Threshold Tuning Using Stochastic Optimization for Graded Signal Control

Article *in* IEEE Transactions on Vehicular Technology · November 2012

DOI: 10.1109/TVT.2012.2209904

CITATIONS
21

READS
122

2 authors:

Prashanth L.A.
University of Maryland, College Park
50 PUBLICATIONS **591** CITATIONS
SEE PROFILE

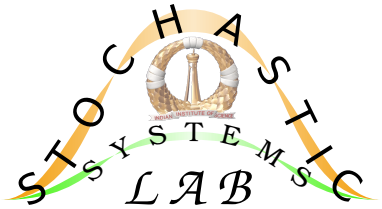
Shalabh Bhatnagar
Indian Institute of Science
254 PUBLICATIONS **2,784** CITATIONS
SEE PROFILE

Threshold tuning using stochastic optimization for graded signal control

Prashanth L. A.[†] S. Bhatnagar[†]

IISc-CSA-SSL-TR-2012-1

http://stochastic.iisc.ernet.in/www/research/files/IISc-CSA-SSL-TR-2012-1.pdf



Stochastic Systems Lab
Computer Science and Automation
Indian Institute of Science, India

April 2012

Threshold tuning using stochastic optimization for graded signal control

Prashanth L. A.[†] S. Bhatnagar[†]

April 25, 2012

Abstract

Many traffic light control (TLC) algorithms are based on graded threshold type policies. However, these algorithms incorporate fixed thresholds that are in general not optimal for all traffic conditions. We present for the first time, an algorithm based on stochastic optimization to tune the associated thresholds. We also present three new TLC algorithms - a full state Q-learning algorithm with state aggregation, a Q-learning algorithm with function approximation involving a novel feature selection scheme and a priority based TLC scheme, respectively. Next, we combine the threshold tuning algorithm with the three aforementioned algorithms. This combination results in several interesting consequences. For instance, in the case of the Q-learning with full state representation, our threshold tuning algorithm gives an optimal way of clustering of states so as to reduce the cardinality of the state space and in the case of the Q-learning algorithm with function approximation, see [1], our (threshold tuning) algorithm provides a novel feature adaptation scheme in order to obtain an 'optimal' selection of features. Our tuning algorithm is online, incremental with proven convergence to the optimal values of thresholds. Further, the additional computational effort due to the integration of the tuning algorithm in a graded threshold based TLC algorithm is minimal. Simulation results show a significant gain in

[†]Department of Computer Science and Automation, Indian Institute of Science {prashanth, shalabh}@csa.iisc.ernet.in

performance when our threshold tuning algorithm is used in conjunction with various TLC algorithms as compared to the TLC algorithms themselves, i.e., without tuning and with fixed thresholds.

Keywords: Threshold tuning, stochastic optimization, simultaneous perturbation stochastic approximation, traffic signal control, intelligent transportation systems.

1 Introduction

Adaptive control of traffic lights forms an integral part of the design of any intelligent transportation system. Information about the queue lengths on the lanes of the road network is a necessary input for many adaptive traffic light control (TLC) algorithms. However, in practice, this information is hard to obtain with precision, but instead certain thresholds on the queue lengths can be used to classify the traffic condition. Thresholds may also be used on the elapsed time, i.e., the amount of time since the signal turned red on any lane. In essence, the TLC algorithm could consider switching a lane to green if either the queue length or the elapsed time exceed certain prescribed thresholds. Designing a TLC algorithm that maximizes the traffic flow in the long term across various junctions in a road network, is a challenging problem. This is because of the variability in traffic patterns and lack of precise state information. Many times, one only has a coarse knowledge of the level of congestion on a lane, such as *low*, *medium* or *high*. Such information can be better obtained using graded or layered feedback policies. For instance, for some suitably chosen threshold levels L_1 and L_2 for each lane with $L_1 < L_2$, one could mark congestion as being in the low range if the queue length on that lane is below L_1. On the other hand, if the queue length is between L_1 and L_2, congestion could be said to be in the medium range, while if it is above L_2, it could be inferred to be high. In such (graded threshold based) policies, however, the choice of the thresholds (such as L_1 and L_2) plays a critical role and a problem is to select such thresholds optimally. Graded threshold policies have been considered for instance in [1]. These policies however consider fixed values for the thresholds. A problem of interest is to find optimal thresholds for such classes of feedback policies.

We consider the problem of designing new TLC algorithms with graded thresholds that are however set optimally. For obtaining optimal thresholds, we develop an algorithm that works in conjunction with the given TLC

algorithm and tunes the thresholds in order to optimize a given cost objective. The problem can be seen to be equivalent to one of finding an optimal feedback policy within a given class of parameterized feedback policies with the underlying parameter in general being a vector of the various thresholds on which the policies depend. In a stochastic dynamic setting as we consider, it is necessary to design an online algorithm to tune the thresholds on queue lengths and/or elapsed times and thereby tune the parameter of the associated feedback policy. The threshold tuning algorithm should be easily implementable, have the necessary convergence properties and most importantly, work for any graded threshold based TLC algorithm and in general for any parameterized class of policies.

In this paper, (a) we design an efficient Simultaneous Perturbation Stochastic Approximation (SPSA) based online threshold tuning algorithm that works with any graded threshold based TLC algorithm, and (b) we propose three new TLC algorithms - a variant of Q-learning with full state representation that incorporates a novel state aggregation scheme, a function approximation based Q-learning TLC algorithm that enhances the scheme proposed in [1] by incorporating a new feature selection procedure, and a novel priority based algorithm. We study our threshold tuning algorithm in conjunction with these TLC algorithms. We describe our contributions in the paragraphs below.

The SPSA based online threshold tuning algorithm that we design here falls in the general category of simulation-based optimization - a collection of methods that do not require a priori knowledge or assumption on the traffic system dynamics. Further, our algorithm has the added advantage that it is easily implementable, possesses the necessary convergence properties and works with any graded threshold based TLC algorithm regardless of the network and traffic conditions. Specifically, we incorporate the one-simulation variant of the SPSA algorithm that uses Hadamard matrix based deterministic perturbations from [2] to tune the graded thresholds used in the sign configuration policy. To the best of our knowledge, we are the first to explore threshold tuning for different classes of graded TLC algorithms. We study the empirical performance of our threshold tuning algorithm when combined with the three new threshold-based TLC algorithms mentioned above. From the simulation experiments, we observe that our tuning algorithm is easily implementable over all road network settings and converges rapidly to the optimal thresholds in any graded threshold based TLC algorithm. Further, the additional computational effort required in finding the optimal thresholds

over algorithms with fixed thresholds is observed to be small.

Amongst the three TLC algorithms that we propose, the Q-learning based TLC algorithm (in the full state case) incorporates a novel state aggregation technique to handle larger state spaces than regular Q-learning (with full state representations). The combination of Q-learning based TLC with state aggregation is seen to result in a significant reduction in the cardinality of the state space, which results in a computational advantage over regular Q-learning (i.e., without state aggregation). Further, we also develop a Q-learning based TLC algorithm with function approximation along the lines of [1], however, with an important difference in the feature selection procedure. The feature selection procedure that we propose intelligently combines the queue lengths and elapsed times with the feasible sign configurations to arrive at the state-action features. This is unlike the features used in [1] in which the state and the action components of the features were almost independent. By obtaining features in a novel manner whereby the state and action components cannot be separated, we observed that the resulting Q-learning algorithm with function approximation significantly outperforms the algorithm proposed in [1]. The priority based TLC that we propose assigns priorities to the various sign configurations based on graded thresholds.

The combination of our threshold tuning algorithm with the other TLC algorithms mentioned above results in several interesting consequences. For instance, in the case of the Q-learning with full state representation, our threshold tuning algorithm results in finding an optimal way of clustering states so as to reduce the cardinality of the state space and in the case of the Q-learning algorithm with function approximation, our threshold tuning algorithm results in tuning online the associated state representation features, and thereby obtains the optimal features within a parameterized class of features (that are parameterized by the threshold parameters). In the context of reinforcement learning (RL), developing algorithms for feature adaptation is currently a hot area of research in itself.

2 Literature Survey

We now review literature in two different areas of related work: (1) techniques pertaining to traffic signal control and (2) developments in stochastic optimization approaches.

2.1 Traffic signal control

We briefly review some traffic light control strategies that have been proposed previously in the literature. The reader is referred to [3, 4] for a detailed discussion of relevant traffic signal control literature. In [5], an off-line traffic signal control technique where the signal timings are generated using a static optimizer, has been proposed. A popular version of this system that has been widely adapted is SCOOT [6] - a traffic responsive signal control scheme that uses traffic volume and occupancy as input. SCATS [7] is a well-deployed scheme that picks one of the pre-calculated control based on the traffic state. Several on-line TLC algorithms that adapt in real-time have also been proposed. For instance, genetic algorithms [8, 9], neural network based algorithms [10, 11], algorithms based on cellular automata [12], stochastic control [13] and dynamic optimization [14, 15, 16, 17] as well as reinforcement learning [18, 1] are all online schemes. In [14, 15, 16, 17], a model of the system is assumed and the optimal signal timings are obtained by solving a dynamic optimization problem in real-time. SPSA gradient estimates have also been used in the neural network based approaches of [10, 11] for optimizing traffic signal timings. A distributed multi-agent model for traffic signal control is presented in [19]. An optimization model of the traffic signal control problem is developed in [20]. The adaptive control based TLC algorithm proposed in [13] incorporates a Markov decision process (MDP) formulation. However, this requires a precise model of the system, which in general is hard to obtain in realistic settings. An adaptive traffic light control algorithm that uses approximate dynamic programming is proposed in [21]. A reinforcement learning based TLC algorithm proposed in [18] uses full state representations that have a high computational complexity. This algorithm has been studied only for an isolated traffic junction scenario in [18]. The RL-based TLC algorithm in [1] incorporates function approximation together with certain fixed graded-feedback policies and is seen to perform well over many road network scenarios with as many as 8 or 9 junctions involving nearly 10^{90} states. Unlike RL algorithms based on full-state representations [18], the computational effort required by the algorithm in [1] remains reasonable even for large scale networks because of the use of function approximation. Traffic signal scheduling in the context of a multi-agent system using a reinforcement learning framework has been explored, for instance, in [22, 23, 24, 25].

2.2 Stochastic optimization

A popular approach for simulation based parameter optimization is Simultaneous Perturbation Stochastic Approximation (SPSA) proposed in [26]. SPSA is an efficient gradient search technique that aims at finding a local minimum of a performance objective. The regular SPSA algorithm perturbs the parameter vector randomly using i.i.d., symmetric, zero-mean random variables and has the critical advantage that it needs only two samples of the objective function for any N-dimensional parameter. A variant of the SPSA algorithm that uses one simulation was proposed in [27]. Unlike its two-simulation counterpart, the algorithm in [27] does not work well in practice. In [2], several variants of the SPSA algorithm that work with deterministic perturbations (in place of randomized) were developed that show performance improvements over their randomized perturbation counterparts. The perturbation variables there are based on either lexicographic or Hadamard matrix based sequences. In particular, the one-simulation variant of the SPSA algorithm that is based on Hadamard matrices has been found to perform significantly better than the one-simulation random-perturbation algorithm presented in [27]. A Newton based SPSA algorithm that requires four system simulations with Bernoulli random perturbations was proposed in [28]. In [29], three SPSA based estimates of the Hessian that require three, two and one system simulation(s), respectively, were proposed. In [30], certain smoothed functional Newton algorithms that incorporate Gaussian-based perturbations were proposed. It is however the case that Newton based algorithms, even though more accurate than gradient based schemes, require significantly higher computational effort resulting from (a) projecting Hessian update at each iteration to the set of positive definite and symmetric matrices and (b) inverting the projected Hessian after each update. Hence, we present in this paper, the application of one-simulation deterministic perturbation SPSA for tuning the threshold parameters.

