Decentralized Learning for Traffic Signal Control

Prabuchandran K.J.[1], Hemanth Kumar A.N[1], Shalabh Bhatnagar[1] *Senior Member, IEEE*,

Abstract— **In this paper, we study the problem of obtaining the optimal order of the phase sequence [14] in a road network for efficiently managing the traffic flow. We model this problem as a Markov decision process (MDP). This problem is hard to solve when simultaneously considering all the junctions in the road network. So, we propose a decentralized multi-agent reinforcement learning (MARL) algorithm for solving this problem by considering each junction in the road network as a separate agent (controller). Each agent optimizes the order of the phase sequence using Q-learning with either ϵ-greedy or UCB [3] based exploration strategies. The coordination between the junctions is achieved based on the cost feedback signal received from the neighbouring junctions. The learning algorithm for each agent updates the Q-factors using this feedback signal. We show through simulations over VISSIM that our algorithms perform significantly better than the standard fixed signal timing (FST), the saturation balancing (SAT) [14] and the round-robin multi-agent reinforcement learning algorithms [11] over two real road networks.**

Index Terms— **traffic signal control, optimal phase sequence, multi-agent reinforcement learning, Q-learning, UCB, VISSIM.**

I. Introduction

Traffic signal control (TSC) is concerned with the problem of intelligently controlling the traffic lights at junctions for minimizing the average delay experienced by the road users. This optimization can be carried out in two possible ways. In one approach, the order of the phase sequence is fixed and the green signal duration of the phases [14] is controlled. In the other approach, the green signal duration of the phases is fixed and the order of the phase sequence is controlled. In this paper, we follow the latter approach. Here, the average delay is reduced by skipping phases appropriately that have low traffic arrival rates.

Modeling the real time traffic along with signal controls at junctions is the first and foremost step towards the solution of the TSC problem. Markov decision process (MDP) with a centralized controller (single agent) provides us a suitable platform for modeling the real time traffic environment. However, the drawback with this modeling is the curse of dimensionality [17], i.e., the total number of states in the MDP exponentially increases with the number of junctions in the road network. Thus, in our paper we decentralize the control to multiple agents and consider each junction as a separate controller (agent). Each agent observes only its own portion of the state space. Nevertheless to decide the order of the phase sequence for its own junction, each agent obtains a feedback cost signal from its neighbors. Note that this facilitates coordination between agents.

The next step in the solution of the TSC problem is to devise suitable algorithms to learn the order of the phase sequence so as to reduce the average delay. The reinforcement learning (RL) based approaches that do not require an explicit model of the traffic environment provide us with simple online algorithms for learning the order of the phase sequences. In this paper, we use the Q-learning algorithm for each controller with exploration mechanism based on either ϵ-greedy method or upper confidence bound (UCB) [3] based method. The cost function for an agent in the Q-learning algorithm uses the information on the congestion-level of its lanes and its neighboring junction's lanes to enable co-ordination (see section II).

Now we review the various approaches in the traffic signal control literature that have been applied to solve the TSC problem. The fixed signal timing (FST) algorithm [15], [10] calculates the signal timings of the phases off-line based on analysis of the traffic at different times during the day. This algorithm incurs large delay as it is not adaptive to traffic fluctuations.

There have been multi-agent based approaches for automatically controlling the traffic lights [5]. In [4], vehicle based representation is used to construct the model and dynamic programming is used to estimate the optimal value function. In [8], [9] max-plus algorithm as co-ordination strategy is utilized in the multi-agent setting. In [2], distributed traffic signal control setting with two types of agents - central agent and outbound agent are studied. Central agent employs Q-learning with function approximation and the outbound agent employs longest queue first algorithm.

In [7], Q-learning based acyclic signal control system for a single intersection with different state representations is studied. In [1], multi-agent Q-learning algorithm with local cost information for large traffic networks is developed. A collaborative RL algorithm is studied in [16] where Q-learning based agents employ Boltzmann action selection scheme. RL with function approximation is considered in [12], [2], [6]. Note that RL algorithms with function approximation scale to large road networks, however these algorithms lack convergence guarantees. Additionally feature selection for function approximation is a challenging task.

