Asynchronous Distributed Reinforcement Learning for Traffic Signal Control[1]

Shalabh Bhatnagar

CSA Department

Joint Affiliation: RBCCPS and CiSTUP

IISc

December 16, 2016

[1] Joint work with Prabuchandran K.J and Hemanth Kumar H.N

Outline of the Presentation

- Introduction and setting
- Basic Framework
- Our algorithm
- Numerical results
- Conclusions and Future work

Traffic Signal Control (TSC) problem

- The problem - intelligently control traffic lights at junctions to reduce the average delay experienced by the road users
- Possible control scenarios:
  - A Centralized control[2][3]
  - B Decentralized control[4]
- We control the green time duration of phases by fixing an order for the phase sequence.

[2] Prashanth L.A and S.Bhatnagar, *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412-421, 2011

[3] Prashanth L.A and S.Bhatnagar, *IEEE Transactions on Vehicular Technology*, 61(9):3865-3860, 2012

[4] Prabuchandran K.J, Hemanth Kumar A.N, and S.Bhatnagar, *IEEE ITSC*, pp.2529-2534, 2014

Problem description

TSC Problem

Find the optimal green signal duration of phases at each intersection to minimize average delay of road users

Existing approaches

- Fixed time signal control - Not adaptive
- Manual signal control - Locally good actions
- Automated adaptive signal control - Learning approaches

Available approaches

- Fixed Timing Control : Signal times calculated offline using traffic characteristics at different times of day[5]
- Adaptive Signal Controls :
  - Model Based : Use mathematical models for traffic dynamics such as fluid patterns of traffic[6]
  - Model Free : Reinforcement Learning algorithms[7]

[5] M.Papageorgiou, *Proceedings of the IEEE*, 91:2043-2067, 2003

[6] N.H.Gartner, C.J.Messer, and A.K.Rathi, *U.S. Department of Transportation, Transportation Research Board*, 1992

[7] B.Abdulhai, R.Pringle and G.Karakoulas, *Journal of Transportation Engineering*, 129:278-290, 2003

Available Information

- We assume that some information about traffic congestion across various lanes is available through sensors
- The available information need not be precise but could be 'coarse' in nature
- We assume the phase-switching in an intersection happens in a round robin manner
  - Ensures fairness
  - Ideally a passenger would like to know which signal will turn green next

Centralized Control Scenario

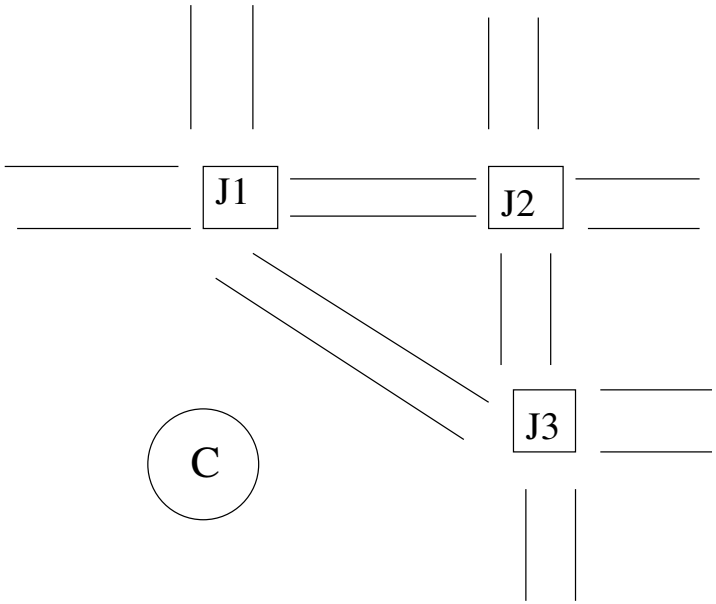


Figure: A Centralized Control Setting – Single Controller

- Controller receives congestion information from all junctions
- Post computation sends feedback on signal configurations to each junction