2.3 Organization

The rest of the paper is organized as follows: In Section 3, we describe in detail the problem framework. In Section 4, we present our threshold tuning algorithm. In Section 5, we present our algorithms for traffic signal control and combine them with the threshold tuning algorithm. In Section 6, we discuss the implementation of the various TLC schemes where our algorithm

was used, in order to find the optimal thresholds in each scheme and present the performance simulation results. Finally, in Section 7, we provide the concluding remarks.

3 Problem Formulation

We study in this paper the problem of maximizing traffic flow through the adaptive control of traffic lights at intersections. Our solution methodology consists of two important components - a threshold-based sign configuration policy obtained from a TLC algorithm for a fixed set of thresholds and another algorithm that operates on top of the TLC algorithm itself for tuning the thresholds. A sign configuration here refers to all the signals associated with a phase, i.e., those that can be switched to green simultaneously. The TLC algorithms, that we consider, indicate when to switch a sign configuration and are all based on graded thresholds. The threshold tuning algorithm, that is common to all the schemes, tunes the graded feedback policy to find the optimal parameters (thresholds) and does not indicate how to switch the sign configurations - a task that is performed by the particular TLC algorithm used. Apart from designing efficient TLC algorithms, an important problem that we address is one of finding the "optimal" set of parameters to use for a given TLC algorithm and which give the optimal feedback policies within the given parameterized class of policies. To the best of our knowledge, this problem has not been addressed in the literature previously.

Each TLC algorithm that we consider uses as input - the queue lengths along the individual lanes leading to the intersection and the time elapsed since the last signal light switch over on each lane. The queue length input is used to minimize (depending on the objective) the average junction waiting times of the road users, while the elapsed time input is used to ensure fairness, i.e., no lane is allowed to stay green for a long time at the cost of other lanes.

We consider a centralized control setting, where control decisions are made by a centralized controller that receives the state information from the various lanes and makes decision on which traffic lights to switch green during a cycle. This decision is then relayed back to the individual junctions. We assume no propagation and feedback delays for simplicity. The elapsed time counter for a lane with green signal stays at zero until the time the signal turns red. For a road network with m junctions and a total of K signalled

lanes across junctions, the state at time n is the vector

$$s_n = (q_1(n), \ldots, q_K(n), t_1(n), \ldots, t_K(n)), \tag{1}$$

where $q_i(n)$ and $t_i(n)$ are respectively the queue length and elapsed time on lane i at time n. The above state formulation is valid for all the TLC algorithms except the Q-learning based TLC that incorporates state aggregation. The details of the state aggregation and the formulation of the state parameterized by thresholds for this algorithm is given in Section 5.1.

The action a_n chosen by the central controller at time n is the sign configuration at each of the m junctions of the road network and has the form: $a_n = (a_1(n), \ldots, a_m(n))$, where $a_i(n)$ is the sign configuration at junction i in time slot n. We assume here that there are a total of m junctions in the road network.

The cost function is designed to maximize traffic flow and at the same time, ensure fairness so that no lane suffers from being red for a long duration. This is achieved by letting the cost function be the weighted sum of queue lengths and elapsed times on the lanes of the road network with the weights chosen suitably to incorporate prioritization of traffic. For instance, traffic on main roads is given a higher priority than that on side roads. Let I_p denote the set of indices of prioritized lanes, i.e., whose traffic is given a higher priority. We let the cost $k(s_n, a_n)$ have the form

$$\begin{aligned} k(s_n, a_n) = \quad & \alpha_1 * (\textstyle\sum_{i \in I_p} \alpha_2 * q_i(n) + \sum_{i \notin I_p} \beta_2 * q_i(n)) \\ + \quad & \beta_1 * (\textstyle\sum_{i \in I_p} \alpha_2 * t_i(n) + \sum_{i \notin I_p} \beta_2 * t_i(n)), \end{aligned} \tag{2}$$

where $\alpha_i, \beta_i \geq 0$ and $\alpha_i + \beta_i = 1, i = 1, 2$. Further, $\alpha_2 > \beta_2$. Thus, lanes in I_p are assigned a higher cost and hence a cost optimizing strategy must assign a higher priority to these lanes in order to minimize the overall cost. Note from the way it is defined (cf. (2)), $k(s_n, a_n)$ depends explicitly on s_n and not a_n. However, tt depends on a_n indirectly as a change in the sign configuration has an impact on the queue lengths and elapsed times on the various lanes.

The sign configuration policy governing the state evolution is based on given queue-length thresholds L_1 and L_2 and elapsed time threshold T_1. We want to find an optimal value for the parameter vector θ that minimizes the long-run average cost (cf. (3)), where θ corresponds to the vector $(L_1, L_2, T_1)^T$ for all the TLC algorithms that we propose in Section 5.

We let the parameter vector θ take values in a compact set

$$C \stackrel{\triangle}{=} [L_{\min}, L_{\max}] \times [L_{\min}, L_{\max}] \times [T_{\min}, T_{\max}] \subset \mathcal{R}^3$$

by making use of the projection operator π defined later, where $L_{\min}, L_{\max}, T_{\min}, T_{\max}$ are certain prescribed thresholds. The objective is to find a θ that minimizes

$$J(\theta) = \lim_{l \to \infty} \frac{1}{l} \sum_{j=0}^{l-1} k(s_j, a_j). \tag{3}$$

The actions a_j are assumed to be governed by one of the policies that we present below, that in turn will be parameterized by the threshold parameter θ. While it is desirable to find a $\theta^* \in C$ that minimizes $J(\theta)$, it is in general difficult to achieve a global minimum. We use therefore a local optimization method for which one needs to evaluate $\nabla J(\theta) \equiv (\nabla_1 J(\theta), \nabla_2 J(\theta), \nabla_3 J(\theta))^T$, for all the algorithms.

We make the following assumption on the function $J(\cdot)$:

Assumption (A1)

The long run average cost $J(\theta)$ is continuously differentiable with bounded second derivative.

Note that (A1) is a technical requirement used to push through a Taylor's argument in the convergence analysis (see Section 5.4). In the case of finite state parameterized Markov chains (as here), one can show using a similar argument as [31] that the parameterized stationary distribution is continuously differentiable if the parameterized transition probabilities are.

The threshold tuning algorithm that we present in the next section finds the optimum parameter θ using only one simulation trajectory, while the TLC schemes that we present estimate the optimal policy for a given set of thresholds. Hence, in combination with the threshold tuning algorithm, the TLC algorithms that we propose are seend to exhibit significant performance improvements.

4 The threshold tuning algorithm

A stochastic iterative algorithm to find the optimal thresholds in the case of a long-run average cost objective would require two nested loops as follows:

1. The inner loop estimates the long-run average cost and also picks actions from the underlying TLC algorithm.

2. The outer loop updates θ along a negative descent direction using an estimate obtained using the outcome of the inner loop procedure.

The above procedure involving two loops has to be performed iteratively until the parameter θ converges to a local minimum. However, such a procedure would typically be computationally expensive considering the fact that one step of the outer loop, i.e., updating θ in the direction of $-\nabla_\theta J(\theta)$, happens only after the convergence of the corresponding inner loop procedure. Using a multi-timescale stochastic approximation procedure [32, Chapter 6], we circumvent this problem as both the inner and outer loops can run in tandem The resulting scheme is shown to converge to the optimal solution for the long-run average cost objective (3).

The threshold tuning algorithm estimates the gradient of the objective function $\nabla_\theta J(\theta)$ using a one-sided SPSA based estimate, i.e.,

$$\nabla_\theta J(\theta) \approx \left(\frac{J(\theta + \delta\triangle)}{\delta} \right) \triangle^{-1}, \tag{4}$$

that incorporates a Hadamard matrix based deterministic construction for the perturbations $\triangle$. While one-simulation SPSA gradient estimates based on randomized perturbations were proposed in [27], these are seen to suffer from a large bias and hence do not give good performance. In [2], a one-simulation SPSA algorithm that incorporates certain Hadamard matrix based deterministic perturbations was proposed. This algorithm from [2] was seen to yield good performance and has a significantly lower bias in comparison to the algorithm in [27].

Fig 1 illustrates the operation of the threshold tuning algorithm. In essence, it is a closed-loop procedure where the system is simulated for a perturbed parameter value $(\theta + \delta\triangle)$ and the average cost estimates obtained via simulation are used to update θ in the negative gradient descent direction using the estimate (4).

In what follows, we use $\hat{s}_l, l \geq 0$ to denote the state-valued process governed by the perturbed parameter sequence $\hat{\theta}_l, l \geq 0$. This process and the corresponding sequence of actions $\hat{a}_l, l \geq 0$ help in performing the parameter updates (5). The threshold tuning algorithm is as follows:

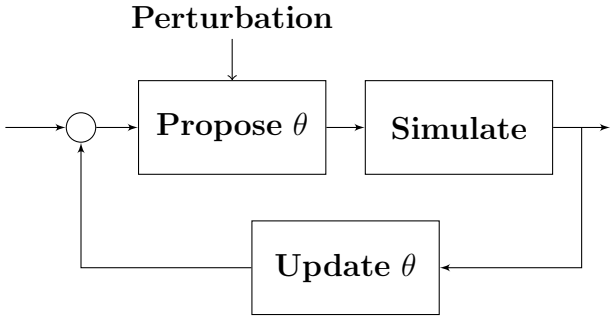


Figure 1: Illustration of the operation of the threshold tuning algorithm

$$\begin{aligned} L_1(n+1) &= \pi_1\left(L_1(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_1(n)}\right)\right), \\ L_2(n+1) &= \pi_1\left(L_2(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_2(n)}\right)\right), \\ T_1(n+1) &= \pi_2\left(T_1(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_3(n)}\right)\right). \end{aligned} \tag{5}$$

In the above,

- $L_1(n), L_2(n), T_1(n)$ denote the n-th updates of the thresholds L_1, L_2 and T_1, respectively.
- $\tilde{Z}(nL)$ represents the cost function averaging term obtained by accumulating the single stage cost over L cycles and is specific to the TLC algorithm being used to obtain the sign configuration policy on the faster timescale. These updates will be explained in the TLC algorithms in the next section.
- $L \geq 1$ is a fixed parameter which controls the rate of update of θ in relation to that of $\tilde{Z}$. This parameter allows for accumulation of updates to $\tilde{Z}$ for L iterations in between two successive θ updates. It is usually observed that allowing L to be greater than 1 improves the algorithm's performance.
- $\delta > 0$ is a given small constant and $\triangle(n) = (\triangle_1(n), \triangle_2(n), \triangle_3(n))^T$ is a vector of ± 1-valued random variables $\triangle_i(n), i = 1, 2, 3$. The $\triangle_i(n)$ themselves correspond to perturbation directions for the three parameter components. For any n, $\triangle(n)$ is obtained from a Hadamard matrix construction described in Section 4.1.

