ADAPTIVE TRAFFIC SIGNAL CONTROL USING APPROXIMATE DYNAMIC PROGRAMMING

Mr. Chen Cai
Research Student
Centre for Transport Studies, University College London

Prof. Benjamin Heydecker
Head of Centre for Transport Studies
University College London

Abstract

This paper presents a concise summary of a study on adaptive traffic signal controller for real time operation. The adaptive controller is designed to achieve three operational objectives: first, the controller adopts a dual control principle to achieve a balanced influence between immediate cost and long-term cost in operation; second, controller switches signals without referring to a preset plan and is acyclic; third, controller adjusts its parameters online to adapt new environment. Not all of these features are available in existing operational controllers. Although dynamic programming (DP) is the only exact solution for achieving the operational objectives, it is usually impractical for real time operation because of demand in computation and information. To circumvent the difficulties, we use approximate dynamic programming (ADP) in conjunction with online learning techniques. This approach can substantially reduce computational burden by replacing the exact value function of DP with a continuous linear approximation function, which is then updated progressively by online learning techniques. Two online learning techniques, which are reinforcement learning and monotonicity approximation respectively, are investigated. We find in computer simulation that the ADP controller leads to substantial savings in vehicle delays in comparison with optimised fixed-time plans. The implications of this study to traffic control are: the ADP controller meet all of the three operational objectives with competitive results, and can be readily implemented for operations at both isolated intersection and traffic networks; the ADP algorithm is computationally efficient, and the ADP controller is an evolving system that requires minimum human intervention; the ADP technique offers a flexible theoretical framework in which a range of functional forms and learning techniques can be further studied.

1. Introduction

Operating traffic signals in urban area requires proper timings in response to traffic demand that varies during a day and across days of a week. Conventional control methods often rely on a library of fixed-time plans that are optimised offline according to certain pattern of traffic demand. A conforming plan is retrieved from the library to accommodate the pattern of demand of hours during a day and across days of a week. The plans are maintained manually; otherwise their performance may degrade at a rate of 3% a year. Studies in responsive control methods, notably SCOOT, OPAC, PRODYN, SCATS, and UPTOPIA, have shown significant advantages over the conventional methods in performance, with reduced human intervention. In this paper, we investigate an adaptive controller that aims for three operational objectives: first, the controller adopts a control principle that aims to achieve a balanced influence between immediate cost and long-term cost in operation; second, controller operates signals without referring to preset stage sequences or stage during; third, controller adjusts its parameters online to adapt new environment. Not all of these features are available in existing operational controllers.

To achieve the identified objectives, we demand a technique that optimises control process over time. Dynamic programming (DP), originally developed by Bellman (1957), is so far the only exact solution to problems as such. With all of its advantages, the DP problem cannot be solved analytically. The standard procedure to solve the DP problem is to recursively calculate *Bellman's*

equation backward, step by step. The key components of the Bellman's equation are the *one-step cost function*, which represents the immediate cost of making an action, and the *value function*, which represents the future cost of implementing a decision. In each iteration, the algorithm has to evaluate the cost of being every possible state of the system by using Bellman's equation. This procedure may make a problem computationally intractable, if the state space of the control problem is large. Bellman called this phenomenon as "the curse of dimensionality." Furthermore, because of the backward recursive calculation, one needs the *complete information* of the control system for the entire time period in concern. In real time operation, such a quest is usually impractical.

Regarding the difficulties, the DP was only used for analytical purpose in previous studies in adaptive traffic signal control, such as in Robertson and Bretherton (1978), OPAC and PRODYN. Specific heuristics are adopted for the actual implementation of adaptive controllers.

In this paper we present a novel approach in exploring the DP in real time traffic signal control. Central to this approach is to replace the true value function of the DP with a continuous approximation function. The approximation function aims to substantially reduce computational requirement, while preserving the fundamental properties of the true value function. The solution procedure is to step forward in time rather than step backward. The approximation function is updated at each step when a new state transition is observed. This approach is frequently denoted as *approximate dynamic programming* or *adaptive dynamic programming*. We use the acronym ADP in the rest of this paper.

The fundamentals of the ADP are discussed in Section 2. The system dynamics and formulations of ADP for traffic signal control are introduced in Section 3. We present numerical experiments in Section 4. The experiments include scenarios of both isolated intersection and traffic network.