Decentralized Control Setting

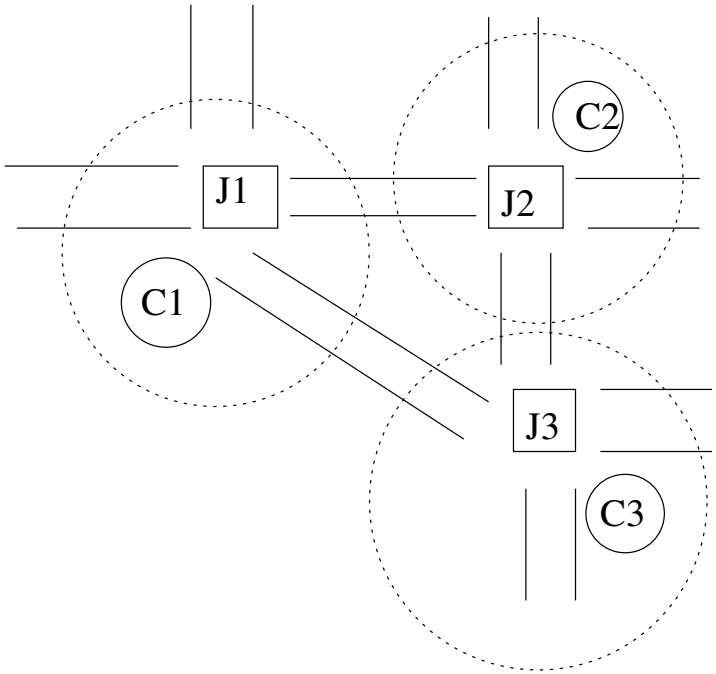


Figure: A Decentralized Control Setting – Multiple Controllers

- Each controller makes decisions using local state information
- Information exchange across junctions factored in local computations

Our approach

- We formulate TSC problem as a co-operative multi-agent system whose goal is to minimize average delay
- Each junction is modelled as an agent and each agent handles the state space associated with it
- Each agent uses Q-learning with either ε-greedy[8] or UCB based[9] exploration
- The feedback cost signal to an agent is obtained from itself and its neighbours

[8] R.Sutton and A.Barto, *Reinforcement Learning*, Cambridge University Press, 1998

[9] P.Auer, N.Cesa-Bianchi, and P.Fischer, *Machine Learning*, 47(2):235-256, 2002

The Setting

Multiagent asynchronous learning with each agent (junction) exchanging congestion information with its one-hop neighbors

Design Issues

- Modeling the problem of optimizing signal duration of phases in a junction with real time traffic - MDP Framework
- Low response time - Coarse information on traffic conditions
- Scalability of the approach - Distributed learning

MDPs[10] and RL[11]

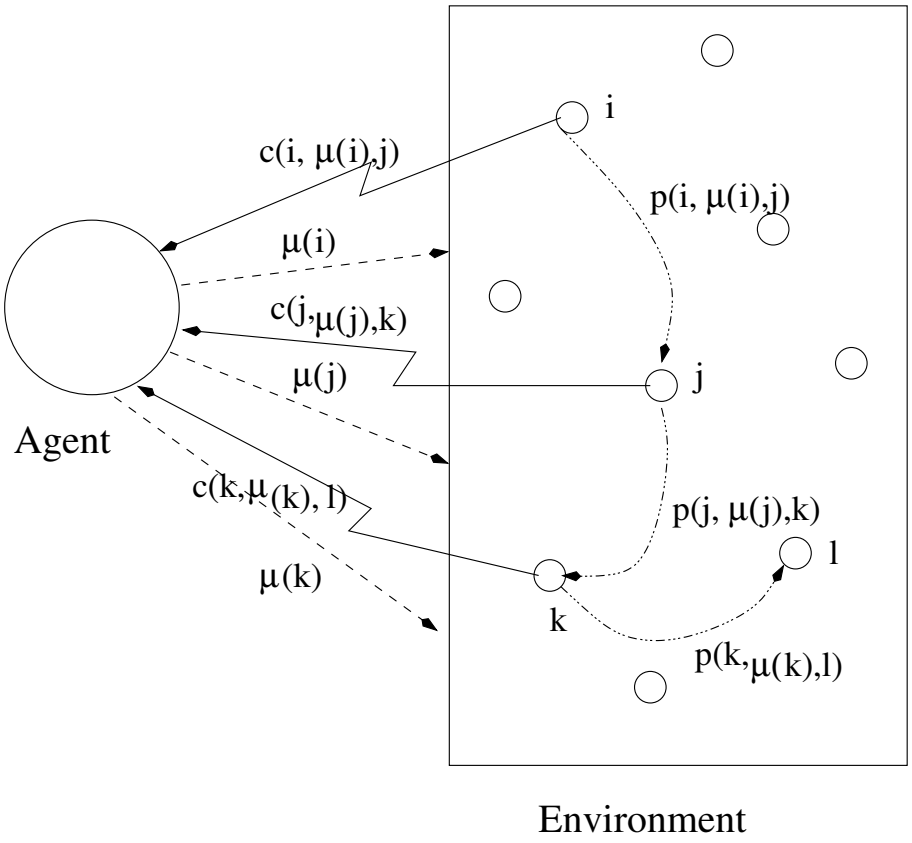


Figure: Agent-Environment Interaction

[10] M.L.Puterman, *Markov Decision Processes*, Wiley, 1994

[11] D.P.Bertsekas and J.N.Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996

