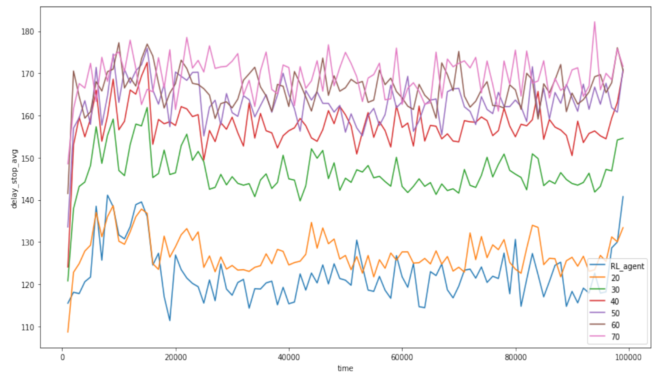

Fig. 12: Delay Stops time comparison (for randomly changing incoming rate) calculated at interval of 1000 secs, we see that our MARL algorithm has lower average delay(Blue line) than any fixed time algorithms

VI. Conclusion

- Agent's primary objective was to reduce the congestion(they got positive reward) in their local environment. We can see in figures(5, 6, 9, 10), In case of MARL algorithm, total congestion is less than fixed scheduling algorithms.
- Decrease in the congestion has led to increase in average speed (fig(7,11)) per vehicle.
- Decrease in congestion has led to decrease in average delay time(fig(8, 12)) per vehicle.
- There are a few manual tuning steps during training, due to multi agent convergence related issues.
- Our solution supports different action space for different agents.
- This is a scalable solution, as all junctions are represented as different (Neural Networks), and they can be trained in a distributed manner independent of each other if the simulation supports distributed access.
- This is a easily deployable solution in real scenario. Model needs queue length data, for that we suggest two feasible solutions.
  1) A high quality camera should be mounted at every incoming lane of junction. A computer vision related trained Neural Net shall be able to process the image and extract the information about how much the queue is filled.
  2) We can put detectors on road ending at a junction at the gap of around (30-40)meters. These detectors can tell how much the road is filled.
  3) In our model every agent communicates with its neighbors, to implement this we need a way of communication(wireless) between all the junctions. We need reliable mode of communication between every neighbors.
- If agents have more information about neighbors(radius 2), they should perform better than current settings.
- We have results for different incoming densities (shown in appendix), and the RL agents have performed better than all fixed time algorithms.

VII. Future Work

A. *Comparison with other Algorithms*

We would like to compare our results with some previous algorithms like Sydney Co-ordinated Adaptive Traffic Control(SCATS) [13], Split Cycle Offset Optimization Technique(SCOOTS) [14], Optimized Policies for Adaptive Control(OPAC) [15].

B. *Training for more real scenario*

We need to test in more realistic conditions. We will build map of some real city's traffic network. These maps are open-source, but road network(with full details) is required to be drawn manually inside simulation software. Data about vehicle incoming rates can be gathered by consulting Google maps.
If in new network, an agent's structure(same number of neighbors) is same as in (4), the agent's weight can be initialized with this weight, and it will take less time during training.

C. Convergence

We saw in figures(9,10) that some junctions are performing far better than fixed time algorithms, where as some are a bit worse. To decrease the training time, we would like to try with off policy algorithms. We would like to work on theoretical convergence of multi agent PPO actor critic algorithms.

References

[1] M. Tan, "Readings in agents," M. N. Huhns and M. P. Singh, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, ch. Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents, pp. 487–494. [Online]. Available: http://dl.acm.org/citation.cfm?id=284860.284934

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, May 1998. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(98)00023-X

[3] P. K.J., H. K. A.N, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 2529–2534.

[4] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[5] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *CoRR*, vol. abs/1611.01142, 2016. [Online]. Available: http://arxiv.org/abs/1611.01142

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[7] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 2496–2505. [Online]. Available: http://doi.acm.org/10.1145/3219819.3220096

[8] L. Li, L. Yisheng, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *ACTA AUTOMATICA SINICA*, vol. 3, pp. 247–254, 06 2016.

[9] D. H. Ballard, "Modular learning in neural networks," in *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'87. AAAI Press, 1987, pp. 279–284. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863696.1863746

[10] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, Apr. 2003. [Online]. Available: https://doi.org/10.1137/S0363012901385691

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[12] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 1057–1063. [Online]. Available: http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

[13] A. Sims and K. Dobinson, "The sydney coordinated adaptive traffic (scat) system philosophy and benefits," *Vehicular Technology, IEEE Transactions on*, vol. 29, pp. 130 – 137, 06 1980.