In [11] multi-agent Q-learning algorithms with round-robin scheduling of phases have been developed. The round-robin scheduling algorithms do not perform very well when there are large variations in the traffic arrivals in different phases. Our algorithms, on the other hand, adaptively skip phases that have low traffic arrival rates and give priority

[1] Department of Computer Science and Automation, Indian Institute of Science, Bangalore-560012, India. {prabu.kj,hemanth.kumar, shalabh} @csa.iisc.ernet.in

to the high traffic phases in order to achieve significant reduction in delay (see section IV).

In this paper, we formulate the problem of intelligently controlling the traffic lights in a road network, namely the TSC problem, as a distributed learning problem. We model each traffic junction as an agent. Each agent updates its Q-factors (see section III) using Q-learning with either an ϵ-greedy or a UCB based exploration strategy using a feedback cost signal obtained from its neighbors. Then, based on the learnt Q-factors, the agent decides the phase that needs to be set to green as a function of the current congestion level and the time elapsed since the signal turned red on each of the lanes. We show that our algorithms yield policies that outperform both the FST and the SAT algorithms. We also show our algorithms perform better than the round-robin multi-agent Q-learning algorithm in [11]. Our main contributions are as follows:

- We model the TSC problem of obtaining the optimal order of the phase sequences in the framework of MDP.
- We assume the traffic congestion level across various lanes is available from sensors. Our algorithms use only coarse information on the traffic. This leads to low response time as it reduces the computation required to collect and process the data from the sensors.
- Instead of having a centralized controller to manage the complete road network, we distribute the control to multiple agents where each agent controls the order of the phase sequence in each junction. This makes our algorithm scalable to large road networks.
- The cost signal in the Q-learning update rule of an agent is constructed based on the traffic congestion level information on the lanes in its junction and its neighbors. This enables the co-ordination between agents.
- We set the initial order of the phases in all the junctions randomly. The agents at the various junctions progressively achieve a self organizing behaviour by interacting with one another.

The rest of the paper is organized as follows. In section II, we formulate the TSC problem as a MDP and propose a decentralized solution to control the traffic lights at the junctions. Section III then presents the distributed Q-learning algorithm with ϵ-greedy and UCB based exploration mechanisms. Section IV provides simulation results on the performance of our algorithms as well as comparisons to the FST, SAT and multi-agent Q-learning [11] algorithms. Finally, Section V provides the concluding remarks.

II. Traffic Signal Control problem as an MDP

Consider a road network that has J junctions. Each junction has certain number of incoming lanes along which the traffic approaches the junction. A phase is a subset of all possible traffic lights at a junction that are set to green (red) at the same time so that the traffic streams along different lanes do not collide with each other. Each junction has multiple phases associated with it. We will denote the total number of phases at junction j by by $P_j \ \forall \ j \in \{1, 2, \ldots, J\}$.

In a standard fixed signal timing (FST) algorithm, phases in a junction are visited in a round-robin manner. The green time duration for phases in the FST algorithm are calculated based on the traffic arrival rates in each of the phases. This algorithm incurs large delay as the duration of phases is not adaptive to the traffic fluctuations. The multi-agent Q-learning algorithm studied in [11] is also a round-robin algorithm. Unlike the FST algorithm, the green signal duration is not fixed for the phases, but instead it is adaptively set based on the state of the system. In [11], it is shown that these algorithms learn good green signal duration of phases adaptively and perform well under traffic fluctuations. However, when the arrival rates in different phases are highly unbalanced, round-robin scheduling of the phases results in blocked lanes. In such scenarios, it is important to skip phases that are not congested in order to clear the long queues of congested lanes at the same time ensuring fairness to the low traffic lanes. Our non-round robin algorithms deal with such situations by skipping such phases and allowing those time slots to be used by high traffic arrival phases so that delay is reduced effectively. In the rest of the section, we will formulate the TSC problem as an MDP and develop a decentralized framework to solve the MDP.

An MDP is a mathematical platform to study sequential decision making problems. In the case of the TSC problem, such decisions turn out to be the choice of the phase that needs to be set to green at a given time. In an MDP setting, we need to specify the states of the MDP, the actions that are feasible in each of the states of the MDP, the cost function for choosing an action in a particular state and the long term objective that needs to be minimized. In the TSC problem, we consider the long term objective as minimizing the average delay of the vehicles in the road network.