The Goal

- Find a policy μ^* that gives the optimal cost

$$V^*(i) \equiv V_{\mu^*}(i) \stackrel{\triangle}{=} \min_{\mu} E\left[\sum_{j=0}^{\infty} \gamma^j c(s_j, \mu(s_j), s_{j+1}) \mid s_0 = i\right], i \in S$$

- V^* satisfies the Bellman equation

$$V^*(i) = \min_{a \in A(i)} \sum_j p(i,a,j)(c(i,a,j) + \gamma V^*(j))$$

The Q-Value Function

- Let $Q^*(i,a)$ be defined according to

$$Q^*(i,a) = \sum_j p(i,a,j)(c(i,a,j) + \gamma V^*(j))$$

- Then

$$V^*(i) = \min_{a \in A(i)} Q^*(i,a)$$

- Moreover,

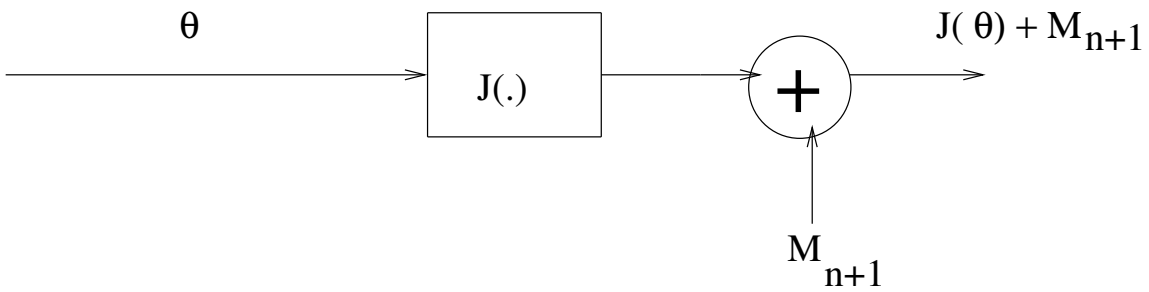
$$Q^*(i,a) = \sum_j p(i,a,j)(c(i,a,j) + \gamma \min_{b \in A(j)} Q^*(j,b))$$

- The above is also known as the Q-Bellman equation

Stochastic Approximation[12]

- Objective: Solve the equation $J(\theta) = 0$ when analytical form of J is not known, however, noisy measurements $J(\theta(n)) + M_{n+1}$ can be obtained

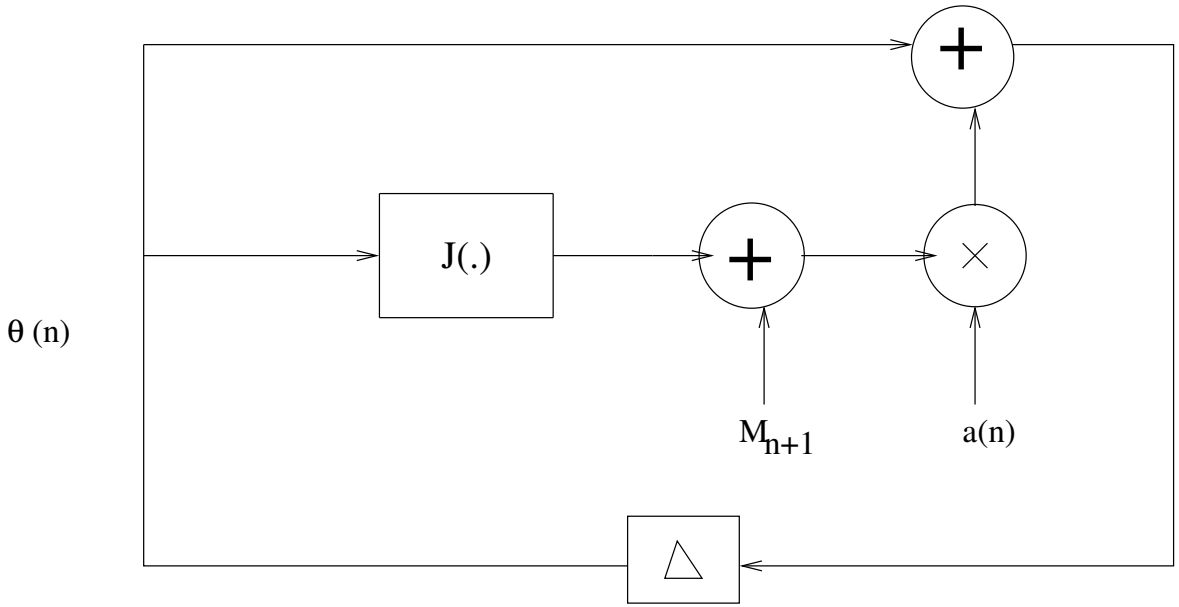
- Here M_n, $n \geq 0$ constitutes the 'noise', zero-mean and uncorrelated with the past



[12] H.Robbins and S.Monro *Annals of Mathematical Statistics*, 22: 400–407, 1951

The Robbins-Monro Algorithm

- Algorithm $\theta(n+1) = \theta(n) + a(n)(J(\theta(n)) + M_{n+1})$



- $a(n) = \frac{1}{n}, \ \frac{1}{n^{\alpha}}, \alpha \in (0.5, 1], \ \frac{\log(n)}{n}, \ \frac{1}{n\log(n)}$ etc.

