Threshold Tuning Using Stochastic Optimization for Graded Signal Control

Prashanth L. A. and Shalabh Bhatnagar, *Senior Member, IEEE*

***Abstract*—Adaptive control of traffic lights is a key component of any intelligent transportation system. Many real-time traffic light control (TLC) algorithms are based on graded thresholds, because precise information about the traffic congestion in the road network is hard to obtain in practice. For example, using thresholds L_1 and L_2, we could mark the congestion level on a particular lane as "low," "medium," or "high" based on whether the queue length on the lane is below L_1, between L_1 and L_2, or above L_2, respectively. However, the TLC algorithms that were proposed in the literature incorporate fixed values for the thresholds, which, in general, are not optimal for all traffic conditions. In this paper, we present an algorithm based on stochastic optimization to tune the thresholds that are associated with a TLC algorithm for optimal performance. We also propose the following three novel TLC algorithms: 1) a full-state Q-learning algorithm with state aggregation, 2) a Q-learning algorithm with function approximation that involves an enhanced feature selection scheme, and 3) a priority-based TLC scheme. All these algorithms are threshold based. Next, we combine the threshold-tuning algorithm with the three aforementioned algorithms. Such a combination results in several interesting consequences. For example, in the case of Q-learning with full-state representation, our threshold-tuning algorithm suggests an optimal way of clustering states to reduce the cardinality of the state space, and in the case of the Q-learning algorithm with function approximation, our (threshold-tuning) algorithm provides a novel feature adaptation scheme to obtain an "optimal" selection of features. Our tuning algorithm is an incremental-update online scheme with proven convergence to the optimal values of thresholds. Moreover, the additional computational effort that is required because of the integration of the tuning scheme in any of the graded-threshold-based TLC algorithms is minimal. Simulation results show a significant gain in performance when our threshold-tuning algorithm is used in conjunction with various TLC algorithms compared to the original TLC algorithms without tuning and with fixed thresholds.**

***Index Terms*—Deterministic perturbation sequences, intelligent transportation systems, simultaneous perturbation stochastic approximation (SPSA), stochastic optimization, threshold tuning, traffic signal control.**

I. 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 has 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 exceeds 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 case is because of the variability in traffic patterns and the lack of precise state information. Many times, we only have 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 feedback policies. For example, for some suitably chosen threshold levels L_1 and L_2 for each lane with $L_1 < L_2$, we 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, whereas 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 one problem is to optimally select such thresholds. Graded-threshold policies have been considered, for example, in [1]. These policies, however, consider fixed values for the thresholds. One 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, optimally set. To obtain optimal thresholds, we develop an algorithm that works in conjunction with the given TLC algorithm and tunes the thresholds to optimize a given cost objective. The problem is shown to be equivalent to the problem 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, we design an efficient simultaneous-perturbation stochastic-approximation (SPSA)-based online

Manuscript received February 15, 2012; revised May 21, 2012 and June 22, 2012; accepted July 13, 2012. Date of publication July 23, 2012; date of current version November 6, 2012. This work was supported in part by the Department of Information Technology, Government of India, through the Automation Systems Technology Center. The review of this paper was coordinated by Dr. A. Chatterjee.

The authors are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India (e-mail: prashanth@csa.iisc.ernet.in; shalabh@csa.iisc.ernet.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2012.2209904

threshold-tuning algorithm that works with any graded-threshold-based TLC algorithm, and we propose the following three new TLC algorithms: 1) a variant of Q-learning with full-state representation that incorporates a novel state aggregation scheme, 2) a function-approximation-based Q-learning TLC algorithm that enhances the scheme that was proposed in [1] by incorporating a new feature selection procedure, and 3) a novel priority-based algorithm. We study our threshold-tuning algorithm in conjunction with these TLC algorithms. We describe our contributions in the following paragraphs.

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. Furthermore, 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. In particular, we incorporate the one-simulation variant of the SPSA algorithm that uses Hadamard-matrix-based deterministic perturbations based on [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 aforementioned three new threshold-based TLC algorithms. Based on the simulation experiments, we observe that our tuning algorithm is easily implementable over all road network settings and rapidly converges to the optimal thresholds in any graded-threshold-based TLC algorithm. Furthermore, the additional computational effort that is required in finding the optimal thresholds over algorithms with fixed thresholds is observed to be small.

Among 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 large state spaces than regular Q-learning (with full-state representations). The combination of Q-learning-based TLC with state aggregation is shown 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). Furthermore, we develop a Q-learning-based TLC algorithm with function approximation along the lines of [1], but 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 that were used in [1], in which the state and the action components of the features are 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 that was proposed in [1]. The priority-based TLC (PTLC) that we propose assigns priorities to the various sign configurations based on graded thresholds.

The combination of our threshold-tuning algorithm with the other aforementioned TLC algorithms results in several interesting consequences. For example, in the case of Q-learning with full-state representation, our threshold-tuning algorithm results in finding an optimal way of clustering states 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.

II. Literature Survey

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

A. Traffic Signal Control

We briefly review some TLC strategies that have been proposed in the literature. See [3] and [4] for a detailed discussion of relevant traffic signal control literature. In [5], an offline traffic signal control technique where the signal timings are generated using a static optimizer has been proposed. One popular version of this system that has been widely adapted is the Split-Cycle Offset Optimization Technique (SCOOT) [6]: a traffic-responsive signal control scheme that uses traffic volume and occupancy as input. The Sydney Coordinated Adaptive Traffic System (SCATS) [7] is another well-deployed scheme that picks one of the precalculated controls based on the traffic state. Several online TLC algorithms that adapt in real time have also been proposed. For example, genetic algorithms [8], neural-network-based algorithms [9]–[11], algorithms that are based on cellular automata [12], stochastic control [13], and dynamic optimization [14]–[16], and RL [1], [17] are all online schemes. In [14]–[16], 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 in [9] and [10] to optimize traffic signal timings.

An interesting account of the evolution of traffic signals, i.e., devices for traffic control, is provided in [18]. In [19], an SPSA-like algorithm that incorporates linear function approximation, in addition, has been proposed for the adaptive optimization of control systems and applied to tune the split module parameters in a traffic-responsive control strategy. Random perturbations are used in the algorithm that was proposed in [19], where the authors note that the basic SPSA algorithm with random perturbations did not perform well. On the other hand, in this paper, we use one-simulation SPSA with deterministic perturbations based on certain Hadamard matrices, and this choice ensures convergence and good empirical performance of the scheme. In [20], the algorithm that was based on [19] has been combined with neural networks. The resulting algorithm has been studied in the context of optimizing the split and cycle times of traffic control strategy through simulation in [21]. The

adaptive-control-based TLC algorithm that was proposed in [13] incorporates a Markov decision process (MDP) formulation. However, this approach requires a precise model of the system, which, in general, is hard to obtain in realistic settings. In [22], a method for automatic red light runner detection on a video is presented. In [23], a three-tier open TLC model is proposed to smooth vehicles' travel to minimize the usage of fuel. An RL-based TLC algorithm that was proposed in [17] uses full-state representations with a high computational complexity. This algorithm has been studied only for an isolated traffic junction scenario in [17]. The RL-based TLC algorithm in [1] incorporates function approximation together with certain fixed graded-feedback policies and is shown to perform well over many road network scenarios, with as many as eight or nine junctions involving nearly 10^{90} states. Unlike RL algorithms that are based on full-state representations [17], the computational effort that is required by the algorithm in [1] remains reasonable, even for large-scale networks, because of the use of function approximation. RL techniques have also been applied in other applications, for example, in [24] for high-speed road following for high-curvature roads.

