Recursive least-squares temporal difference learning for adaptive traffic signal control at intersection

Biao Yin, Mahjoub Dridi, Abdellah El Moudni

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

Recursive least-squares temporal difference learning for adaptive traffic signal control at intersection

Biao Yin[1,*], Mahjoub Dridi[2], Abdellah El Moudni[2]

[1]LVMT-City Mobility Transport Laboratory, École des Ponts ParisTech, IFSTTAR, UPEM, 77455 Champs-sur-Marne, France

[2]NIT-O2S, Université de technologie de Belfort-Montbéliard, 90000 Belfort, France

[*]Corresponding author: Biao Yin; e-mail: biao.yin@enpc.fr; telephone: +33(0)787130511

Abstract This paper presents a new method to solve the scheduling problem of adaptive traffic signal control at intersection. The method involves recursive least-squares temporal difference (RLS-TD(λ)) learning that is integrated into approximate dynamic programming. The learning mechanism of RLS-TD(λ) is to make an adaptation of linear function approximation by updating its parameters based on environmental feedback. This study investigates the method implementation after modelling a traffic dynamic system at intersection in discrete time. In the model, different traffic control schemes regarding signal phase sequence are considered, especially the defined adaptive phase sequence (APS). By simulating traffic scenarios, RLS-TD(λ) is superior to TD(λ) for updating functional parameters in the approximation, and APS outperforms other conventional control schemes on reducing traffic delay. By comparing with other traffic signal control algorithms, the proposed algorithm yields satisfying results in terms of traffic delay and computation time.

Keywords adaptive traffic signal control; recursive least-squares temporal difference; approximate dynamic programming; adaptive phase sequence

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

1 Introduction

Traffic congestion has critical impacts on people's daily life and environment. It leads to excess delays, reduced safety, and increased environmental pollution. Traffic signal timing aims to schedule flows at intersection efficiently to reduce congestion. In earlier research, a fixed-time (or pre-timed) cyclical signal plan is made for traffic management. Control parameters, such as cycle length, phase splits, and offset, are off-line optimized by using historical traffic data, and then are set in advance for signal controller. Fixed-time control is limited to low flow fluctuations, and is inefficient for traffic changes. Later, adaptive control occurs that it can adjust the control parameters in real time, responding to traffic demands. Adaptive control is more flexible in operation than fixed-time control. Adaptive traffic signal control systems have been available in practice, but most of them use conventional control methods. As the development of intelligent transportation systems, intelligent algorithm plays an important role to be implemented in signal controller.

One of the most well-known intelligent algorithms is reinforcement learning [1-3], which can make an agent (signal controller) do goal-directed learning and decision making by interacting with unknown environment. Over the past decade, many research has been worked on reinforcement learning with function approximation, which has been brought together with approximate dynamic programming (ADP) community [4-6]. Reinforcement learning mainly focuses on control problems based on Markov decision process (MDP) without model information, such as Q-learning. Differently, ADP usually obtains near-optimal solutions of MDP with some model information, which can be viewed as planning cases for sequential decision making [3]. From literature [7-9], MDP can describe the traffic signal control problems well in a discrete-time dynamical way. However, conventional dynamic programming (DP) algorithm meets "curse of dimensionality" due to a large state space, according to the MDP description of this problem. ADP offers a near-optimal solution and reduces computation cost tremendously using an approximate technique in its learning mechanism. In order to update the approximate function in ADP, there are several candidates for the learning mechanism, such as gradient descent [10], TD(λ) [11], RLS-TD(λ) [12], and kernel-based learning [13].

Here, we focus on RLS-TD(λ) learning. Bradtke and Barto [14] firstly proposed Least-Squares TD (LS-TD) and its recursive version RLS-TD in a linear regression. Afterward, in [15] and [12], LS-TD(λ) and RLS-TD(λ) were presented, respectively, as the extensions of LS-TD and RLS-TD with trace-decay parameter λ (from λ=0 to general $0 \le \lambda \le 1$). As mentioned by Boyan [15], LS-TD(λ) offers several significant advantages by comparing with TD(λ), such as fewer training samples, no step-size coefficient, and independence on arbitrary initial values. Similar advantages also appear in RLS-TD(λ). Moreover, RLS-TD(λ) has a recursive computation so that it is more suitable for online learning than LS-TD(λ).

In the other aspect, signal controller operates a traffic control scheme to schedule different directional flows, in order to make them evacuate from intersection without conflicts. Based on an MDP model framework, we study three kinds of traffic control schemes, namely fixed phase sequence (FPS), variable phase sequence (VPS), and adaptive phase sequence (APS), to know the effectiveness of the control strategies. All schemes take into account the properties of compatible traffic flows and signal phase sequences. In FPS, both of traffic flow combination and phase sequence are fixed. In VPS, traffic flow combination is fixed but phase sequence is variable. In APS, both of traffic flow combination and phase

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

sequence are variable. In other words, compatible traffic movements can be grouped optionally in APS to occupy the intersection zone. But control performance is unknown when APS is integrated into a related algorithm.

Considering the advantages of RLS-TD(λ) learning and the flexibility of APS scheme, the objective of this study is to design an related algorithm to make some improvements in control performances regarding traffic delay and computing efficiency. Two main contributions are made in the paper. First, we propose an algorithm that integrates RLS-TD(λ) learning into ADP in an application to real-time adaptive traffic signal control at a typical intersection. Second, APS control scheme is implemented to compare with the other schemes based on the proposed algorithm, and its effectiveness for adaptive control is verified.

The rest of the paper is organized as follows. A literature review of related works is presented in Section 2. Main principles of ADP approach with RLS-TD(λ) learning in linear function approximation are described in Section 3. Section 4 presents an on-line control algorithm after modelling a typical signalized intersection with the three kinds of traffic control schemes. In Section 5, we do experiments and analyse simulation results. Finally, we draw conclusions and give the future work.

2 Literature review

Nowadays, adaptive traffic control systems are widely used around the world. From the view of system deployment, more and more adaptive traffic control systems are decentralized, although centralized systems (e.g., SCOOT [16], SCATS [17]) have been higher degree of control logic on the intersection level [18]. In methodology, many distributed systems, such as OPAC [19], PRODYN [20], and RHODES [21], uniformly recognized the importance of DP in solving the multi-stage decision-making problems. DP is not implemented directly for these complex systems because of its weakness in computation. There are two ways often taken into account for computation efficiency. One is to apply heuristic techniques and the other one is to simplify state variables, for example, state aggregation method. Control performances in these ways are not guaranteed. By reducing search states, forward DP algorithms are proposed to obtain exact solutions of traffic control problems [22, 23]. However, related research has common limitations of the short-term planning and the requirement of perfect information about traffic arrivals.

There is considerable literature focusing on intelligent algorithms for adaptive traffic signal control systems [24-28]. A literature overview on this subject can be found in [29]. We introduce some influential references as follows. Park and Chang [24] used genetic algorithm for preliminary study of adaptive signal control at isolated intersection, provided that perfect knowledge of individual vehicle arrivals is available. Lee *et al*. [26] developed the work in [24] using rolling horizon approach for real-time adaptive signal optimization. García-Nieto *et al*. [30] proposed a swarm intelligence approach to find successful cycle programs of traffic lights, using a microscopic traffic simulator. However, these evolutionary algorithms are generally limited by the explorations of micro-evolutionary processes and the planning models based on cellular processes, which cost much computation in simulation. Srinivasan *et al.* [31] presented an approach about hybrid neural networks with fuzzy rules for real-time traffic signal control by introducing multi-agent system. The proposed system is proven to control a large-scale traffic network effectively. This approach is fine, but it is not sensitive to traffic environmental impacts. Recently,

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

reinforcement learning and ADP methods obtain much attention in applications to traffic signal control problems, especially in micro-simulation models [7, 32-35]. These methods are sensitive to environment due to their learning mechanisms for updating values in time. With regard to the distinctions of approximation processes, there are several learning methods to update the value of approximate function. In [36], Q-learning is used to update the cooperative multi-agent, according to the best-response of Q-value at next state. In each intersection agent, the large state-space still remains. The likelihood of Q-value evaluated by the count of visit states need to take a long simulation time to converge. In the works of Arel *et al.* [32], Box and Waterson [34], and Li *et al*. [37], reinforcement learning with the approximation of neural networks, namely action network and critic network, are implemented to adaptive traffic signal control. The parameters in these neural networks are updated by gradient descent. Cai *et al*. [7] suggest that a simple linear approximation is sufficient for online operation. Because non-linear functions, such as neural networks, for exploring the complex approximation may not prove cost effective due to some difficulties in network training [38], especially for the complex and discrete-time traffic control problem [39]. In [7, **Erreur ! Signet non défini.**], linear approximations based on feature-extraction function were successfully employed, using TD(0) (λ=0) learning to update parameters. As mentioned before, RLS-TD(λ) is superior to TD(λ) in theory study. However, the effectiveness of RLS-TD(λ) has not been researched in field study, for example, in the application to adaptive traffic signal control problem.