More General Recursions

- Consider the recursion

$$\begin{aligned} \theta(n+1) &= \theta(n) + a(n)h(\theta(n), \mu(n)) \\ &= \theta(n) + a(n)(J(\theta(n)) + M_{n+1}), \end{aligned} \tag{1}$$

  where $J(\theta(n)) = E[h(\theta(n), \mu(n)) \mid \theta(n)]$

- Under some assumptions, algorithm has the same asymptotic behaviour as

$$\dot{\theta}(t) = J(\theta(t))$$

- Important instances:
  - Fixed point iteration: $J(\theta) = f(\theta) - \theta$
  - Stochastic gradient scheme: $J(\theta) = -\nabla f(\theta)$

An SA Algorithm for Q-Bellman Equation

- Recall the Q-Bellman equation

$$Q^*(i,a) = \sum_j p(i,a,j)(c(i,a,j) + \gamma \min_{b \in A(j)} Q^*(j,b))$$

- Let
  (1) $\theta = Q(i,a)$
  (2) $J(\theta) = \sum_j p(i,a,j)(c(i,a,j) + \gamma \min_{b \in A(j)} Q(j,b)) - Q(i,a)$

- Goal: Find $\theta^* \stackrel{\triangle}{=} Q^*(i,a)$ such that $J(\theta^*) = 0$

Q-Learning[13]

- An iterative RL algorithm which solves Q-Bellman equation using following update rule:

$$Q_{n+1}(i,a) = Q_n(i,a) + a(n)(c(i,a,j) + \gamma \min_{b \in A(j)} Q_{n+1}(j,b) - Q_n(i,a))$$

$$= Q_n(i,a) + a(n)(J(\theta(n)) + M_{n+1})$$

where $a(n)$: step size at time 'n'

[13] C.J.C.H.Watkins and P.Dayan, *Machine Learning*, 8(3):279-292, 1992

Q-ODE

- ODE associated with Q-learning:

$$\dot{Q}(i,a) = \sum_j p(i,a,j)(c(i,a,j) + \gamma \min_{b \in A(j)} Q(j,b)) - Q(i,a)$$

- The above ODE has $Q^*(i,a)$ as its unique globally asymptotically stable equilibrium
- Then $Q_n(i,a) \to Q^*(i,a)$ almost surely as $n \to \infty$
- *Problem*: The procedure is computationally expensive and requires huge memory

Multi-Agent Algorithms[14]

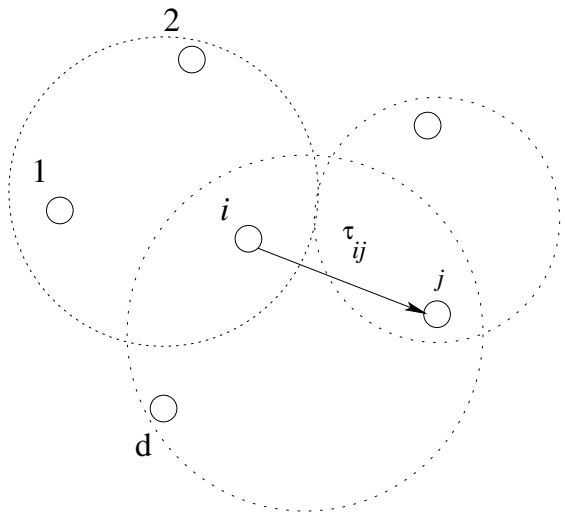


Figure: Asynchronous Updates

- Let $\theta_i(n)$ be nth parameter update of ith processor
- Each processor updates its parameter component on its own 'local' clock

[14] S.Bhatnagar, *Systems and Control Letters*, 60:472-478, 2011

Stability and Convergence

- The Distributed Stochastic Update: For $i = 1, \ldots, d$,

$$\theta_i(n+1) = \theta_i(n) + a(\nu(i,n))I\{i \in Y_n\}$$

$$(J_i(\theta_{1-\tau_{i1}(n)}(n), \ldots, \theta_{d-\tau_{id}(n)}(n)) + M_{n+1}(i))$$

