Reinforcement Learning With Function Approximation for Traffic Signal Control

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

***Abstract*—We propose, for the first time, a reinforcement learning (RL) algorithm with function approximation for traffic signal control. Our algorithm incorporates state-action features and is easily implementable in high-dimensional settings. Prior work, e.g., the work of Abdulhai *et al.*, on the application of RL to traffic signal control requires full-state representations and cannot be implemented, even in moderate-sized road networks, because the computational complexity exponentially grows in the numbers of lanes and junctions. We tackle this problem of the curse of dimensionality by effectively using feature-based state representations that use a broad characterization of the level of congestion as low, medium, or high. One advantage of our algorithm is that, unlike prior work based on RL, it does not require precise information on queue lengths and elapsed times at each lane but instead works with the aforementioned described features. The number of features that our algorithm requires is linear to the number of signaled lanes, thereby leading to several orders of magnitude reduction in the computational complexity. We perform implementations of our algorithm on various settings and show performance comparisons with other algorithms in the literature, including the works of Abdulhai *et al.* and Cools *et al.*, as well as the fixed-timing and the longest queue algorithms. For comparison, we also develop an RL algorithm that uses full-state representation and incorporates prioritization of traffic, unlike the work of Abdulhai *et al.* We observe that our algorithm outperforms all the other algorithms on all the road network settings that we consider.**

***Index Terms*—Q-learning with full-state representation (QTLC-FS), Q-learning with function approximation (QTLC-FA), reinforcement learning (RL), traffic signal control.**

I. Introduction

With increasing traffic in urban areas and limitations of road infrastructure, any attempt to improve the traffic flow of the system would involve an intelligent design of traffic signal timing for the junctions. The traffic junctions play a very important role in determining the congestion state of the road network. Many traffic junctions worldwide currently use fixed signal timings, i.e., they periodically cycle through the sign configurations in a round-robin manner. Although such a strategy is easy to implement, it does not consider the actual traffic conditions and may result in more congestion. In this paper, we study the use of a reinforcement learning (RL)-based traffic control system. The objective is to maximize traffic flow by performing adaptive control of traffic lights at intersections. Sensors, which are placed along the lanes that lead to a junction, periodically convey the traffic information to a controller or agent at the junction, which then decides on the signal timings.

In some body of work on adaptive traffic control, the usage of neural network (NN)-based controllers is recommended. For instance, in [3], simultaneous perturbation stochastic approximation (SPSA)-based gradient estimates are used in an NN feedback controller to optimize system performance. The idea is to develop a function that takes in current traffic information and outputs the signal timings. This function is approximated through an NN. In [4] and [5], the aforementioned NN-SPSA algorithm was studied through simulations on the mid-Manhattan, New York City, network, and it was found to give a 10% reduction in the vehicle waiting times compared to the previously used strategy employed. In [6], a NN-based traffic signal control approach in a multiagent system is presented and compared against an existing traffic control algorithm, and an SPSA-NN multiagent traffic control method is developed.

In some other work, [7] discusses a 0–1 mixed-integer linear programming formulation of the traffic signal control problem. A distributed-multiagent-based approach for traffic signal control is presented in [8]. In [9] and [10], the traffic management problem is formulated as an optimization problem and genetic algorithms are used to solve this problem. Genetic algorithms provide a heuristic optimization technique for such problems. In [11], a Markov decision process (MDP) framework for adaptive control of traffic lights is considered. However, to directly be applicable, we require complete information on the transition probabilities of the system, which is often not available. In [12], a commercial software package (TRANSYT) that uses a static optimization technique is designed to generate the signal timings offline based on the traffic conditions measured at different periods of the day. This technique is, however, not adaptive. In [13], the authors propose the SCOOT method, where inductive sensors are used to collect cyclic flow profiles (CFPs) and relay the same to the central SCOOT optimizer. The CFPs are then used to estimate the queues, particularly to calculate the effect of alterations in the predicted signal timings. The SCOOT optimizer then makes many split and offset alterations to coordinate the traffic flows. In [14], an RL algorithm for the case of nonstationary traffic conditions in a decentralized framework is presented. In [15], an adaptive dynamic programming technique for traffic signal control is presented. A controller at each intersection adjusts its signal

Manuscript received December 22, 2009; revised August 20, 2010 and September 20, 2010; accepted October 30, 2010. Date of publication December 6, 2010; date of current version June 6, 2011. This work was supported in part by the Automation Systems Technology Center, which is a program of the Department of Information Technology, Government of India. The Associate Editor for this paper was S. S. Nedevschi.

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).

Digital Object Identifier 10.1109/TITS.2010.2091408

timing based on traffic information and the performance in the neighboring intersections.

In [1], Q-learning with lookup table representation has been applied to traffic light control (TLC) on a single junction. The case of a road network with multiple junctions has not been considered. Furthermore, the algorithm in [1] has been developed for the case when all the lanes are given equal priority for switching lights. Full-state representations cannot directly be used in the case of road networks with multiple traffic junctions because of the exponential blowup in the computational requirements as the number of junctions and, thereby, the cardinalities of the state and action spaces increase. In [2], a TLC method, i.e., self-organizing traffic lights (SOTL), is presented, which switches a lane to green if the elapsed time since the signal turned red on that lane crosses a certain threshold, provided that the number of vehicles on the lane is above another threshold. Thus, although queue lengths on the signaled lanes of the network are not directly considered in deciding the sign configuration, an estimate of the congestion level is used.

In this paper, we use an RL-based algorithm (see [16] and [17]) to solve the traffic signal control problem. The reason for using this approach is that RL allows for learning the optimal strategy for signal timing without assuming any model of the system. The benefits of using RL are twofold. First, it is model-free, i.e., it learns and adapts the policy through interaction with the environment. RL algorithms are online, incremental, and easy to implement. Second, on high-dimensional state–action spaces, function approximation techniques in RL can be used to achieve computational efficiency. Q-learning [18] is an important and well-studied RL algorithm that is efficient and is known to converge to the optimal policy. In this paper, we develop a Q-learning-based TLC algorithm that incorporates function approximation. Such an algorithm has been applied for the first time in traffic signal control. Among its many advantages, it is shown to easily be implementable on a range of high-dimensional network settings and gives far superior performance compared with other related algorithms in the literature. We compare the performance of our algorithm with a range of algorithms that assign a certain (fixed) traffic light duration to the various sign configurations. In addition, we compare the performance of our algorithm with an algorithm that switches traffic lights to green for lanes with the longest queue. We also show performance comparisons of our algorithm with the algorithms in [1] and [2]. Furthermore, we develop another algorithm along the lines of [1] that requires full-state information and compare the performance of our algorithm with this new algorithm.