First we will describe the states for our TSC MDP. A state $s^j(t)$ for a given junction j at time t corresponds to a vector $s^j(t) = (q_1^j(t), q_2^j(t), ..., q_{L_j}^j(t), e_1^j(t), e_2^j(t), ..., e_{L_j}^j(t))^T$ of dimension $2L_j$, where L_j denotes the number of incoming lanes in that junction, $q_i^j(t),\ i \in \{1, 2 \ldots, L_j\}$ denotes the queue-length of the traffic in the ith lane at that junction and $e_i^j(t)$ denotes the elapsed time since the i^{th} lane turned red. Let S^j denote the set of all states at junction j. The state space of our TSC MDP is the cartesian product of the local state spaces of the junctions present in the network, i.e., $S \triangleq \bigtimes_{j=1}^{J} S^j$ (the cartesian product of the S^js). This poses a problem as the state space becomes large when there are many junctions in the network. We alleviate this problem by modeling each junction as a separate traffic signal controller (or agent). This is equivalent to saying that each agent observes only a portion of the total state space. Each agent then performs computations and controls the junction associated with it.

Now consider all possible states for a given junction after this decentralization step. For instance, consider a junction with 4 incoming lanes each of which accommodates 100 vehicles. The total possible states in that junction is at least 4×10^8. Note that we have not considered the time elapsed since red ($e_i^j(t)\ i \in \{1, 2 \ldots, L_j\}$) portion of the state vector for this computation. It is evident that the size of the state space of a junction is still considerably large even after reducing it to a decentralized setting. Hence, we quantize the queue lengths as well as the elapsed times. We consider three levels of discretization $\{lowQ = 0, mediumQ = 1\ and\ highQ = 2\}$ for the queue lengths and two levels of discretization for the elapsed times $\{lowE = 0,\ highE = 1\}$. This representation has two distinct advantages - the state space becomes manageable to be handled by the RL algorithms and the difficulty of obtaining precise measurements on queue lengths is alleviated. In order to discretize the queue lengths, we place two arrays of sensors at distances D_1 and D_2 along each lane. If the sensors at both D_1 and D_2 detect vehicles, we set the queue length to high. If only the sensors at D_1 detect vehicles then we set the queue length to medium and if none of the sensors detect vehicles, we set the queuelength to low. This discretization of queue lengths into three bins and elapsed times into two bins is given below:

$$\bar{q}_i^j(t) = \begin{cases} 0, & \text{if } q_i^j(t) < D_1, \\ 1, & \text{if } D_1 \leq q_i^j(t) \leq D_2, \quad i \in \{1, 2, \ldots, L\} \\ 2, & \text{if } q_i^j(t) > D_2. \end{cases}$$

$$\bar{e}_i^j(t) = \begin{cases} 0, & \text{if } e_i^j(t) \leq \bar{T}, \\ 1, & \text{if } e_i^j(t) > \bar{T}, \quad i \in \{1, 2, \ldots, L\} \end{cases} \tag{1}$$

Thus, the state vector for a given junction $j \in \{1, 2, \ldots, J\}$, $s^j(t) = (\bar{q}_1^j(t), \ldots, \bar{q}_{L_j}^j(t), \bar{e}_1^j(t), \ldots, \bar{e}_{L_j}^j(t))$ at time t is the vector of quantized queue lengths $\bar{q}_i^j(t),\ i \in \{1, \ldots, L_j\}$ on each of the incoming lanes and the vector of quantized elapsed times $\bar{e}_i^j(t)\ i \in \{1, \ldots, L_j\}$ of the incoming lanes.