- $\pi : \mathcal{R}^3 \rightarrow C$ is the projection operator defined by $\pi(\theta) \stackrel{\triangle}{=} (\pi_1(\theta_1), \pi_1(\theta_2), \pi_2(\theta_3))^T, \theta \in \mathcal{R}^3$. Here for any $x \in \mathcal{R}$, $\pi_1(x) \stackrel{\triangle}{=} \min(\max(L_{\min}, x), L_{\max})$ and $\pi_2(x) \stackrel{\triangle}{=} \min(\max(T_{\min}, x), T_{\max})$, respectively.

The complete algorithm is described as under.

Algorithm 1 The threshold tuning algorithm

Input:

- R, a large positive integer; θ_0, initial parameter vector; $\delta > 0$; $\triangle$;
- UpdateTheta(), the stochastic update rule discussed in (5)
- Simulate(θ) $\rightarrow X$: the function that performs one time-step of the road traffic simulation and output the single-stage cost value $k(\hat{s}_n, \cdot)$ (cf. (2))
- UpdateAverageCost(): the function that updates the average cost estimate $\tilde{Z}(\cdot)$ used in (5) and is specific to the TLC-algorithm.
- UpdateTheta(): the function that updates the threshold parameter θ according to (5).

Output: $\theta^* \stackrel{\triangle}{=} \theta_R$.

$\theta \leftarrow \theta_0$, $n \leftarrow 1$
loop
  $\hat{X} \leftarrow$ Simulate($\theta + \delta\Delta$)
  UpdateAverageCost()
  if $n \,\%\, L = 0$ **then**
    UpdateTheta()
  end if
  $n \leftarrow n + 1$
  if $n = R$ **then**
    Terminate with θ.
  end if
end loop

We make the following assumption on the step-sizes $a(n)$ and $b(n)$:

Assumption (A2)

$$\sum_n a(n) = \sum_n b(n) = \infty; \sum_n (a^2(n) + b^2(n)) < \infty,$$
$$\text{and } \lim_{n\to\infty} \frac{a(n)}{b(n)} = 0.$$

The first two requirements in (A2) are standard in stochastic approximation algorithms. In particular, the first requirement ensures that the recursions do not converge prematurely while the second requirement aids in canceling the effect of stochastic noise. As described previously, the third requirement in (A2) essentially gives rise to the desired separation of timescales between the recursion (5) that is common to all algorithms and the recursions (8), (11) and (12) of the various TLC algorithms that we describe below. As we do in our experiments for all algorithms, one can select the following step-sizes $a(n)$ and $b(n), n \geq 0$:

$$a(0) = \hat{a},\ b(0) = \hat{b},\ a(n) = \hat{a}/n,\ b(n) = \hat{b}/n^{\alpha},\ n \geq 1,$$

with $1/2 < \alpha < 1, 0 < \hat{a}, \hat{b} < \infty$. The choice of step-sizes above can be seen to satisfy the requirements in (A2).

4.1 The Hadamard matrix based construction for $\{\triangle(n)\}$

An $m \times m$ ($m \geq 2$) matrix H is called a **Hadamard matrix** of order m if its entries belong to $\{1, -1\}$ and $H^T H = mI_m$, where I_m denotes the $m \times m$ identity matrix. Further, a Hadamard matrix is said to be normalized if all the elements of its first row and column are 1. A simple and systematic way of constructing normalized Hadamard matrices of order $m = 2^k$ is as follows:

For $k = 1$,

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

and for general $k > 1$,

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}.$$

Let $P = 2^{\lceil log_2(N+1) \rceil}$, where, as mentioned before, N is the parameter dimension. This implies $P \geq N + 1$. Now construct a normalized Hadamard matrix H_P of order P using the above procedure. Let $h(1), \ldots, h(N)$ be any N columns other than the first column of H_P. The first column is not

considered because all elements in the first column are 1, while all the other columns have an equal number of $+1$ and -1 elements. The latter property aids in cancelling the bias terms. Form a new matrix $\widetilde{H}_P$ of order $P \times N$ with $h(1), \ldots, h(N)$ as its columns. Let $\widetilde{\triangle}(k), k = 1, \ldots, P$ denote the rows of $\widetilde{H}_P$. The perturbation sequence $\{\triangle(m)\}$ is now generated by cycling through the rows of $\widetilde{H}_P$, i.e.,

$$\triangle(n) = \widetilde{\triangle}(n \bmod P + 1), \forall n \geq 0.$$

Having described the problem framework and the threshold tuning algorithm, we now present three TLC algorithms - each corresponding to a given class of graded threshold based sign configuration policies with given thresholds. The thresholds L_1, L_2 and T_1 are held fixed and used in various ways (as we explain below) in the these algorithms. When combined with the threshold tuning algorithm (5), these algorithms represent the most effective traffic light control strategies in their respective classes. The first two algorithms that we present below are based on Q-learning and incorporate state aggregation and function approximation, respectively, while the last algorithm is a priority based scheme.

5 Traffic Light Control algorithms

5.1 Q-learning TLC with State Aggregation (QTLC-SA)

Q-learning is an important RL algorithm that has the following incremental update rule:

$$Q_{n+1}(i,a) = \quad Q_n(i,a) + a(n)\left(k(i,a) + \gamma \min_{b \in \mathcal{A}(j)} Q_n(j,b) - Q_n(i,a)\right), \tag{6}$$

for all feasible tuples (i,a) of states and actions. Here j is the next state after state i that is observed via simulation and follows the distribution $p(i,.,a)$. Also, a (resp. b) is a feasible action in state i (resp. j) and $0 < \gamma < 1$ is a given discount factor. While traditional methods to solve the MDP (see [33]) formulated in the previous section would require the specification of $p(i,.,a)$, the Q-learning solution (6) works with only simulation based sample estimates of the single stage cost function (2) to obtain the optimal sign configuration policy, and explicit knowledge of the transition probabilities $p(i,\cdot,a)$ is not required. Quantities $Q_n(i,a)$ are estimates of the Q-value

when the feasible state-action tuple is (i, a), i.e., $a \in \mathcal{A}(i), i \in \mathcal{S}$. $Q_0(i, a)$ can be initialized arbitrarily, a simple choice is to set them all to zero. Note that recursion (6) is run for all feasible state-action tuples (i, a).

An MDP framework [33] requires the identification of states, actions and costs. While the actions and single-stage cost function were identified in Section 3, the state formulation in our algorithm below is different and uses state aggregation to tackle the curse of dimensionality. Consider the process $\{s'_n(\theta), n \geq 0\}$, parameterized by $\theta = (L_1, L_2, T_1)^T$, defined as follows:

$$\begin{aligned} s'_n(\theta) &= (q'_1(n), \ldots, q'_K(n), t'_1(n), \ldots, t'_K(n)), \\ \text{where} & \\ q'_i(n) &= \begin{cases} 0 & \text{if } q_i(n) < L_1 \\ 0.5 & \text{if } L_1 \leq q_i(n) \leq L_2 \\ 1 & \text{if } q_i(n) > L_2 \end{cases} \\ t'_i(n) &= \begin{cases} 0 & \text{if } t_i(n) \leq T_1 \\ 1 & \text{if } t_i(n) > T_1 \end{cases} \end{aligned} \tag{7}$$

Note that, as before, $q_i(n)$ and $t_i(n)$ denote the waiting queue lengths and elapsed times corresponding to the lane i at time n. Thus, $q'_i(n)$ and $t'_i(n)$ serve as proxies for $q_i(n)$ and $t_i(n), i = 1, \ldots, K$ respectively. By only allowing $q'_i(n)$ to take three possible values and $t'_i(n)$ to take two values depending on $q_i(n), t_i(n)$ and θ, we cluster together states s_n into $s'_n(\theta), n \geq 0$ in order to make computation more manageable. The problem of the curse of dimensionality associated with large state spaces is the primary reason why regular Q-learning with full state representations (cf. recursion (6)) is not implementable in the case of large networks, see [1].

As discussed above, the states in this algorithm are parameterized by θ via the thresholds L_1, L_2 and T_1. Our state formulation clusters together states based on the thresholds. The Q-learning algorithm with the update rule (6) when applied to our setting with state aggregation will be referred to as the QTLC-SA algorithm.

5.1.1 QTLC-SA with threshold tuning (QTLC-SA-TT)

The QTLC-SA-TT algorithm is a two timescale stochastic approximation algorithm that updates θ on the slower timescale and learns the optimal sign configuration policy (parameterized by θ) on the faster timescale. The update of the threshold parameters is done using the threshold tuning algorithm (5), whereas the cost function averaging (required for the threshold tuning

algorithm) and update of the Q-learning algorithm are performed as follows: For $m = nL, \ldots, (n+1)L - 1$,

$$\tilde{Z}(m+1) = \tilde{Z}(m) + b(n)(k(\tilde{s}_m(\theta), \tilde{a}_m) - \tilde{Z}(m)), \tag{8a}$$

$$Q_{m+1}(\hat{s}(\theta), \hat{a}) = Q_m(\hat{s}(\theta), \hat{a}) + b(n)\left(k(\hat{s}, \hat{a}) + \gamma \min_{b \in \mathcal{A}(j(\theta))} Q_m(j(\theta), b) - Q_m(\hat{s}(\theta), \hat{a})\right) \tag{8b}$$

- Here, $\{\hat{s}_l(\theta)\}$ denotes the aggregated state-valued process that is governed by $\{\hat{\theta}_l\}$, where $\hat{\theta}_l = \theta_n + \delta\triangle(n)$ for $n = \left\lfloor \frac{l}{L} \right\rfloor$, with a given $L \geq 1$.
- Whereas the Q-function update (cf. (8b)) is performed for all feasible state action tuples (where states are now aggregated states), the action $\tilde{a}_m$ in the argument of the single-stage cost $k(\hat{s}_m(\theta), \cdot)$ in (8a) is chosen to be the one that minimizes $Q_m(\hat{s}_m(\theta), \cdot)$. This ensures that threshold tuning is done based on the optimal action selection at each stage.
- The perturbations $\triangle(n)$ are generated using Hadamard matrices as outlined in Section 4.1.

We make the following assumption on the underlying process, when the actions are obtained from the QTLC-SA algorithm:

Assumption (A3)

(A3a) The basic underlying process $\{\hat{s}_n(\theta), n \geq 1\}$ is an MDP that is parameterized by θ

(A3b) The Markov chain $\{\hat{s}_n(\theta), n \geq 1\}$ under a given parameter $\theta \in C$ and any fixed stationary sign configuration policy is ergodic.

Assumption (A3) ensures in particular that the long-run average cost in (3) is well-defined for any given $\theta \in C$ under any stationary sign configuration policy.