2. Fundamentals of the ADP

Let $i \in X$ be the state variable of the system, and $u \in U$ the decision variable. Given the initial state i_0 and a sequence of decisions u_t at discrete time t, a dynamic programming algorithm is to solve

$$\min_{u_t \in U} \underset{w_t}{E}\left\{\sum_{t=0}^{m-1} \alpha^t g_t\left(i_t, i_{t+1}\right) \middle| i_0 = i\right\}. \tag{1}$$

The backward dynamic programming solution recursively computes the Bellman's equation

$$J\left(i_t\right) = \min_{u_t = U} \underset{w_t}{E}\left\{g_t\left(i_t, i_{t+1}\right) + \alpha J\left(i_{i+1} \middle| i_t\right)\right\}, \text{ for } t = m-1, m-2, ..., 0, \tag{2}$$

and implement at each time step t

$$u_t^* = \arg\min_{u_t = U} \underset{w_t}{E}\left\{g_t\left(i_t, i_{t+1}\right) + \alpha J\left(i_{i+1} \middle| i_t\right)\right\}, \tag{3}$$

where

g_t is the one-step cost function,

$\alpha \in (0,1]$ is the discount factor,

$J(i)$ is the value function representing the future cost of being in state i,

and the expectation operator is taken in respect to the probability in state transition from i_t to i_{t+1} influenced by random information w_t.

It is not difficult to show that we have to evaluate $J(i)$ for each $i \in X$ at time t so that the recursive calculation (2) may continue to the next step. This is the core problem of dimensionality, but is further complicated by the transition probability from i_t to i_{t+1} and the decision space U. In the ADP approach, we define a continuous approximation function $\tilde{J}(\cdot, r): X \times \mathbb{R}^K \to \mathbb{R}$ to replace the true $J(\cdot): X \to \mathbb{R}$, where parameter vector r of $\tilde{J}$ is K-dimensional. At each time step t, we calculate

$$\hat{J}(i_t) = \min_{u_t \in U} \underset{w_t}{E}\left\{ g(i_t, i_{t+1}) + \alpha \tilde{J}(i_{t+1}, r_t) \right\}, \text{ for } t = 0,1,...,m . \quad (4)$$

and implement at each time step *t*

$$u_t^* = \arg\min_{u_t \in U} \underset{w_t}{E}\left\{ g(i_t, i_{t+1}) + \alpha \tilde{J}(i_{t+1}, r_t) \right\} . \quad (5)$$

By using function $\tilde{J}(\cdot, r)$ we avoid a look-up table of $J(i)$ that contains the true value of being each state *i*. We calculate (4) only upon visiting the actual state *i*, thus substantially reducing computational requirement.

Also by using function $\tilde{J}(\cdot, r)$ we calculate (4) by stepping forward into time. Upon each state transition from i_t to i_{t+1} we obtain a new estimate of function parameter Δr_t, which is then used to update the current estimation of parameter vector *r* as

$$r_{t+1} = r_t + \eta_t \Delta r_t , \quad (6)$$

where η_t is the stepsize that satisfies $0 < \eta_t \leq 1$.

A general algorithm of the ADP is summarised in Fig.1.

Step 1.	Initialisation a) Initialise r_0. b) Choose an initial state i_0, $i \in X$. c) Set t = 1.
Step 2.	System receives random information w_t.
Step 3.	For t = 1,2,…, m-1, a) Calculate (4), (5) b) Calculate Δr_t , c) Update parametric vector r using (6), d) Implement u_t^* to transfer system to new state i_{t+1}.
Step 4.	If $t < m$ go to step 2.

Fig. 1 The general algorithm for the ADP

The remaining question here is how to obtain the estimation Δr_t online and update the functional parameter accordingly. Bertsekas and Tsitsiklis (1996) use artificial neural network to formulate approximation function and update the functional parameters using learning paradigms of neural networks, such as *supervised learning* or *reinforcement learning* (Sutton and Barto, 1998). In the context of online operation, where information of the system is not known a priori, reinforcement learning is a practical solution. One of the key technique of reinforcement learning is *temporal difference* (TD) *learning* (Sutton 1988), which constantly tracks the *difference* between estimation and the actual observation. Tsitsiklis and Van Roy (1997) prove that a linear approximation function trained by TD learning converges with a probability of 1, if a few outstanding assumptions are met. A different technique to update functional parameter is present by Papadaki and Powell (2002, 2003). They firstly identify the structural properties of the true value function,and then use perturbation learning to estimate the partial gradients of the linear approximation function. In this paper we investigate both TD learning and the perturbation learning for a linear approximation function.