We now describe comparisons of this paper with prior work in the literature. The algorithm in [1], despite being a Q-learning algorithm, requires full-state representations and cannot be implemented even on road networks of moderate size. Indeed, the implementation of this algorithm has been shown in [1] only for the case of a single-junction road network. On the other hand, we use function approximation, and hence, our algorithm is found to easily be implementable on larger road networks such as a 3×3-grid and an eight-junction corridor. We observe that, on a 3×3 grid, the cardinality of the state–action space is nearly 10^{101}, whereas the number of features that our algorithm requires is only about 200. Thus, algorithms such as in [1] are not implementable on such road networks, whereas our algorithm is found to easily be implementable and gives fast convergence. Although the algorithm in [2] also requires information on the level of congestion and elapsed times on the various lanes, it is not based on RL, unlike our algorithm. Hence, the algorithm in [2] does not have the advantage of an RL-based algorithm, because it does not adaptively update its policy using information from interactions with the environment. The state–action features that we incorporate require information on whether the level of congestion on any given lane is low, medium, or high and whether the elapsed time is below or above a threshold.

For comparison, we further develop another RL algorithm that requires full-state representations and, unlike [1], incorporates prioritization in its cost objective. This case is because our experimental settings are based on scenarios in which the volume of traffic on the main road is significantly higher compared to the side roads. Hence, to give higher priority to main-road traffic, we develop a variant of the algorithm in [1] that assigns a higher cost to traffic on the main roads than on the side roads. We implemented this aforementioned algorithm and the algorithms proposed in [1] only on a two-junction corridor setting, because these algorithms could not be implemented on larger road networks due to the exponential increase in the aforementioned computational complexity. However, we found that, even on a two-junction corridor, our algorithm with function approximation outperformed both algorithms.

Next, unlike [9], [19], and [20], Q-learning in the case of full-state representation can be shown to converge to an optimal policy. The same under function approximation converges to an approximately optimal policy under some conditions; see [21]. Although we also assume that the underlying process is an MDP, unlike [11], we do not require information on the transition probabilities of the system, which are hard to obtain in any (real) system. On the other hand, our algorithm directly works with observed data in terms of the level of congestion and elapsed times on the signaled lanes of the road network. Furthermore, unlike [1] that considers only a single-junction scenario, we develop and apply our algorithm on road networks with multiple junctions. As aforementioned, we propose and apply Q-learning with function approximation in the case of larger road networks, where full-state representations cannot be used because of the curse of dimensionality.

A. Our Contributions

Our contributions are listed as follows.

- We consider the problem of adaptive signal control of traffic lights at junctions and develop a Q-learning algorithm with feature-based state–action representations and function approximation. To the best of our knowledge, RL with function approximation for traffic signal control has been proposed here for the first time in the literature.
- For comparison, we also develop a Q-learning-based TLC algorithm that uses full-state representations. Unlike [1],

we incorporate prioritization of traffic in the cost function for this algorithm.

- We study the performance of our function-approximation-based RL algorithm on the following road network settings: 1) a two-junction corridor; 2) a 2×2 grid network; 3) a 3×3 grid network; and 4) an eight-junction corridor. We compare the performance of our algorithm with various existing TLC algorithms—fixed timing, longest queue (LTLC), and SOTL from [2]—as well as the Q-learning-based TLC algorithms that use full-state representations, i.e., the algorithm that we develop for prioritized traffic and the algorithm based on [1]. Our algorithm is shown to easily be implementable over all network settings, converges fast, and consistently outperforms all the other TLC algorithms considered.

We implement our algorithms on an open-source Java-based software, i.e., Green Light District (GLD) [22]. The other TLC algorithms with which we compare the performance of our algorithm were also implemented in GLD.

The rest of this paper is organized as follows. In Section II, we describe in detail the traffic signal control problem formulation using the MDP framework. To put things in perspective, we first present in Section III the Q-learning algorithm based on full-state representation for prioritized traffic, which we develop for comparison. Next, in Section IV, we present our Q-learning algorithm with function approximation. In Section V, we discuss the implementation of the various TLC algorithms and present the performance simulation results. Finally, in Section VI, we provide the concluding remarks.

II. Traffic Control Problem

We consider the problem of finding an optimal schedule for the sign configurations at traffic junctions with the aim of maximizing traffic flow. The signals associated with a phase, i.e., signals that can simultaneously be switched to green form a sign configuration.

We formulate the traffic signal control problem in the MDP framework and assume that control decisions are made by a centralized controller. For a network of traffic junctions, centralized control has been considered, e.g., in [23]. An MDP formulation requires the characterization of states, actions, costs, and the objective criterion. We explain the basic MDP framework as follows.

A. MDP Framework

A stochastic process $\{X_n\}$ that takes values in a set S is called an MDP if its evolution is governed by a control-valued sequence $\{Z_n\}$ so that the following controlled Markov property is satisfied:

$$Pr(X_{n+1} = j | X_n = i, Z_n = a, X_{n-1} = i_{n-1}, Z_{n-1} = a_{n-1}, \ldots, X_0 = i_0, Z_0 = a_0) = p(i,j,a) \quad (1)$$

for any $i_0, \ldots, i_{n-1}$, i, j, $a_0, \ldots, a_{n-1}$, a, in appropriate sets. We assume here that, if $X_n = i$ for any n, the set of feasible actions or controls is $A(i)$. Thus, in (1), $a \in A(i)$, $a_{n-1} \in A(i_{n-1})$, and so on. Let $A = \cup_{i \in S} A(i)$ denote the control space (i.e., the set of all controls). We assume that both S and A are finite sets. Depending on the current system state, the decision maker or controller picks a control in a way to minimize a long-term cost. We consider the infinite-horizon discounted-cost criterion for this purpose. Let $k(i,a)$ denote the single-stage cost incurred when the system is in state i and action $a \in A(i)$ is chosen. Furthermore, when the state is i and action a is chosen, the next state is j, with a probability of $p(i,j,a)$. These probabilities satisfy $p(i,j,a) \in [0,1]$, $\forall i$, $j \in S, a \in A(i)$ and that $\sum_{j \in S} p(i,j,a) = 1$, for any given $i \in S$ and $a \in A(i)$. Let $\gamma \in (0,1)$ denote the discount factor.

A sequence of functions $\pi = \{\mu_1, \mu_2, \ldots\}$, with each $\mu_n : S \rightarrow A$, $n \geq 1$, is said to be an admissible policy if $\mu_n(i) \in A(i)$, $\forall i \in S$. This condition corresponds to the choice of control $Z_n = \mu_n(X_n)$, $\forall n$. Let Π denote the set of all admissible policies. An admissible policy $\pi = \{\mu_1, \mu_2, \ldots\}$ with each $\mu_n = \mu$, $n \geq 1$ is said to be a stationary deterministic policy (SDP). By a common abuse of notation, we simply refer to μ as an SDP.