B. Stochastic Optimization

One popular approach for simulation-based parameter optimization is SPSA, as proposed in [25]. SPSA is an efficient gradient search technique that aims at finding a local minimum of a performance objective. The regular SPSA algorithm randomly perturbs the parameter vector using independent and identically distributed, symmetric, zero-mean random variables and has the critical advantage that it needs only two samples of the objective function for any N-D parameter. A variant of the SPSA algorithm that uses one simulation was proposed in [26]. Unlike its two-simulation counterpart, the algorithm in [26] 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, which show performance improvements over their randomized perturbation counterparts. The perturbation variables are based on either lexicographic or Hadamard matrix 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 [26]. A Newton-based SPSA algorithm that requires four system simulations with Bernoulli random perturbations was proposed in [27]. In [28], three SPSA-based estimates of the Hessian that require three, two, and one system simulation(s), respectively, were proposed. In [29], certain smoothed functional Newton algorithms that incorporate Gaussian-based perturbations were proposed. It is, however, the case that Newton-based algorithms, although more accurate than gradient-based schemes, require significantly higher computational effort that results from projecting a Hessian update at each iteration to the set of positive definite and symmetric matrices and 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.

C. Organization

The rest of this paper is organized as follows. In Section III, we describe in detail the problem framework. In Section IV, we present our threshold-tuning algorithm. In Section V, we present our algorithms for traffic signal control and combine them with the threshold-tuning algorithm. In Section VI, we discuss the implementation of the various TLC schemes, where our algorithm was used to find the optimal thresholds in each scheme, and present the performance simulation results. Finally, in Section VII, we provide the concluding remarks.

III. 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 the following two important components: 1) a threshold-based sign configuration policy obtained from a TLC algorithm for a fixed set of thresholds and 2) another algorithm that operates on top of the TLC algorithm itself for tuning the thresholds. A sign configuration here refers to all the signals that are associated with a phase, i.e., signals that can simultaneously be switched to green. The TLC algorithms that we consider indicate when to switch a sign configuration and are all based on graded thresholds. The threshold-tuning algorithm, which is common to all the schemes, tunes the graded feedback policy to find the optimal parameters (thresholds) and does not indicate how we can switch the sign configurations, a task that is performed by the particular TLC algorithm used. Apart from designing efficient TLC algorithms, one important problem that we address is to find the "optimal" set of parameters to use for a given TLC algorithm and that gives 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.

Each TLC algorithm that we consider uses as input the queue lengths along the individual lanes that lead to the intersection and the time elapsed since the last signal light has switched 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, whereas 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 signaled 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 the elapsed time on lane i at time n. The aforementioned state

formulation is valid for all the TLC algorithms, except for 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 are given in Section V-A.

The action a_n that is 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 suitably chosen to incorporate prioritization of traffic. For example, traffic on main roads is given a higher priority than on side roads. Let I_p denote the set of indices of prioritized lanes, i.e., whose traffic is given higher priority. We let the cost $k(s_n, a_n)$ have the form

$$k(s_n, a_n) = \alpha_1 * \left(\sum_{i \in I_p} \alpha_2 * q_i(n) + \sum_{i \notin I_p} \beta_2 * q_i(n) \right) + \beta_1 * \left(\sum_{i \in I_p} \alpha_2 * t_i(n) + \sum_{i \notin I_p} \beta_2 * t_i(n) \right) \quad (2)$$

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

The sign configuration policy that governs 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 V.

We let the parameter vector θ take values in a compact set $C \stackrel{\Delta}{=} [L_{\min}, L_{\max}] \times [L_{\min}, L_{\max}] \times [T_{\min}, T_{\max}] \subset \mathcal{R}^3$ by making use of the projection operator π, as will be defined later, where $L_{\min}$, $L_{\max}$, $T_{\min}$, and $T_{\max}$ are certain prescribed thresholds. The objective is to find a θ that minimizes

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

The actions a_j are assumed to be governed by one of the policies that we present as follows, which, in turn, will be parameterized by the threshold parameter θ. Although it is desirable to find a $\theta^* \in C$ that minimizes $J(\theta)$, it is, in general, difficult to achieve a global minimum. We therefore use a local optimization method, for which we need 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 a bounded second derivative.

Note that (A1) is a technical requirement that is used to push through a Taylor's argument in the convergence analysis (see Section VII). In the case of finite-state parameterized Markov chains, we can show, using a similar argument as given in [30], 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, whereas 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 shown to exhibit significant performance improvements.

IV. Threshold-Tuning Algorithm

A stochastic iterative algorithm for finding 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 aforementioned procedure that involves two loops has to iteratively be performed until the parameter θ converges to a local minimum. However, such a procedure would typically be computationally expensive, considering 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 multiple-time-scale stochastic approximation procedure [31, Ch. 6], we circumvent this problem, because 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.,

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

which incorporates a Hadamard-matrix-based deterministic construction for the perturbations $\triangle$. One-simulation SPSA gradient estimates based on randomized perturbations were proposed in [26], but these are shown 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. The algorithm that was based on [2] was shown to yield good performance and has significantly lower bias compared with the algorithm in [26].

Fig. 1 illustrates the operation of the threshold-tuning algorithm. In essence, it is a closed-loop procedure where the

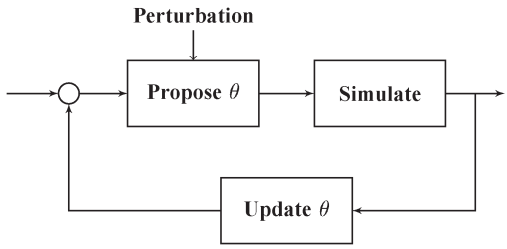

Fig. 1. Operation of the threshold-tuning algorithm.

system is simulated for a perturbed parameter value $(\theta + \delta\triangle)$ and where the average cost estimates that were obtained through 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), shown below. The threshold-tuning algorithm is given as follows:

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

The quantities used in (5) are the following:

- $L_1(n)$, $L_2(n)$, and $T_1(n)$ denote the nth 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 used to obtain the sign configuration policy on the faster time scale. These updates will be explained in the TLC algorithms in the next section.
- $L \geq 1$ is a fixed parameter that controls the rate of update of θ in relation to that of $\tilde{Z}$. This parameter allows for the 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, although convergence can be proven for any value of L, including 1.
- $\delta > 0$ 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 as described in Section IV-A.
- $\pi : \mathcal{R}^3 \to C$ is the projection operator defined by $\pi(\theta) \stackrel{\Delta}{=} (\pi_1(\theta_1), \pi_1(\theta_2), \pi_2(\theta_3))^T$, $\theta \in \mathcal{R}^3$. Here, for any $x \in \mathcal{R}$, $\pi_1(x) \stackrel{\Delta}{=} \min(\max(L_{\min}, x), L_{\max})$, and $\pi_2(x) \stackrel{\Delta}{=} \min(\max(T_{\min}, x), T_{\max})$, respectively.

Remark 1: Our threshold-tuning algorithm is different from the algorithms in [19] and [20] in many ways. An expected cost objective is considered in the latter approach, and the objective function is parameterized, in the spirit of RL, using a linear function approximation architecture, with the parameter being tuned in SPSA-like fashion, whereas in our case, we consider the long-run average cost objective (3). Furthermore, we directly consider the objective function without resorting to parameterized approximations of the cost, and using SPSA with deterministic perturbations, we ensure convergence to a locally optimal point of the objective. Note that our tuning algorithm requires only one simulation for estimating the objective function, unlike the algorithms that were proposed in [19] and [20]. In addition, we provide three new TLC algorithms, all based on graded thresholds, and combine our threshold-tuning algorithm with these TLC algorithms.

The complete algorithm is described as follows.