In addition, to authors' knowledge, works on traffic control scheme at intersection are usually discussed on fixed flow combinations. It means that compatible flows always remain the same group, owning the right-of-way to occupy the conflict zone. On the other hand, signal phase sequence mostly operates in a fixed or variable way (see the survey in [40]). In the paper, FPS and VPS control schemes do in this way, too. Moreover, we will test the performance of the defined APS scheme, and make the comparisons with FPS and VPS in experiments.

As discussed above, the parameter learning techniques in ADP algorithm and the traffic control schemes of signal plan should be further studied, aiming to find an appropriate solution of traffic signal control problem.

3 Approximate dynamic programming with RLS-TD(λ)

Although DP can achieve an optimal control policy under MDP model framework. Powell [4] has mentioned that DP has computation burden as “three curses of dimensionality”, which refers to the dimensions of state space, information space (space of random noise), and decision space. While, ADP approach with machine learning is computationally efficient to obtain a near-optimal solution. This section presents RLS-TD(λ) learning that is integrated into ADP approach due to the computational benefits of its learning efficiency.

3.1 Markov decision process

The environment of the decision problem we discuss is described by a finite MDP in [41]. A finite MDP is a tuple $< S, A, p, R >$, where S is the finite set of environment states, A is the finite set of actions,

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

$p: S \times A \times S \to [0,1]$ is the state transition probability function, and $R: S \times A \times S \to \mathbb{R}$ is the reward function. In detail, the state $s_t \in S$ describes the environment at each time *t*. The controller can choose the state at each time by taking actions $a_t \in A$. As a result of the action a_t, the environment changes its state from s_t to some $s_{t+1} \in S$, according to the state transition probability given by *p* which is represented as $p(s_t, a_t, s_{t+1})$. The controller receives immediately a scalar reward $r_t \in \mathbb{R}$ according to the reward function $R: r_t = R(s_t, a_t, s_{t+1})$.

For a given admissible policy π involving a sequence of decisions $a_t: S \to A$, the expected total reward is given by the following infinite-horizon discounted value function J^π, which is expressed as

$$J^\pi(s_t) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}, \tag{1}$$

where γ $(0 < \gamma < 1)$ is a discount factor. The objective is to find an optimal policy by minimizing Eq. (1). The following Bellman's equation offers a recursive solution to this goal. That is,

$$J^*(s_t) = \min_{a_t \in A(s_t)} \left\{ r_t + \gamma \sum_{s_{t+1} \in S} p(s_t, a_t, s_{t+1}) J^*(s_{t+1}) \right\} \tag{2}$$

for all $s_t \in S$.

3.2 ADP with linear function approximation

Traditional DP algorithms, such as value iteration and policy iteration, can solve the Bellman's equation given in Eq. (2) [42]. From practical experiences, there are two limitations of DP algorithms. First, the knowledge of transition probability $p(s_t, a_t, s_{t+1})$ is required, and it is subject to the complete information from the environment. This makes DP in a short-term planning, which affects the performance over the whole horizon. Although some methods weaken this impact, such as rolling horizon approach and model predictive control, they will cost additional computation especially in large-scale DP problems. Second, it has to loop over the entire state space to evaluate the decisions such that an optimal stationary policy is finally obtained. The computation complexity is in an exponential order to the size of state variables. The computation requirement grows extremely even for a small size problem. For example, there is an isolated intersection with 8 lanes in 4-phase signal plan. Each lane can contain at most 19 queuing vehicles. Arrival information is either 0 or 1 for each lane and signal decision for each lane is either 0 remaining green or 1 switching current status. Thus, the computation order (without any simplification by control scheme) is $4 \times 20^8 \times 2^8 \times 2^8 \approx 6.71 \times 10^{15}$. It is impractical to evaluate the states by all iterations for model convergence.

Whereas in ADP, the characteristics of function approximation, parameter adaptation by learning, and real-time forward operation can address the DP drawbacks mentioned above. The structure of ADP refers to a continuous approximation function that is defined to replace the exact cost-to-go value function in DP. From literature [7, 37], ADP with linear function approximation is easier to be trained than neural

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

network, and is more suitable to be implemented for discrete state-space problems. Generally, a linear function approximation can be expressed as

$$\tilde{J}(s_t, \theta_t) = \phi_t^{\mathrm{T}} \theta_t \tag{3}$$

where ϕ_t refers to the feature-extraction function that maps state s_t to valued feature (column) vector with *N*-dimension, and θ_t refers to the associated parameter (column) vector as the same dimension with ϕ_t. Using linear function approximation, consequently, the objective value function in ADP is given by

$$\hat{J}(s_t) = \min_{a_t \in A(s_t)} E\{r_t + \gamma \tilde{J}(s_{t+1}, \theta_t)\} \tag{4}$$

where $\hat{J}(s_t)$ is the observed value of current state according to environment.

In ADP, the estimated value $\tilde{J}$ of visiting actual state by following a learning process will approximate the optimal value J^{π} in convergence. Instead of computing the optimal J^{π} with huge-dimensional states, we compute the low-dimensional parameter vector θ_t. As a result, computation requirement is substantially reduced. For linear function approximation, it has been proven that there is only one optimum θ^*, which is achieved ultimately in convergence [11]. Due to limited state resources and solutions from this approximation, the parameter θ_t is updated to an ideal goal θ^* by an incremental process to minimize some error metric between the estimated value $\tilde{J}$ and observed value $\hat{J}$. In the following part, we will use RLS-TD(λ) learning technique to find θ^*.

3.3 RLS-TD(λ)

Suppose that we observe a sequence of states s_t based on the simulation with random input information w_t, i.e., $(s_0, a_0, w_1, s_1, a_1, w_2, \ldots, s_T, a_T, w_{T+1})$. Temporal difference δ_t (also called TD error) is defined corresponding to the transition from s_t to s_{t+1} by

$$\delta_t = r_t + \gamma \tilde{J}(s_{t+1}, \theta_t) - \tilde{J}(s_t, \theta_t) \tag{5}$$

Since the linear case $\tilde{J}(s_t, \theta_t) = \phi_t^{\mathrm{T}} \theta_t$, TD error in Eq. (5) at time *t* can be rewritten as

$$\delta_t = r_t - (\phi_t - \gamma \phi_{t+1})^{\mathrm{T}} \theta_t \tag{6}$$

Then, for $t = 0, 1, \ldots, T$, multi-step TD (also called TD(λ)) updates θ_t according to the formula

$$\theta_{t+1} = \theta_t + \eta_t \delta_t \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \nabla_{\theta_t} \tilde{J}(s_t, \theta_t) \tag{7}$$

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

where θ_0 is initialized to an arbitrary vector, η_t is a sequence of scalar step-size, λ is a parameter in $[0,1]$, and $\nabla_{\theta_t}\tilde{J}(s_t,\theta_t)$ is the vector of partial derivatives with respect to the components of θ_t.

In the case of linear function approximation, a more convenient representation of TD(λ) is obtained by defining a sequence of eligibility vectors z_t. That is,

$$\begin{aligned} z_t &= \sum_{k=0}^{t}(\gamma\lambda)^{t-k}\nabla_{\theta_t}\tilde{J}(s_t,\theta_t) \\ &= \sum_{k=0}^{t}(\gamma\lambda)^{t-k}\phi_t. \end{aligned} \tag{8}$$

With this notation, Eq. (7) can be rewritten as

$$\theta_{t+1} = \theta_t + \eta_t\delta_t z_t \tag{9}$$

and the eligibility vectors can be updated recursively according to

$$z_{t+1} = \gamma\lambda z_t + \phi_{t+1} \tag{10}$$

initialized with $z_0 = \mathbf{0}$.

In the study of Tsitsiklis and Van Roy [11], the above linear TD(λ) algorithm is proved to converge with probability 1 under certain assumptions. For special case λ=0, TD(0) is an equivalent to single-step TD algorithm where only the most recent observation matters to calculate the value function as well as the update of approximation.

To improve the efficiency of linear TD(λ) learning, LS-TD and RLS-TD learning methods are suggested in [14]. In LS-TD, the least-squares approximation to θ^* at time $t\,(t \geq 1)$ is the vector θ_t that minimizes the quadratic objective function

$$O(\theta_t) = \frac{1}{t}\sum_{i=1}^{t}\left(r_i - (\phi_i - \gamma\phi_{i+1})^{\mathrm{T}}\theta_t\right)^2. \tag{11}$$

By taking the partial derivative of $O(\theta_t)$ with respect to θ_t, as well as employing the instrumental variable approach, LS-TD solution of Eq. (11) gives us the t^{th} estimate to θ^*. That is,

$$\theta_t = \left(\frac{1}{t}\sum_{i=1}^{t}\phi_i(\phi_i - \gamma\phi_{i+1})^{\mathrm{T}}\right)^{-1}\left(\frac{1}{t}\sum_{i=1}^{t}\phi_i r_i\right). \tag{12}$$

LS-TD method requires the computation of a matrix inverse at each time step. Thus, recursive least-squares technique is used to derive a modified algorithm, namely RLS-TD, to reduce the computational complexity of LS-TD. The weight update rules of RLS-TD are as follows:

$$\tilde{\delta}_t = r_t - (\phi_t - \gamma\phi_{t+1})^{\mathrm{T}}\theta_{t-1} \tag{13}$$

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

$$P_t = P_{t-1} - \frac{P_{t-1}\phi_t(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}}{1+(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}\phi_t} \tag{14}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1+(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}\phi_t}\tilde{\delta}_t\phi_t. \tag{15}$$

Notice that Eq. (15) looks like the TD(0) learning rule for function approximation that is linear in the parameters, except that the scalar step-size parameter in TD(0) is replaced by a gain matrix. To use RLS-TD method, users must specify θ_0 and P_0. θ_0 is normally set to be vector **0**. P_0 is typically to use $P_0 = \varepsilon I$, where *I* is the identity matrix and ε is a small positive constant.