- Here $\nu(i,n) = \sum_{m=0}^{n} I\{i \in Y_m\}$ and Y_n is the (random) subset of components updated at time n
- Key requirement: $\liminf_{n\to\infty} \frac{\nu(i,n)}{n} > 0$ for all i
- Key result: $\sup_n \| \theta(n) \| < \infty$ almost surely and $\{\theta(n)\}$ asymptotically tracks the ODE

$$\dot{\theta}(t) = \Lambda(t)J(\theta(t))$$

TSC as MDP

State space of MDP

- L_j - number of incoming lanes to a junction j.
- State of junction j at time 't' is defined as a vector

$$s^j(t) = (q_1^j(t), q_2^j(t), ..., q_{L_j}^j(t), p^j(t))$$

$q_i^j(t)$ - queue length in the i^{th} lane at time t
$p^j(t)$ - index of the phase that has to be set to green

- State space of the network: $S \triangleq \times_{j=1}^{J} S^j$
  J - total number of junctions and S^j state space of junction j
- This poses a problem as state space increases rapidly with number of junctions in the network

Solution: Decentralization

State space

- We consider each junction as a separate learning agent
- Each agent observes only a portion of the state space
- This consideration improves scalability

State Space Issues

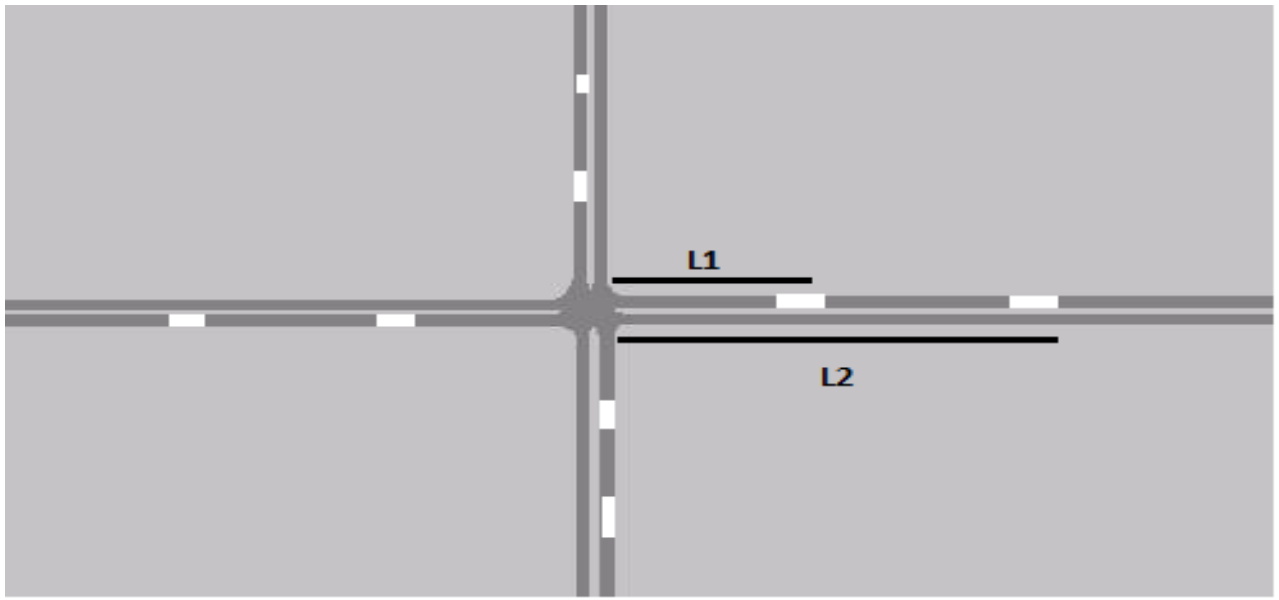
State space of a junction

- Consider a junctions with 4 incoming lanes, each of which accommodates 100 vehicles
- Total possible states for the junction - $4 * 10^8$
- State space for a junction is still considerably large

Solution: State Discretization

Segregate the queue-length in a lane as {low=0, medium=1, high=2} using sensors

$$q_i^j(t) = \begin{cases} low, & \text{if } q_i^j(t) < L1 \\ medium, & \text{if } L1 \leq q_i^j(t) \leq L2 \\ high, & \text{if } q_i^j(t) > L2 \end{cases}$$



Action space and Cost function

- **Actions** :
  - Action a_j of agent j in state s_j is to decide the green light duration of current phase
  - The action set is $\{low = 10s, medium = 20s, high = 30s\}$ same for all states

- **Cost Function** : Cost $c_j(t)$ at time t incurred by agent $j \in J$ for taking an action a_t^j at state s_t^j :

$$c_j(s_t^j, a_t^j, s_{t+1}^j) = \frac{1}{|N_j|} \sum_{k \in N_j} \sum_{i=1}^{L_k} q_i^k(t+1), \tag{2}$$