Let d_t denote the temporal difference at time step *t,* we have

$$d_t = \hat{J}(i_t) - \tilde{J}(i_t, r_t), \quad (7)$$

where $\tilde{J}$ is calculated by (4). The functional parameter *r* is then updated by

$$r_{t+1} = r_t + \eta_t d_t \sum_{k=1}^{t} \lambda^{t-k} \nabla_r \tilde{J}\left(i_k, r_t\right), \tag{8}$$

where λ is an exponential weighting factor that satisfies $0 \le \lambda \le 1$. Stepsize factor η_t has to satisfy

$$\sum_{t=0}^{\infty} \eta_t = \infty \text{ and } \sum_{t=0}^{\infty} \eta_t^2 < \infty, \tag{9}$$

to ensure that r_t eventually converge to the unique r^* that satisfies

$$\left\| \tilde{J}\left(\cdot, r^*\right) - J \right\|_D \le \frac{1-\alpha\lambda}{1-\alpha} \left\| \prod J - J \right\|_D,$$

where $\prod J$ is the solution to the least-squares problem of

$$r^* = \arg\min \sum_{i \in X} \left(J(i) - \tilde{J}(i, r) \right)^2.$$

and D is the diagonal matrix with diagonal entries $\pi(i)$, $i = 1, \ldots, n$,

$$D = \begin{pmatrix} \pi(1) & 0 & \ldots & 0 \\ 0 & \pi(2) & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \ldots & \pi(n) \end{pmatrix}.$$

The entries of D denotes the steady-state probability for the process i_t, and $\pi(i) > 0, \forall i \in X$.

On the other hand, we may also seek to estimate Δr by numerically calculate partial gradient $\Delta r(k)$, $k = 1,2, K$. The partial estimation is obtained by perturbing the state i with an artificial increment $\Delta i(k)$, and calculate

$$\Delta r_t(k) = \frac{\hat{J}\left(i_t + \Delta i(k)\right) - \hat{J}\left(i_t\right)}{\left\| \Delta i(k) \right\|}. \tag{10}$$

We then smooth to obtain an updated estimate of the functional parameter

$$r_{t+1}(k) = (1-\eta_t) r_t(k) + \eta_t \Delta r_t(k), \text{ for } k = 1, 2, \ldots, K. \tag{11}$$

Papadaki and Powell (2002, 2003) use (10) and (11) to update the linear approximation function designated for a batch dispatch problem, which exhibits considerable similarities to traffic signal control problem.

3. Formulation of ADP for traffic signal control

The outstanding assumptions for the traffic signal control problem include:

1). Signal phases are composed of effective greens and effective reds only, and no amber interval is considered. Lost time is not consider either.

2). There are no constraints on the maximum duration of a green period. A signal switch is immediately followed by mandatory inter-green and minimum green.

3). Queue lengths are calculated at the end of each time interval, neglecting the detail of vehicle behaviour during the interval. Signals may only be switched at the boundary between intervals

4). The saturation flow on all lanes is 1 vehicles per 2 seconds. This is equivalent to 1440 vehicles per hour, a rate that is sufficiently close to the saturation flow of a single traffic lane.

5). Upstream roadside sensors provide information of arriving traffic of the next 10 seconds

The state *i* of a traffic intersection is primary determined by the queues remaining in each traffic link and the signal indication each link receives. We denote queue remaining in traffic intersection by column vector *l* and signal indications by column vector *s*. For an intersection of *N* links, vector *l* and *s* can be expressed as

$$l = \begin{bmatrix} l(1) \\ \vdots \\ l(N) \end{bmatrix}, \quad s = \begin{bmatrix} s(1) \\ \vdots \\ s(N) \end{bmatrix},$$

where $l(n)$ denotes the actual number of vehicles queuing in link *n*, and each element of *s* is a binary variable depending on traffic signal indication such that

$$s(n) = \begin{cases} 1 & \text{if signal is green for link } n \\ 0 & \text{if signal is red for link } n \end{cases}$$

The system state *i* therefore can be expressed as $i \{l, s\}$. To construct the approximation function, we employ a feature-extraction function $\phi(i)$ such that,

$$\phi(i) = \begin{bmatrix} \phi(i(1)) \\ \vdots \\ \phi(i(N)) \end{bmatrix}, \text{ where } \phi(i(n)) = \begin{cases} \begin{bmatrix} l(n) \\ 0 \end{bmatrix} & \text{if } s(n) = 1 \\ \begin{bmatrix} 0 \\ l(n) \end{bmatrix} & \text{if } s(n) = 0. \end{cases}$$

The linear approximation function is formed by

$$\tilde{J}(i, r) = \sum_{n=1}^{N} \phi_n(i) r(n), \tag{12}$$

where

$$r(n) = \begin{bmatrix} r^-(n) \\ r^+(n) \end{bmatrix}. \tag{13}$$

In such a way, we differentiate the signal status, and assign r^- to queue length variable $l(n)$ if link *n* receives green signal, or r^+ if otherwise.