According to [12], RLS-TD(λ) can be viewed as the extension of RLS-TD with $0 \le \lambda \le 1$. Besides Eq. (13), the weight update rules of standard RLS-TD(λ) are given by

$$P_t = P_{t-1} - \frac{P_{t-1}z_t(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}}{1+(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}z_t} \tag{16}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1+(\phi_t - \gamma\phi_{t+1})^{\mathrm{T}} P_{t-1}z_t}\tilde{\delta}_t z_t. \tag{17}$$

The detailed derivation of RLS-TD(λ) and convergence proof can be found in [12]. In the next section, we will focus on adaptive traffic signal control system, in which the ADP with RLS-TD(λ) learning is implemented.

4 Adaptive traffic signal control algorithm

In this section, firstly, we present modeling framework of traffic signal control at intersection based on MDP. Then, traffic control schemes and related decisions are discussed. Finally, we summarize the algorithm using a multi-step planning for online adaptive control.

4.1 Traffic dynamic system

Traffic signal control at intersection is considered as a stochastic discrete event system. Time intervals refer to identical divisions of a planning horizon. Assume that one interval is from *t* to *t*+1. At first, the following principal assumptions are given:

(1) Traffic signal durations are divided by discrete unit intervals. The size of one interval is 2 seconds, which is usually set to be a safe headway.
(2) Signal phases are composed of effective greens and reds only, excluding amber intervals. There is no pedestrian phase.
(3) Each phase contains at least mandatory intervals including inter-green intervals and minimum green intervals, during which no signal switching is admissible. The extension of green signal is one interval per step. There is no lost time for vehicle receiving green signal.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

(4) The discharge rate (saturation flow) on each lane is one vehicle per interval. This rate is equivalent to 1800 vehicles per hour.

Under the framework of MDP, as mentioned in Section 3.1, model formulation requires the characterizations of state, action, transition probability, and reward (cost) function. At time *t*, for an isolated intersection with total *N* lanes, traffic state can be expressed by $s_t = (k_t, x_t), s_t \in S$, where k_t is a vehicle state vector with element $k_t(n)$ representing actual number of queuing vehicles on lane *n* ($n = 1,...,N$); x_t is a signal state vector with element $x_t(n)$ representing the signal state on lane *n*. Assign $x_t(n) = 1$ to green signal and $x_t(n) = 0$ to red signal. The decision or action of the system is $a_t \in A(s_t)$. Let $A = \cup_{s_t \in S} A(s_t)$ denote action space. We define $a_t(n)$ to be a binary variable that $a_t(n) = 1$ means signal switches on lane *n*, otherwise, $a_t(n) = 0$. During the mandatory intervals, we have $a_t(n) = 0$ as well. Once the system makes a decision at time *t*, the traffic state s_t will change. The transitions of signal state and vehicle state are expressed respectively by

$$x_{t+1}(n) = (x_t(n) + a_t(n)) \bmod 2 \tag{18}$$

$$k_{t+1}(n) = k_t(n) + w_t(n) - y_t(n) \tag{19}$$

where $w_t(n)$ denotes traffic arrival satisfying a distribution according to traffic arrival rates. It adopts the value of either 0 or 1 vehicle/interval (veh/int). The traffic departure rate $y_t(n)$ is also a binary variable restricted by

$$y_t(n) = \begin{cases} 1, & \text{if } x_t(n) = 1 \text{ and } k_t(n) + w_t(n) \geq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Because transition probabilities are hard to achieve for all state transitions, we can use the power of computer to generate random observations, which satisfy a specific distribution. The process is referred as Monte Carlo sampling. We have deterministic state transitions described in Eqs. (18) and (19). Vehicle state at next time step is determined by the system state s_t, information of vehicle arrival w_t, and policy decision a_t at the current step *t*. Since state transitions are influenced by random arriving vehicles, the process of vehicle state can be seen as a stochastic process with Markov property. As for the reward function r_t, in our study, it is defined by the sum of queue lengths at the next time $t+1$. That is,

$$r_t = R(s_t, a_t, s_{t+1}) = \sum_{n=1}^{N} k_{t+1}(n). \tag{21}$$

According to Eq. (2), the objective of the control problem is to minimize the discounted total queue lengths corresponding to traffic delay.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

4.2 Traffic control schemes

A typical four-approach isolated intersection with conflicts is depicted in Fig. 1. The right-turn movement is integrated into the straight one sharing the same signal. Compatible traffic flows are grouped into combinations that have the right-of-way to occupy the conflict zone. In this section, three traffic control schemes, namely FPS, VPS, and APS are discussed.

As shown in Fig. 2 (a), the intersection is controlled by typical four signal phases. The eight movements are divided into four flow combinations, denoted by G_1, G_2, G_3, and G_4. In FPS scheme, signal controller organizes these combinations in a continuous loop, e.g., $G_1 \to G_2 \to G_3 \to G_4 \to G_1$. Generally, the inter-green interval or red clearance time represented by the barrier is used to separate phases for different flow directions. Adaptive signal control operates FPS to either extend or terminate current green phase, according to the detected traffic arrival information. In a particular state s_t, the action space $A_{\text{FPS}}(s_t)$ has only 2 possible action vectors such that $A_{\text{FPS}}(s_t) = \{a_t^0, a_t^1\}$. The action of signal unchanged is $a_t^0 = (0,0,0,0,0,0,0,0)^{\text{T}}$ and the action of signal switch is, for example, $a_t^1 = (1,1,0,0,1,1,0,0)^{\text{T}}$ for the next particular phase. The action a_t^1 indicates that signals on lane 1 and lane 5 will change from green to red, and on lane 2 and lane 6 they do contrarily. As calculated by Eq. (18), the next transferred signal state is $x_{t+1} = (x_t + a_t^1) \bmod 2 = (0,1,0,0,0,1,0,0)^{\text{T}}$.

The flow combinations for signal phases in VPS are the same with those in FPS. Differently, the phase in VPS may operate one after another in an uncertain or unordered way. For example, in Fig. 2 (b), the phase sequence operates as $G_1 \to G_3 \to G_2 \to G_4 \to G_1$. With VPS scheme, signal controller makes the phase sequence optimization and skips to another one considering its performances, such as queue length and vehicle waiting time. Therefore, the action space $A_{\text{VPS}}(s_t)$ has 4 possible action vectors, one for remaining the same phase and three actions for switching to other phases.

In APS scheme, the possibilities of phase choice are more than those in four-phase mechanism referring to FPS and VPS. It means that there are additional traffic flow combinations which can avoid flow conflicts, e.g., lane 1 can be combined with one of lanes 4, 5, and 6. We list all combination possibilities, as shown in Fig. 2 (c). For the relationship (combination coefficient) between two lanes, assign "1" if they are non-conflicting, and assign "0" otherwise. One possible phase sequence may operate like (1, 5) $\to$ (1, 6) $\to$ (3, 8) $\to$ (2, 6) $\to$ (4, 7) $\to$ …. In total, there are 12 combinations considering the upper triangular matrix as symmetry. APS groups all possible compatible traffic flows for signal phases. In other words, the action space $A_{\text{APS}}(s_t)$ has 12 possible action vectors. This scheme is firstly considered in our study for the adaptive traffic signal control algorithm.

The control behaviors of traffic flows organized by FPS, VPS, and APS get sequentially more and more flexible. For numerous adaptive traffic signal control systems, signal phase duration is not fixed and phase sequence is cyclic, such as FPS scheme. When next alternative phase does not work in a particular phase order but in an acyclic way, it operates like VPS scheme. Moreover, in APS, any compatible traffic flows can be serviced at intersection. It does not have strict phase sequence requirement, which is significantly different from the conventional four-phase mechanism.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

4.3 Control algorithm

Regarding the RLS-TD(λ) learning in ADP approach and our traffic system modeling above, some specifics are required to design the algorithm for traffic signal control.