5.2 Q-learning with Function Approximation with a Novel Feature Selection scheme (QTLC-FA-NFS)

It turns out that with QTLC-SA, the curse of dimensionality using state aggregation cannot be fully controlled for large networks with high-dimensional

states, as the cardinality of the aggregated state space increases significantly with the dimension of the states. To alleviate the curse of dimensionality problem, the use of function approximation has been advocated in [1] and the QTLC-FA algorithm developed therein is a computationally efficient TLC algorithm that is based on Q-learning. However, the state-action features used in [1] do not incorporate a clear dependence between states and actions. We further improve the performance of the QTLC-FA algorithm proposed in [1] by incorporating a novel feature selection scheme that incorporates dependence between states and actions and assigns priorities to various state-action combinations in the features.

In a function approximation setting, corresponding to every feasible state-action tuple, a feature vector has to be specified. The inputs for the feature selection scheme would be the broad estimates of congestion levels and elapsed times on the lanes of the road network. In [1], the feature vector contained a bit each for the congestion estimate, elapsed time estimate and the sign configuration portion, respectively, for each lane of the road network. While each of these attributes are important, the approximation architecture used in [1] did not take into account the dependence between features. In our novel feature selection scheme that we propose below, we incorporate dependence between the state and the action features while using graded thresholds. Another advantage of our methodology is that the dimension of the feature vector is now reduced by more than half as compared to the scheme in [1]. Numerical experiments show that this new choice of features significantly outperforms the one used in [1]. The various aspects of the Q-learning TLC algorithm that incorporates function approximation with the new feature selection scheme, are described below.

In the algorithm below, a state-action feature denoted by $\sigma_{i,a}$ is associated with each feasible state-action tuple (i, a). The Q-function is then approximated as

$$Q(i, a) \approx \omega^T \sigma_{i,a}. \tag{9}$$

Let s_n and s_{n+1} denote the states at instants n and $n+1$, respectively, and ω_n be the estimate of the parameter ω at instant n. The Q-learning algorithm with function approximation has the following update rule:

$$\omega_{n+1} = \omega_n + b(n)\sigma_{s_n,a_n}\left(k(s_n, a_n) + \gamma \min_{v \in A(s_{n+1})} \omega_n^T \sigma_{s_{n+1},v} - \omega_n^T \sigma_{s_n,a_n}\right), \tag{10}$$

where ω_0 is set arbitrarily. In (10), the action a_n is chosen in state s_n according to

Table 1: Feature selection ($\sigma_i(n)$) scheme for lane i

State	**Action**	**Feature**
$q_i(n) < L_1$ and $t_i(n) < T_1$	RED	0
	GREEN	1
$q_i(n) < L_1$ and $t_i(n) \geq T_1$	RED	0.2
	GREEN	0.8
$L_1 \leq q_i(n) < L_2$ and $t_i(n) < T_1$	RED	0.4
	GREEN	0.6
$L_1 \leq q_i(n) < L_2$ and $t_i(n) \geq T_1$	RED	0.6
	GREEN	0.4
$q_i(n) \geq L_2$ and $t_i(n) < T_1$	RED	0.8
	GREEN	0.2
$q_i(n) \geq L_2$ and $t_i(n) \geq T_1$	RED	1
	GREEN	0

$a_n = arg\min_{v \in A(s_n)} \omega_n^T \sigma_{s_n,v}$.

The features are chosen as follows: $\sigma_{s_n,a_n} = (\sigma_1(n), \ldots, \sigma_K(n))^T$, where the scheme for selection of the feature value $\sigma_i(n)$ corresponding to lane i is explained in Table. 1.

The feature selection scheme is graded and assigns a value for each lane based on whether the queue length on the lane is below L_1, is between L_1 and L_2, or is above L_2, on whether the elapsed time is below T_1 or above it and also on whether the sign configuration indicates a RED or a GREEN light for the lane. For instance, if both queue length and elapsed time are above the "highest" threshold level for the lane, then an action of GREEN would result in a feature value of 0 and an action of RED would result in the value 1. In essence, this choice indicates that the TLC algorithm should attempt to switch this lane to green because regardless of the value of the weight vector, the Q-value in such a case would be 0. By a similar argument, if both the queue length and the elapsed time are below the "lowest" threshold level for the lane, then the feature value chosen is just the opposite, i.e., 0 for RED and 1 for GREEN, implying that it is better to keep this lane red. The feature values corresponding to other decision choices are again suitably graded. It is clear that the assignment of feature values here takes into account the dependence between states and actions. This is unlike [1] where feature values have been assigned separately to states and actions.

While the cardinality of the state-action space may be high so that storing/updating Q-values for each (s, a)-tuple may be impossible, the above

feature-based algorithm estimates Q-values using the parameterization (9), making its implementation feasible even on large road networks as the parameter ω_n has the same dimension as that of σ_{s_n,a_n}. The Q-learning with function approximation algorithm with the update rule (9) and our novel feature selection scheme described above, will be referred to as the QTLC-FA-NFS algorithm.

5.2.1 QTLC-FA-NFS with threshold tuning (QTLC-FA-NFS-TT)

As with QTLC-SA-TT, the combination of QTLC-FA-NFS with the threshold tuning algorithm (5) gives the QTLC-FA-NFS-TT algorithm. The recursions on the faster timescale in the case of QTLC-FA-NFS-TT algorithm are as follows: Let $\{\tilde{s}_n, n \geq 0\}$ denote a state-valued process that depends on both the tunable policy as well as the tunable parameter $\tilde{\theta}_l, l \geq 0$, where $\tilde{\theta}_l = \theta_n + \delta\triangle(n)$ for $n = \left[\frac{l}{L}\right]$, and updates of $\theta_n \equiv (L_1(n), L_2(n), T_1(n))^T$ are governed according to (5). For $m = nL, \ldots, (n+1)L - 1$,

$$\tilde{Z}(m+1) = \tilde{Z}(m) + b(n)\left(k(\tilde{s}_m, \hat{a}_m) - \tilde{Z}(m)\right), \tag{11a}$$

$$\omega_{m+1} = \omega_m + b(n)\sigma_{\tilde{s}_m,\hat{a}_m}\left(k(\tilde{s}_m, \hat{a}_m) + \gamma \min_{v \in \mathcal{A}(\tilde{s}_{m+1})} \omega_m^T \sigma_{\tilde{s}_{m+1},v} - \omega_m^T \sigma_{\tilde{s}_m,\hat{a}_m}\right). \tag{11b}$$

The action $\hat{a}_m$ in (11a)-(11b) is chosen to be the one that minimizes $\omega_m^T \sigma_{\tilde{s}_m,v}$ over all $v \in \mathcal{A}(\tilde{s}_m)$.

We make the following assumption here:

Assumption (A4)

(A4a) For any given $\theta \in C$, the basic underlying process $\{s_n, n \geq 1\}$ is an MDP.

(A4b) For any given policy and parameter $\theta \in C$, the process $\{s_n, n \geq 1\}$ is ergodic Markov.

5.3 Priority based TLC (PTLC)

The sign configuration policy here is a graded threshold-based policy that assigns different priorities to different policy levels. The thresholds here are on the queue lengths (L_1 and L_2) and elapsed times since the last switch

Table 2: Priority assignment for each lane in the TLC policy

Condition	**Priority value**
$q_i < L_1$ and $t_i < T_1$	1
$q_i < L_1$ and $t_i \geq T_1$	2
$L_1 \leq q_i(n) < L_2$ and $t_i(n) < T_1$	3
$L_1 \leq q_i(n) < L_2$ and $t_i \geq T_1$	4
$q_i \geq L_2$ and $t_i < T_1$	5
$q_i \geq L_2$ and $t_i \geq T_1$	6

over of lights to red (T_1) on individual lanes. The cost assigned to each lane is decided based on whether the queue length on that lane is below L_1, is between L_1 and L_2, or is above L_2 at any instant and also on whether the elapsed time is below T_1 or above it. For instance, if both queue length and elapsed time are above the "highest" threshold levels (L_2 and T_1 respectively) on a given lane, then the policy assigns the highest priority value (of 6) to that lane. The priority assignment for any lane i of the road network based on the queue length q_i and elapsed time t_i is shown in Table. 2. The policy then selects the sign configuration with the maximum (over all feasible sign configurations) sum of lane priority values. In essence, the TLC algorithm flushes the traffic on lanes with long waiting queues, while also giving higher priority to lanes that have been waiting on a red signal for a long time. This helps to combine efficiency with fairness.

5.3.1 PTLC with threshold tuning (PTLC-TT)

As with the previous TLC algorithms, we combine the threshold tuning algorithm (5) with PTLC to obtain the PTLC-TT algorithm. The state-valued process $\{\hat{s}_n, n \geq 0\}$ in this case under the priority based policy described above depends on the tunable parameter sequence $\hat{\theta}_l = \theta_n + \delta\triangle(n), n \geq 0$, where $\theta_n \equiv (L_1(n), L_2(n), T_1(n))^T, n \geq 0$ are updated according to (5). The faster timescale recursions here are given as follows: For $m = nL, \ldots, (n+1)L - 1$,

$$\tilde{Z}(m+1) = \tilde{Z}(m) + b(n)(k(\hat{s}_m, \hat{a}_m) - \tilde{Z}(m)). \tag{12}$$

The action $\hat{a}_m$ above is selected in state $\hat{s}_m$ based on the priority assignment policy (described above), i.e., select the sign configuration that has the maximum sum of priority values (where the maximum is over all feasible

sign configurations) and switch the lanes in the chosen sign configuration to green.

We now make the following assumption:

Assumption (A5)

(A5a) The basic underlying process $\{\hat{s}_n, n \geq 0\}$ is a parameterized MDP.

(A5b) For any given $\theta \in C$ and the specified priority-based policy, $\{\hat{s}_n, n \geq 0\}$ is an ergodic Markov process.

Having described the three TLC algorithms and their combination with the threshold tuning algorithm (5), we now provide a sketch of the convergence of θ to its optimal value using the analysis from [2].

5.4 Convergence analysis

We show here the convergence of the threshold tuning algorithm, viz., recursions (5), when the updates of $\tilde{Z}$ are governed according to (12) in the PTLC-TT scheme. A similar analysis works in the case when these are governed as per the other schemes – (8) and (11), respectively. Note that because the action selection in the case of the PTLC algorithm is via the policy as prescribed in Table 2, one can write $\hat{a}_n = f(\hat{s}_n)$, $n \geq 0$, where the function f is a deterministic function that specifies the aforementioned policy. For simplicity, we shall consider here the case of $L = 1$, i.e., of no additional averaging on top of the two-timescale averaging. The case of general (finite) L can also be handled, see for instance, [2].

We will assume for simplicity that the elapsed times on any lane remain bounded even though the bound can be large. Moreover, the queue lengths on any lane are bounded as well because each lane (between two junctions) can at most accommodate a bounded number of vehicles. The single-stage cost function can be seen to be a linear function of the state and remains bounded in this case. If the elapsed times are allowed to become unbounded, then the single-stage cost can become unbounded as well and one will require additional assumptions such as the existence of a stochastic Lyapunov function in order to ensure that the second moment of the system state remains uniformly bounded almost surely.