Algorithm 1: 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$(\theta) \to X$: 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: $\theta^* \stackrel{\Delta}{=} \theta_R$.

```
θ ← θ_0, n ← 1
loop
  X̂ ← Simulate(θ + δΔ)
  UpdateAverageCost()
  if n%L = 0 then
    UpdateTheta()
  end if
  n ← 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): We have

$$\sum_n a(n) = \sum_n b(n) = \infty, \sum_n \left(a^2(n) + b^2(n)\right) > \infty$$

$$\text{and } \lim_{n\to\infty} \frac{a(n)}{b(n)} = 0.$$

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

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

with $(1/2) < \alpha < 1$, $0 < \hat{a}$, and $\hat{b} < \infty$.

A. 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. Furthermore, 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 described as follows.

For $k = 1$, we have

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

and for general $k > 1$, we have

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

Let $P = 2^{\lceil log_2(N+1) \rceil}$, where, as aforementioned, N is the parameter dimension. This implies $P \geq N + 1$. Now, construct a normalized Hadamard matrix H_P of order P using the aforementioned 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, whereas all the other columns have an equal number of $+1$ and -1 elements. The latter property aids in canceling the bias terms. Form a new matrix $\widetilde{H}_P$ of order $P \times N$, with $h(1), \ldots, h(N)$ being 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 in the next section) in the three algorithms. When combined with the threshold-tuning algorithm (5), these algorithms represent the most effective TLC strategies in their respective classes. The first two algorithms that we present in the following section are based on Q-learning and incorporate state aggregation and function approximation, respectively, whereas the last algorithm is a priority-based scheme.

V. Traffic Light Control Algorithms

A. Q-Learning Traffic Light Control With State Aggregation (QTLC-SA)

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

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

for all feasible tuples (i, a) of states and actions. Here, j is the next state after state i that is observed through simulation and follows the distribution $p(i, ., a)$. In addition, a (respectively, b) is a feasible action in state i (respectively, j), and $0 < \gamma < 1$ is a given discount factor. Traditional methods for solving the MDP (see [32]) formulated in the previous section would require the specification of $p(i, ., a)$, whereas 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 arbitrarily be initialized, and 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 [32] requires the identification of states, actions, and costs. Although the actions and single-stage cost function were identified in Section III, the following state formulation in our algorithm is different and uses state aggregation to tackle the curse of dimensionality. Consider the process $\{s'_n(\theta), n \geq 0\}$, parameterized by $\theta = (L_1, L_2, T_1)^T$, which is defined as follows:

$$s'_n(\theta) = (q'_1(n), \ldots, q'_K(n), t'_1(n), \ldots, t'_K(n)) \quad (7)$$

where

$$q'_i(n) = \begin{cases} 0, & \text{if } q_i(n) < L_1 \\ 0.5, & \text{if } L_1 \leq q_i(n) \leq L_2 \\ 1, & \text{if } q_i(n) > L_2 \end{cases}$$

$$t'_i(n) = \begin{cases} 0, & \text{if } t_i(n) \leq T_1 \\ 1, & \text{if } t_i(n) > T_1. \end{cases}$$

Note that, as aforementioned, $q_i(n)$ and $t_i(n)$ denote the waiting queue lengths and elapsed times that correspond to the lane i at time n, respectively. 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 $s'_n(\theta)$, $n \geq 0$ to make computation more manageable. The problem of the curse of dimensionality associated with large state spaces is the primary reason that regular Q-learning with full-state representations [cf., recursion (6)] is not implementable in the case of large networks; see [1].

The Q-learning algorithm (6) when applied to our setting with state aggregation will be referred to as the QTLC-SA algorithm.

1) QTLC-SA-TT: The Q-learning traffic light control with state aggregation with threshold tuning (QTLC-SA-TT) algorithm is a two-time-scale stochastic approximation algorithm that updates θ on the slower time scale and learns the optimal sign configuration policy (parameterized by θ) on the faster time scale. 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 $m = nL, \ldots, (n+1)L - 1$, we have

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

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

where the following conditions hold.

- $\{\hat{s}_l(\theta)\}$ denotes the aggregated state-valued process that is governed by $\{\hat{\theta}_l\}$, where $\hat{\theta}_l = \theta_n + \delta\triangle(n)$ for $n = [(l/L)]$, with a given $L \geq 1$.
- Although 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 based on the policy that is suggested by the Q-value update.
- The perturbations $\triangle(n)$ are generated using Hadamard matrices, as outlined in Section IV-A.

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

2) Assumption (A3):

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

B) 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.

B. 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 fully be controlled for large networks with high-dimensional states, because the cardinality of the aggregated state space significantly increases 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 that was developed therein is a computationally efficient TLC algorithm that is based on Q-learning. However, the state–action features that were used in [1] do not incorporate 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, the elapsed-time estimate, and the sign configuration portion, respectively, for each lane of the road network. Although each of these attributes is important, the approximation architecture used in [1] did not take into account the dependence between features. In the novel feature selection scheme that we propose, 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, compared with the scheme in [1]. Numerical experiments show that this new choice of features significantly outperforms the one used in [1]. Various aspects of the Q-learning TLC algorithm that incorporates function approximation with the new feature selection scheme are now described.