The linear function approximation $\tilde{J}(\cdot)$ containing feature-extraction function ϕ_t and associated parameter θ_t is related to the traffic state $s_t(n) = (k_t(n), x_t(n))$. That can be expressed as

$$\tilde{J}(s_t, \theta_t) = \sum_{n=1}^{N} \phi_t^{\mathrm{T}}(k_t(n), x_t(n))\theta_t(n) = \phi_t^{\mathrm{T}}\theta_t \tag{22}$$

where $\phi_t = (\phi_t(k_t(n), x_t(n)), n = 1, 2, \ldots, N)^{\mathrm{T}}$ and $\theta_t = (\theta_t(n), n = 1, 2, \ldots, N)^{\mathrm{T}}$. We define $\phi_t(k_t(n), x_t(n))$ as

$$\phi_t(k_t(n), x_t(n)) = \begin{cases} (k_t(n), 0)^{\mathrm{T}}, & \text{if } x_t(n) = 1 \text{ (signal green)} \\ (0, k_t(n))^{\mathrm{T}}, & \text{otherwise.} \end{cases} \tag{23}$$

and define $\theta_t(n) = (\theta_t^g(n), \theta_t^r(n))^{\mathrm{T}}$, allowing $\theta_t^g(n)$ and $\theta_t^r(n)$ to the corresponding queue length variable $k_t(n)$ if lane *n* receives green signal and red signal, respectively.

As mandatory intervals include inter-green intervals and minimum green intervals, we adopt multi-step planning forward in the dynamic process. The objective function in *M*-step planning is formulated as

$$J^*(s_t) = \min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+M-1} \gamma^{k-t} r_k + \gamma^M J^*(s_{t+M}) \right\}. \tag{24}$$

With linear function approximation $\tilde{J}(\cdot)$ in ADP, the above Eq. (24) can be modified by

$$\hat{J}(s_t) = \min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+M-1} \gamma^{k-t} r_k + \gamma^M \tilde{J}(s_{t+M}, \theta_t) \right\}. \tag{25}$$

And signal controller can implement the following optimal action greedily

$$a_t^* = \arg\min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+M-1} \gamma^{k-t} r_k + \gamma^M \tilde{J}(s_{t+M}, \theta_t) \right\}. \tag{26}$$

According to Eqs. (13), (16), and (17), the update of θ_t in RLS-TD(λ) learning rule by *M*-step planning is calculated as

$$\tilde{\delta}_t = \sum_{k=t}^{t+M-1} \gamma^{k-t} r_k - (\phi_t - \gamma^M \phi_{t+M})^{\mathrm{T}} \theta_{t-1} \tag{27}$$

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

$$P_t = P_{t-1} - \frac{P_{t-1} z_t (\phi_t - \gamma^M \phi_{t+M})^{\mathrm{T}} P_{t-1}}{1 + (\phi_t - \gamma^M \phi_{t+M})^{\mathrm{T}} P_{t-1} z_t} \tag{28}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma^M \phi_{t+M})^{\mathrm{T}} P_{t-1} z_t} \tilde{\delta}_t z_t . \tag{29}$$

Derivation of RLS-TD(λ) in *M*-step planning is located in Appendix. Finally, on-line adaptive traffic signal control algorithm using ADP with RLS-TD(λ) learning is summarized in Algorithm 1.

Algorithm 1: Adaptive traffic signal control algorithm by ADP_RLS-TD(λ)

1. Choose an initial state $s_0(n) = (k_0(n), x_0(n))$, parameter $\theta_0(n) = (\theta_0^g(n), \theta_0^r(n))^{\mathrm{T}}$ for each lane *n*; set time $t = 0$, planning step $m = M$;
2. Receive the traffic arrivals w_t during *M* intervals from detected information;
3. Choose the action space A_{APS} (or $A_{\mathrm{FPS}}, A_{\mathrm{VPS}}$);
4. **while** $t \le T$ **do**
5. **if** $m > 0$ **then**
6. Signal unchanged with $a_t^* = 0$, and $m = \max(m-1, 0)$;
7. **else**
8. **for** each $a_t \in A_{\mathrm{APS}}(s_t)$ **do**
9. Pre-calculate and store the accumulated rewards and estimated values;
10. **end for**
11. Find the optimal decision a_t^* using Eq. (26);
12. **if** $a_t^* = 1$ **then**
13. Change signal to all-red, and set $m = M - 1$;
14. **end if**
15. **end if**
16. Update functional parameter vector θ_t using Eqs. (27), (28), and (29);
17. Implement optimal decision a_t^* at time interval *t*;
18. Transfer system state $s_t(n)$, including signal state $x_t(n)$ and vehicle state $k_t(n)$ by Eqs. (18) and (19), respectively;
19. $t = t + 1$;
20. **end while**

5 Simulation and analysis

In this section, traffic signal control at isolated intersection depicted in Fig. 1 is simulated. Traffic scenarios in different traffic arrival rates are tested, using the three different traffic control schemes, i.e., FPS, VPS, and APS. To validate the proposed ADP_RLS-TD(λ) algorithm, its experimental results are compared with other different methods.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

5.1 Data preparation

In practice, traffic data is detected from inductive loops embedded upstream of each lane or other detecting techniques. In our experiments, random traffic arrival data is generated in each step by computer simulation. A simple 0-1 Bernoulli distribution is adopted. By using inverse transformation method (ITM), random data consist of binary values 1 and 0, where value 1 represents vehicle arrival during one interval, and 0 otherwise. The probability of number "1" in distribution means an expected traffic arrival rate. Because vehicle arrivals generated by Monte Carlo simulation, traffic volumes referring to the sum of number "1" are a little different given the same traffic arrival rates.

For a finer planning, the value of either 1 or 0 in each unit increment, by which one interval can be divided integrally, is randomly chosen. The sum of values in all increments per interval is subject to the value of being either 1 or 0. For example, if there are 4 increments per interval, the situation of vehicle arrivals may be $(0,0,1,0)$ or $(0,0,0,0)$, etc.

Simulator generates data according to the traffic arrival rate vector, which corresponds to the combinations (G_1, G_2, G_3, G_4). Traffic Scenario A and B that have asymmetric and symmetric rates, respectively, as well as the corresponding traffic volumes, are shown in Table 1. Traffic Scenario C gives symmetric time-varying arrival rates, which are processed smoothly ranging from 0.10 veh/int to 0.20 veh/int, as shown in Fig. 3. Notice that the highest traffic arrival rate in Traffic Scenario B has the intensity (calculated by the method in [43]) almost 0.9, which is already close to the road saturation.

Table 1 Traffic scenarios of asymmetric and symmetric average arrival rates.

	Arrival rate (veh/int)	Traffic volume by ITM (veh/h)
Traffic Scenario A	(0.10, 0.20, 0.10, 0.20)	(350, 722, 365, 735)
Traffic Scenario B	(0.20, 0.20, 0.20, 0.20)	(742, 725, 740, 735)

Table 2 Details of system parameters.

Parameters	Definitions	Value setting
T	simulation period	40000 intervals
N	total lanes at intersection	8
$g_{\min}$	minimum green time	3 intervals
$g_{\max}$	maximum green time	30 intervals
g_{int}	inter-green (all-red) time	1 interval
M	mandatory multi-step	4 intervals
$\theta_0^g(n)$	initial parameter to green signal	5 in Scenario A and B; 3 in Scenario C
$\theta_0^r(n)$	initial parameter to red signal	5
γ	discount factor	0.90
η	learning rate constant	0.001
ε	parameter in matrix P_0	0.01
s_a	saturation (departure) flow	1 veh/int=1800 veh/h

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

5.2 Performance measure

The main performance measure refers to the term of average traffic delay denoted by *D*, which is expressed in seconds and calculated according to the following formulation in OPAC system [43].

$$D = \frac{T_{\mathrm{D}}}{T_{\mathrm{A}}} = \frac{2\sum_{n}^{N}\sum_{t}^{T} k_t(n)}{\sum_{n}^{N}\sum_{t}^{T} w_t(n)} \tag{30}$$

where T_{D} (veh·int) is to measure the total vehicle-intervals (in 2 s units) that is the sum of queue lengths of all lanes N during the period T. T_{A}(veh) is the total number of vehicle arrivals at intersection.

Moreover, average queue length of waiting vehicles at intersection is concerned, and computation time is considered for the algorithm efficiency.

5.3 Implementation

For the simulation of traffic signal control system, system parameter setting is given in Table 2. Actually, learning parameter η in the comparative TD(λ) is step-size scheduling in a time-varying form [11]. From the experience of related study [7], we use a constant leaning rate at η=0.001. By numerical experiments, the functional parameters $\theta_0^g(n)$ and $\theta_0^r(n)$ are initially set to guarantee the control performance at the beginning of simulation. Regarding the performance of ADP with RLS-TD(λ) learning in FPS, VPS, and APS schemes, we compare the proposed algorithm with a fixed-time control (FC) method and certain adaptive control (AC) algorithms, i.e., Greedy algorithm, Heuristic algorithm, and Q-learning. They are introduced as follows.