We decide on the choice of the phase that needs to be set to green after every T seconds. The action $a^j(t)$ of the controller at junction j then corresponds to the choice of the phase to be set to green, i.e., $a^j(t) \in A_j \triangleq \{1, 2, \ldots, P_j\}$, where P_j as mentioned before corresponds to the total number of phases at junction j. Note that the same phase that has been currently set to green can be chosen as the valid next phase by the algorithm. A policy $\pi^j = \{\mu^j(t), t \geq 0\}$ followed at the junction j is a sequence of maps $\mu^j(t)$ from the state space to the action space such that when the state is $s^j(t)$ at instant t, $\mu^j(t)(s^j(t))$ specifies the phase that needs to be chosen as the current phase. We only consider stationary deterministic policies Π that do not change with time. We thus denote a policy π^j by $\pi^j = \{\mu^j, \mu^j, \ldots\}$, where $\mu^j(s^j(t))$ is the action in state $s^j(t)$. All actions here are assumed feasible in every state. The goal in our setting is to obtain policies for each of the junctions so that the long-term average delay of the road users is minimized.

The immediate cost $c^j(t)$ incurred by an agent at junction $j \in J$ for taking an action $a^j(t)$ when the current state is $s^j(t)$ and the next state is $s^j(t+1)$, is a convex combination of the cost due to the elapsed times and the cost due to the queue lengths at time $t + 1$ and is given by

$$c^j(t) = \gamma_1\big(\frac{1}{P_j}\sum_{k=1}^{L_j} \bar{e}_k^j(t+1))\big) + \gamma_2\big(\frac{1}{|N_j|}\sum_{k \in N_j}\sum_{i=1}^{L_k} \frac{\bar{q}_i^k(t+1)}{L_k}\big), \tag{2}$$

where $\gamma_1, \gamma_2 > 0$ such that $\gamma_1 + \gamma_2 = 1$ and N_j corresponds to the set consisting of neighboring junctions to junction j including itself. The first term in the R.H.S of (2) is the average elapsed time of the incoming lanes into junction j at time $t+1$. Inclusion of this component in the cost function ensures that none of the phases go on starvation. The second term in the R.H.S of (2) is the average queue length of the incoming lanes in the junction and the neighbors at time $t+1$. This component considers the effect of action $a^j(t)$ on the congestion level of the incoming lanes in its junction and its neighbors. Note that if $\gamma_1 < \gamma_2$ then more importance is given to minimizing the average delay without considering about the fairness for the low traffic phases. On the other hand, if $\gamma_1 > \gamma_2$ then we will compromise on our objective to minimize the delay to ensure fairness. We set the weights γ_1 and γ_2 to 0.5 each in our experiments.

The next section describes our optimal phase sequence multi-agent reinforcement learning algorithm along with its exploration strategies.

III. Algorithm

Q-learning is a stochastic approximation algorithm that learns the optimal policy for an MDP by experiencing the simulation samples online. In this algorithm, we update the Q-factors [17] associated with each state-action pairs. Q-factors defined for each policy provide a way to differentiate policies by measuring the effectiveness of actions in states under each policy. In our formulation, instead of having one single controller for the entire road network, we have multiple controllers at every junction each one of them controlling the traffic lights at its junction. Hence, in our algorithm, each controller updates its Q-factors based on the cost information obtained from its neighbors and itself.

The agent at junction j follows the following update rule: $\forall t \geq 0$,

$$Q_{t+1}^j(s^j(t), a^j(t)) = Q_t^j(s^j(t), a^j(t)) + \gamma(t)(c^j(t) + \alpha \min_{b \in A_j} Q_t^j(s^j(t+1), b) - Q_t^j(s^j(t), a^j(t)))\ , \tag{3}$$

where $s^j(t)$ and $s^j(t+1)$ correspond to the quantized queue-length vectors at the junction j at times t and $t+1$, respectively, when we set the current phase $a^j(t)$ at time t to one of the phases $\{1, 2, \ldots, P_j\}$ according to the ε-greedy or the UCB based exploration strategies. We initialize this update rule with $Q_0^j(s^j(0), a^j(0)) = 0,\ \forall\ s^j(0) \in S^j,\ a^j(0) \in A_j$. The cost $c^j(t)$ is computed from the neighbouring junctions as mentioned in (2). The step sizes $\gamma(t),\ t \geq 0$ in (3) satisfy the requirement, $\gamma(t) > 0\ \forall t$ and further,

$$\sum_t \gamma(t) = \infty, \sum_t \gamma(t)^2 < \infty. \tag{4}$$

The description of our optimal phase sequence multi-agent reinforcement learning algorithm is given below:

Algorithm 1 MARL algorithm for scheduling phases

```
1: for $time = 1$ to $\infty$ do
2:   if $time$ mod $T = 0$ then
3:     for $j = 1$ to $J$ do
4:       Collect the local and neighbor's traffic conditions to compute cost
5:       Update Q-factors for junction using (3)
6:       Set current phase using one of the exploration mechanisms
7:     end for
8:   end if
9:   $time = time - 1$
10: end for
```

In the case of ε-greedy exploration strategy, we choose the action $a^j(t) = \arg\min_c Q_t(s^j(t), c)$ with probability $1 - \varepsilon$. A small value of ε is required to explore the actions that have not been chosen by the algorithm.

In the case of UCB-based exploration strategy, the action selected is based on the upper confidence bound index. The index is constructed as a sum of two terms. The first term corresponds to the learnt Q-factor for the state-action pairs. The second term is inversely proportional to the number of times an action has been chosen in a given state. Note that the UCB algorithm setting for stateless MDP (the multi-armed bandit) is given in [3]. Here

$$a^j(t) = \arg\max_{c \in A} \left\{ -Q_t^j(s^j(t), c) + \sqrt{\frac{\ln R_{s^j(t)}(t)}{R_{s^j(t),c}(t)}} \right\}, \quad (5)$$

where $R_{s^j(t)}(t)$ denotes the number of times the state $s^j(t)$ has been visited till time t and $R_{s^j(t),c}(t)$ denotes the number of times the action c has been chosen in state $s^j(t)$ until time t. Note that during the start of the algorithm we need to select all actions once in each state. The second term helps in exploration. If action c has not been chosen sufficient number of times in state $s^j(t)$ then the second term will dominate the first term and action c will be explored. As learning progresses, the first term dominates the second term and the action selection is based only on the Q-factors.

For a single agent (junction) setting, the Q-learning update rule (3) with both the exploration strategies converges to the optimal policy (see [19]). In the multi-agent setting when we construct the immediate cost based on the neighbouring junctions (as mentioned in (2)), the Q-learning update rule yields a policy that has significantly better performance over both the FST and the SAT algorithms (see Section IV).

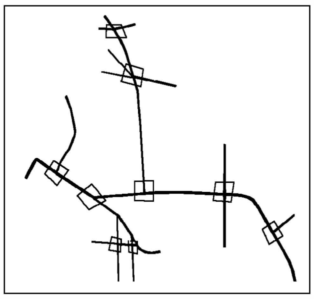
IV. Simulation results

We used the VISSIM traffic simulator for performance comparisons of the algorithms. VISSIM is a microscopic, time step and behaviour based simulation software developed to model urban traffic and public transit operations. VISSIM allows users to analyze private and public transport operations under constraints such as lane configuration, vehicle composition, traffic signals etc., thus making it a useful tool for the evaluation of various alternatives. It provides users a GUI to construct various road layouts and another interface to carry out simulations and obtain performance statistics.

Experimental setup: We considered the following two different settings for the road networks:

- A nine-junction road network (9Jn) - a real road network around the Indian Institute of Science campus in Bangalore, India.
- A twenty-junction road network (20Jn)- a real road network in Bangalore, India.

TABLE I
Different road network layouts


(a) Nine-Junction Network

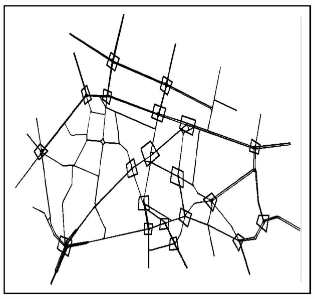

(b) Twenty-Junction Network

Table I gives the two different road geometries we considered in our experiments. In all our experiments, we take the decisions after $T = 10$ seconds. The average number of vehicles entering the 9Jn and the 20Jn road networks per hour were set to $7,400$ and $10,050$ respectively. In the plots shown, the X-axis corresponds to the iterations of the algorithm and the Y-axis corresponds either to the average delay experienced by the road users or the average stopped delay. Note that the average delay of the vehicles is computed as the average difference between (a) the time taken for a vehicle to travel from the source to the destination when there is no traffic and (b) the same when there is traffic. Similarly, average stopped delay is computed as the average stand still time per vehicle near junctions.