N_j - set including itself and neighbors of junction j

Multiagent Reinforcement-Learning (MARL) Algorithm

- Each agent updates its Q-factor using the Q-learning update rule
- Agent at junction j follows :

$$Q^j_{n+1}(s^j, a^j) = Q^j_n(s^j, a^j) + a(n)\left(c_j(n) + \gamma \min_{b \in A} Q^j_n(s^{j'}, b) - Q^j_n(s^j, a^j)\right),$$

s^j, $s^{j'}$ - queue-length vectors at junction j at times n and $n+1$

action a^j at time n is set according to one of the exploration mechanisms (ε-greedy exploration or UCB exploration)

- The cost feedback signal $c_j(n)$ is obtained from neighbours for taking action a^j

Exploration mechanisms

ε-greedy exploration strategy

With probability $1-\varepsilon$, choose action $a^j = \arg\min_c Q_n(s^j, c)$ and with probability ε, choose random feasible action a^j in state s^j

Exploration mechanisms

UCB-based exploration strategy

Compute action a^j based on upper confidence bound index

$$a^j = \arg\max_{c \in A} \left\{ -Q_n^j(s^j, c) + \sqrt{\frac{\ln R_{s^j}(n)}{R_{s^j,c}(n)}} \right\}, \tag{3}$$

$R_{s^j}(n)$ - Number of visits to state s^j until time n

$R_{s^j,c}(n)$ - Number of times action c is chosen in state s^j until time n

If action c is not chosen sufficient number of times in state s^j then second term will dominate and action c will be explored. With learning, first term dominates and action is selected only based on Q-factors.

MARL algorithm for Round Roubin Scheduling

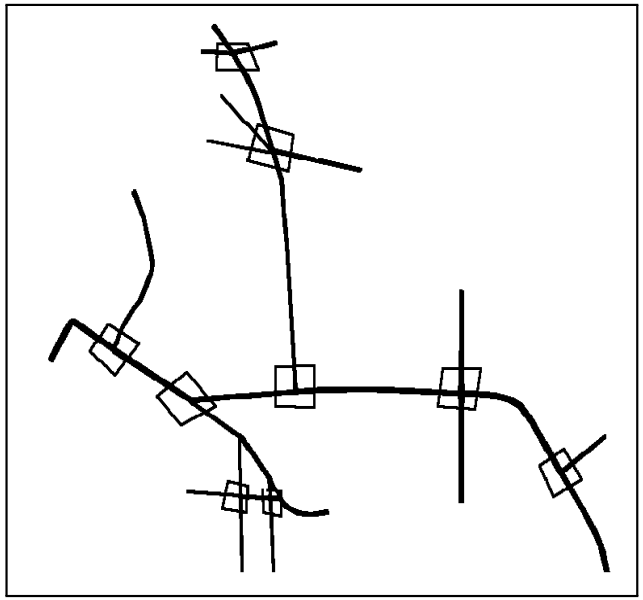
Algorithm 1

for $time = 1$ to ∞ **do**
  for $j = 1$ to $number\_of\_junctions$ **do**
    if $green\_time(j) = 0$ **then**
      a. $curr\_phase(j) = (curr\_phase(j)+1)$ mod $number\_of\_phases(j)$
      b. Set $green\_time(j)$ using one of the exploration mechanisms
      c. Collect local and neighbor's traffic conditions to compute cost
      d. Update Q-factors for junction j using Q-learning update rule
    end if
    $green\_time(j) = green\_time(j) - 1$
  end for
end for

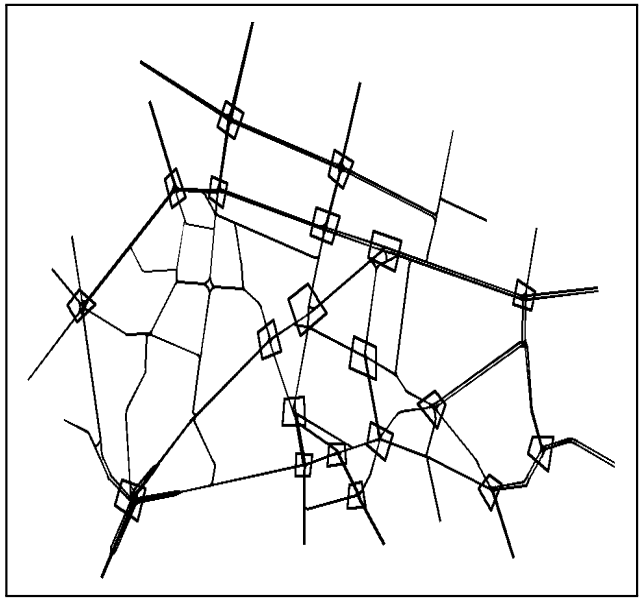
Experimental Setup

- Simulations are run on VISSIM, a microscopic traffic simulator
- Two different 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.