- FC. In this control method, we consider the algorithm in [8], called Haijema-MDP in this paper. The optimal-fixed-cycle method starts with the minimum-cycle-length algorithm which increases the cycle length by unit increment based on the evaluation of Markov chains, according to the expected traffic arrival rates. Algorithm stops until no better cycle length can be found. In the literature, authors mentioned that Haijema-MDP was better than Webster's method, especially in unbalanced arrival rates.
- Greedy algorithm. In this method, the *M*-step planning evaluation function in Eq.(24) lacks the part of exact state values and only has *M* steps of immediate rewards for searching forward. The decisions are greedily chosen by evaluating the immediate reward function.
- Heuristic algorithm. This is a forward search dynamic programming algorithm, shorten by FSDP [44]. Because this method has a global optimization solution, computation complexity is very high by increasing the planning forward steps. We make this algorithm for two tests with planning horizon T_p =16 intervals and T_p =25 intervals, respectively.
- Q-learning. In this method, the normal rule for updating value function is used as follows,

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

$$Q_{t+1}(s,a) = Q_t(s,a) + \eta_t(r_t(s,a) + \gamma \min_{a' \in A} Q_t(s',a') - Q_t(s,a)), \quad (31)$$

where η_t is the learning rate, s' is the transferred state from s taken action a. Entire traffic states need to be looped because their values need to be updated overall. Here, states are reduced by using three flow density levels, i.e., low, medium, and high. From numerical experiments, various settings of thresholds between the levels are confirmed, concerning the different traffic arrival rates.

All the experiments are implemented in computer simulation. The program is built by MATLAB 64 bits and executed by processor Intel Core i5, CPU 2.67GHz × 4. Assume that the detected traffic arrival information in the future can be received in *M* intervals. The information data is processed in rolling horizon approach, which can make the system decision per interval with the rolling *M* steps forward. In real-time operation, the executive time of algorithm should be guaranteed. That is to say, the run time per step while making a decision must be less than one interval in reality.

5.4 Results and analysis

To know the effectiveness of RLS-TD(λ) learning for the traffic control field, firstly, we analyse parameter evolutions in the algorithm. By implementing RLS-TD(λ) learning, the parameter evolutions of selected lanes are illustrated in Fig. 4. All of these parameters trend to relative steady levels after some steps. Obvious differences appear in the parameters θ_t^g, which associate to the feature-extraction function regarding vehicles receiving green signal. The values of steady levels related to $\theta_t^g(1)$ in Fig. 4(a) and $\theta_t^g(7)$ in Fig. 4(d) are smaller than those related to $\theta_t^g(2)$ in Fig. 4(b) and $\theta_t^g(6)$ in Fig. 4(c). This could be explained as follows. In Traffic Scenario A, the arrival rates of lane 1, 2, 6, and 7 are 0.1, 0.2, 0.2, and 0.1 veh/int, respectively. Lane 1 and 7 receive less green durations than lane 2 and 6, and go faster to reach the lower values at steady level. While the parameters θ_t^r give almost the same values at steady levels. That means a little difference of queue lengths on the lanes where vehicles are waiting for red signal. Theoretically, RLS-TD(λ) is better than the conventional TD(λ) learning, according to studies in [15, 12]. We investigate their applications in the traffic control field and compare their experimental results, as shown in Fig. 5. Obviously, the parameters $\theta_t^g(n)$ and $\theta_t^r(n)$ in RLS-TD(λ) achieve much less variances than those in TD(λ). Moreover, RLS-TD(λ) makes earlier stable trends in their evolutions. From the subfigures, different λ values in RLS-TD(λ) generate almost the same parameter evolutions. By contrary, big differences appear among the performances of TD(λ). Especially, in Fig. 5(d), the parameter evolutions of TD(1) have the largest fluctuations. Similar aforementioned results can be obtained by other parameter settings of λ and *n*.

In Fig. 6, the results of average traffic delay are compared between RLS-TD(λ) and TD(λ) learning with different settings of λ. Traffic Scenario A, B, and C are tested. The performance of RLS-TD(λ) learning is better than TD(λ) learning, by reason of its advantages for updating parameters. In Traffic Scenario B and C, TD(1) even gets unreasonable values, which are removed.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

Table 3 Results of average delay (s) in Traffic Scenario A, B, and C.

Traffic Scenario	A			B			C		
Haijema-MDP	23.68			49.02			30.69		
Phase sequence	FPS	VPS	APS	FPS	VPS	APS	FPS	VPS	APS
Greedy	23.21	21.88	12.37	59.90	55.85	24.52	30.08	28.56	18.87
FSDP (T_p =16)	18.65	14.80	8.74	45.30	40.53	12.58	22.12	20.25	9.23
FSDP (T_p =25)	17.13	13.76	8.25	39.76	38.68	11.80	20.74	18.82	9.01
Q-learning	20.03	18.23	12.02	49.32	48.90	22.10	28.14	26.88	15.42
ADP_RLS-TD(0)	20.06	17.44	10.57	42.24	41.91	19.43	25.67	23.94	12.33
ADP_RLS-TD(0)*[1]	14.78	11.16	9.34	20.09	18.36	13.41	14.38	12.27	9.72
Improvement (%)	0	15.43	45.96	0	5.04	57.34	0	8.11	46.95

Note 1: simulation step is 0.25 interval (0.5 s) in a fine solution.

Table 4 Comparisons of run time in Scenario A.

Methods	FSDP (T_p =16)			FSDP (T_p =25)			ADP_RLS-TD(λ)			
Phase sequence	FPS	VPS	APS	FPS	VPS	APS	FPS	VPS	APS	APS*[1]
Run time (min)	1.3	6.3	99.6	3.2	23.9	582.5	0.3	0.7	1.1	4.4

Note 1: the same in Table 3.

From the analysis of parameter evolutions and the performance of average delay, RLS-TD(λ) is superior to TD(λ) learning in ADP approach for adaptive traffic signal control. By the following experiments, ADP with RLS-TD(λ) learning comparing with other control methods are discussed.

Table 3 shows the results of average traffic delay after implementing different algorithms. The performances of FPS, VPS, and APS schemes are also involved. As a whole, AC methods that are able to adjust phase duration and phase sequence, are better than FC, except the results of Greedy algorithm for the FPS and VPS schemes in Traffic Scenario B. Greedy algorithm with limited information to guide decision making is not suitable for adaptive traffic signal control in these cases. With more model information (more planning steps), FSDP (T_p=16) and FSDP (T_p=25) are much better than Greedy algorithm. By comparing the learning methods, ADP_RLS-TD(λ) (λ=0) yields better results than Q-learning. Especially under the severe traffic density situation - Traffic Scenario B, ADP_RLS-TD(0) obtains delay reductions about 7 s in FPS and VPS, and 2.7 s in APS. On the other hand, FPS, VPS, and APS schemes have significant impacts on the performance of AC algorithms. Comparing with FPS, APS can obtain the 45.96%, 57.34%, and 46.95% improvements of average delay reduction in Traffic Scenario A, B, and C, respectively. Followed by VPS, average delay reductions are 15.43% and 8.11% in Traffic Scenario A and C, and 5.04% for Traffic Scenario B. The detailed improvements related to these control schemes in AC methods are shown in Fig. 7. Integrated into appropriate algorithms, APS outperforms on average traffic delay than FPS and VPS, which are related to cyclic and acyclic signal control, respectively. APS can operate in a highly adaptive way. It means that not only the phase sequence is acyclic but also all the possible phases are evaluated by utility function.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

As shown in Table 3 and Fig. 7, FSDP (T_p=25) method looks like the appropriate approach with lower traffic delays and higher improvements. Actually, see Table 4, FSDP (T_p=25) costs much time during the whole simulation, especially in APS, where run time is 582.5 min. In VPS, it costs about 23.9 min. Because the action space of APS is much larger than VPS, FSDP must search optimal values in a huge state-space when using APS scheme. Meanwhile, the computing time increases considerably as T_p increases. By contrary, ADP_RLS-TD(λ) algorithm costs a little time in computation, even in APS where time consumption is about 1.1 min for the whole simulation (about 0.002 s per simulation step). In this algorithm, a forward process and an estimation by linear function approximation are the key points to reduce the computation complexity. Based on the simulation results, we argue that the optimal DP algorithms such as FSDP cannot conveniently apply to complex applications. For example, it is hard to work in the APS scheme, finer planning solution, and traffic network control. While, a near-optimal DP algorithm integrated with learning techniques, such as ADP_RLS-TD(λ) in this paper, indicates great potential to solve the high-dimensional problem. The computational efficiency of ADP with RLS-TD(λ) learning is fully capable for online control at isolated intersection.

Additionally, in [2], the authors presented the evidence that planning in very small steps may be the most efficient approach even on pure planning problems if the problem is too large to be solved exactly. In the work of [7], the fine solution was achieved from perturbation learning which was better than the coarse solution. We investigate the finer step planning to know if this planning is suitable for ADP with RLS-TD(λ) learning in its application. Let the small step be 0.25 interval (0.5 s) in the experiments. Considering ADP with RLS-TD(0) algorithm, consequently, the results of traffic delay are better than those by the original step of being one interval, as shown in Table 3. On the other hand, the performances of average queue length are shown in Fig. 8, which indicates the relevant effectiveness by the different control schemes as well. Obviously, APS works very well in less variance and lower queue length. Although these fine solutions cost more computation time (see in Table 4), it is enough to guarantee the online operation by using ADP with RLS-TD(λ) learning.