The PTLC-TT algorithm (in the case of $L = 1$) can be rewritten as

follows:

$$L_1(n+1) = \pi_1\left(L_1(n) - a(n)\left(\frac{\tilde{Z}(n)}{\delta\triangle_1(n)}\right)\right), \tag{13}$$

$$L_2(n+1) = \pi_1\left(L_2(n) - a(n)\left(\frac{\tilde{Z}(n)}{\delta\triangle_2(n)}\right)\right), \tag{14}$$

$$T_1(n+1) = \pi_3\left(T_1(n) - a(n)\left(\frac{\tilde{Z}(n)}{\delta\triangle_3(n)}\right)\right), \tag{15}$$

$$\tilde{Z}(n+1) = \tilde{Z}(n) + b(n)(k(\hat{s}_n, f(\hat{s}_n)) - \tilde{Z}(n)). \tag{16}$$

Lemma 1. *Each of the recursions (13)-(16) is uniformly bounded with probability one.*

Proof. Recursions (13)-(15) stay uniformly bounded as a consequence of the projection operators π_1 and π_2 respectively. As described previously, the single-stage cost function $k(\cdot,\cdot)$ remains uniformly bounded almost surely. Now note that since $b(n) \to 0$ as $n \to \infty$, there exists an integer $N_0 > 0$ such that for all $n \geq N_0$, $b(n) < 1$. Thus, for $n \geq N_0$, $\tilde{Z}(n+1)$ is a convex combination of $\tilde{Z}(n)$ and $k(\hat{s}_n, f(\hat{s}_n))$ (a uniformly bounded quantity). Suppose $\sup_n |k(\hat{s}_n, f(\hat{s}_n))| \leq \hat{K}$, almost surely for some $\hat{K} > 0$. Thus,

$$\sup_n |\tilde{Z}(n)| \leq \max\{|\tilde{Z}(0)|, \cdot, |\tilde{Z}(N_0)|, \hat{K}\} < \infty,$$

almost surely. The claim follows. □

We analyze first the convergence of the faster timescale recursion (16). Define two sequences of time points $\{s(n)\}$ and $\{t(n)\}$ according to $s(0) = t(0) = 0$ and for $n \geq 1$, $s(n) = \sum_{m=0}^{n} a(m)$, $t(n) = \sum_{m=0}^{n} b(m)$. Let $\Delta(t), t \geq 0$ be defined by $\Delta(t) = \Delta(n), t \in [s(n), s(n+1)], n \geq 0$.

Let $\mathcal{F}_n = \sigma(\hat{s}_m, \theta(m), m \leq n), n \geq 0$ be a sequence of associated sigma fields. Consider the sequence $N_n, n \geq 0$, defined according to $N_0 = 0$ and for $n \geq 1$,

$$N_n = \sum_{m=0}^{n-1} b(m)\left(k(\hat{s}_m, f(\hat{s}_m)) - E\left[k(\hat{s}_m, f(\hat{s}_m)) \mid \mathcal{F}_{m-1}\right]\right).$$

We have the following result.

Lemma 2. $(N_n, \mathcal{F}_n), n \geq 0$ *forms an almost surely convergent martingale sequence.*

Proof. It is easy to see that $(N_n, \mathcal{F}_n), n \geq 0$ forms a martingale sequence. Now consider the quadratic variation process associated with this martingale. Note that

$$\sum_{n=0}^{\infty} E[(N_{n+1} - N_n)^2 \mid \mathcal{F}_n] = \sum_{n=0}^{\infty} b(n)^2 \xi^2(n) < \infty,$$

almost surely, since $\xi^2(n)$ remains uniformly bounded because the single-stage cost function $k(\cdot, \cdot)$ is uniformly bounded and moreover, $\sum_n b(n)^2 < \infty$. The claim follows from the martingale convergence theorem (cf. Theorem 3.3.4 of [34]). □

Define now the operators $\bar{\pi}_1$ and $\bar{\pi}_2$ as follows:

$$\bar{\pi}_1(v(x)) = \lim_{\eta \to 0} \left(\frac{\pi_1(x + \eta v(x)) - x}{\eta} \right), \bar{\pi}_2(w(x)) = \lim_{\eta \to 0} \left(\frac{\pi_2(x + \eta w(x)) - x}{\eta} \right), \tag{17}$$

for any continuous functions $v : [L_{\min}, L_{\max}] \to \mathcal{R}$ and $w : [T_{\min}, T_{\max}] \to \mathcal{R}$, respectively. The limits in (17) exist and are unique since $[L_{\min}, L_{\max}]$ and $[T_{\min}, T_{\max}]$ are convex sets. From definition, $\bar{\pi}_1(v(x)) = v(x)$ (resp. $\bar{\pi}_2(w(y)) = w(y)$) if $x \in (L_{\min}, L_{\max})$ (resp. $y \in (T_{\min}, T_{\max})$.

Consider now the following set of ODEs associated with (13)-(16).

$$\dot{L}_1(t) = 0, \quad \dot{L}_2(t) = 0, \quad \dot{T}_1(t) = 0, \tag{18}$$

$$\dot{\tilde{Z}}(t) = J(\theta(t) + \delta\Delta(t)) - \tilde{Z}(t), \tag{19}$$

respectively. In view of (18)-(19) and the manner in which $\Delta(t)$ is defined, it is sufficient to consider the following ODE in place of (19) for the faster timescale recursion:

$$\dot{\tilde{Z}}(t) = J(\theta + \delta\Delta) - \tilde{Z}(t). \tag{20}$$

Note above that we let $\theta(t) \equiv \theta$ and $\Delta(t) \equiv \Delta$, viz., time invariant quantities (as they get updated on the slower timescale). Before we proceed further, we recall a key result from [35]. Consider an ODE

$$\dot{x}(t) = f(x(t)), \tag{21}$$

where $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$ (for some $N \geq 1$) is a Lipschitz continuous function. Given T, $\gamma > 0$, we call a bounded, measurable $y(\cdot) : \mathcal{R}^+ \cup \{0\} \rightarrow \mathcal{R}^N$, a (T, γ)-perturbation of (21) if there exist $0 = T_0 < T_1 < T_2 < \cdots < T_r \uparrow \infty$ with $T_{r+1} - T_r \geq T \ \forall r$ and solutions $\theta^r(t)$, $t \in [T_r, T_{r+1}]$ of (21) for $r \geq 0$, such that

$$\sup_{t \in [T_r, T_{r+1}]} \| \theta^r(t) - y(t) \| < \Delta.$$

We recall the following result given as Theorem 1, pp.339 of [35].

Let H denote the set of globally asymptotically stable attractors of the ODE (21) and given $\epsilon > 0$, let H^ϵ denote the set of points that are within an ϵ-neighborhood from the set H. In particular, $H \subset H^\epsilon$.

Lemma 3 (Hirsch Lemma)**.** *Given ϵ, $T > 0$, $\exists \bar{\gamma} > 0$ such that for all $\gamma \in (0, \bar{\gamma})$, every (T, γ)-perturbation of (21) converges to H^ϵ.*

We now return to the analysis of the faster timescale recursion. Given $\hat{T} > 0$, let $\hat{T}_n, n \geq 0$ be defined according to $\hat{T}_0 = 0$ and $\hat{T}_n = \min\{t(n) \mid t(n) \geq \hat{T}_{n-1} + \hat{T}\}$, $n \geq 1$. Then $\hat{T}_{n+1} - \hat{T}_n \approx \hat{T}$, $\forall n \geq 0$ and there exists a subsequence $\{l(n)\}$ of $\{n\}$ such that $\hat{T}_n = t(l(n)) \forall n$. Define $\bar{L}_1(t), \bar{L}_2(t), \bar{T}_1(t), \bar{Z}(t), t \in [t(n), t(n+1)], n \geq 0$ in the following manner: $\bar{L}_1(t(n)) = L_1(n)$, $\bar{L}_2(t(n)) = L_2(n)$, $\bar{T}_1(t(n)) = T_1(n)$, $\bar{Z}(t(n)) = \tilde{Z}(n)$, respectively, with suitable continuous linear interpolations in between intervals $[t(n), t(n+1)]$.

Proposition 1. *The functions $\bar{L}_1(t), \bar{L}_2(t), \bar{T}_1(t), \bar{Z}(t), t \geq 0$ form $(\hat{T}, \gamma)$-perturbations of the ODEs (18)-(19).*

Proof. Note that along the timescale $t(n), n \geq 0$, i.e., using the sequence of step sizes $b(n), n \geq 0$, one can rewrite (13)-(15) as follows:

$$L_1(n+1) = \pi_1 \left(L_1(n) - b(n)\chi_1(n) \right), \tag{22}$$

$$L_2(n+1) = \pi_1 \left(L_2(n) - b(n)\chi_2(n) \right), \tag{23}$$

$$T_1(n+1) = \pi_3 \left(T_1(n) - b(n)\chi_3(n) \right), \tag{24}$$

where $\chi_i(n) = \dfrac{a(n)}{b(n)} \left(\dfrac{\tilde{Z}(n)}{\delta \triangle_i(n)} \right), i = 1, 2, 3$, respectively. Now, from Assumption (A2), we have that $\dfrac{a(n)}{b(n)} \rightarrow 0$ as $n \rightarrow \infty$. Hence, $\chi_j(n) = o(1), j = 1, 2, 3$. Finally, consider the recursion (16). Note that by Lemma 2, we have that

$\sum_{j=l(n)}^{l(n+1)-1} b(j)M_j \to 0$ as $n \to \infty$, where $M_j = k(\hat{s}_m, f(\hat{s}_m)) - E\left[k(\hat{s}_m, f(\hat{s}_m)) \mid \mathcal{F}_{m-1}\right]$, $j \geq 1$, is the associated martingale difference sequence.

One can now rewrite (16) as

$$\tilde{Z}(n+1) = \tilde{Z}(n) + b(n)(J(\theta(n) + \delta\Delta(n)) + \chi_4(n)) + (N_{n+1} - N_n),$$

where $\chi_4(n) = E\left[k(\hat{s}_m, f(\hat{s}_m)) \mid \mathcal{F}_{m-1}\right] - J(\theta(n) + \delta\Delta(n))$. From Assumption (A5), $\chi_4(n) \to 0$ on the 'natural timescale' that is clearly faster than the timescale of the algorithm, see Chapter 6.2 of [32] for a detailed treatment of natural timescale recursions involving Markov noise. The claim follows. □

Lemma 4. *With probability one,* $|\tilde{Z}(n) - J(\theta(n) + \delta\Delta(n)| \to 0$ *as* $n \to \infty$.

Proof. Follows by applying Lemma 3 for every $\epsilon > 0$. □

We now consider the slower timescale recursions (13)-(15). Recall that the parameter dimension in our case is $N = 3$. Thus, $P = 2^{\lceil log_2(N+1) \rceil} = 4$. Let $\Delta(n) = (\Delta_1(n), \Delta_2(n), \Delta_3(n))^T$, $n \geq 0$ be the perturbations obtained using the Hadamard matrix based procedure.