[14] P. Hunt, D. Robertson, and R. Bretherton, "The scoot on-line traffic signal optimisation technique (glasgow)." *Traffic Engineering Control*, vol. 23, pp. 190–192, 01 1982.

[15] N. Gartner, "Opac: A demand-responsive strategy for traffic signal control," *Transportation Research Record Journal of the Transportation Research Board*, vol. No. 906, pp. 75–81, 01 1983.

Appendix

A. Results and Analysis

1) Incoming Rate = 1000 vehicles per hour

Total Vehicle input is 10,000 vehicles per hour. Figures(13,14) show Mean Occupancy comparison. Figures(15,16) show average speed and delay comparison respectively.

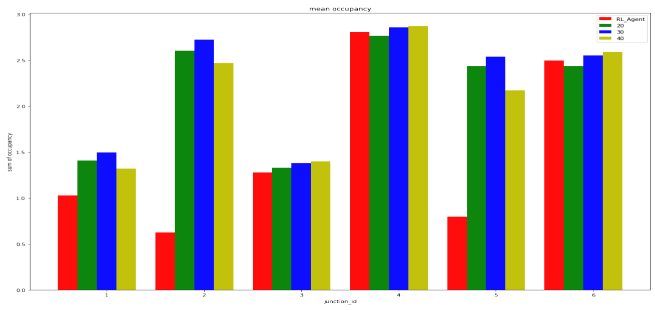

Fig. 13: Comparison of congestion at all junctions for RL, and fixed time(20,30,40) algorithms.(Red-RL, Green-20, Blue-30, Yellow-40). Summing up differences at all the junctions, RL agents are performing better than fixed time algorithms for fixed incoming rate 1000 vehicles per hour at each input

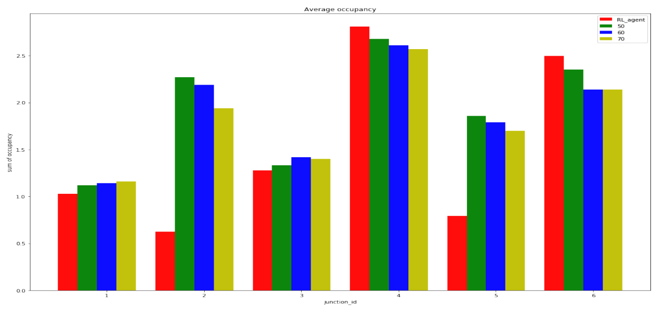

Fig. 14: Comparison of congestion at all junctions for RL, and fixed time(50,60,70) algorithms.(Red-RL, Green-50, Blue-60, Yellow-70). Summing up differences at all the junctions, RL agents are performing better than fixed time algorithms for 1000 vehicles per hour at each input

2) Incoming Rate = 3000 vehicles per hour

Total Vehicle input is 30,000 vehicles per hour. Figures(17,18) show Mean Occupancy comparison. Figures(19,20) show average speed and delay comparison respectively.

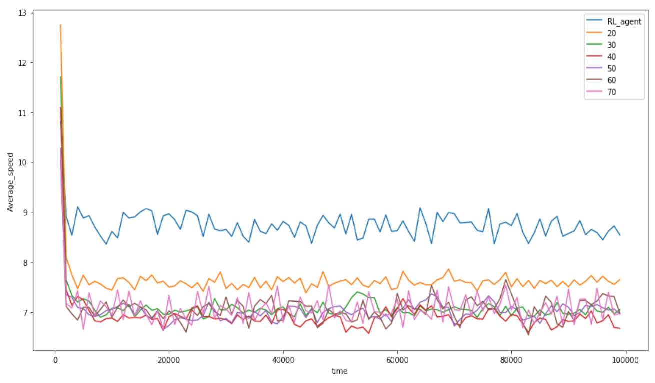

Fig. 15: Average speed comparison calculated for interval of 1000 secs, Our Algorithm has higher average speed(blue) for 1000 vehicles per hour at each input

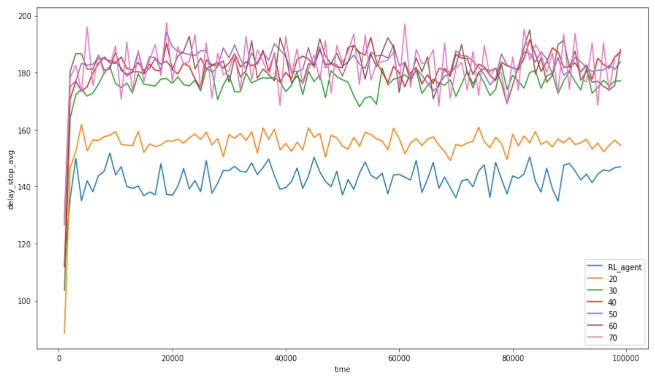

Fig. 16: Delay Stops time comparison at interval of 1000 secs, our MARL algorithm has lower delay(blue) for 1000 vehicles per hour at each input