6 Conclusions

This study has investigated ADP with RLS-TD(λ) learning in an application to real-time adaptive traffic signal control at intersection. Regarding the evolutions of linear functional parameters in the tested scenarios, RLS-TD(λ) learning converged faster to the steady state, and the parameters fluctuated less than TD(λ) learning. Meanwhile, less traffic delays were observed by RLS-TD(λ) learning with different settings of λ. On the other hand, we compared ADP_RLS-TD(λ) algorithm with other control methods from experimental results. Although the heuristic algorithm FSDP achieved low traffic delays, it consumed too much time to operate in the fine scheduling case. While ADP with RLS-TD(λ) learning reduced largely the run time and yielded acceptable traffic delays, especially in the fine solution. Moreover, from the performances of the algorithms with FPS, VPS, and APS strategies, APS outperforms with 45%~58% delay reduction than FPS, especially under the high traffic density situation. However, it is impractical to implement the APS strategy in current real-world traffic operation due to the security factor when merging flows from different directions. If protocols among vehicles are able to guarantee the path security, flexible

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

flow organizations, such as APS, would likely come true.

In the near future, the proposed algorithm will be extended to two aspects. First, to make more reasonable and realistic, the model side need enrich the definitions of traffic state and control reward. As objective function with single control variable in learning methods often has limited performances, multi-objective signal timing control is a potential solution of this problem. Encouraged work could be found in references [45-46]. Second, the algorithm could also be promoted, aiming to find a solution of coordinated network control. By the reason of effects of upstream traffic, it is very important to consider the coordination among intersections so as to improve signal control efficiency in a global view.

Appendix: Derivation of RLS-TD(λ) in *M*-step planning

In *M*-step planning, the objective function Eq. (11) is modified as:

$$O(\theta_t)=\frac{1}{t}\sum_{i=1}^{t}\left(\sum_{k=i}^{i+M-1}\gamma^{k-i}r_k-(\phi_i-\gamma^M\phi_{i+M})^{\mathrm{T}}\theta_t\right)^2. \tag{32}$$

According to related theories [14, 47], we can rewrite Eq. (12) by using ϕ_t as the instrumental variable in LS-TD. That is,

$$\theta_t=\left(\frac{1}{t}\sum_{i=1}^{t}\phi_i(\phi_i-\gamma^M\phi_{i+M})^{\mathrm{T}}\right)^{-1}\left(\frac{1}{t}\sum_{i=1}^{t}\phi_i\sum_{k=i}^{i+M-1}\gamma^{k-i}r_k\right). \tag{33}$$

In LS-TD(λ), θ_t can be estimated as

$$\begin{aligned}\theta_t&=\left(\frac{1}{t}\sum_{i=1}^{t}z_i(\phi_i-\gamma^M\phi_{i+M})^{\mathrm{T}}\right)^{-1}\left(\frac{1}{t}\sum_{i=1}^{t}z_i\sum_{k=i}^{i+M-1}\gamma^{k-i}r_k\right)\\&\approx\left(\sum_{i=1}^{t}z_i(\phi_i-\gamma^M\phi_{i+M})^{\mathrm{T}}\right)^{-1}\left(\sum_{i=1}^{t}z_i\sum_{k=i}^{i+M-1}\gamma^{k-i}r_k\right)\end{aligned} \tag{34}$$

where using the eligibility vector z_t in Eq. (10) substitutes the variable ϕ_t. According to matrix inverse lemma and RLS-TD(λ) [12], the parameter vector θ_t updated by RLS-TD(λ) in *M*-step planning (in Eqs. (27), (28), and (29)) can be guaranteed.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

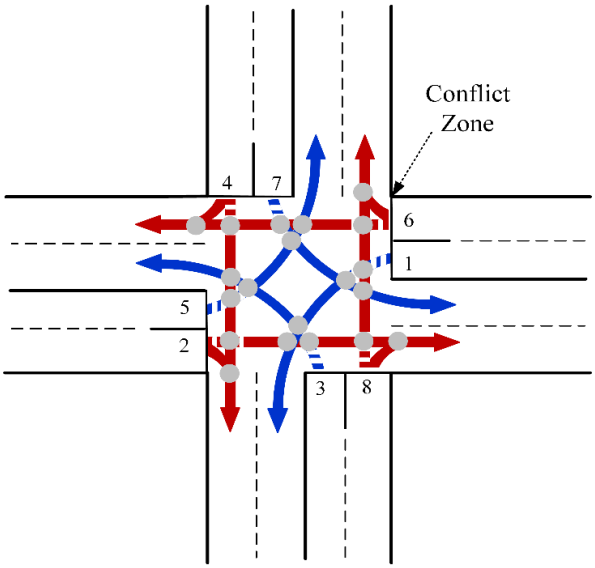
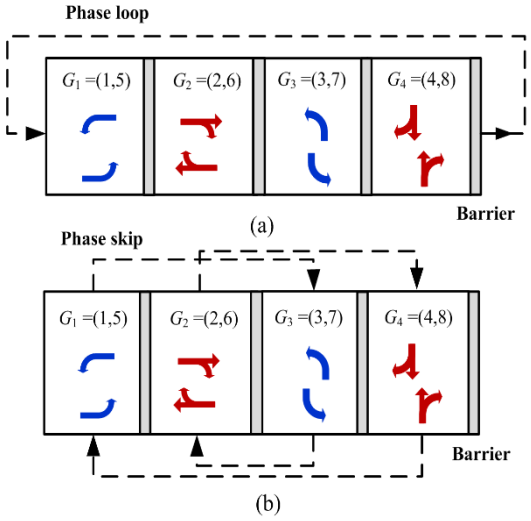


Fig. 1 Illustration of a typical intersection with conflicts.



Lane	1	2	3	4	5	6	7	8
1		0	0	1	1	1	0	0
2	0		0	0	1	1	1	0
3	0	0		0	0	1	1	1
4	1	0	0		0	0	1	1
5	1	1	0	0		0	0	1
6	1	1	1	0	0		0	0
7	0	1	1	1	0	0		0
8	0	0	1	1	1	0	0	

(c)

Fig. 2 Illustrations of traffic control scheme in (a) FPS, (b) VPS, and (c) APS.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

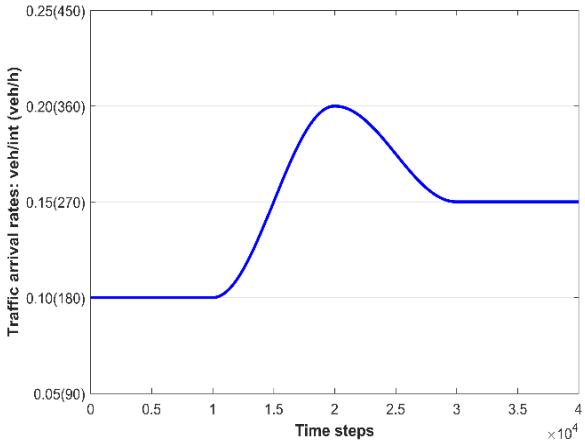


Fig. 3 Traffic Scenario C-flow profile on average arrival rates during the simulation.

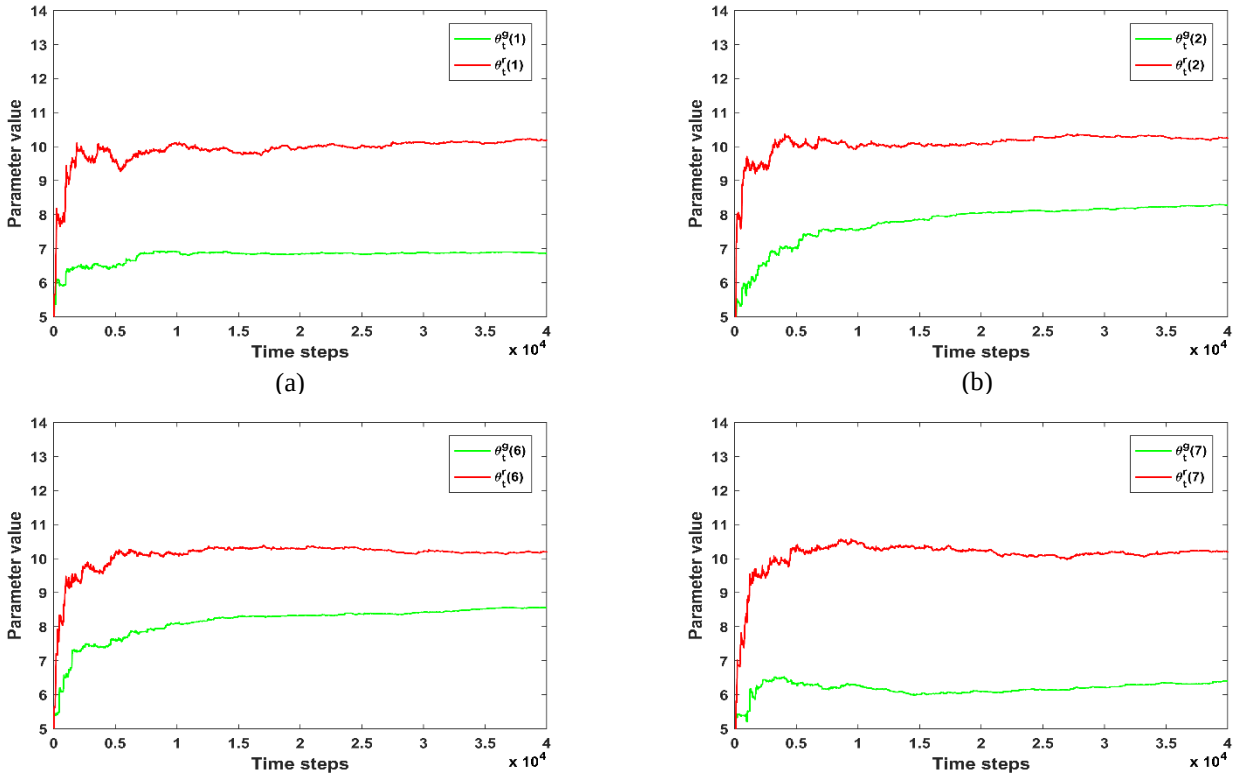


Fig. 4 Evolutions of functional parameters by using RLS-TD(λ) learning (in Traffic Scenario A, APS with λ=0, n=1, 2, 6, and 7).