Lemma 5. *The vectors* $\Delta(n)$, $n \geq 0$ *satisfy the following properties:*

1. *For any* $s \geq 0$ *and all* $k \in \{1, \ldots, 3\}$, $\sum_{n=s+1}^{s+P} \frac{1}{\Delta_k(n)} = 0.$

2. *For any* $s \geq 0$ *and all* $i, j \in \{1, \ldots, 3\}$, $i \neq j$, $\sum_{n=s+1}^{s+P} \frac{\Delta_i(n)}{\Delta_j(n)} = 0.$

Proof. The claim is obvious from the construction. □

Let $\theta(n) = (L_1(n), L_2(n), T_1(n))^T$. For any $x = (x_1, x_2, x_3)^T \in \mathcal{R}^3$, let $\pi(x) \stackrel{\triangle}{=} (\pi_1(x_1), \pi_1(x_2), \pi_2(x_3))^T$. Similarly, let $\bar{\pi}(x) = (\bar{\pi}_1(x_1), \bar{\pi}_1(x_2), \bar{\pi}_2(x_3))^T$. In view of Lemma 4, one can consider the following recursions in place of (13)-(15):

$$\theta(n+1) = \pi\left(\theta(n) - a(n)J(\theta(n) + \delta\Delta(n))(\Delta(n))^{-1}\right). \tag{25}$$

Lemma 6. *Given any fixed integer* $K > 0$, *for all* $r \in \{1, \ldots, K\}$

(i) $\lim_{m\to\infty} \| \theta(m+r) - \theta(m) \| = 0$, *w.p.* 1, *and*

(i) $\lim_{m\to\infty} \| \nabla\theta(m+r) - \nabla\theta(m) \| = 0$, *w.p.* 1.

Proof. (i) The proof follows in a similar manner as Lemma 2.2 of [2].
(ii) Follows from (i) and Assumption (A1). □

The proof of the following result was not shown in [2] for Hadamard matrix perturbations. Hence, we provide it below.

Lemma 7. *The following hold for any* $k, l \in \{1, \ldots, 3\}$, $k \neq l$*:*

$$\| \sum_{n=m}^{m+P-1} \frac{a(n)}{a(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J(\theta(n)) \| \;\to\; 0 \;\; as \;\; m \to \infty, \tag{26}$$

$$\| \sum_{n=m}^{m+P-1} \frac{a(n)}{a(m)} \frac{1}{\triangle_l(n)} J(\theta(n) + \delta\Delta(n)) \| \;\to\; 0 \;\; as \;\; m \to \infty. \tag{27}$$

Proof. We first show that (26) holds. By letting $K = P$ in Lemma 6, it follows that $a(j)/a(m) \to 1$ as $m \to \infty$ for any $j \in \{m, \ldots, m+P-1\}$. Note also that P is an even integer.As a consequence of Lemma 5, one can split any set $A_m \stackrel{\triangle}{=} \{m, m+1, \ldots, m+P-1\}$ into two disjoint subsets $A^+_{m,k,l}$ and $A^-_{m,k,l}$ each having the same number of elements, with $A^+_{m,k,l} \cup A^-_{m,k,l} = A_m$ and such that $\dfrac{\triangle_k(n)}{\triangle_l(n)}$ takes value $+1$ $\forall n \in A^+_{m,k,l}$ and -1 $\forall n \in A^-_{m,k,l}$, respectively. Thus,

$$\| \sum_{n=m}^{m+P-1} \frac{a(n)}{a(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J(\theta(n)) \| = \| \sum_{n\in A^+_{m,k,l}} \frac{a(n)}{a(m)} \nabla_k J(\theta(n)) - \sum_{n\in A^-_{m,k,l}} \frac{a(n)}{a(m)} \nabla_k J(\theta(n)) \| .$$

The claim in (26) now easily follows. Finally, the claim in (27) follows from Lemma 6 and Assumption (A1), in a similar manner as (26). □

Consider now the following ODE:

$$\dot{\theta}(t) = \bar{\pi}(-\nabla J(\theta(t))). \tag{28}$$

Let $R \subset I = \{\theta \in C \mid \nabla J(\theta) = 0\}$ denote the set of stable fixed points of (28). Note that I denotes the set of all fixed points of (28) that would include

not just stable equilibria but also unstable ones such as local maxima, saddle points etc. The set of stable equilibria (i.e., the local minima) is in general a subset of I. For $\eta > 0$, let $R^\eta \stackrel{\triangle}{=} \{\theta \in C | \parallel \theta - \theta_0 \parallel < \eta \text{ for some } \theta_0 \in R\}$, denote the set of points that are within an η-neighborhood of R.

Theorem 5.1. *Given $\eta > 0$, there exists $\delta_0 > 0$ such that for all $\delta \in (0, \delta_0)$, the recursions $\theta(n)$ converge almost surely to R^η.*

Proof. The recursion (25) can be rewritten as follows: For $i = 1, 2, 3$,

$$\theta_i(n+1) = \gamma_i \left(\theta_i(n) - a(n) \frac{J(\theta(n) + \delta\Delta(n))}{\Delta_i(n)} \right), \tag{29}$$

where $\gamma_1 = \gamma_2 = \pi_1$ and $\gamma_3 = \pi_2$, respectively. The recursion (29) can be rewritten as follows:

$$\theta_i(n+1) = \theta_i(n) - a(n) \frac{J(\theta(n) + \delta\Delta(n))}{\Delta_i(n)} - a(n) Z_i(n), \tag{30}$$

where $Z_i(n)$ is an error term that results from the projection.

By a Taylor series expansion of $J(\theta(m) + \delta\triangle(m))$ around the point $\theta(m)$, one obtains: For $i = 1, 2, 3$,

$$\theta_i(m+2^P) = \theta_i(m) - \sum_{l=m}^{m+2^P-1} a(l) \nabla_i J(\theta(l)) - a(m) \sum_{l=m}^{m+2^P-1} \sum_{j=1, j\neq i}^{3} \frac{a(l)}{a(m)} \frac{\triangle_j(l)}{\triangle_i(l)} \nabla_j J(\theta(l))$$

$$- a(m) \sum_{l=m}^{m+2^P-1} \frac{a(l)}{a(m)} \frac{1}{\triangle_i(l)} J(\theta(l)) + a(m) O(\delta) - \sum_{j=m}^{m+2^P-1} a(j) Z_i(j). \tag{31}$$

The third and the fourth terms on the RHS of (31) vanish asymptotically as a consequence of Lemma 7. The rest now follows from Theorem 5.3.1 on pp.191-196 of [36]. □

6 Simulation Experiments

We now provide numerical results to illustrate the performance of the various threshold tuning TLC algorithms. For the purpose of traffic simulation and performance comparison of the various graded threshold based TLC algorithms, we used the Green Light District (GLD) traffic simulation software

[37]. The TLC algorithms were compared using the performance metrics of average junction waiting time and the total number of road users who reached their destination, as functions of the number of cycles. We now recall the QTLC-FA algorithm proposed in [1] and describe its tuning variant QTLC-FA-TT, that we also implemented for the sake of comparisons.

6.1 Q-learning with Function Approximation and Threshold Tuning (QTLC-FA-TT)

The sign configuration policy here is based on the QTLC-FA TLC algorithm from [1]. The function approximation architecture is similar to that used in the QTLC-FA-NFS algorithm (see section 5.2). The difference between QTLC-FA and QTLC-FA-NFS lies in the feature selection procedure. The features in QTLC-FA are chosen as follows:

$$\begin{aligned}
\sigma_{s_n,a_n} &= (\sigma_{q_1(n)},\ldots,\sigma_{q_K(n)},\sigma_{t_1(n)},\ldots,\sigma_{t_K(n)},\sigma_{a_1(n)},\ldots,\sigma_{a_m(n)})^T, \text{ where} \\
\sigma_{q_i(n)} &= \begin{cases} 0 & \text{if } q_i(n) < L_1 \\ 0.5 & \text{if } L_1 \leq q_i(n) \leq L_2 \\ 1 & \text{if } q_i(n) > L_2 \end{cases} \\
\sigma_{t_i(n)} &= \begin{cases} 0 & \text{if } t_i(n) \leq T_1 \\ 1 & \text{if } t_i(n) > T_1 \end{cases}
\end{aligned} \tag{32}$$

Further $\sigma_{a_1(n)},\ldots,\sigma_{a_m(n)}$ correspond to the sign configurations chosen at each of the m junctions.

The QTLC-FA-TT algorithm combines the QTLC-FA algorithm with threshold tuning in a similar manner as the QTLC-FA-NFS-TT algorithm. The update rule (5) is again followed to tune the thresholds. The sign configuration policy used in this algorithm is from the QTLC-FA algorithm described above.

6.2 Implementation

We implemented the threshold tuning algorithm for all four algorithms that incorporate graded thresholds - PTLC, QTLC-SA, QTLC-FA and QTLC-FA-NFS, respectively. We compared the performance of the tuned variants of the TLC algorithms against their counterparts that involved fixed thresholds (no tuning). The algorithms studied are outlined below:

- **PTLC**: This is a graded threshold based TLC algorithm described in Section 5.3 and uses a priority assignment scheme based on waiting

queue lengths and elapsed times in order to arrive at a sign configuration. Note that this algorithm uses fixed thresholds, and the performance of this algorithm is studied against its tuning variant PTLC-TT (below).

- **PTLC-TT**: This algorithm combines threshold tuning with the priority based TLC mentioned above. In other words, the sign configuration policy here is the same as PTLC, except that the thresholds used ($\theta = (L_1, L_2, T_1)^T$ are tuned using the online incremental algorithm described in Section 5.3.

- **QTLC-SA**: This is the Q-learning based TLC algorithm that incorporates state aggregation and is described in Section 5.1. Note that this algorithm uses fixed thresholds and the performance of this algorithm is compared against its tuning variant (QTLC-SA-TT) below.

- **QTLC-SA-TT**: This algorithm combines threshold tuning with the QTLC-SA TLC mentioned above. As in PTLC-TT, the parameter of thresholds $\theta = (L_1, L_2, T_1)^T$ used in the QTLC-SA TLC algorithm are tuned using the online incremental algorithm described in Section 5.1.

- **QTLC-FA**: This is the TLC algorithm presented in [1] and uses Q-learning with function approximation. The feature selection scheme here is as described in (32).

- **QTLC-FA-TT**: This is the tuning variant of the QTLC-FA algorithm mentioned above and incorporates the update (5) to tune the threshold parameter θ.

- **QTLC-FA-NFS**: This is the Q-learning based TLC algorithm with function approximation that incorporates the novel feature selection strategy described in Table 1. This algorithm is described in detail in Section 5.2. Note that this algorithm uses fixed thresholds and the performance of this algorithm is compared against its tuning variant (QTLC-FA-NFS-TT) below.

- **QTLC-FA-NFS-TT**: This algorithm is the tuning counterpart of QTLC-FA-NFS and tunes the threshold parameter θ using the recursions (5).