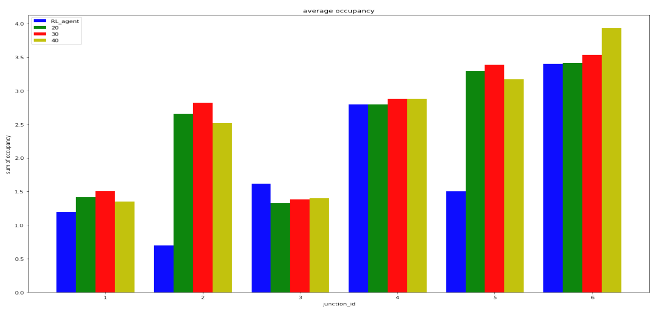

Fig. 17: Comparison of congestion at all junctions for RL, and fixed time(20,30,40) algorithms.(Blue-RL, Green-20, Red-30, Yellow-40). Summing up differences at all the junctions, RL agents are performing better than fixed time algorithms for 3000 vehicles per hour at each input

B. Phasing analysis

Figure(21) shows a common junction. It has 4 lanes each containing incoming and outgoing. Definitions:
Let $I_k, k \in \{1,2,3,4\}$ denote incoming lanes.
Let $O_k, k \in \{1,2,3,4\}$ denote outgoing lanes.
Let $R_{j,k}$ denote a route from incoming lane I_j to outgoing lane O_k.

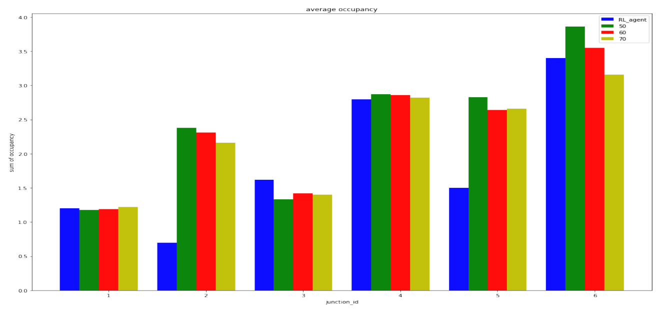

Fig. 18: Comparison of congestion at all junctions for RL, and fixed time(50,60,70) algorithms.(Blue-RL, Green-50, Red-60, Yellow-70). Summing up differences at all the junctions, RL agents are performing better than fixed time algorithms for 3000 vehicles per hour at each input

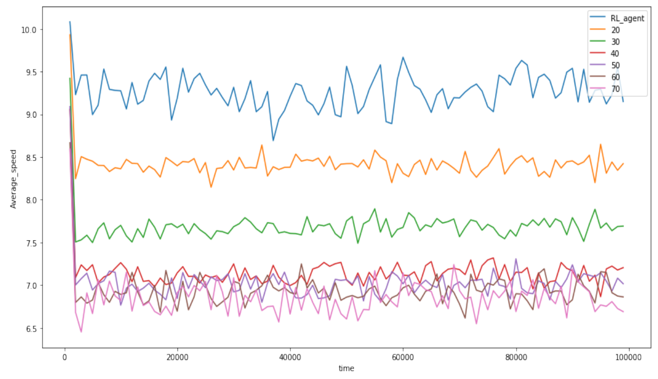

Fig. 19: Average speed comparison calculated for interval of 1000 secs, our MARL algorithm has higher speed(blue) for 3000 vheicles per hour at each input

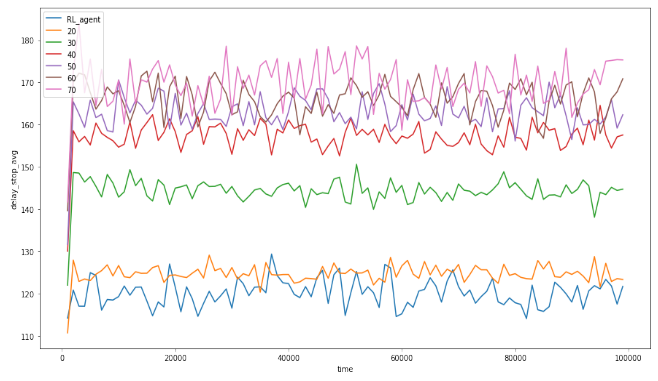

Fig. 20: Delay Stops time comparison at interval of 1000 secs,our MARL algorithm has lower delay(blue), for 2000 vehicles per hour at each input

We want to analyze total vehicle flow density in different phasing styles shown in figures(3, 2). We will assume left turn at junction is always open. On any incoming I_k having f as flow rate, αf goes straight, $(1-\alpha)f$ takes right turn.