In the FST algorithm, the time durations of the various phases have been chosen to be the best (in terms of average delay) over a range of fixed timing values. In the SAT algorithm, we have set the saturation level to 90% and the minimum phase duration to 20 seconds. Also, we computed the (parameter) maximum cycle length as in [16] by using the factor parameter set to 2. We refer the reader to [14] for a complete description of the SAT algorithm. The MARL algorithm in [11] with UCB-based exploration given is denoted as Q-UCB-RR and with ε-greedy exploration is denoted as Q-ε-greedy-RR. We similarly denote our non-round-robin algorithm with UCB-based exploration Q-UCB and with ε-greedy exploration as Q-ε-greedy. We set the discount factor to $\alpha = 0.9$ and the exploration parameter in the Q-ε-greedy algorithm to $\epsilon = 0.1$, respectively. We set the

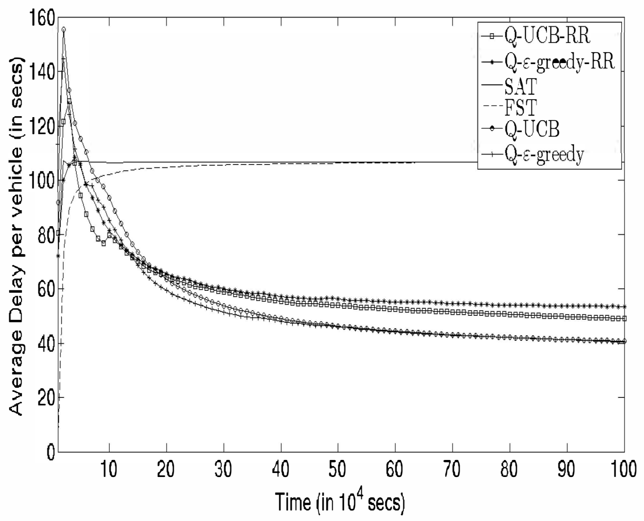


Fig. 1. Average delay performance comparison for the nine junction road network

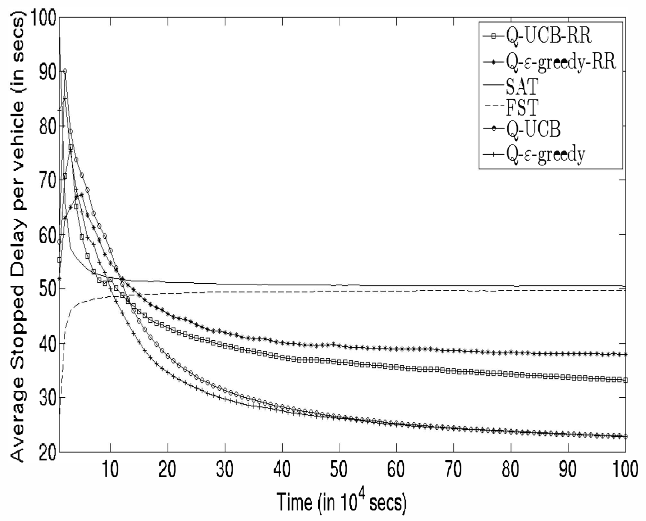


Fig. 3. Average stopped delay performance comparison for the nine junction road network

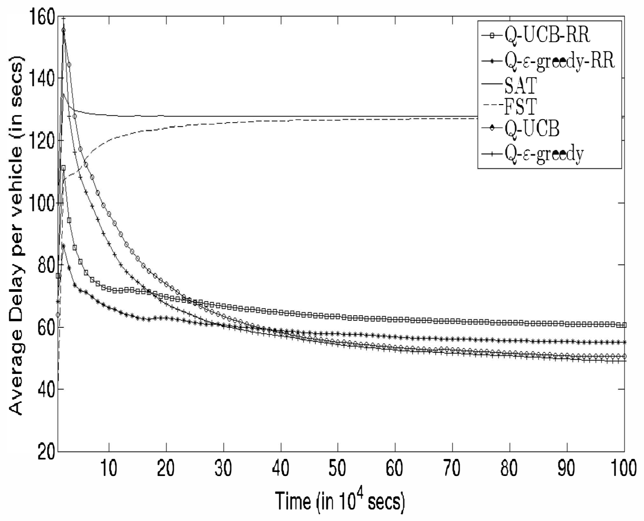


Fig. 2. Average delay performance comparison for the twenty junction road network