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

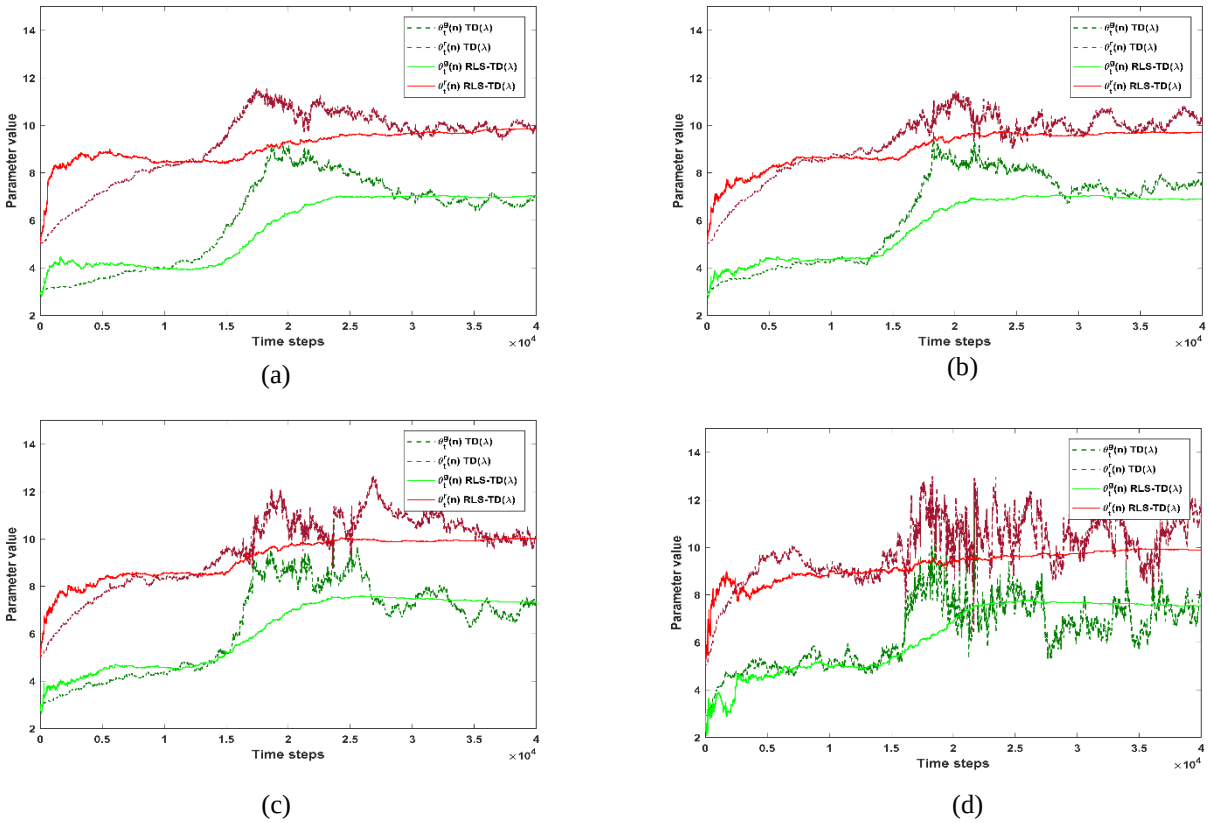


Fig. 5 Comparisons of functional parameters by using ADP approach between RLS-TD(λ) and TD(λ) learning (in Traffic Scenario C, APS with n=1): (a) λ=0, (b) λ=0.2, (c) λ=0.5, and (d) λ=1.

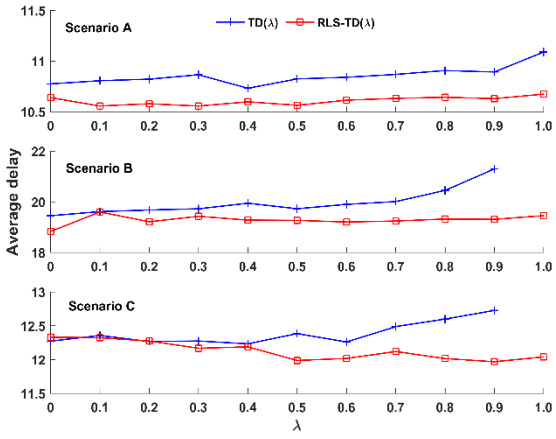


Fig. 6 Comparisons of average delays by ADP between RLS-TD(λ) and TD(λ) learning in APS.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

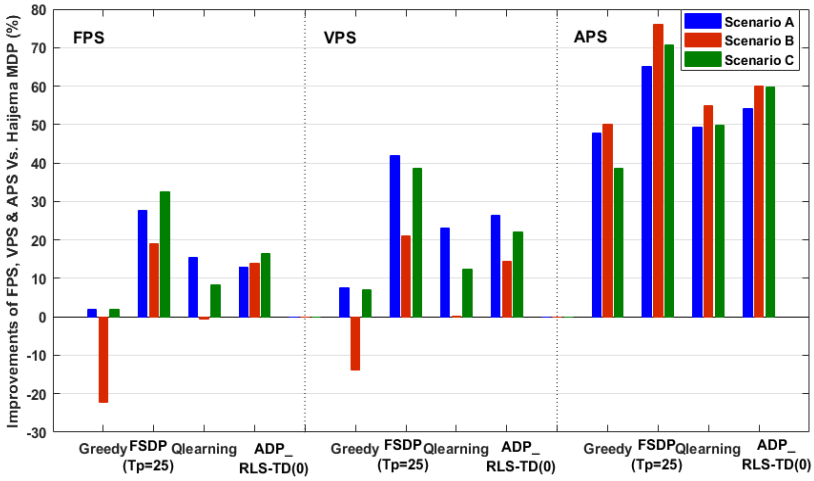


Fig. 7 Improvements of average delays by different methods in FPS, VPS, and APS, comparing with Haijema-MDP.