For a given admissible policy $\pi \in \Pi$, the value function $V^\pi : S \rightarrow \mathcal{R}$ is defined by

$$V^\pi(i) = E\left[\sum_{m=0}^{\infty} \gamma^m k\left(X_m, \mu_m(X_m)\right) | X_0 = i\right] \quad (2)$$

for all $i \in S$. Then, the aim is to find an optimal policy π^* that gives the optimal value function $V^* : S \rightarrow \mathcal{R}$, which is defined by

$$V^*(i) = \min_{\pi \in \Pi} V^\pi(i). \quad (3)$$

It is well known (see [24]) that an SDP achieves the optimal policy, i.e., the policy that corresponds to the optimal value $V^*(i)$, $\forall i \in S$. Furthermore, the optimal value function $V^*(\cdot)$ satisfies the Bellman equation of optimality as

$$V^*(i) = \min_{a \in A(i)} \left(k(i,a) + \gamma \sum_{j \in S} p(i,j,a) V^*(j) \right) \quad (4)$$

for all $i \in S$.

To solve (4), we require knowledge of the transition probabilities $p(i,j,a)$. Moreover, the number of states (i.e., the cardinality of S) can be very large such that solving (4) becomes computationally infeasible. The QTLC-FS that we describe in Section III tackles the first problem alone (not the second), whereas the Q-learning algorithm with function approximation that we subsequently present in Section IV tackles both problems, i.e., the lack of model information and the curse of dimensionality.

B. Traffic Control Problem as an MDP

We consider a road network with m junctions, $m > 1$. Each junction has multiple crossroads, with each road having j lanes. Our algorithms require a description of states, actions, and costs. The state is the vector of queue lengths and the elapsed times. The elapsed time on a lane is the time since the signal

turned red on that lane. This quantity is zero for lanes on which the signal is green. 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 a green signal stays at zero until the time that the signal turns red. For a network with a total of N signaled lanes, the state at time n is

$$s_n = (q_1(n), \ldots, q_N(n), t_1(n), \ldots, t_N(n))^T$$

where $q_i(n)$ is the queue length on lane i at time n, and $t_i(n)$ is the elapsed time for the red signal on lane i at time n.

The actions a_n comprise the sign configuration (which is a feasible combination of traffic lights to switch) in the m junctions of the road network and have the form $a_n = (a_1(n), \ldots, a_m(n))^T$, where $a_i(n)$ is the sign configuration at junction i in time slot n. We consider only sign configurations that are feasible in the action set and not all possible red–green combinations of traffic lights (which would exponentially grow with the number of traffic lights). Thus, the action set $A(s_n) = \{\text{feasible sign configurations in state } s_n\}$.

The cost function here has two components. The first component is the sum of the queue lengths of the individual lanes, and the second component is the sum of the elapsed times on all lanes. The idea here is to regulate the flow of traffic to minimize the queue lengths while, at the same time, ensuring fairness so that no lane suffers from being red for a long duration. Lanes on the main road are given higher priority over other lanes. We achieve the prioritization of main-road traffic as follows. Let I_p denote the set of indexes of lanes whose traffic should be given higher priority. Then, the single-stage cost $k(s_n, a_n)$ has the form

$$k(s_n, a_n) = r_1 * \left(\sum_{i \in I_p} r_2 * q_i(n) + \sum_{i \notin I_p} s_2 * q_i(n) \right) + s_1 * \left(\sum_{i \in I_p} r_2 * t_i(n) + \sum_{i \notin I_p} s_2 * t_i(n) \right) \quad (5)$$

where r_i, $s_i \geq 0$, and $r_i + s_i = 1$, $i = 1, 2$. Furthermore, $r_2 > s_2$. Thus, lanes in I_p are assigned a higher cost, and hence, a cost-optimizing strategy must assign a higher priority to these lanes to minimize the overall cost.

As aforementioned, we consider the infinite-horizon discounted-cost framework. The discount factor γ plays a crucial role, because a lower γ serves to discount the future costs more, thereby putting less emphasis on these costs, as opposed to a higher value of γ. We let $\gamma = 0.9$ in our experiments.

For ease of exposition and to put things in perspective, we first present in Section III a Q-learning algorithm based on full-state representations that we develop for the case of prioritized traffic. Next, in Section IV, we present for the first time in the literature our Q-learning-based TLC algorithm, which incorporates function approximation and is shown to easily be implementable on high-dimensional settings and to outperform other well-known TLC algorithms.

III. Q-Learning With Full-State Representation

The idea in RL is that, to solve (4), we run a stochastic iterative algorithm using observations obtained from online samples. It is then shown, using the theory of stochastic approximation, that the algorithm asymptotically converges to an optimal value function and policy tuple. An important RL algorithm goes by the name Q-learning [18]. Here, we define Q-values $Q(i, a)$, $i \in S$, and $a \in A(i)$ as follows:

$$Q(i, a) = \left(k(i, a) + \gamma \sum_{j \in S} p(i, j, a) V^*(j) \right). \quad (6)$$

Based on (4), it is easy to see that

$$V^*(i) = \min_{a \in A(i)} Q(i, a). \quad (7)$$

Based on (7) and (6), we obtain the following form of the Bellman equation of optimality, which is also oftentimes called the Q-Bellman equation:

$$Q(i, a) = \left(k(i, a) + \gamma \sum_{j \in S} p(i, j, a) \min_{b \in A(j)} Q(j, b) \right). \quad (8)$$

Although, in (4), the minimization is immediately to the right of the equality, in (8), the same gets pushed inside the summation on the right. The reason for this approach is that we are no longer looking at just state values but at values of state–action tuples. We obtain, as a result, the following online incremental update stochastic (Q-learning) algorithm:

$$Q_{n+1}(i, a) = Q_n(i, a) + a(n) \times \left(k(i, a) + \gamma \min_{b \in A(\eta_n(i,a))} Q_n\left(\eta_n(i, a), b\right) - Q_n(i, a) \right). \quad (9)$$

We can start this algorithm by arbitrarily initializing values of all $Q_0(i, a)$, and a simple choice is to set them all to zero. In the aforementioned, $\eta_n(i, a)$, $n \geq 0$, are independent and identically distributed (i.i.d.) random variables that have the distribution $p(i, \cdot, a)$, i.e., $\eta_n(i, a) = j$ with probability $p(i, j, a)$. In addition, $a(n)$, $n \geq 0$ are (positive) step sizes that satisfy the following conditions:

$$\sum_n a(n) = \infty \quad \sum_n a(n)^2 < \infty. \quad (10)$$

The aforementioned first condition ensures that the algorithm does not prematurely converge, whereas the second condition ensures that the noise in the algorithm asymptotically vanishes. Most often, as we do in our experiments, the step sizes are simply chosen to be $a(n) = 1/n$, $n \geq 1$. The convergence

of this algorithm has been analyzed in [18] and [25]. Upon convergence, we obtain the optimal Q-value function and, hence, the optimal policy as well.

This algorithm requires a full-state representation as it updates over all feasible state–action tuples. It addresses the case when the system model is not known; however, the state and action spaces are manageable. Q-learning finds the optimal policy, even without knowledge of the transition probabilities of the underlying MDP. It does so by iteratively updating $Q(s,a)$ using (9) to obtain the optimal sign configuration policy.

We refer to the Q-learning algorithm with full state representation when applied to our setting as the QTLC-FS algorithm.