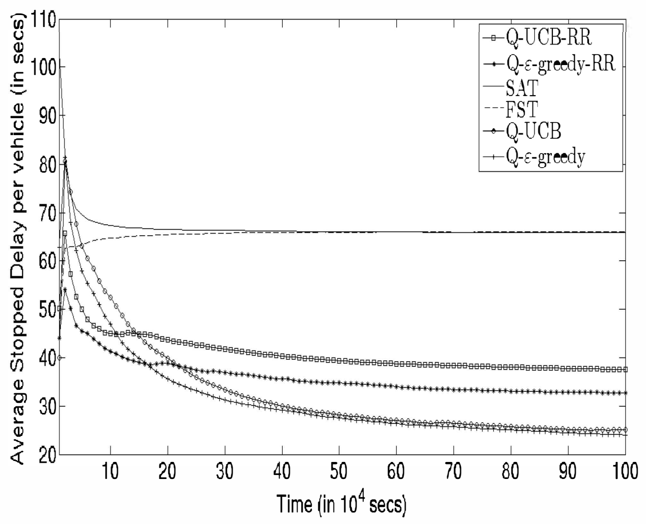


Fig. 4. Average stopped delay performance comparison for the twenty junction road network

step-size $\gamma = 0.1$ for the first 10^5 seconds and then gradually decrease it as $\frac{C}{n}$, for $n > 10^5$ seconds where $C = 10^4$. Note that the chosen step-sizes satisfy (4).

From Figs. 1 and 2, it can be seen that our algorithms exhibit significantly better performance over the FST and SAT algorithms in terms of average delay in both the road network configurations. Note that both of our algorithms also perform better than Q-UCB-RR and Q-ε-greedy-RR algorithms. In the initial stages of learning, our algorithms have higher average delay compared to other algorithms. However, as learning progresses, the average delay of the vehicles is greatly reduced after 3×10^5 seconds. Figs. 3 and 4 demonstrate the effectiveness of the dynamic TSC policies obtained by our MARL algorithms in terms of the average stopped delay. The fixed timing algorithm due to its static nature causes a large delay for the road users. The SAT algorithm despite being dynamic in nature is unable to adapt quickly to the underlying traffic. Our algorithms, on the other hand, by utilizing the coarse congestion level and elapsed time information adapt to the traffic condition to decide on the right choice of the phase. Note that in the plots the performance of the Q-UCB and Q-ε-greedy algorithms is close.

We have summarized the performance indices like average stopped delay, average number of stops, and average delay of all the algorithms in Table II. The entries in the first column in Table II correspond to the road network as well as the algorithm used. For instance, 9JnQ-UCB-RR corresponds to the 9Jn road network using the MARL algorithm in [11] with UCB based exploration. It can be easily seen from the table that our algorithms perform significantly better as compared to the FST and the SAT algorithms. Also, the performance of our algorithms is better than the round robin MARL algorithms. For instance, in the twenty junction scenario, we get approximately 33% and 26% reduction in average delay for Q-UCB and Q-ε-greedy algorithms compared to the round-robin counterparts in [11]. Note that in terms of the average number of stops Q-UCB-RR and Q-ϵ-greedy have slightly better performance compared to our algorithms for the 9 junction road network. The reason for this could be attributed to the non-round robin nature of our algorithms.