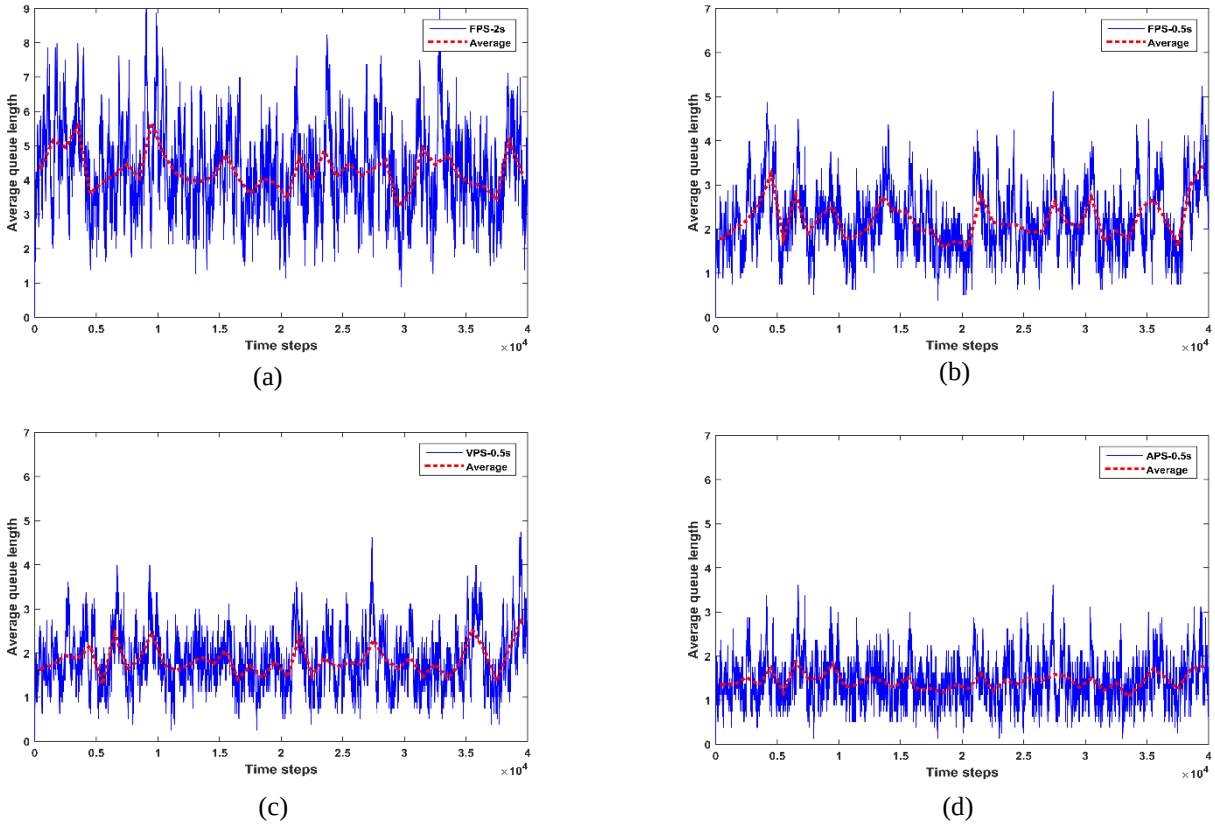


Fig. 8 Evolutions of average queue length by ADP with RLS-TD(λ) (in Traffic Scenario B, λ=0).

Conflict of Interest:

The authors declare that they have no conflict of interest.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

Reference

[1] Khan SG, Herrmann G, Lewis FL, Pipe T, Melhuish C (2012) Reinforcement learning and optimal adaptive control: An overview and implementation examples. Annu Rev Control, 36(1), 42–59.

[2] Sutton RS, Barto AG (1998) Reinforcement Learning: An Introduction. Cambridge MA: MIT Press.

[3] Xu X, Zuo L, Huang Z (2014) Reinforcement learning algorithms with function approximation: recent advances and applications. Inform Sciences, 261, 1–31.

[4] Powell WB (2007) Approximate Dynamic Programming: Solving the curses of dimensionality. John Wiley & Sons,USA.

[5] Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. IEEE Comput Intell M, 4(2), 39–47.

[6] Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, 15, 493–525.

[7] Cai C, Wong CK, Heydecker BG (2009) Adaptive traffic signal control using approximate dynamic programming. Transport Res C: Emer, 17(5), 456–474.

[8] Haijema R, van der Wal J (2008) An MDP decomposition approach for traffic control at isolated signalized intersections. Proba Eng Inform Sc, 22(4), 587–602.

[9] Yu XH, Recker WW (2006) Stochastic adaptive control model for traffic signal systems. Transport Res C: Emer, 14(4), 263–282.

[10] Baird L, Moore AW (1999) Gradient descent for general reinforcement learning. Adv Neur In, 968–974.

[11] Tsitsiklis JN, Van Roy B (1997) An analysis of temporal-difference learning with function approximation. IEEE T Automat Contr, 42(5), 674–690.

[12] Xu X, He H, Hu D (2002) Efficient reinforcement learning using recursive least-squares methods. J Artif Intell Res, 16(1), 259–292.

[13] Ormoneit D, Sen Ś (2002) Kernel-based reinforcement learning. Mach Learn, 49(2-3), 161–178.

[14] Bradtke SJ, Barto AG (1996) Linear least-squares algorithms for temporal difference learning. Mach Learn, 22(1-3), 33–57.

[15] Boyan JA (2002) Technical update: Least-squares temporal difference learning. Mach Learn, 49(2-3), 233–246.

[16] Hunt PB, Robertson DI, Bretherton RD, Winton RI (1981) SCOOT - a traffic responsive method of coordinating signals. Transport and Road Research Laboratory, Crowthorne, U.K., Technique Report.

[17] Lowrie PR (1982) The Sydney coordinated adaptive traffic system-principles, methodology, algorithms. In Proceddings of International Confonference on Road Traffic Signalling.

[18] Mladenovic MN, Stevanovic A, Kosonen I, Glavic D (2015) Adaptive Traffic Control Systems: Guidelines for Development of Functional Requirements. mobil.TUM. Munich, Germany.

[19] Gartner NH, Pooran FJ, Andrews CM (2001) Implementation of the OPAC adaptive control strategy in a traffic signal network. Proceedings of IEEE Conference Intelligent Transportation Systems, 195–200.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

[20] Henry J, Farges J, Tuffal J (1984) The PRODYN real time traffic algorithm. IFACIFIP-IFORS Conference on Control in Transportation System. http://trid.trb.org/view.aspx?id=339694

[21] Mirchandani P, Head L (2001) A real-time traffic signal control system: architecture, algorithms, and analysis. Transport Res C: Emer, 9(6), 415–432.

[22] Heung TH, Ho TK, Fung YF (2005) Coordinated road-junction traffic control by dynamic programming. IEEE T Intell Transp, 6(3), 341-350.

[23] Wu J, Abbas-Turki A, El Moudni A (2009) Discrete methods for urban intersection traffic controlling. In Proceedings of IEEE Vehicular Technology Conference, 1-5.

[24] Park B, Chang M (2002) Realizing benefits of adaptive signal control at an isolated intersection. Transport Res Rec, (1811), 115-121.

[25] Abdulhai B, Pringle R, Karakoulas GJ (2003) Reinforcement learning for true adaptive traffic signal control. J Transp Eng-ASCE, 129(3), 278-285.

[26] Lee J, Abdulhai B, Shalaby A, Chung EH (2005) Real-time optimization for adaptive traffic signal control using genetic algorithms. J Intell Transport S, 9(3), 111-122.

[27] Kergaye C, Stevanovic A, Martin PT (2010) Comparative evaluation of adaptive traffic control system assessments through field and microsimulation. J Intell Transport S, 14(2), 109-124.

[28] Li L, Lv Y, Wang FY (2016) Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 3(3), 247-254.

[29] Araghi S, Khosravi A, Creighton D (2015) A review on computational intelligence methods for controlling traffic signal timing. Expert Syst Appl, 42(3), 1538–1550.

[30] García-Nieto J, Alba E, Carolina Olivera A (2012) Swarm intelligence for traffic light scheduling: Application to real urban areas. Eng Appl Artif Intel, 25(2), 274–283.

[31] Srinivasan D, Choy MC, Cheu RL (2006) Neural networks for real-time traffic signal control. IEEE T Intell Transp, 7(3), 261–272.

[32] Arel I, Liu C, Urbanik T, Kohls AG (2010) Reinforcement learning-based multi-agent system for network traffic signal control. IET Intell Transp Sy, 4(2), 128–135.

[33] Bazzan ALC (2009) Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Auton Agent Multi-Ag, 18(3), 342–375.

[34] Box S, Waterson B (2013) An automated signalized junction controller that learns strategies by temporal difference reinforcement learning. Eng Appl Artif Intel, 26(1), 652–659.

[35] Prashanth LA, Bhatnagar S (2011) Reinforcement learning with function approximation for traffic signal control. IEEE T Intell Transp, 12(2), 412–421.

[36] El-Tantawy S, Abdulhai B, Abdelgawad H (2013) Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto. IEEE T Intell Transp, 14(3), 1140–1150.

[37] Li T, Zhao D, Yi J (2008) Adaptive dynamic programming for multi-intersections traffic signal intelligent control. In Proceedings of IEEE Conference Intelligent Transportation Systems, 286–291.

[38] Zhao D, Hu Z, Xia Z, Alippi C, Zhu Y, Wang D (2014) Full-range adaptive cruise control based on supervised adaptive dynamic programming. Neurocomputing, 125, 57-67.

This is the final version published in Neural Computing and Applications, 2017
DOI10.1007/s00521-017-3066-9

[39] Huang, YS, Weng YS, Zhou MC (2014). Modular design of urban traffic-light control systems based on synchronized timed Petri nets. IEEE T Intell Transp, 15(2), 530-539.

[40] El-Tantawy S, Abdulhai B, Abdelgawad H (2014) Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. J Intell Transport S, 18(3), 227–245.

[41] Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. IEEE T Syst Man Cy C, 38(2), 156–172.

[42] Bertsekas DP (1995) Dynamic programming and optimal control (Vol. 1, No. 2). MA: Athena Scientific.

[43] Gartner NH, Tarnoff PJ, Andrews CM (1991) Evaluation of optimized policies for adaptive control strategy. Transport Res Rec, 105–114.

[44] Yin B, Dridi M, El Moudni A (2015) Forward search algorithm based on dynamic programming for real-time adaptive traffic signal control. IET Intell Transp Sy, 9(7), 754-764.

[45] Khamis MA, Gomaa W (2012) Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In Proceedings of IEEE Conference Machine Learning and Applications, 586-591.

[46] Khamis, MA, Gomaa W(2014) Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. Eng Appl Artif Intel, 29, 134-151.

[47] Söderström T, Stoica P (2002) Instrumental variable methods for system identification. Circ Syst Signal Pr, 21(1), 1–9.