We performed experiments with the aforementioned TLC algorithms on the road networks shown in Fig. 2. In essence, we consider four different

settings - a single junction road network, a four-junction corridor, a 4x4-grid network and a seven-junction corridor, respectively for testing the TLC algorithms. Thus, we test the performance of our algorithms on both small road networks (such as the single-junction network and the four-junction corridor) as well as large size road network (such as the 4x4-grid). The 4x4-grid network consists of 16 edge nodes (where traffic is generated), 16 junctions with traffic lights, 40 roads, with each being 4 lanes wide and when full can house upto 1500 vehicles. Further, the cardinality of the state-action space is of the order of 10^{130} for the 4x4-grid network. We tested the performance of all the proposed TLC algorithms - PTLC, QTLC-SA and QTLC-FA-NFS (and their tuning counterparts) as well as the QTLC-FA algorithm from [1] on a single junction and four-junction corridor. However, on a 4x4-grid network as well as a seven-junction corridor, we could not implement the QTLC-SA algorithm. As explained before, QTLC-SA uses full state representations and hence is not implementable on high-dimensional state-action spaces (as with a 4x4-grid network and a seven-junction corridor). However, QTLC-SA is an enhancement to the Q-learning based TLC algorithm proposed for instance in [18]. In fact, the Q-learning based TLC with full state representations in [18] was proposed only for a single-junction road network. Use of state aggregation allowed the resulting QTLC-SA algorithm to scale up to a four-junction corridor. The results in this scenario are presented here.

The simulations are conducted for 25000 cycles for all algorithms and in each simulation, the destination of the road user is fixed randomly (using a discrete uniform distribution). At each time-step, vehicles are inserted into the road network as per the spawn frequencies of the edge nodes, where spawn frequency specifies the rate at which traffic is generated randomly in GLD. The spawn frequencies of the edge nodes are set in a way that ensures that the proportion of cars flowing on the main road to those on the side roads is in the ratio $100 : 5$. This ratio is close to what is practically observed and has also been used for instance in [38]. The results presented below are the averages over ten independent simulations with different initial seeds. For all the algorithms, the weights in the single-stage cost function (2) are set as $\alpha_1 = \beta_1 = 0.5$, $\alpha_2 = 0.6$ and $\beta_2 = 0.4$, respectively. This assignment essentially gives equal weightage to queue length and elapsed time components of (2), while according a higher weightage to main road traffic over side road traffic. The threshold parameter θ was set to $(6, 14, 90)^T$ for all the TLC algorithms. The performance of the TLC algorithms with the above tuning parameter was then compared with the counterparts of these

algorithms that incorporate tuning. This value of θ has been used in the results reported in [1]. The parameters δ and L used in the threshold tuning algorithm (5) were set to 0.5 and 10 respectively. The discount factor γ used in (6) and (9) was set to 0.9.

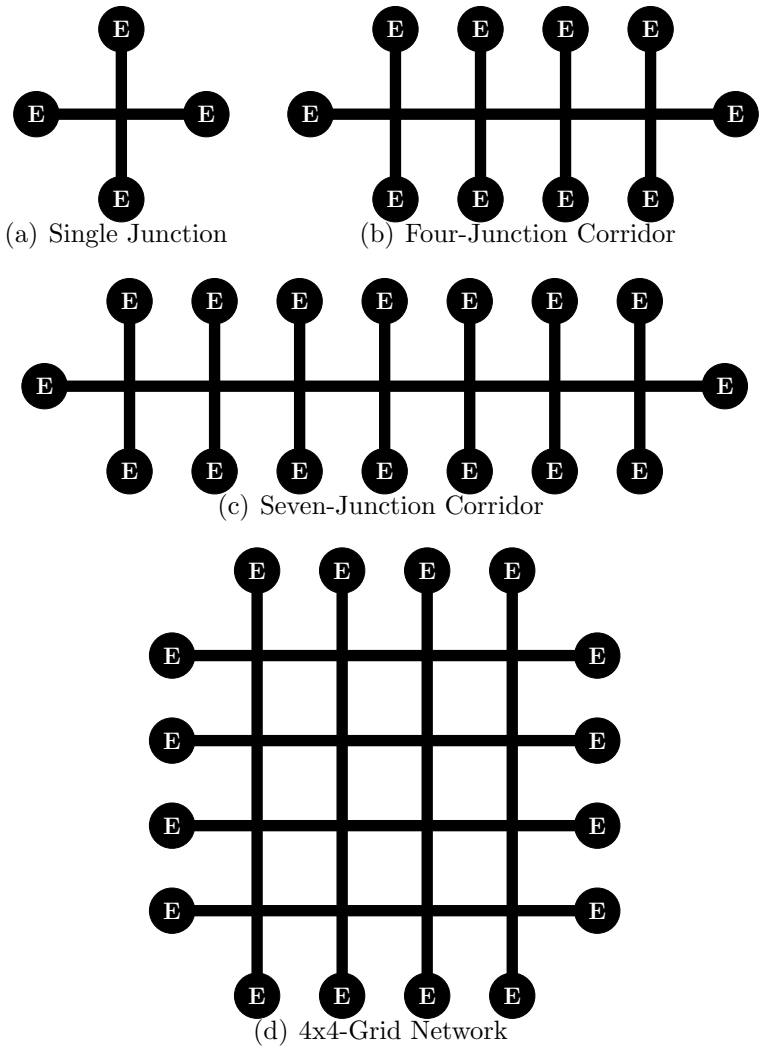
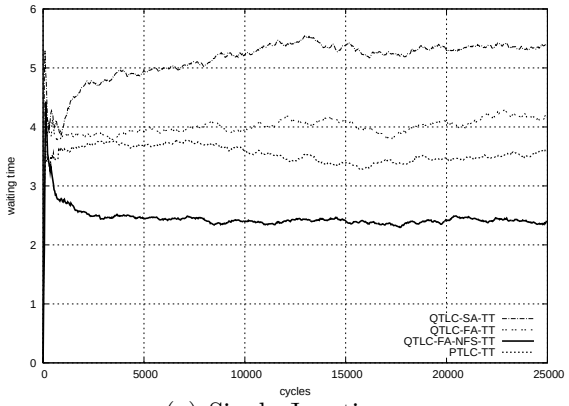


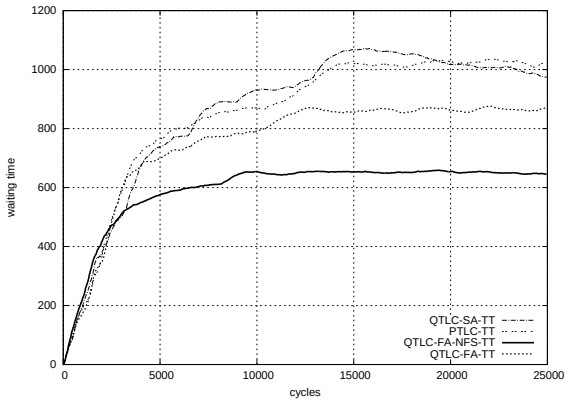
Figure 2: Road Networks used for our Experiments

6.3 Results

Fig. 3 shows the average trip waiting time (ATWT) plots on a single-junction and a four-junction corridor, respectively and compares the four TLC al-

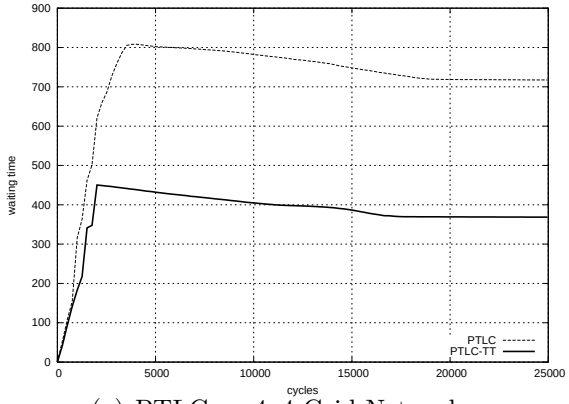


(a) Single Junction

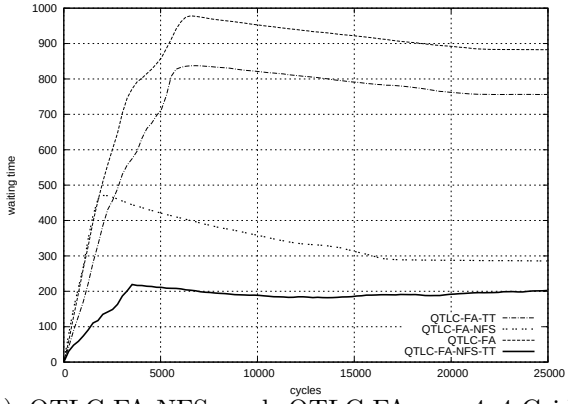


(b) Four-Junction Corridor

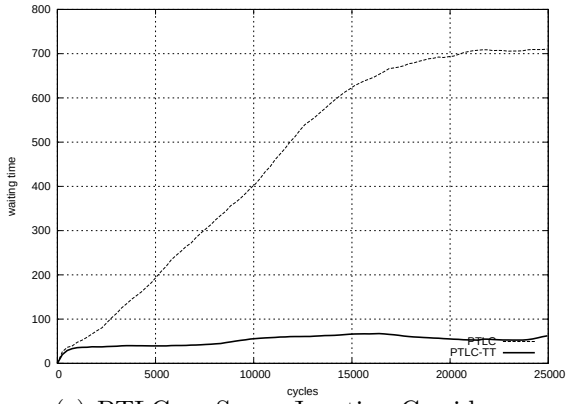
Figure 3: Performance Comparison of TLC Algorithms using Average Trip Waiting Time as the metric on two small road networks



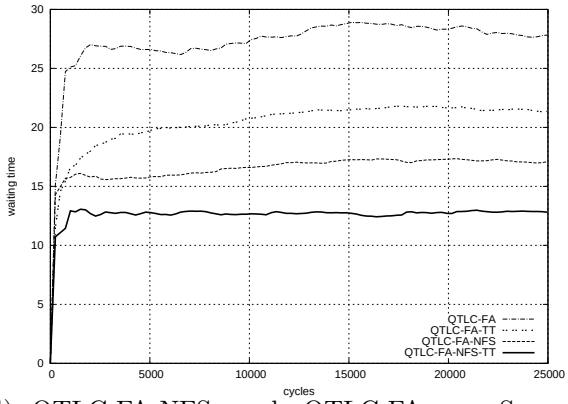
(a) PTLC on 4x4-Grid Network



(b) QTLC-FA-NFS and QTLC-FA on 4x4-Grid Network



(c) PTLC on Seven-Junction Corridor



(d) QTLC-FA-NFS and QTLC-FA on Seven-Junction Corridor

Figure 4: Performance Comparison of TLC Algorithms with their tuning counterparts on a 4x4-Grid and a Seven-Junction Corridor network

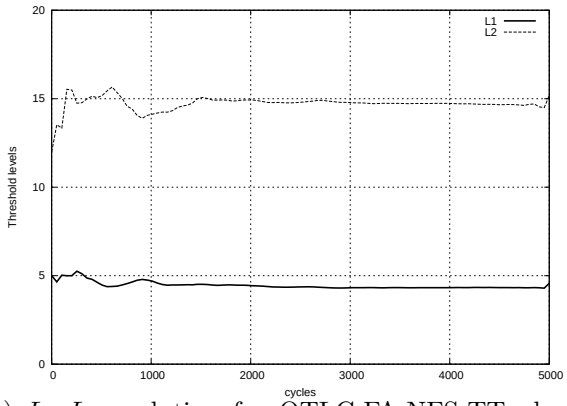
Table 3: Total Arrived Road Users (TAR) for various TLC algorithms on two road networks