In the following algorithm, 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:

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

where ω_0 is arbitrarily set. In (10), the action a_n is chosen in state s_n according to $a_n = arg\min_{v \in A(s_n)} \omega_n^T \sigma_{s_n,v}$, whereas the features are

$$\sigma_{s_n,a_n} = (\sigma_1(n), \ldots, \sigma_K(n))^T$$

where the scheme for the selection of the feature value $\sigma_i(n)$ that corresponds to lane i when action a is chosen is explained in Table I.

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, between L_1 and L_2, or above L_2, whether the elapsed time is below T_1 or above it, as well as whether the sign configuration indicates a red or green light for the lane. For example, if both the queue length and the 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

TABLE I
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

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 that correspond 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, which is unlike [1], where such dependence was not incorporated.

Although the cardinality of the state–action space may be so high such that storing or updating Q-values for each (s, a)-tuple may be impossible, the aforementioned feature-based algorithm estimates Q-values using the parameterization (9), making its implementation feasible, even on large road networks, because 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 will be referred to as the QTLC-FA-NFS algorithm.

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

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

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

The action $\hat{a}_m$ in (11a) and (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.

TABLE II
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

2) Assumption (A4):

A) For any given $\theta \in C$, the basic underlying process $\{s_n, n \geq 1\}$ is an MDP.
B) For any given policy and parameter $\theta \in C$, the process $\{s_n, n \geq 1\}$ is ergodic Markov.

C. 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, because the last switchover of lights to red (T_1) on individual lanes. The cost that is assigned to each lane is decided based on whether the queue length on that lane is below L_1, between L_1 and L_2, or above L_2 at any instant and also on whether the elapsed time is below or above T_1. For example, if both the queue length and the 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 II. 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 approach helps combine efficiency with fairness.

1) PTLC-TT: Similar to the previous TLC algorithms, we combine the threshold-tuning algorithm (5) with PTLC to obtain the priority-based traffic light control with threshold tuning (PTLC-TT) algorithm. The state-valued process $\{\hat{s}_n, n \geq 0\}$ in this case under the aforementioned priority-based policy 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 time-scale recursions here are given as follows. For $m = nL, \ldots, (n+1)L - 1$, we have

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

The aforementioned action $\hat{a}_m$ is selected in state $\hat{s}_m$ based on the aforementioned priority assignment policy, i.e., select the sign configuration with 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.

2) Assumption (A5):

A) The basic underlying process $\{\hat{s}_n, n \geq 0\}$ is a parameterized MDP.
B) For any given $\theta \in C$ and the specified priority-based policy, $\{\hat{s}_n, n \geq 0\}$ is ergodic Markov.

D. Convergence Result

Let $\bar{\pi}(x) = (\bar{\pi}_1(x_1), \bar{\pi}_1(x_2), \bar{\pi}_2(x_3))^T$ for any $x = (x_1, x_2, x_3)^T \in \mathcal{R}^3$. Define now the operators $\bar{\pi}_1$ and $\bar{\pi}_2$ as follows:

$$\bar{\pi}_1(v(x)) = \lim_{\eta \to 0} \left(\frac{\pi_1(x + \eta v(x)) - x}{\eta} \right)$$
$$\bar{\pi}_2(w(x)) = \lim_{\eta \to 0} \left(\frac{\pi_2(x + \eta w(x))\, x}{\eta} \right) \quad (13)$$

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 (13) exist and are unique, because $[L_{\min}, L_{\max}]$ and $[T_{\min}, T_{\max}]$ are convex sets. Based on the definition, $\bar{\pi}_1(v(x)) = v(x)$ (respectively, $\bar{\pi}_2(w(y)) = w(y)$) if $x \in (L_{\min}, L_{\max})$ (respectively, $y \in (T_{\min}, T_{\max})$.

Consider the following ordinary differential equation (ODE):

$$\dot{\theta}(t) = \bar{\pi}(-\nabla J(\theta(t))). \quad (14)$$

Let $R \subset I = \{\theta \in C | \nabla J(\theta) = 0\}$ denote the set of stable fixed points of (14). Note that I denotes the set of all fixed points of (14) that would include not only stable equilibriums but also unstable equilibriums such as local maxima and saddle points. The set of stable equilibriums (i.e., the local minima) is, in general, a subset of I. For $\eta > 0$, let $R^\eta \triangleq \{\theta \in C | \|\theta - \theta_0\| < \eta$ for some $\theta_0 \in R\}$ denote the set of points that are within an η-neighborhood of R.

Theorem 1: Given $\eta > 0$, there exists $\delta_0 > 0$ such that, for all $\delta \in (0, \delta_0)$, the recursions $\theta(n)$ almost surely converge to R^η.

The proof of the aforementioned theorem is given in the Appendix.

VI. 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 [33]. 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 Q-learning with function approximation and threshold tuning (QTLC-FA-TT), which we also implemented for comparison.

A. QTLC-FA-TT

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

$$\sigma_{s_n,a_n} = \big(\sigma_{q_1(n)}, \ldots, \sigma_{q_K(n)}, \sigma_{t_1(n)}, \ldots, \sigma_{t_K(n)} \sigma_{a_1(n)}, \ldots, \sigma_{a_m(n)}\big)^T \quad (15)$$

where

$$\sigma_{q_i(n)} = \begin{cases} 0, & \text{if } q_i(n) < L_1 \\ 0.5, & \text{if } L_1 \leq q_i(n) \leq L_2 \\ 1, & \text{if } q_i(n) > L_2 \end{cases}$$
$$\sigma_{t_i(n)} = \begin{cases} 0, & \text{if } t_i(n) \leq T_1 \\ 1, & \text{if } t_i(n) > T_1. \end{cases}$$

Furthermore, $\sigma_{a_1(n)}, \ldots, \sigma_{a_m(n)}$ correspond to the sign configurations that were 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 in 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 based on the aforementioned QTLC-FA algorithm.