IV. Q-Learning With Function Approximation

The QTLC-FS algorithm runs an incremental stochastic algorithm (9) to obtain the optimal sign configuration policy. However, this approach requires a lookup table to store the Q-values for every possible (s,a)-tuple. Although this condition is useful in small state and action spaces, it becomes computationally expensive for larger road networks that involve multiple junctions. For instance, in the case of a small road network (e.g., a two-junction corridor) with ten signaled lanes, with each lane accommodating 20 vehicles, the number of state–action tuples (and, hence, the size of the $Q(s,a)$ lookup table) is on the order of 10^{14}. This condition leads to an extraordinary computation time and space, because lookup table representation requires much memory, and second, the lookup and update operation of $Q(s,a)$ for any (s,a) tuple is expensive because of the number of (s,a)-tuples. For instance, in the case of the aforementioned ten-lane example, (9) would correspond to a system of 10^{14} equations needed to update $Q_n(i,a)$ for each feasible (i,a)-tuple once. The situation is aggravated when we consider larger road networks such as a grid or a corridor with several junctions, because the sizes of the state and action spaces exponentially blow up. It is precisely for this reason that the Q-learning algorithm proposed in [1] is not even implementable on medium- and large-sized road networks. To alleviate this problem of the curse of dimensionality, we incorporate feature-based methods. These methods handle the aforementioned problem by making computational complexity manageable. An introduction to feature-based methods is given in the following section before we describe our Q-learning-based TLC algorithm, which uses function approximation.

A. Feature-Based Representations

Note that the Bellman equation for optimality (4) requires solving a system of equations in $|S|$ variables. Similarly, a solution to the Q-Bellman equation (8) requires solving a system of equations in $|S \times A(S)|$ unknowns. Here, $S \times A(S)$ denotes the set of all feasible state–action tuples. As previously noted, the number of variables in these equations is of a large order, even for road networks of small sizes. Thus, algorithms that require full-state representations, e.g., QTLC-FS or the algorithm in [1], are not even implementable on reasonably sized road networks. This case is the prime reason for resorting to function-based approximations.

In the setting of Q-learning with function-based approximation, the idea is to approximate the Q-value function $Q(s,a)$ as

$$Q(s,a) \approx \theta^T \sigma_{s,a} \tag{11}$$

where $\sigma_{s,a}$ is a d-dimensional feature (column) vector that corresponds to the state–action tuple (s,a), with $s \in S$, and $a \in A(s)$. The dimension d is significantly less compared to the cardinality of the set of feasible state–action tuples (s,a). Here, θ is a tunable parameter whose dimension is the same as in $\sigma_{s,a}$.

Let Φ denote a matrix with rows $\sigma_{s,a}^T, s \in S, a \in A(s)$. The number of rows of this matrix is thus $|S \times A(S)|$, whereas the number of columns is d. Let

$$\sigma_{s,a} = (\sigma_{s,a}(1), \ldots, \sigma_{s,a}(d))^T.$$

Then, $\Phi = (\Phi(i), i = 1, \ldots, d)$, where $\Phi(i)$ is the column vector and that is defined by

$$\Phi(i) = (\sigma_{s,a}(i), s \in S, a \in A(s))^T, \quad i = 1, \ldots, d.$$

Let $\theta = (\theta_1, \ldots, \theta_d)^T$. Then

$$Q \approx \sum_{i=1}^{d} \Phi(i)\theta_i, \quad \text{or alternatively, } Q \approx \Phi\theta$$

where $Q = [Q(s,a), s \in S, a \in A(s)]^T$ is the vector of the Q-values $Q(s,a)$ over all feasible (s,a) tuples. In other words, Q is approximated using $\Phi\theta$. Note that the gradient of the approximate Q-value function with regard to θ is

$$\nabla_\theta Q(s,a) \approx \sigma_{s,a}.$$

The Q-learning algorithm with function approximation that we present in (12) is shown to perform a gradient search in $\mathbf{R}^d$.

A routine requirement used to prove the convergence of function-approximation-based algorithms is that the columns $\Phi(i), i = 1, \ldots, d$ of the feature matrix Φ are linearly independent. We expect this requirement to hold good in our setting, because the size of the state–action space is very large, and in comparison, the dimension (d) of the feature vector is small. For instance, in a 3×3 grid setting, whereas the size of the state–action space is on the order of 10^{101}, the size of d is only about 200.

B. QTLC-FA

We now describe the Q-learning-based TLC algorithm with function approximation (QTLC-FA). Although the QTLC-FS algorithm as such requires complete state information and is computationally less efficient, its function-approximation-based variant parameterizes the value function and requires significantly less computation in terms of space and time requirements while giving good performance. The QTLC-FA algorithm, which is a variant of the QTLC-FS algorithm, updates the parameter θ, which is a d-dimensional quantity. Thus, instead of solving a system in $|S \times A(S)|$ variables, we solve here a system in only d variables. As aforementioned, in the case of the 3×3 grid road network that we consider in our

experiments, it is shown that, although $|S \times A(S)| \sim 10^{101}$, d is only about 200. This results in significant speedup in the computation time when feature-based representations are used.

Let s_n, s_{n+1}, denote the state at instants n and $n+1$, respectively. Let θ_n be the nth update of the parameter θ. The QTLC-FA algorithm uses the following update rule:

$$\theta_{n+1} = \theta_n + \alpha(n)\sigma_{s_n,a_n} \times \left(k(s_n,a_n) + \gamma \min_{v \in A(s_{n+1})} \theta_n^T \sigma_{s_{n+1},v} - \theta_n^T \sigma_{s_n,a_n}\right) \quad (12)$$

where θ_0 is arbitrarily set. In (12), the action a_n is chosen in state s_n according to $a_n = arg\min_{v \in A(s_n)} \theta_n^T \sigma_{s_n,v}$.

Although the algorithm (12) updates the parameter $\theta \in \mathbf{R}^d$, this approach results in updating the projected Q-value functions, i.e., those that are obtained according to $Q \approx \Phi\theta$. Note that the set $\{\Phi\theta | \theta \in \mathbf{R}^d\}$ forms a subspace of the set of all functions on $|S \times A(S)|$ (i.e., the original Q-value functions).

The features are chosen based on the queue lengths and elapsed times of each signaled lane of the road network. In particular, we select features σ_{s_n,a_n} to have the following form:

$$\sigma_{s_n,a_n} = \big(\sigma_{q_1(n)}, \ldots, \sigma_{q_N(n)}, \sigma_{t_1(n)}, \ldots, \sigma_{t_N(n)} \sigma_{a_1(n)}, \ldots, \sigma_{a_m(n)}\big)^T$$

where

$$\sigma_{q_i(n)} = \begin{cases} 0, & \text{if } q_i(n) < L1 \\ 0.5, & \text{if } L1 \le q_i(n) \le L2 \\ 1, & \text{if } q_i(n) > L2 \end{cases}$$

$$\sigma_{t_i(n)} = \begin{cases} 0, & \text{if } t_i(n) \le T1 \\ 1, & \text{if } t_i(n) > T1. \end{cases} \quad (13)$$

Furthermore, $\sigma_{a_1(n)}, \ldots, \sigma_{a_m(n)}$ corresponds to the actions or sign configurations chosen at each of the m junctions. As before, N is the total number of lanes (inclusive of all junctions) in the network. $L1$ and $L2$ are thresholds on the queue lengths, and $T1$ is a threshold on the elapsed time. Note that the parameter θ_n has the same dimension as in σ_{s_n,a_n}. Again, the advantage here is that, instead of updating the Q-values for each feasible (s,a)-tuple, as before, we estimate these Q-values according to the parameterization (11). In the case of a fixed policy, the algorithm (12) is analogous to the well-studied temporal-difference learning algorithm. A proof of convergence of the algorithm (12) under some conditions is provided in [21].