	Average stopped delay [s]	Average delay [s]	Average number of stops
9JnFST	49.69	106.73	8.34
9JnSAT	50.51	106.67	7.95
9JnQ-ε-greedy-RR	37.95	53.47	1.99
9JnQ-UCB-RR	33.17	48.96	2.05
9JnQ-ε-greedy	22.69	40.26	2.30
9JnQ-UCB	22.81	40.69	2.27
20JnFST	65.99	109.34	18.53
20JnSAT	65.45	109.45	20.67
20JnQ-ε-greedy-RR	32.68	54.98	2.42
20JnQ-UCB-RR	37.53	60.72	2.54
20JnQ-ε-greedy	23.99	48.95	2.18
20JnQ-UCB	25.01	50.04	2.26

TABLE II

AVERAGE PERFORMANCE STATISTICS COMPARISONS FOR THE TSC ALGORITHMS FOR TWO ROAD NETWORK LAYOUTS

V. CONCLUSION

We formulated the TSC problem of obtaining the optimal order of phase sequence as an MDP and applied the Q-UCB and Q-ϵ-greedy algorithms. Our algorithms are seen to converge to policies that significantly outperform the FST, SAT and MARL algorithms in [11]. Our algorithms work with coarse queue-length information and schedule phases in a non-round robin manner. Also, our algorithms are simple to implement and scalable to large road networks. Q-UCB and Q-ϵ-greedy show nearly similar performance on both road networks. Our algorithms do not require the model parameters of the traffic environment and the policies are learnt online. In our algorithms, we discretized and fixed our elapsed times, queue lengths and the time duration after which the decisions are made. One could tune these discrete parameters on a slower timescale [13] for optimal performance using DSPSA [18] algorithm. Our algorithms could also be extended by including eligibility traces [17] for faster convergence. In addition to decentralized learning one could utilize function approximation based techniques with suitable basis functions for road networks with many junctions. We assumed zero delay in communication between junctions, however, in practice this may not be true, so one could develop algorithms by taking these delays into consideration.

ACKNOWLEDGEMENT

This work was supported in part through projects from Department of Science and Technology (DST) and the Robert Bosch Centre, IISc.

REFERENCES

[1] M. Abdoos, N. Mozayani, and A.L.C. Bazzan. Traffic light control in non-stationary environments based on multi agent q-learning. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1580–1585. IEEE, 2011.

[2] I. Arel, C. Liu, T. Urbanik, and A.G. Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *Intelligent Transport Systems, IET*, 4(2):128–135, 2010.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[4] B. Bakker, S. Whiteson, L. Kester, and F. Groen. Traffic light control by multiagent reinforcement learning systems. *Interactive Collaborative Information Systems*, pages 475–510, 2010.

[5] A.L.C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375, 2009.

[6] C. Cai, C.K. Wong, and B.G. Heydecker. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 17(5):456–474, 2009.

[7] S. El-Tantawy and B. Abdulhai. An agent-based learning towards decentralized and coordinated traffic signal control. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 665–670. IEEE, 2010.

[8] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. *Machine Learning and Knowledge Discovery in Databases*, pages 656–671, 2008.

[9] J.C. Medina and R.F. Benekohal. Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 596–601. IEEE, 2012.

[10] Markos Papageorgiou. Review of road traffic control strategies. *Proceedings of the IEEE*, 91:20432067, 2003.

[11] K.J. Prabuchandran, A.N. Hemanth Kumar, and Shalabh Bhatnagar. Multi-agent reinforcement learning for traffic signal control. In *Intelligent Transportation Systems (ITSC), 2014 17th International IEEE Conference on*. IEEE, accepted, 2014.

[12] L.A. Prashanth and S. Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421, 2011.

[13] L.A. Prashanth and S. Bhatnagar. Threshold tuning using stochastic optimization for graded signal control. *Vehicular Technology, IEEE Transactions on*, 61(9):3865–3880, 2012.

[14] S Richter. Learning traffic control-towards practical traffic control using policy gradients. *Albert-Ludwigs-Universitat Freiburg, Tech. Rep*, 2006.

[15] D. Robertson. Transyt: a traffic network study tool. *Road Research Laboratory Crowthorne*, 1969.

[16] A. Salkham, R. Cunningham, A. Garg, and V. Cahill. A collaborative reinforcement learning approach to urban traffic control optimization. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 2, pages 560–566. IEEE, 2008.

[17] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998.

[18] Qi Wang and James C Spall. Discrete simultaneous perturbation stochastic approximation on loss function with noisy measurements. *Proceedings of the American Control Conference, San Francisco*, 29:4520–4525, 2011.

[19] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.