B. Implementation

We implemented the threshold-tuning algorithm for all four algorithms that incorporate graded thresholds, i.e., 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, which involved fixed thresholds (no tuning). The algorithms studied are outlined as follows.

- **PTLC**. This graded-threshold-based TLC algorithm, as described in Section V-C, uses a priority assignment scheme based on waiting queue lengths and elapsed times 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.
- **PTLC-TT**. This algorithm combines threshold tuning with the aforementioned PTLC. 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 V-C.
- **QTLC-SA**. This Q-learning-based TLC algorithm incorporates state aggregation and is described in Section V-A. Note that this algorithm uses fixed thresholds and the performance of this algorithm is compared against its tuning variant (QTLC-SA-TT).
- **QTLC-SA-TT**. This algorithm combines threshold tuning with the aforementioned QTLC-SA TLC. 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 V-A.

- **QTLC-FA**. This is the TLC algorithm presented in [1] and uses Q-learning with function approximation.
- **QTLC-FA-TT**. This is the tuning variant of the aforementioned QTLC-FA algorithm 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 I. This algorithm is described in detail in Section V-B and uses fixed thresholds. Its performance is compared to its tuning variant QTLC-FA-NFS-TT.
- **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, i.e., a single junction road network, a four-junction corridor, a 4×4 grid network, and a network of nine signalized junctions with 24 roads around the Indian Institute of Science campus in Bangalore (hereafter referred to as the IISc network), respectively, to test 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) and large-size road networks (such as the 4×4-grid and the IISc network). The 4×4-grid network consists of 16 edge nodes (where traffic is generated), 16 junctions with traffic lights, and 40 roads, with each being four lanes wide and, when full, capable of housing up to 1500 vehicles. Furthermore, the cardinality of the state–action space is on the order of 10^{130} for the 4×4-grid network. We tested the performance of all the aforementioned TLC algorithms and their tuning counterparts on a single junction and four-junction corridor. However, on a 4×4-grid network and the IISc network, we could not implement the QTLC-SA algorithm. As explained, QTLC-SA uses full-state representations and hence is not implementable on high-dimensional state–action spaces (as with the 4×4-grid network and the IISc network). However, QTLC-SA is an enhancement to the Q-learning based TLC algorithm proposed, for example, in [17]. In fact, the Q-learning based TLC with full-state representations in [17] was proposed only for a single-junction road network. The 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 25 000 cycles for all algorithms, and in each simulation, the destination of the road user is randomly fixed (using a discrete uniform distribution). At each time step, vehicles are inserted into the road network based on the spawn frequencies of the edge nodes, where spawn frequency specifies the rate at which traffic is randomly generated in GLD. The spawn frequencies of the edge nodes are set in a way that ensures that the proportion of cars that flow on the main road to the cars that flow 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 example, in [34]. The following results 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 weighting to the queue length and elapsed-time components of (2) while according higher weighting to the main-road traffic over the 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 aforementioned 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.

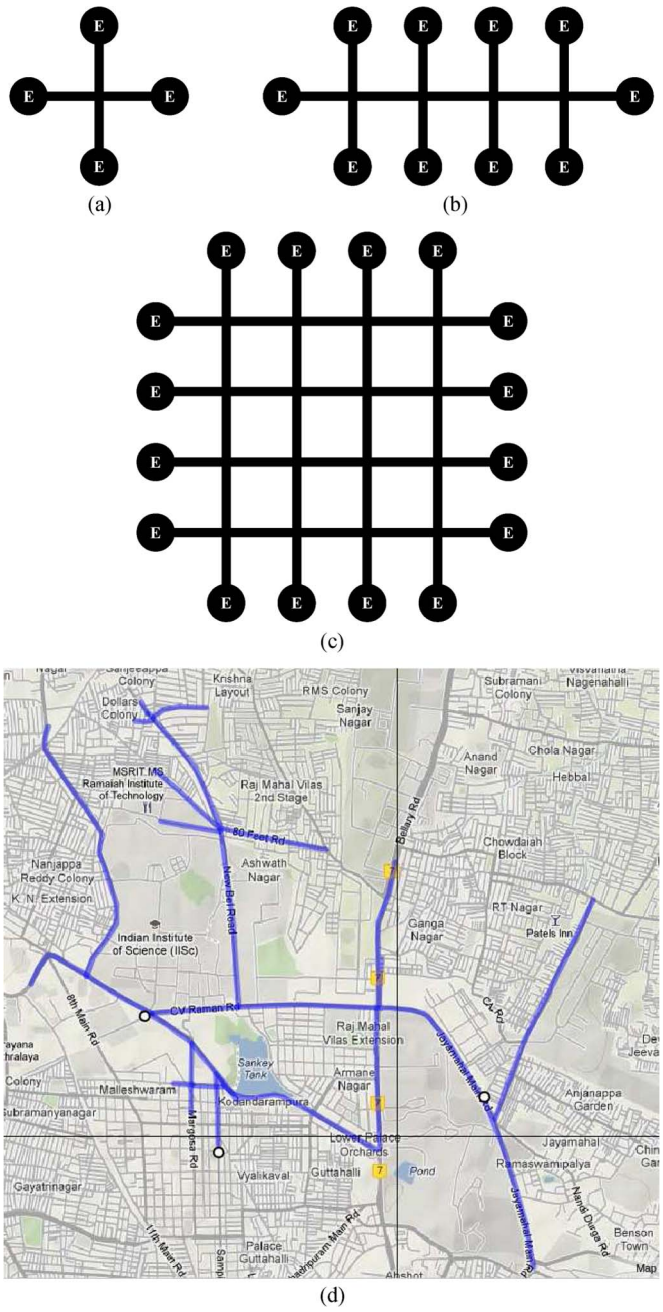

Fig. 2. Road networks used for our experiments. (a) Single junction. (b) Four-junction Corridor. (c) 4×4 grid network. (d) IISc network.

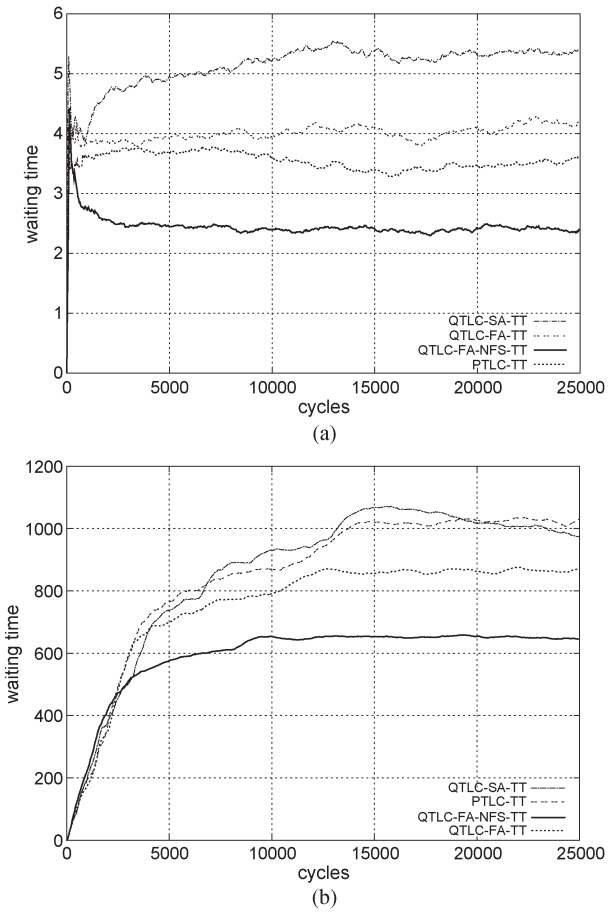

Fig. 3. Performance comparison of the TLC algorithms using ATWT as the metric on two small road networks. (a) Single junction. (b) Four-junction corridor.

C. Results

Fig. 3 shows the average trip waiting time (ATWT) plots on a single- and a four-junction corridor, respectively, and compares the four TLC algorithms, i.e., PTLC, QTLC-SA, QTLC-FA, and QTLC-FA-NFS, when combined with the threshold-tuning algorithm (5). Fig. 4(a)–(d) shows ATWT plots that compare the PTLC, QTLC-FA, and QTLC-FA-NFS algorithms with their tuning counterparts on a 4×4-grid network and the IISc network, respectively. Table III(b) presents the total arrived road users (TAR) for the various TLC algorithms on a 4×4-grid network and the IISc network, respectively. As aforementioned, because 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- and a four-junction corridor. The choice of weights $\alpha_1 = \beta_1 = 0.5$ is justified by the results obtained for different choices of α_1 in Table III(a).

Although the default setting in GLD for the traffic arrival pattern is uniform, we also conducted an experiment where the traffic arrived according to a Poisson process. The results from this experiment on two road networks for the PTLC and QTLC-FA-NFS algorithms are presented in Fig. 5. We observe that, even in this case, our tuning scheme results in a performance improvement for both the algorithms. This result is significant,

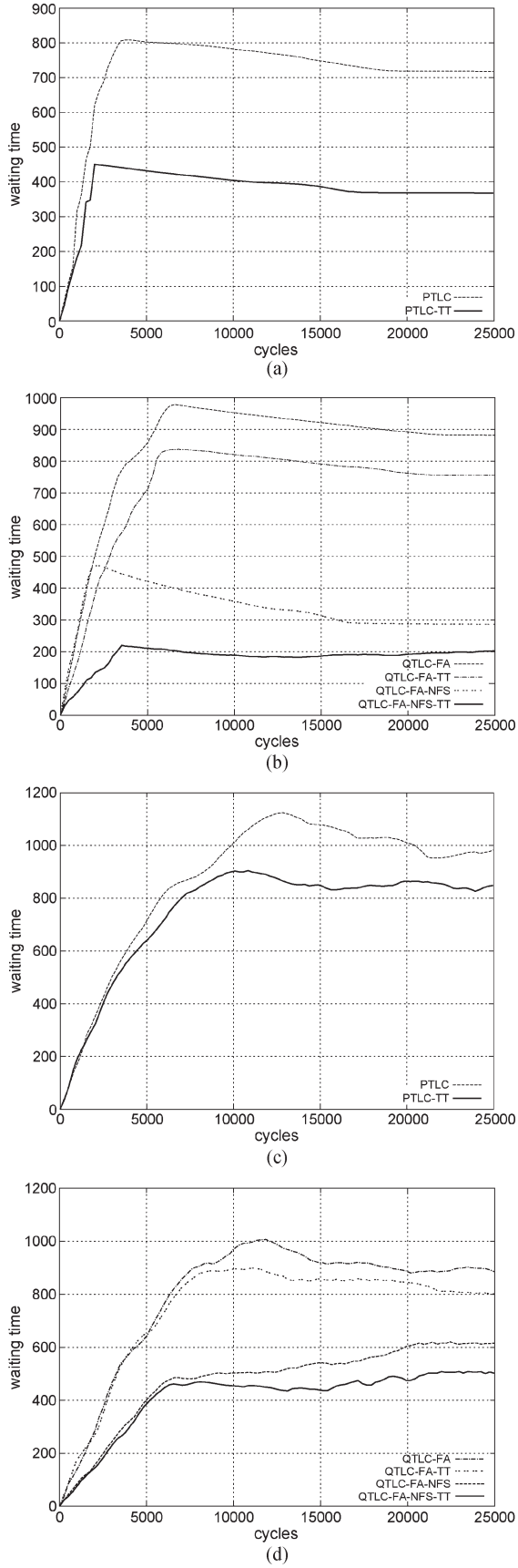

Fig. 4. Performance comparison of the TLC algorithms with their tuning counterparts on a 4×4 grid and the IISc network. (a) PTLC on 4×4 grid network. (b) QTLC-FA-NFS and QTLC-FA on 4×4 grid network. (c) PTLC on the IISc network. (d) QTLC-FA-NFS and QTLC-FA on the IISc network.

TABLE III
TAR RESULTS FROM TWO EXPERIMENTS. (a) TAR FOR A RANGE OF WEIGHTS α_1 ON A SINGLE JUNCTION. (b) TAR FOR VARIOUS TLC ALGORITHMS ON TWO ROAD NETWORKS

(a)

α_1	PTLC	QTLC-FA-NFS
0.2	16514.38±674.09	26675.27±660.33
0.3	17563.46±683.51	25632.40±669.58
0.4	17585.98±663.99	27627.77±669.28
0.5	**18365.80±662.31**	**27779.77±672.60**
0.6	17486.90±669.88	27571.61±670.46
0.7	17683.87±659.67	26454.72±669.99