One advantage of using the aforementioned features is that we do not require full information on the queue lengths or the elapsed times. Thresholds $L1$ and $L2$ can be marked on the lanes and used to estimate low (less than $L1$), medium (between $L1$ and $L2$), or high (above $L2$) traffic. Likewise, the elapsed time can be categorized as being below the threshold $(T1)$ or above it. More gradations of the queue lengths and elapsed times can also be considered. Thus, another advantage of our QTLC-FA algorithm over the algorithm in [1] or the QTLC-FS algorithm is that it does not require precise queue length information. Such information is often hard to obtain, whereas a characterization of traffic at any time as low, medium, or high is easier.

Practical implementation of QTLC-FA would require the placement of sensors along the lanes of the road network. Because we require only information on whether the traffic congestion is below threshold $L1$, in between thresholds $L1$ and $L2$, or above threshold $L2$, we can use two loops of sensors: one loop placed along $L1$ and another loop along $L2$. If sensors at $L1$ do not detect congestion, it can be inferred that the congestion level on that lane is "low," i.e., below $L1$. If, on the other hand, sensors at $L1$ detect congestion but sensors at $L2$ do not, then congestion can be inferred to be in the "medium" range (i.e., above $L1$ but below $L2$). Similarly, if sensors at $L2$ also detect congestion, then the congestion level can be inferred to be in the "high" range (i.e., at the level of $L2$ or more). Elapsed times are usually measured by time counters placed at signal intersections. Again, we only need information on whether the elapsed time on a lane is above or below a threshold $T1$. Information on congestion levels and elapsed times below or above a threshold will then have to be communicated to the central controller, which would then run the QTLC-FA algorithm to obtain the sign configuration policy. As observed from our experiments, QTLC-FA has a very short transient phase and is computationally very efficient (both in time and space complexities) and, hence, can easily be implemented to obtain the sign configurations online in a real system. Note that, although we do not require precise information on vehicle count, the problem of getting precise estimates of vehicle count is an interesting problem in itself and has independently been addressed, e.g., in [26].

There is another advantage of placing sensors along fixed distances (e.g., $L1$ and $L2$) from the traffic junction and using the distance of "detected" congestion (from the junction) as a proxy for queue length thresholds. For instance, in the case of the traffic situation prevalent in India, i.e., highly congested traffic with a high diversity of vehicles, getting precise queue length information is extremely difficult. Using passenger car units (PCUs) as a measure of queue length [23] (e.g., three motorbikes could correspond to one PCU), we can estimate the number of PCUs (could be a fraction) that can be accommodated in a unit distance (e.g., 1 m) of the lane. That number multiplied by $L1$ or $L2$ would roughly correspond to the number of PCUs that can be packed in $L1$ or $L2$ m of the lane. Thus, $L1$ and $L2$ could be used as proxies for queue thresholds, except for a multiplication factor. The form of our cost objective (5) is such that the multiplication factor does not play a role because the form of the "optimal" policy obtained would still be the same.

V. Simulation Experiments

We use the GLD simulator (see [22]) for the implementation and evaluation of our TLC algorithms. GLD allows users to build road networks (involving lanes, junctions, and road users), simulate traffic from various road users, and obtain performance statistics. The crucial part is that it allows the implementation and evaluation of traffic light algorithms. It consists of an interface for constructing the road layouts and a traffic simulator for conducting the experiments with existing and new TLC algorithms.

A. Implementation

We implement our QTLC-FA algorithm. For comparison, we also implement the following algorithms.

- **Fixed-timing TLC**. This algorithm periodically cycles through the list of feasible sign configurations while not considering the traffic load on the lanes of the road network. The cycling period in this algorithm is a tunable parameter, and we show the results of its performance for various cycling periods.
- **SOTL** [2]. This algorithm uses the elapsed times to decide the sign configuration, i.e., the traffic light switches to green when the elapsed time crosses a threshold, provided that the number of vehicles crosses another threshold L. We set $L = 5$ in our experiments, as has been done in [2].
- **LTLC**. Here, the number of road users waiting for a traffic light to turn green is counted and used to decide on the combination of traffic lights to be turned green in the next time slot. In essence, LTLC attempts to switch the lane with the highest number of waiting road users to green.
- **QTLC-FS**. This algorithm has been described in Section III.
- **Q-learning with no priority** (QTLC-NP) [1]. This algorithm has a similar update rule as QTLC-FS; however, the cost function here is

$$k(s_n, a_n) = \sum_{i=1}^{N} q_i(n) + \sum_{i=1}^{N} t_i(n). \tag{14}$$

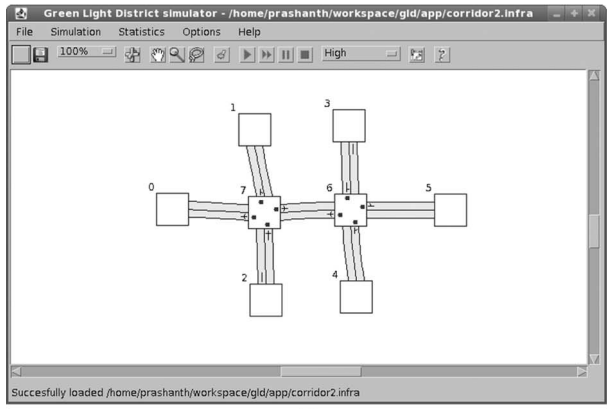
  Thus, in particular, unlike QTLC-FS, this algorithm does not assign a higher priority to main-road traffic.

We consider the following four different network scenarios:

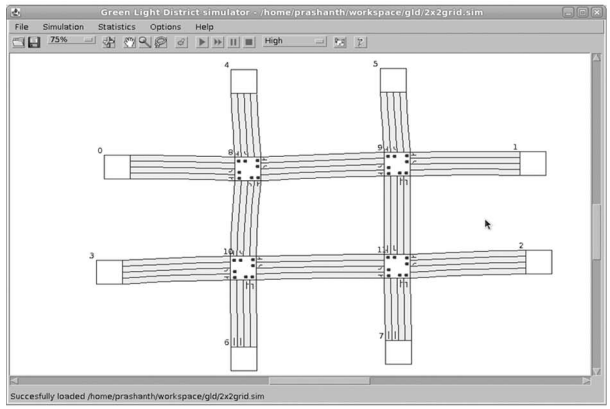
1) a two-junction corridor;
2) a 2×2 grid network
3) a 3×3 grid network;
4) an eight-junction corridor.