	4x4-Grid Network	**Seven-Junction Corridor**
QTLC-FA-NFS	17408.56 ± 84.20	28137.27 ± 700.45
QTLC-FA-NFS-TT	22917.40 ± 80.83	30078.96 ± 698.25
PTLC	3807.14 ± 70.26	13705.74 ± 321.66
PTLC-TT	7662.64 ± 82.38	18699.59 ± 415.35
QTLC-FA	14418.95±80.15	21715.25 ± 511.17
QTLC-FA-TT	19369.61 ± 77.96	26832.66 ± 703.69

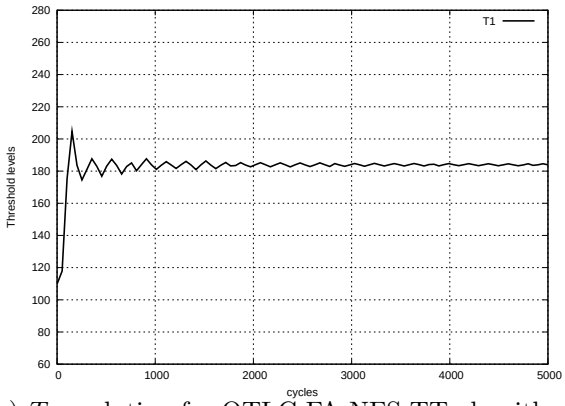
gorithms PTLC, QTLC-SA, QTLC-FA (see [1]) and QTLC-FA-NFS when combined with the threshold tuning algorithm (5). Figs. 4(a) – 4(d) show ATWT plots comparing PTLC, QTLC-FA and QTLC-FA-NFS algorithms with their tuning counterparts on a 4x4-grid network and a seven-junction corridor respectively. Table. 3 presents the total arrived road users (TAR) for the various TLC algorithms on a 4x4-grid network and a seven-junction corridor respectively. As mentioned before, since QTLC-SA is not implementable on high-dimensional state spaces (and hence, larger networks), performance comparisons of this algorithm are presented only on a single junction and a four-junction corridor.

We observe that incorporating our threshold tuning algorithm results in performance enhancements for all the TLC algorithms. Further, among the TLC algorithms studied, it is evident that QTLC-FA-NFS algorithm shows the best overall performance on all network settings considered. In particular, it outperformed the algorithm from [1], which establishes the superiority of the feature selection procedure of QTLC-FA-NFS over QTLC-FA from [1]. While PTLC-TT algorithm worked second-best to QTLC-FA-NFS-TT on the single-junction road network, on larger road networks it is found to perform poorly in comparison to other TLC algorithms.

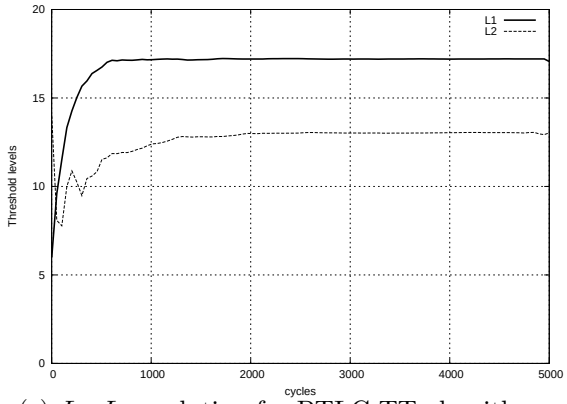
We also observe that the parameter θ converges to the optimal threshold value for each TLC algorithm - PTLC, QTLC-SA and QTLC-FA - considered in each of the road networks (See Fig. 2). This is illustrated by the convergence plots in Fig. 5. These plots as well as the ATWT plots also show that the transient phase of the threshold tuning algorithm (5), i.e., the initial period when the threshold parameter θ has not converged, is short.



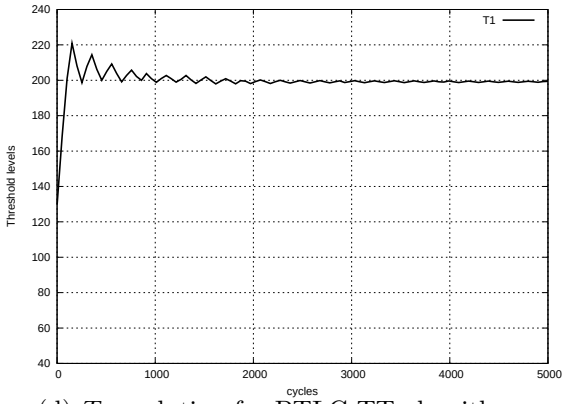
(a) L_1, L_2 evolution for QTLC-FA-NFS-TT algorithm



(b) T_1 evolution for QTLC-FA-NFS-TT algorithm



(c) L_1, L_2 evolution for PTLC-TT algorithm



(d) T_1 evolution for PTLC-TT algorithm

Figure 5: Convergence of θ parameter - illustration for QTLC-FA-NFS-TT and PTLC-TT algorithms in the case of a Four-Junction Corridor

7 Conclusions and Future Work

Our goal in this paper was to design an algorithm for tuning the thresholds to their optimal values in any graded threshold based TLC algorithm. Towards this end, we introduced and analyzed three new traffic light control algorithms with threshold tuning. We developed and applied, for the first time, an SPSA-based threshold tuning algorithm that incorporates Hadamard matrix based deterministic perturbation sequences and with proven convergence to the optimal thresholds for the various algorithms. We developed three new TLC algorithms - a priority-based TLC algorithm, an enhanced version of the Q-learning with full-state representation TLC algorithm that incorporates state aggregation and a Q-learning algorithm with function approximation that incorporates a novel feature selection scheme. We combined our threshold tuning algorithm with the above mentioned TLC algorithms. In the case of the Q-learning with full state representations, our threshold tuning algorithm finds an "optimal" way of clustering states based on the aforementioned thresholds and in the case of the Q-learning with function approximation based TLC, our algorithm amounts to feature adaptation for an RL algorithm using function approximation (a topic that is of independent research interest in the RL community). Empirical observations indicate a significant gain in performance when our threshold tuning algorithm is used in conjunction with each of the TLC algorithms considered. The tuning variants of the TLC algorithms (above) clearly outperform their counterparts that use (arbitrarily set) fixed thresholds.

References

[1] L. A. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412 – 421, 2011.

[2] S. Bhatnagar, M. Fu, S. Marcus, and I. Wang, "Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 13, no. 2, pp. 180–209, 2003.

[3] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[4] M. Papageorgiou, M. Ben-Akiva, J. Bottom, P. Bovy, S. Hoogendoorn, N. Hounsell, A. Kotsialos, and M. McDonald, "ITS and traffic management," *Handbooks in Operations Research and Management Science*, vol. 14, pp. 715–774, 2007.

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

[6] D. Robertson and R. Bretherton, "Optimizing networks of traffic signals in real time-the SCOOT method," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, Part 2, pp. 11–15, 1991.

[7] A. Sims and K. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[8] M. Girianna and R. Benekohal, "Using genetic algorithms to design signal coordination for oversaturated networks," *Journal of Intelligent Transportation Systems*, vol. 8, no. 2, pp. 117–129, 2004.

[9] J. Sanchez-Medina, M. Galan-Moreno, and E. Rubiyo-Royo, "Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 132–141, 2010.

[10] J. Spall and D. Chin, "Traffic-responsive signal timing for system-wide traffic control," *Transportation Research Part C*, vol. 5, no. 3-4, pp. 153–163, 1997.

[11] D. Srinivasan, M. Choy, and R. Cheu, "Neural networks for real-time traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261–272, 2006.

[12] J. Wei, A. Wang, and N. Du, "Study of self-organizing control of traffic signals in an urban network based on cellular automata," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 2, pp. 744–748, 2005.

[13] X. Yu and W. Recker, "Stochastic adaptive control model for traffic signal systems," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 4, pp. 263–282, 2006.

[14] N. Gartner, "Opac: A demand-responsive strategy for traffic signal control," *Transportation Research Record*, no. 906, 1983.

[15] J. Henry, J. Farges, and J. Tuffal, "The PRODYN real time traffic algorithm," in *Int. Fed. of Aut. Control (IFAC) Conf*, 1984.

[16] F. Boillot, J. Blosseville, J. Lesort, V. Motyka, M. Papageorgiou, and S. Sellam, "Optimal signal control of urban traffic networks." in *Proc. 6th IEE Int. Conf. Road Traffic Monitoring and Control*, 1992, pp. 75–79.

[17] S. Sen and K. Head, "Controlled optimization of phases at an intersection," *Transportation science*, vol. 31, no. 1, pp. 5–17, 1997.

[18] B. Abdulhai, R. Pringle, and G. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, p. 278, 2003.

[19] B. Gokulan and D. Srinivasan, "Distributed geometric fuzzy multiagent urban traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 714–727, 2010.

[20] W. Lin and C. Wang, "An enhanced 0-1 mixed-integer LP formulation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 238–245, 2004.

[21] T. Li, D. Zhao, and J. Yi, "Adaptive dynamic programming for multi-intersections traffic signal intelligent control," in *11th International IEEE Conference on Intelligent Transportation Systems, 2008. ITSC 2008*, 2008, pp. 286–291.

[22] L. De Oliveira and E. Camponogara, "Multi-agent model predictive control of signaling split in urban traffic networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 1, pp. 120–139, 2010.

[23] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *Intelligent Transport Systems, IET*, vol. 4, no. 2, pp. 128–135, 2010.

[24] A. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, 2008, pp. 560–566.

[25] B. Bakker, M. Steingrover, R. Schouten, E. Nijhuis, and L. Kester, "Cooperative multi-agent reinforcement learning of traffic lights," in *Proceedings of the Workshop on Cooperative Multi-Agent Learning, European Conference on Machine Learning, ECML*, vol. 5, 2005, p. 65.

[26] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[27] ——, "A one-measurement form of simultaneous perturbation stochastic approximation," *Automatica*, vol. 33, no. 1, pp. 109–112, 1997.

[28] ——, "Adaptive stochastic approximation by the simultaneous perturbation method," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1839–1853, 2000.

[29] S. Bhatnagar, "Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 15, no. 1, pp. 74–107, 2005.

[30] ——, "Adaptive Newton-based multivariate smoothed functional algorithms for simulation optimization," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 18, no. 1, pp. 1–35, 2007.

[31] P. J. Schweitzer, "Perturbation theory and finite Markov chains," *Journal of Applied Probability*, vol. 5, pp. 401–413, 1968.

[32] V. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Cambridge University Press, 2008.

[33] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996.

[34] V. S. Borkar, *Probability Theory: An Advanced Course*. New York: Springer, 1995.

[35] M. W. Hirsch, "Convergent activation dynamics in continuous time networks," *Neural Networks*, vol. 2, pp. 331–349, 1989.

[36] H. Kushner and D. Clark, *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag New York, 1978, vol. 6.

[37] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman, "Simulation and optimization of traffic in a city," in *IEEE Intelligent Vehicles Symposium*, June 2004, pp. 453–458.

[38] S. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," *Advances in Applied Self-organizing Systems*, pp. 41–50, 2008.