0.8	14722.40±646.74	25385.50±653.01

(b)

	4x4-Grid Network	IISc Network
QTLC-FA-NFS	17408.56 ± 84.20	21751.17 ± 525.41
QTLC-FA-NFS-TT	**22917.40 ± 80.83**	**24773.38 ± 626.48**
PTLC	3807.14 ± 70.26	16235.74 ± 472.31
PTLC-TT	7662.64 ± 82.38	18878.87 ± 461.66
QTLC-FA	14418.95±80.15	18699.59 ± 415.35
QTLC-FA-TT	19369.61 ± 77.96	20531.24 ± 520.26

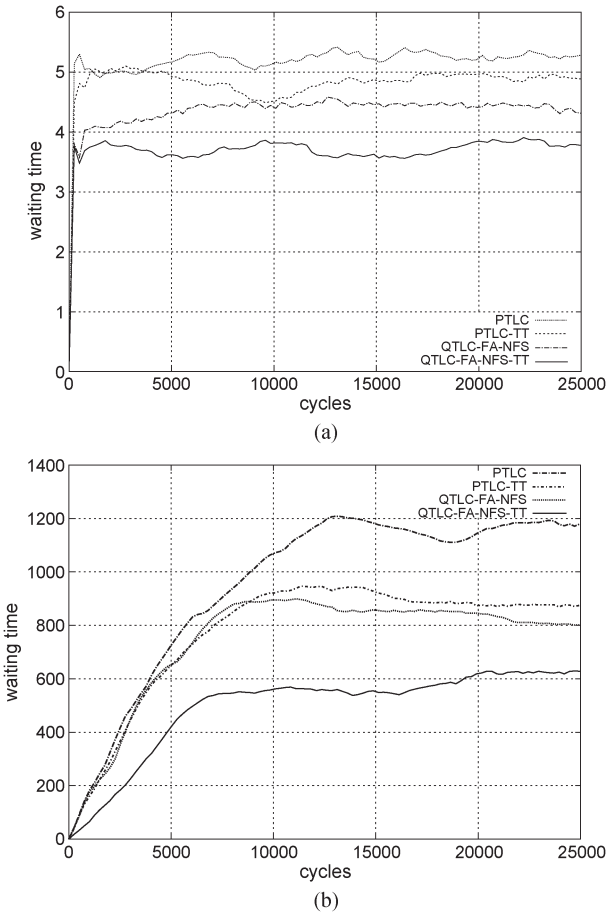

Fig. 5. Performance comparison of the TLC algorithms with a Poisson vehicle arrival pattern on two road networks. (a) PTLC and QTLC-FA-NFS on a single junction. (b) PTLC and QTLC-FA-NFS on the IISc network.

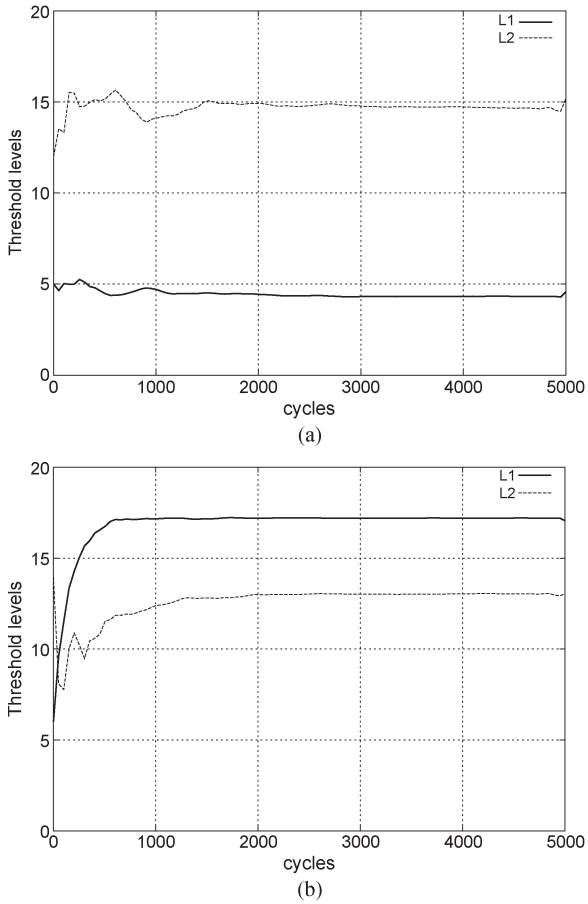

Fig. 6. Convergence of parameter θ: Illustration for the QTLC-FA-NFS-TT and PTLC-TT algorithms in the case of a four-junction corridor. (a) L_1, L_2 evolution for the QTLC-FA-NFS-TT algorithm. (b) L_1, L_2 evolution for the PTLC-TT algorithm.

because our tuning scheme is shown to perform well, even for cases where the arrival distribution is not uniform. Note that, in the basic problem formulation, we made no assumption about the nature of the arrival distribution, except for assuming a Markovian evolution for the system.

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. 6. These plots and 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.

We observe that incorporating our threshold-tuning algorithm results in performance enhancements for all the TLC algorithms. Furthermore, among the TLC algorithms studied, it is evident that the QTLC-FA-NFS algorithm shows the best overall performance on all network settings considered. In particular, it outperformed the algorithm in [1], which establishes the superiority of the feature selection procedure of QTLC-FA-NFS over QTLC-FA in [1]. Although the 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 compared to the QTLC-FA-NFS algorithm and its tuning variant.

VII. Conclusion

Our goal in this paper has been to design an algorithm for tuning the thresholds to their optimal values in any graded-threshold-based TLC algorithm. Toward this end, we introduced and analyzed three new TLC 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 the following three new TLC algorithms: 1) the PTLC algorithm; 2) QTLC-SA; and 3) QTLC-FA-NFS. We combined our threshold-tuning algorithm with the aforementioned TLC algorithms. In the case of 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 clearly outperform their counterparts that use (arbitrarily set) fixed thresholds.

It may be noted that SPSA is a local-search-based optimization technique. Although we could possibly consider methods such as simulated annealing to find the global minimum, the flip side of such approaches is the computational complexity requirement, which is usually very high, unlike SPSA. Furthermore, in this particular application scenario, we have observed that the deterministic-perturbation SPSA-based threshold-tuning algorithm works well, and this case is demonstrated by the results of the simulation experiments.

Appendix
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 based on the other schemes—(8) and (11), respectively. Note that, because the action selection in the case of the PTLC algorithm is through the policy as prescribed in Table II, we 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-time-scale averaging. The case of general (finite) L can also be handled; for example, see [2].

We will assume, for simplicity, that the elapsed times on any lane remain bounded, although the bound can be large. Moreover, the queue lengths on any lane are bounded, because each lane (between two junctions) can at most accommodate a bounded number of vehicles. The single-stage cost function is shown to be a linear function of the state and remains bounded in this case. The PTLC-TT algorithm (in the case of $L = 1$) can be rewritten as follows:

$$L_i(n+1) = \pi_1\left(L_i(n) - a(n)\left(\frac{\tilde{Z}(n)}{\delta\triangle_i(n)}\right)\right) \quad i = 1, 2 \tag{16}$$

$$T_1(n+1) = \pi_3\left(T_1(n) - a(n)\left(\frac{\tilde{Z}(n)}{\delta\triangle_3(n)}\right)\right) \tag{17}$$

$$\tilde{Z}(n+1) = \tilde{Z}(n) + b(n)\left(k\left(\hat{s}_n, f(\hat{s}_n)\right) - \tilde{Z}(n)\right). \tag{18}$$

Lemma 1: Each of the recursions (16)–(18) is uniformly bounded with probability one.

Proof: Recursions (16) and (17) stay uniformly bounded as a consequence of the projection operators π_1 and π_2, respectively. As aforementioned, the single-stage cost function $k(\cdot,\cdot)$ almost surely remains uniformly bounded. Now, note that, because $b(n) \to 0$ 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 that $\sup_n |k(\hat{s}_n, f(\hat{s}_n))| \leq \hat{K}$ almost surely for some $\hat{K} > 0$. Thus, $\sup_n |\tilde{Z}(n)| < \infty$ almost surely. The claim follows. ■

We first analyze the convergence of the faster time-scale recursion (18). 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 $n \geq 1$

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

$$\triangleq \sum_{m=0}^{n-1} b(m) M_m.$$

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 that is associated with this martingale. Note that

$$\sum_{n=0}^{\infty} E\left[(N_{n+1} - N_n)^2 | \mathcal{F}_n\right] = \sum_{n=0}^{\infty} b(n)^2 M_m^2 < \infty$$

almost surely, because M_m^2 remains uniformly bounded, because the single-stage cost function $k(\cdot,\cdot)$ is uniformly bounded. Moreover, $\sum_n b(n)^2 < \infty$. The claim follows from the martingale convergence theorem (cf., [35]). ■

Consider now the following set of ODEs associated with (16)–(18):

$$\dot{L}_1(t) = 0 \quad \dot{L}_2(t) = 0 \quad \dot{T}_1(t) = 0 \tag{19}$$

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

respectively. Based on (19) and (20) and the manner in which $\Delta(t)$ is defined, it is sufficient to consider the following ODE in place of (20) for the faster time-scale recursion:

$$\dot{\tilde{Z}}(t) = J(\theta + \delta\Delta) - \tilde{Z}(t). \tag{21}$$