The road networks are shown in Fig. 1, which are snapshots obtained from the GLD software. Although we consider all roads to be of two lanes in the two-junction corridor, we consider all roads to be of four lanes in all the other settings. This has been done because we could implement the QTLC-FS and QTLC-NP algorithms only on the two-junction corridor setting when all roads had two lanes. When the number of lanes is increased, both algorithms could not be implemented because of the aforementioned curse of dimensionality effect. On the other hand, our algorithm with function approximation, i.e., QTLC-FA, was found to easily be implementable on all the settings that we considered. The simulations were conducted for 5000 cycles for all algorithms. Each road user's destination was randomly fixed using a discrete uniform distribution to choose one of the edge nodes.

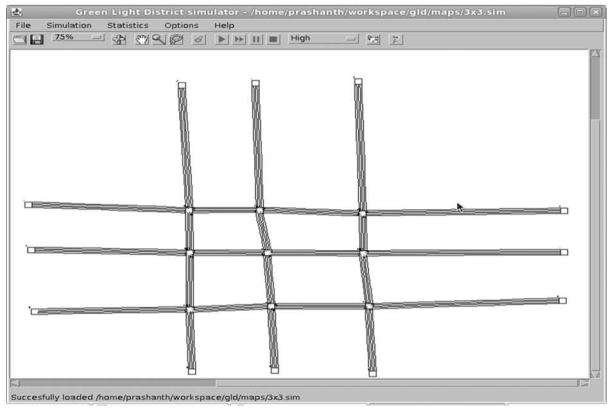
In all the road networks, we set the spawn frequencies (the average rate at which traffic is generated randomly in GLD) so that the proportion of cars flowing on the main road to those on the side roads is in the ratio of 100:5. This setting is close to real-life traffic scenarios on many busy corridor and grid networks and has also been used, e.g., in [2].



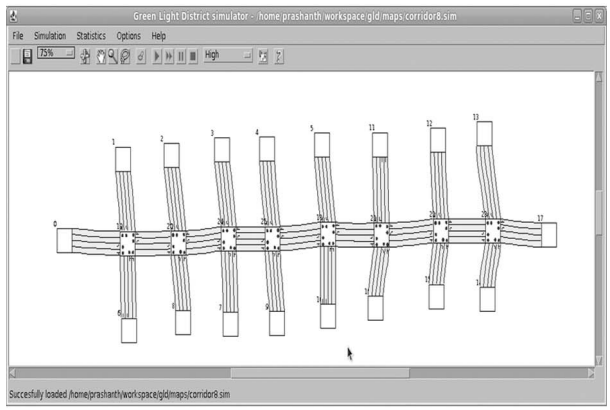
(a)



(b)



(c)



(d)

Fig. 1. Road networks used for our experiments. (a) Two-junction corridor, (b) 2×2 grid network, (c) 3×3 grid network, and (d) eight-junction corridor.

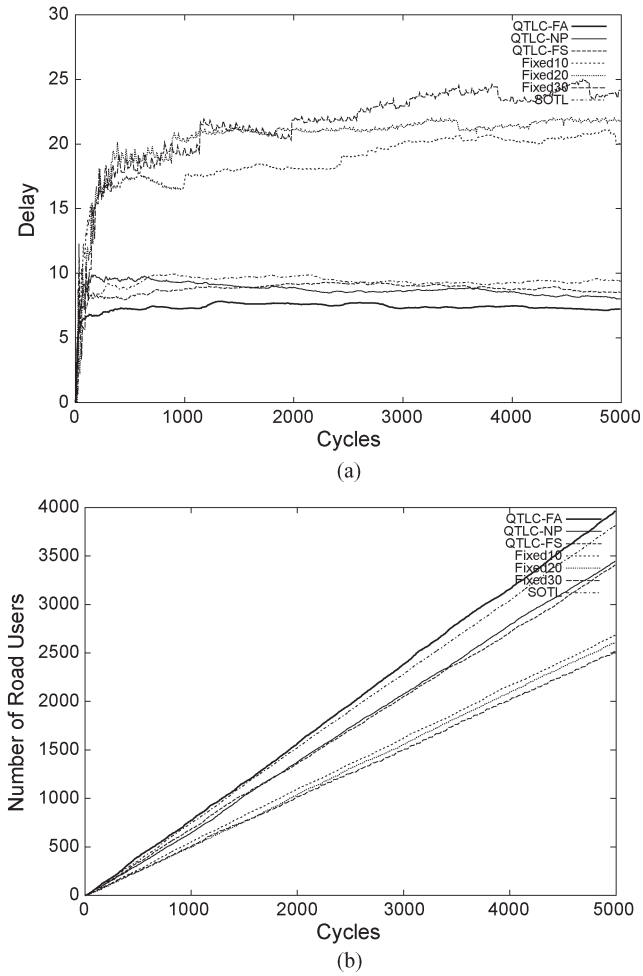


Fig. 2. Performance comparison of the TLC algorithms for the two-junction corridor case. (a) AJWT. (b) TAR.

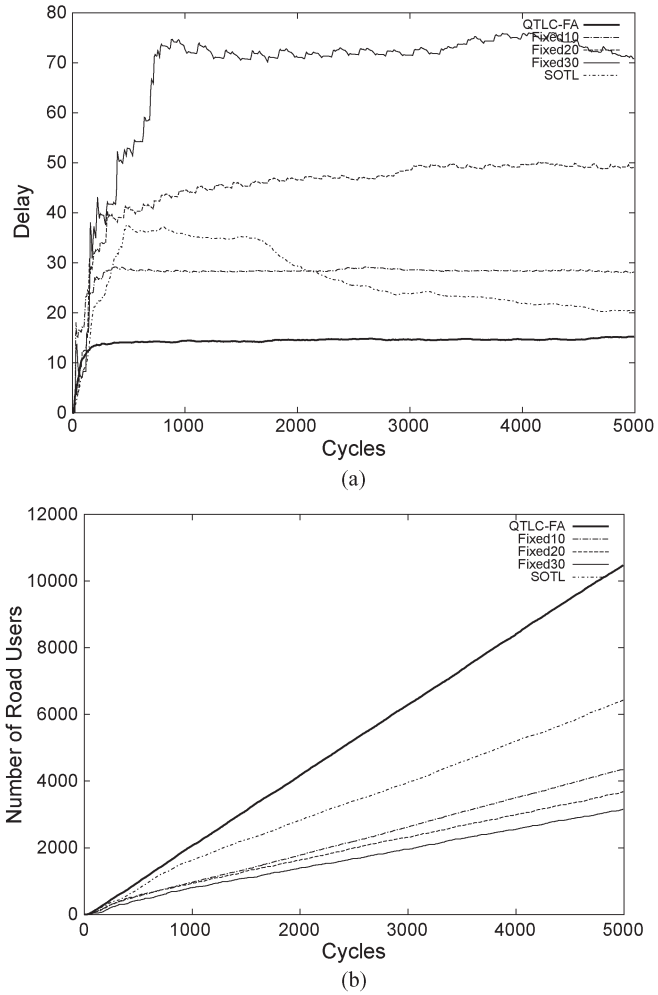


Fig. 3. Performance comparison of the TLC algorithms for the 2×2 grid network case. (a) AJWT. (b) TAR.