For our analysis, incoming lanes and flow rates are shown in table (III). Routes that will be active during a phase are

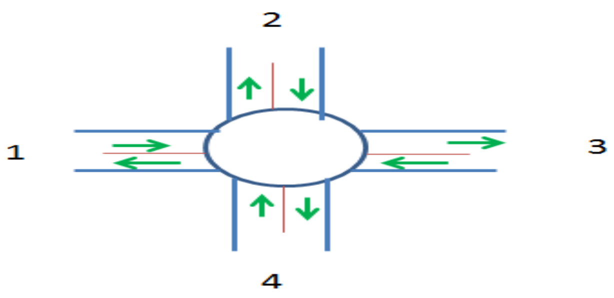

Fig. 21: 4 lane incoming junction

IncomingLane	FlowRate
I_1	$kf, k \geq 2$
I_2	f
I_3	f
I_4	f

TABLE III: Flow rate at incoming lanes of junction in figure(21). $k \geq 2$ shows asymmetric nature in traffic movement. Generally during peak hours(office opening time/office closing time), incoming flow from residential area/office area will be higher as compared to all other regions.

Phase	Allowed Route in fig(3)	Allowed Route in fig(2)
A	$R_{1,3}, R_{1,4}$	$R_{1,3}, R_{3,1}$
B	$R_{2,4}, R_{2,1}$	$R_{1,4}, R_{3,2}$
C	$R_{3,1}, R_{3,2}$	$R_{2,4}, R_{4,2}$
D	$R_{4,2}, R_{4,3}$	$R_{2,1}, R_{4,3}$

TABLE IV: Allowed route in different phases for phasing techniques shown in figures(3,2). $R_{i,j}$ denotes route from incoming I_i to outgoing O_j.

Phase	Allowed Route	time given	flow on allowed route	total flow
A	$R_{1,3}, R_{1,4}$	kt	kf	k^2ft
B	$R_{2,4}, R_{2,1}$	t	f	ft
C	$R_{3,1}, R_{3,2}$	t	f	ft
D	$R_{4,2}, R_{4,3}$	t	f	ft

TABLE V: total flow analysis for phasing in figure(3), cycle time is $(k+3)t$, since flow on I_1 is k times all other flows, so phase A gets kt time. Others get t time.

shown in table (IV). We consider a cycle time of $(k+3)t$. For cycle time $(k+3)t$, Table(VI) shows total flow for phasing in figure(2), Table(V) shows total flow for phasing in figure(3).

1) $\alpha \in (0.5, 0.7)$ in case of traffic for most junctions.
2) Sum of total flow in table(VI) = $(\alpha^2 + (1-\alpha)^2)((k+1)^2 + 4)ft$
3) Sum of total flow in table(V) = $(k^2 + 3)ft$
4) $(\alpha^2 + (1-\alpha)^2) \in (0.50, 0.58) if \alpha \in (0.5, 0.7)$. Because of this reduction factor almost by half, for $k = 3, 4, 5, ..$ simple phasing shown in fig(3) wins over two lane movement at one time shown in figure(2).

Phase	Allowed Route	time given	flow on allowed route	total flow
A	$R_{1,3},R_{3,1}$	$\alpha(k+1)t$	$\alpha(k+1)f$	$\alpha^2(k+1)^2ft$
B	$R_{1,4},R_{3,2}$	$(1-\alpha)(k+1)t$	(1-α)$(k+1)f$	$(1-\alpha)^2(k+1)^2ft$
C	$R_{2,4},R_{4,2}$	$2\alpha t$	$2\alpha f$	$4\alpha^2 ft$
D	$R_{2,1},R_{4,3}$	$2(1-\alpha)t$	$2(1-\alpha)f$	$4(1-\alpha)^2 ft$

TABLE VI: total flow analysis for phasing in figure(2). Since total flow on lanesI_1,I_3 are $(k+1)f$, so total time for lanes (1,3) should be $(k+1)t$. Since in phase A, α times incoming on lanes(I_1,I_3) move, so phase A gets $\alpha(k+1)t$. Phase B gets remaining of $(k+1)t$. Phases C and D share $2t$ in ratio $\alpha/(1-\alpha)$.