Road Network Layouts



(a) Nine-Junction Network



(b) Twenty-Junction Network

Table: Different road network layouts

Parameter Settings

- Our action set is $\{low = 10s,\ medium = 20s,\ high = 30s\}$
- Traffic conditions for the road network

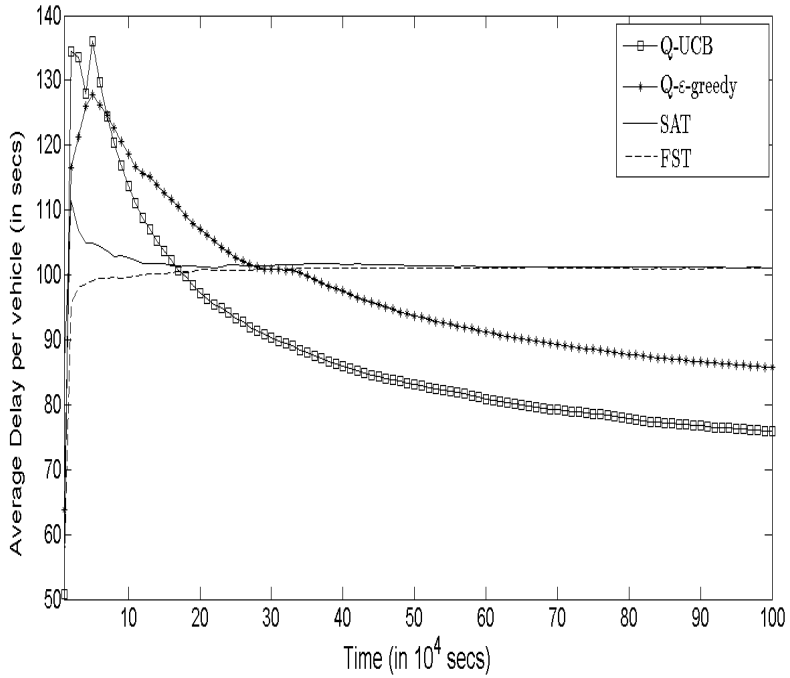
Road network	Arrival rate per hour
9 Jn	5500
20 Jn	9650

Algorithms Considered

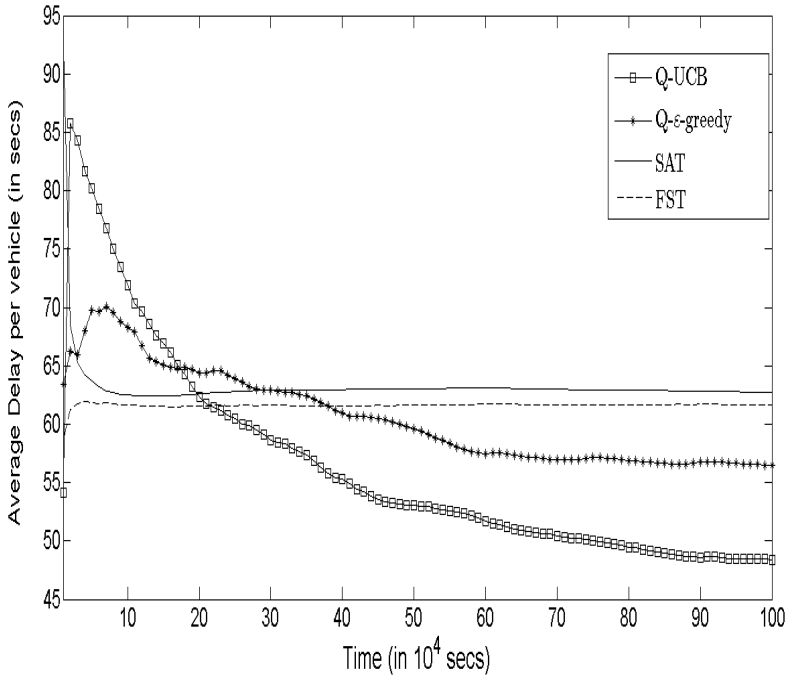
- We compare performance of our MARL algorithms with Fixed Signal Timing (FST) and Saturation Balancing (SAT)[15] algorithms
  - FST algorithm - Signal timing tuned for best performance
  - SAT algorithm - an adaptive algorithm that controls signal timings based on the vehicle density during last cycle
- Performance metrics - average delay and average stopped delay