For both QTLC-FS and QTLC-FA, we set the weights in the single-stage cost function $k(s, a)$ in (5) as $r_1 = s_1 = 0.5$. We thus give equal weight to both the queue length and elapsed time components. Furthermore, we set $r_2 = 0.6$ and $s_2 = 0.4$. This assignment gives a higher priority to the lanes on the main road than on the side roads. The thresholds $L1$ and $L2$ were set to 6 and 14, respectively, considering that the length of the roads in all the road networks that we study is 20. The threshold $T1$ was set to 90.

B. Results

We compare the performance of the TLC algorithms using the average junction waiting times (AJWT) and total arrived road users (TAR), i.e., the number of road users who have reached their destination. The performance plots of AJWT and TAR versus the number of cycles in all four road networks studied are shown in Figs. 2–5.

Based on the aforementioned plots, we observe that QTLC-FA consistently shows the best results in all the four road networks studied. We now discuss the performance results in more detail. Based on the AJWT and TAR plots, we observe that QTLC-FA performs better than the fixed-timing TLC algorithm for all the cycling periods considered in the latter case. Using a broad estimate of queue lengths and elapsed times of the signaled lanes enables the QTLC-FA algorithm to adapt the sign configuration policy to the traffic situation, whereas fixed-timing algorithms are unmindful of the current traffic situation, leading to longer waiting times.

We do not show the plots of the LTLC algorithm, because it performed very poorly compared to our algorithms. In fact, when LTLC was used, the traffic invariably entered a deadlock situation, because the algorithm always arrived at a sign configuration that did not allow traffic to pass across junctions. The longest queue status of the lanes then remained the same because of the gridlock, leaving the sign configuration unchanged.

QTLC-FA also outperformed the SOTL algorithm [2] in all the four road networks. Using a broad estimate of congestion on the signaled lanes apart from the elapsed times, the QTLC-FA algorithm found an approximately optimal sign configuration policy that minimized the long-term discounted cost, which, in essence, ensured smooth traffic flow. Although both SOTL and QTLC-FA used a rough measure of the level of congestion based on certain thresholds, QTLC-FA outperformed SOTL, because it adaptively tunes the feedback policy, unlike SOTL.

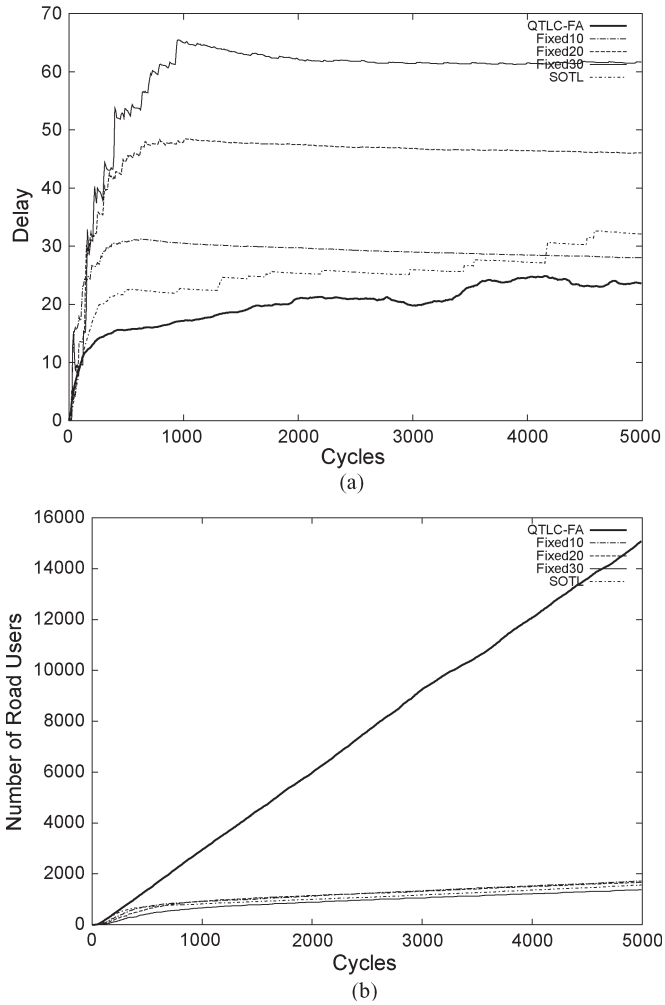


Fig. 4. Performance comparison of the TLC algorithms for the 3 × 3 grid network case. (a) AJWT. (b) TAR.

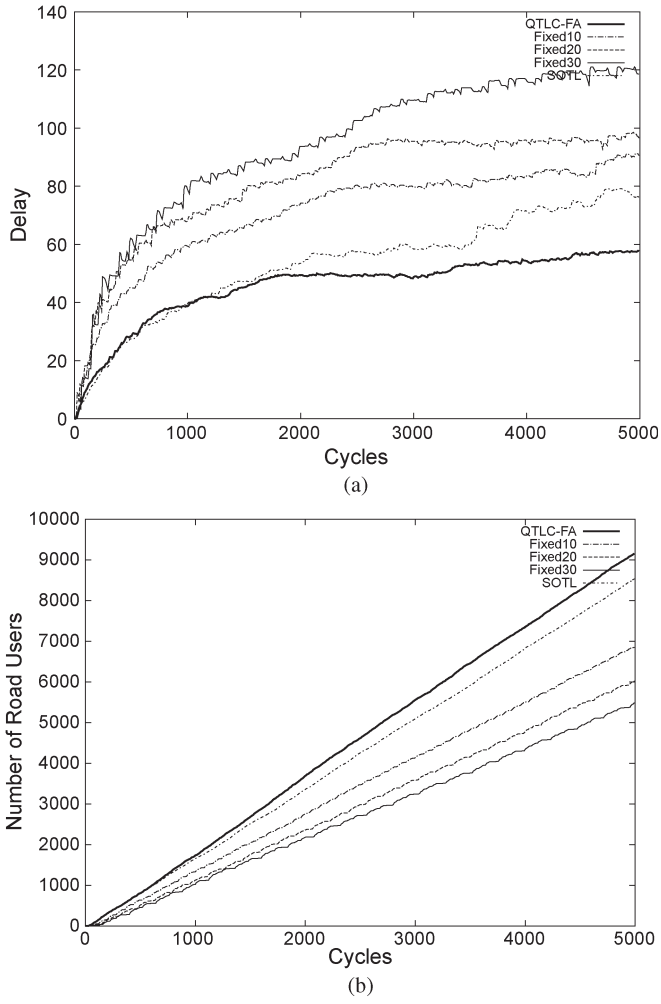


Fig. 5. Performance comparison of the TLC algorithms for the eight-junction corridor case. (a) AJWT. (b) TAR.