Note that we let $\theta(t) \equiv \theta$ and $\Delta(t) \equiv \Delta$, viz., time-invariant quantities (as they get updated on the slower time scale). Given $\hat{T} > 0$, let $\hat{T}_n$, $n \geq 0$, be defined according to $\hat{T}_0 = 0$ and $\hat{T}_n = \min\{t(n) | t(n) \geq \hat{T}_{n-1} + \hat{T}\}$, $n \geq 1$. 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 $\bar{L}_1(t)$, $\bar{L}_2(t)$, $\bar{T}_1(t)$, $\bar{Z}(t)$, $t \in [t(n), t(n+1)]$, $n \geq 0$, as follows: $\bar{L}_1(t(n)) = L_1(n)$, $\bar{L}_2(t(n)) = L_2(n)$, $\bar{T}_1(t(n)) = T_1(n)$, $\bar{Z}(t(n)) = \tilde{Z}(n)$, respectively, with suitable continuous linear interpolations in between intervals $[t(n), t(n+1)]$. Given T, $\epsilon > 0$, we say that $\bar{x}(\cdot)$ is a (T, ϵ)-perturbation of the ODE, $\dot{x}(t) = f(x(t))$, if there exist T_n, $n \geq 0$, such that $T_{n+1} - T_n \geq T \forall n$ and $\sup_{t \in [T_n, T_{n+1}]} \|\bar{x}(t) - x(t)\| < \epsilon, \forall n$.

Proposition 1: The functions $\bar{L}_1(t)$, $\bar{L}_2(t)$, $\bar{T}_1(t)$, $\bar{Z}(t)$, and $t \geq 0$ form $(\hat{T}, \gamma)$-perturbations of ODEs (19) and (20).

Proof: Note that, along the time scale $t(n)$, $n \geq 0$, i.e., using the sequence of step sizes $b(n)$, $n \geq 0$, we can rewrite (16) and (17) as follows:

$$L_i(n+1) = \pi_1 \left(L_i(n) - b(n)\chi_i(n) \right), \quad i = 1, 2 \tag{22}$$

$$T_1(n+1) = \pi_3 \left(T_1(n) - b(n)\chi_3(n) \right) \tag{23}$$

where $\chi_i(n) = (a(n)/b(n))((\tilde{Z}(n)/\delta\triangle_i(n)))$, $i = 1, 2, 3$, respectively. Now, based on Assumption (A2), we have $(a(n)/b(n)) \to 0$ as $n \to \infty$. Hence, $\chi_j(n) = o(1)$, $j = 1, 2, 3$. Finally, consider the recursion (18). Note that, by Lemma 2, we have $\sum_{j=l(n)}^{l(n+1)-1} b(j) M_j \to 0$ as $n \to \infty$. We can now rewrite (18) as

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

where $\chi_4(n) = E[k(\hat{s}_m, f(\hat{s}_m)) | \mathcal{F}_{m-1}] - J(\theta(n) + \delta\Delta(n))$. Based on Assumption (A5), $\chi_4(n) \to 0$ on the "natural time scale," which is clearly faster than the time scale of the algorithm; see [31] for a detailed treatment of natural time-scale recursions that involve Markov noise. The claim follows. ■

Lemma 3: With probability one, $|\tilde{Z}(n) - J(\theta(n) + \delta\Delta(n)| \to 0$ as $n \to \infty$.

Proof: The proof follows by applying the theorem in [36] for every $\epsilon > 0$. ■

We now consider the slower time-scale recursions (16) and (17). 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 4: The vectors $\Delta(n)$, $n \geq 0$, satisfy the following properties.

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

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 $\pi(x) \stackrel{\Delta}{=} (\pi_1(x_1), \pi_1(x_2), \pi_2(x_3))^T$. In view of Lemma 3, we can consider the following expression in place of (16) and (17):

$$\theta(n+1) = \pi \left(\theta(n) - a(n) J\left(\theta(n) + \delta\Delta(n)\right) \left(\Delta(n)\right)^{-1} \right). \tag{24}$$

Lemma 5: Given any fixed integer $K > 0$, for all $r \in \{1, \ldots, K\}$, the following hold.

1) $\lim_{m \to \infty} \|\theta(m + r) - \theta(m)\| = 0$, with probability 1 (wp1).
2) $\lim_{m \to \infty} \|\nabla\theta(m + r) - \nabla\theta(m)\| = 0$, wp1.

Proof:

1) The proof follows in a similar manner as in the lemma in [2].
2) The proof follows from 1) and Assumption (A1). ■

The proof of the following result was not shown in [2] for Hadamard matrix perturbations.

Lemma 6: The following conditions hold for any k, $l \in \{1, \ldots, 3\}$, $k \neq l$:

$$\left\| \sum_{n=m}^{m+P-1} \frac{a(n)}{a(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J\left(\theta(n)\right) \right\| \tag{25}$$

$$\left\| \sum_{n=m}^{m+P-1} \frac{a(n)}{a(m)} \frac{1}{\triangle_l(n)} J\left(\theta(n) + \delta\Delta(n)\right) \right\| \to 0 \text{ as } m \to \infty. \tag{26}$$

Proof: We first show that (25) holds. By letting $K = P$ in Lemma 5, it follows that $a(j)/a(m) \to 1$ as $m \to \infty$ for any $j \in \{m, \ldots, m + P - 1\}$. In addition, note that P is an even integer. As a consequence of Lemma 4, we can split any set $A_m \stackrel{\Delta}{=} \{m, m+1, \ldots, m+P-1\}$ into two disjoint subsets $A^+_{m,k,l}$, $A^-_{m,k,l}$ each having the same number of elements, with $A^+_{m,k,l} \cup A^-_{m,k,l} = A_m$ and such that $(\triangle_k(n)/\triangle_l(n))$ takes value $+1 \forall n \in A^+_{m,k,l}$ and $-1 \forall n \in A^-_{m,k,l}$, respectively. Thus

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

The first claim in (25) now easily follows, and the second claim follows from Lemma 5 and (A1). ■

Proof of Theorem 1: The recursion (24) can be rewritten as follows. For $i = 1, 2, 3$, we have

$$\theta_i(n+1) = \gamma_i \left(\theta_i(n) - a(n) \frac{J\left(\theta(n) + \delta\Delta(n)\right)}{\Delta_i(n)} \right) \tag{27}$$

where $\gamma_1 = \gamma_2 = \pi_1$, and $\gamma_3 = \pi_2$, respectively. The recursion (27) can be rewritten as follows:

$$\theta_i(n+1) = \theta_i(n) - a(n)\frac{J\left(\theta(n) + \delta\Delta(n)\right)}{\Delta_i(n)} - a(n)Z_i(n) \tag{28}$$

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 $\theta(m)$, we obtain, for $i = 1, 2, 3$

$$\begin{aligned}\theta_i(m+2^P) = \theta_i(m) &- \sum_{l=m}^{m+2^P-1} a(l)\nabla_i J\left(\theta(l)\right) \\ &- a(m)\sum_{l=m}^{m+2^P-1}\sum_{j=1,j\neq=i}^{3}\frac{a(l)}{a(m)}\frac{\triangle_j(l)}{\triangle_i(l)}\nabla_j J(\theta(l)) \\ &- a(m)\sum_{l=m}^{m+2^P-1}\frac{a(l)}{a(m)}\frac{1}{\triangle_i(l)}J\left(\theta(l)\right) \\ &+ a(m)O(\delta) - \sum_{j=m}^{m+2^P-1} a(j)Z_i(j).\end{aligned}$$

The third and fourth terms on the aforementioned right-hand side asymptotically vanish as a consequence of Lemma 6. The rest now follows from the theorem in [37]. □

ACKNOWLEDGMENT

The authors would to thank the anonymous reviewers for their comments, which helped to significantly enhance the overall quality of this paper.

REFERENCES

[1] L. A. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 412–421, Jun. 2011.

[2] S. Bhatnagar, M. Fu, S. Marcus, and I. Wang, "Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences," *ACM Trans. Model. Comput. Simul.*, vol. 13, no. 2, pp. 180–209, Apr. 2003.

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

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

[5] D. Robertson, *TRANSYT: A Traffic Network Study Tool*. Crowthorne, U.K.: Road Res., 1969.

[6] D. Robertson and R. Bretherton, "Optimizing networks of traffic signals in real time: The SCOOT method," *IEEE Trans. Veh. Technol.*, vol. 40, pt. 2, no. 1, pp. 11–15, Feb. 1991.