[15] S.Richter, D.Aberdeen and J.Yu, *NIPS*, 2007

Performance Comparison - Average Delay

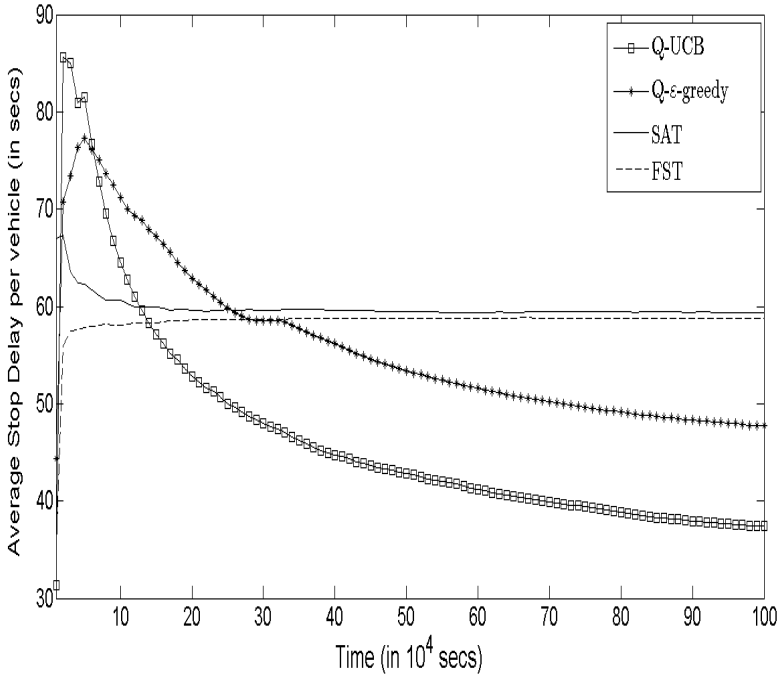


(a) 20 Jn road network

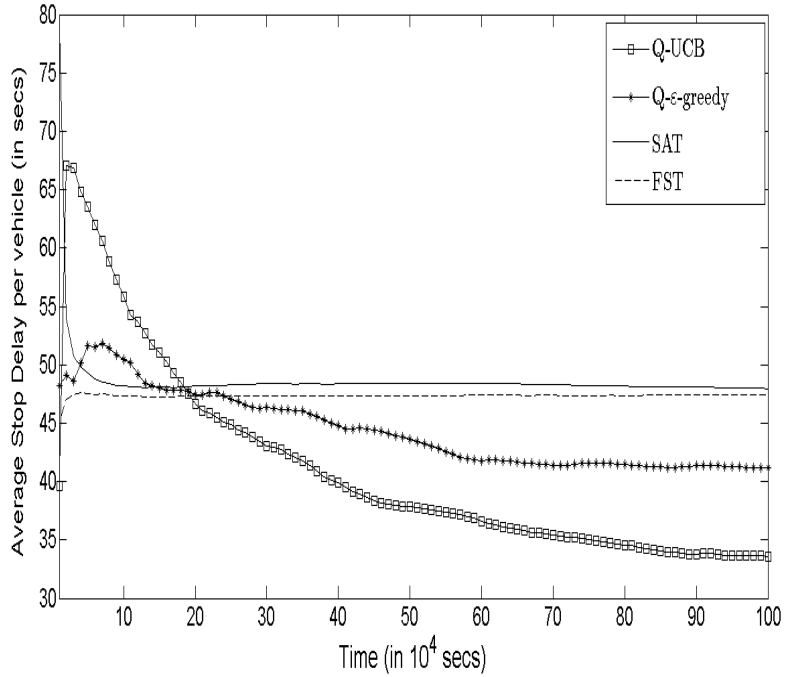


(b) 9 Jn road network

Performance Comparison - Average Stopped Delay



(a) 20 Jn road network



(b) 9 Jn road network

Performance Statistics

	Average stopped delay [s]	Average delay [s]	Average number of stops
9JnFST	47.40	61.72	2.14
9JnSAT	47.92	62.74	2.11
9JnQ-ε-greedy	41.13	56.46	1.82
9JnQ-UCB	33.60	48.41	1.63
20JnFST	58.78	101.06	4.03
20JnSAT	59.30	101.10	4.01
20JnQ-ε-greedy	47.61	85.74	3.56
20JnQ-UCB	37.31	75.82	3.04

Table: Average performance comparisons for the road network layouts

Future work

- The threshold parameters in the policies can be tuned adaptively for optimal performance[16]
- Non-zero communication delays between junctions need be considered
- Function approximation approaches can be explored together with asynchronous distributed techniques
- Signalling for pedestrian movement
- Modeling driver behaviour

[16] Prashanth L.A and S.Bhatnagar, *IEEE Transactions on Vehicular Technology*, 61(9):3865-3880, 2012