As previously explained, we implemented the QTLC-FS and QTLC-NP [1] algorithms only on the two-junction corridor, with each road having two lanes, and not on the other networks because of the exponential increase in computational complexity in the case of networks with more lanes and junctions. On the other hand, QTLC-FA is easily implementable on larger network scenarios and requires much less computation. Based on the performance plots in Fig. 2, we observe that QTLC-FA did better than both QTLC-FS and QTLC-NP, apart from the other TLC algorithms with which it was compared. A judicious choice of features that take into account both the congestion levels on the lanes of the road network and the elapsed times to ensure that no lane waits for a long duration for its signal to turn green resulted in an improved performance for QTLC-FA, compared with both QTLC-FS and QTLC-NP.

Based on the AJWT plots, we observe that the transient phase, i.e., the initial period when QTLC-FA tunes its parameters before stabilizing on a policy, is only a few cycles, and hence, QTLC-FA rapidly converges to a near-optimal sign configuration policy. In addition, QTLC-FA has the advantages of any RL algorithm, i.e., it adapts well to traffic conditions on different types of road networks. This case is evident in the superior performance of QTLC-FA on all road networks considered, with no specific tuning of the QTLC-FA algorithm done for a particular road network.

VI. Conclusion

Designing a road traffic management system based on wireless sensor networks that achieves high traffic flow rates with minimum congestion is a challenging task. RL presents an interesting paradigm for solving such problems. We have designed and evaluated two Q-learning-based algorithms for road traffic control on a network of junctions. Q-learning TLCs have the advantages of model-free learning algorithms that adapt in real time to the traffic conditions. Our Q-learning algorithm with function approximation is proposed for the first time in the literature for traffic signal control. Based on the performance simulations, it is observed that our QTLC-FA algorithm consistently outperforms all the other algorithms with which we showed performance comparisons over all the network settings considered.

Our future work would involve the application of other efficient RL algorithms with function approximation (see [27] and [28]) to the problem of traffic signal control. Furthermore, we shall also consider feature adaptation schemes that would

update the features to obtain the "best possible" features. Moreover, we shall develop RL algorithms for constrained MDPs and adapt them to the setting of traffic signal control. Finally, it would be interesting to incorporate the effects of driver behavior in our framework (see [29]).

ACKNOWLEDGMENT

The authors would like to thank the reviewers of this paper for their comments, which helped in significantly enhancing the scope and overall quality of this paper, Prof. A. Kumar and Prof. K. V. S. Hari for their helpful discussions, and Mr. Ravikumar for his useful inputs and for sharing his work.

REFERENCES

[1] 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/Jun. 2003.
[2] 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.
[3] J. Spall and D. Chin, "Traffic-responsive signal timing for systemwide traffic control," *Transp. Res. Part C: Emerging Technol.*, vol. 5, no. 3/4, pp. 153–163, Aug. 1997.
[4] R. Smith and D. Chin, "Evaluation of an adaptive traffic control technique with underlying system changes," in *Proc. Winter Simul. Conf.*, 1995, pp. 1124–1130.
[5] D. Chin, J. Spall, and R. Smith, "Evaluation of systemwide traffic signal control using stochastic optimization and neural networks," in *Proc. Amer. Control Conf.*, 1999, vol. 3, pp. 2188–2194.
[6] 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.
[7] W. Lin and C. Wang, "An enhanced 0–1 mixed-integer LP formulation for traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 238–245, Dec. 2004.
[8] B. Gokulan and D. Srinivasan, "Distributed geometric fuzzy multiagent urban traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 714–727, Sep. 2010.
[9] 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, Apr. 2004.
[10] 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 Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 132–141, Mar. 2010.
[11] X. Yu and W. Recker, "Stochastic adaptive control model for traffic signal systems," *Transp. Res. Part C: Emerging Technol.*, vol. 14, no. 4, pp. 263–282, Aug. 2006.
[12] D. Robertson, *TRANSYT: A Traffic Network Study Tool*. Crowthorne, U.K.: Road Res. Lab., 1969.
[13] 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.
[14] D. de Oliveira, A. Bazzan, B. da Silva, E. Basso, L. Nunes, R. Rossetti, E. de Oliveira, R. da Silva, and L. Lamb, "Reinforcement learning based control of traffic lights in nonstationary environments: A case study in a microscopic simulator," in *Proc. 4th EUMAS*, 2006, pp. 31–42.
[15] T. Li, D. Zhao, and J. Yi, "Adaptive dynamic programming for multi-intersections traffic signal intelligent control," in *Proc. 11th Int. IEEE ITSC*, 2008, pp. 286–291.
[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. Cambridge, MA: MIT Press, Mar. 1998.
[17] D. P. Bertsekas and J. N. Tsitsiklis, *Neurodynamic Programming (Optimization and Neural Computation Series 3)*. Belmont, MA: Athena Scientific, May 1996.
[18] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00992698
[19] G. Abu-Lebdeh and R. Benekohal, "Design and evaluation of dynamic traffic management strategies for congested conditions," *Transp. Res. Part A: Policy Pract.*, vol. 37, no. 2, pp. 109–127, Feb. 2003.
[20] I. Yun and B. Park, "Application of stochastic optimization method for an urban corridor," in *Proc. 38th Winter Simul. Conf.*, 2006, pp. 1493–1499.
[21] F. Melo and M. Ribeiro, "Q-learning with linear function approximation," in *Proc. Learn. Theory*, 2007, pp. 308–322.
[22] 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.
[23] P. Ravikumar, Area traffic control system for heterogeneous traffic having limited lane discipline," Department of Civil Engineering, Indian Institute of Technology Bombay, Mumbai, India, Tech. Rep., 2009.
[24] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.
[25] J. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, no. 3, pp. 185–202, Sep. 1994.
[26] K. Kwong, R. Kavler, R. Rajagopal, and P. Varaiya, "Real-time measurement of link vehicle count and travel time in a road network," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 4, pp. 814–825, Dec. 2010.
[27] V. Konda and J. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2004.
[28] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009.
[29] J. Pauwelussen and P. Feenstra, "Driver behavior analysis during ACC activation and deactivation in a real traffic environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 329–338, Jun. 2010.

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

He was with Texas Instruments, Bangalore, India, for more than six years as a Senior Software Systems Engineer. His research interests include stochastic control and optimization, reinforcement learning and its application to road traffic control, and wireless networks.

Shalabh Bhatnagar (SM'05) received the B.S. (Hons.) degree in physics from the University of Delhi, Delhi, India, in 1988 and the M.S. 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 an Associate Professor. He has also held visiting positions with the Indian Institute of Technology, Delhi, and the University of Alberta, Edmonton, AB, Canada. He is the author or a coauthor of more than 90 research papers in various journals and conference proceedings. His research interests include reinforcement learning, stochastic control, and simulation optimization, in particular applications in communications, wireless networks, and, more recently, vehicular traffic control.

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 has 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. He is an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.