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

[8] M. Girianna and R. Benekohal, "Using genetic algorithms to design signal coordination for oversaturated networks," *J. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 117–129, 2004.

[9] J. Spall and D. Chin, "Traffic-responsive signal timing for system-wide traffic control," *Transp. Res. C*, vol. 5, no. 3/4, pp. 153–163, Aug.–Oct. 1997.

[10] D. Srinivasan, M. Choy, and R. Cheu, "Neural networks for real-time traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 261–272, Sep. 2006.

[11] M. Patel and N. Ranganathan, "IDUTC: An intelligent decision-making system for urban traffic-control applications," *IEEE Trans. Veh. Technol.*, vol. 50, no. 3, pp. 816–829, May 2001.

[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 Trans. Veh. Technol.*, vol. 54, no. 2, pp. 744–748, Mar. 2005.

[13] X. Yu and W. Recker, "Stochastic adaptive control model for traffic signal systems," *Transp. Res. C: Emerg. Technol.*, vol. 14, no. 4, pp. 263–282, Aug. 2006.

[14] N. Gartner, *OPAC: A demand-responsive strategy for traffic signal control*. Washington, DC: Transp. Res. Rec., 1983.

[15] J. Henry, J. Farges, and J. Tuffal, "The PRODYN real-time traffic algorithm," in *Proc. IFAC Conf.*, 1984, pp. 305–310.

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

[17] B. Abdulhai, R. Pringle, and G. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, May 2003.

[18] E. Mueller, "Aspects of the history of traffic signals," *IEEE Trans. Veh. Technol.*, vol. VT-19, no. 1, pp. 6–17, Feb. 1970.

[19] E. Kosmatopoulos, M. Papageorgiou, A. Vakouli, and A. Kouvelas, "Adaptive fine-tuning of nonlinear control systems with application to the urban traffic control strategy TUC," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 6, pp. 991–1002, Nov. 2007.

[20] E. Kosmatopoulos and A. Kouvelas, "Large-scale nonlinear control system fine-tuning through learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1009–1023, Jun. 2009.

[21] A. Kouvelas, K. Aboudolas, E. Kosmatopoulos, and M. Papageorgiou, "Adaptive performance optimization for large-scale traffic control systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1434–1445, Dec. 2011.

[22] N. Yung and A. Lai, "An effective video analysis method for detecting red light runners," *IEEE Trans. Veh. Technol.*, vol. 50, no. 4, pp. 1074–1084, Jul. 2001.

[23] C. Li and S. Shimamoto, "An open traffic light control model for reducing vehicles CO_2 emissions based on ETC vehicles," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 97–110, Jan. 2012.

[24] S. Oh, J. Lee, and D. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Trans. Veh. Technol.*, vol. 49, no. 3, pp. 997–1005, May 2000.

[25] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992.

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

[27] J. Spall, "Adaptive stochastic approximation by the simultaneous perturbation method," *IEEE Trans. Autom. Control*, vol. 45, no. 10, pp. 1839–1853, Oct. 2000.

[28] S. Bhatnagar, "Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization," *ACM Trans. Model. Comput. Simul.*, vol. 15, no. 1, pp. 74–107, Jan. 2005.

[29] S. Bhatnagar, "Adaptive Newton-based multivariate smoothed functional algorithms for simulation optimization," *ACM Trans. Model. Comput. Simul.*, vol. 18, no. 1, pp. 1–35, Dec. 2007.

[30] P. J. Schweitzer, "Perturbation theory and finite Markov chains," *J. Appl. Probab.*, vol. 5, no. 2, pp. 401–413, Aug. 1968.

[31] V. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

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

[33] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman, "Simulation and optimization of traffic in a city," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2004, pp. 453–458.

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

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

[36] M. W. Hirsch, "Convergent activation dynamics in continuous-time networks," *Neural Netw.*, vol. 2, no. 5, pp. 331–349, 1989.

[37] H. Kushner and D. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York: Springer-Verlag, 1978.

Prashanth L. A. was born in Gauribidanur, India. He received the B.E. degree in computer science in 2002 from the National Institute of Technology, Surathkal, India, and the M.Sc. (M.Eng.) degree in 2008 from the Indian Institute of Science, Bangalore, India, where he is currently working toward the Ph.D. degree with the Department of Computer Science and Automation.

For more than six years, he was with Texas Instruments (India) Pvt. Ltd., Bangalore, as a Senior Software Systems Engineer. His research interests include stochastic control and optimization, reinforcement learning and its application to road traffic control, service systems, and wireless networks.

Shalabh Bhatnagar (SM'05) received the B.Sc. (Hons.) degree in physics from the University of Delhi, Delhi, India, in 1988 and the M.Sc. and Ph.D. degrees in electrical engineering from the Indian Institute of Science, Bangalore, India, in 1992 and 1997, respectively.

From 1997 to 2000, he was a Research Associate with the Institute for Systems Research, University of Maryland, College Park. From 2000 to 2001, he was a Divisional Postdoctoral Fellow with the Free University, Amsterdam, The Netherlands. He is currently with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, as a Professor. He has also held visiting positions with the Indian Institute of Technology, Delhi, and the University of Alberta, Edmonton, AB, Canada. His research interests include stochastic control, reinforcement learning, and simulation optimization. He is interested in applications in communication and wireless networks, as well as in vehicular traffic control. He has been the author or a coauthor of more than 100 research articles in various journals and conference proceedings.

Dr. Bhatnagar is a Senior Associate of the Abdus Salam International Center for Theoretical Physics, Trieste, Italy, and a Professional Member of the Association for Computing Machinery. He was an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING from 2007 to 2011. He received the Young Scientist Award from the Systems Society of India in 2007 and two Outstanding Young Faculty Awards from Microsoft Research India in 2007 and 2008.