We further denote random arriving traffic by column vector *w*, where

$$w = \begin{bmatrix} w(1) \\ \vdots \\ w(N) \end{bmatrix}.$$

Let column vector *y* denote the departing traffic from the *N*-link intersection, it follows

$$y = \begin{bmatrix} y(1) \\ \vdots \\ y(n) \end{bmatrix}.$$

Finally, the transition of system state during time increment from *t* to *t*+1 can be then described as

$$l_{t+1}(n) = l_t(n) - y_t(n) + w_t(n), \tag{14}$$

and signal vector *s* is transferred

$$s_{t+1}(n) = \left(s_t(n) + u_t(n)\right) \bmod_2, \tag{15}$$

where decision variable u_t takes

$$u_t(n) = \begin{cases} 1 & \text{for signal switch} \\ 0 & \text{unchanged.} \end{cases}$$

Equation (14) and (15) describe the state transitions of a single step. The number steps in a certain time period depends the resolution of the discrete time system. Let Δt denote the time increment of discrete time step, we assume that there is a total number of M steps in the planning period of the signal controller, and consequently the actual duration of the planning period is $M\Delta t$ seconds. Since we assume that the upstream detector provides 10-second data of future traffic, the planning period $M\Delta t$ is normally between 10 and 20 seconds. Traffic data for the period beyond 10-second can be predicted by using Monte Carlo simulation. For a M-step planning period, the ADP controller aims to find

$$\mathbf{u}_t^* = \arg\min_{w_k} E\left[\sum_{k=t}^{t+M-1} \alpha^{k-t} g\left(i_k, i_{k+1}\right) + \alpha^M \tilde{J}_{t-1}\left(i_{t+M-1}, r_{t-1}\right)\right],\ i \in X \tag{16}$$

and calculate

$$\hat{J}_M\left(i_t\right) = \min_{\mathbf{u}} E\left[\sum_{k=t}^{t+M-1} \alpha^{k-t} g\left(i_k, i_{k+1}\right) + \alpha^M \tilde{J}_{t-1}\left(i_{t+M-1}, r_{t-1}\right)\right],\ i \in X\ , \tag{17}$$

where the one-step cost function g is given by

$$g\left(i_t, i_{t+1}\right) = \sum_{n=1}^{N}\left[l_t(n) - y_t(n) + w_t(n)\right]\Delta t\ . \tag{18}$$

The M-step temporal difference can be expressed as

$$\begin{aligned} d_M &= \hat{J}_M\left(i_t\right) - \tilde{J}\left(i_t, r_t\right) \\ &= \sum_{k=t}^{t+M-1} \alpha^{k-t} d_k\left(i_k, i_{k+1}\right), \end{aligned} \tag{19}$$

and the functional parameter is updated by

$$r_{t+1} = r_t + \eta_t \phi\left(i_t\right) \sum_{k=t}^{t+M-1} \alpha^{k-t} d_k\left(i_k, i_{k+1}\right),\ t = 0,1,.... \tag{20}$$

Equation (20) can be regarded as a special variant of (8). With a large M, Eq. (20) comes closer to (8) with $\lambda = 1$, and a smaller M makes (20) closer to (8) with $\lambda = 0$.

For the perturbation technique presented by (10) and (11), in M-step planning, we simply use (17) instead of one-step equation (4) to calculate $\hat{J}_M$.

The traffic signal control algorithm using ADP can be summarised as the following:

Step 0: Initialisation

0.1 Choose an initial system state i_0;

0.2 Initialise functional parameter vector r_0;

0.3 Initiate learning rate (or stepsize) η_0;

0.4 Set time index $t = 0$.

Step 1: Receiving new information

1.1 Set time index $t = t + 1$;

1.2 Receive detected information w_t ;

1.3 Predict the information vector w_t' for the extra part of the planning period, if necessary.

Step 2: Evaluate control decisions

2.1 If signal change is not admissible, set $\mathbf{u}^*_t = \mathbf{0}$;

2.2 If signal change is admissible, for the planning period of M-steps, find the optimal decision $\mathbf{u}^*_t$ using (14).

Step 3: Update approximation function

a) TD option:

3.a.1 Calculate new observation $\hat{J}_M\left(i_t\right)$ using (17)

3.a.2 Calculate current approximation $\tilde{J}_{t-1}\left(i_t, r_{t-1}\right)$ using (12);

3.a.3 Calculate *M*-step temporal difference using (19)

3.a.4 Update functional parameter vector r_{t-1} using (20).

b) Perturbation option:

3.b.1 Numerically calculate partial gradient for $n = 1, 2, \ldots, N$ by perturbing queue $l_t(n)$ of state i_t by $\Delta l(n)$,

if $s(n) = 0$ (green signal in link n), using (10) and (17) to obtain $\Delta r_t^-\left(n\right)$,

if $s(n) = 1$ (red signal in link n), using (10) and (17) to obtain $\Delta r_t^+\left(n\right)$,

3.b.2 Update functional parameter vector r_{t-1} accordingly by using (11) for $n = 1, 2, \ldots, N$.

Step 4: Implement optimal decision $\mathbf{u}_t^*$ for the first Δt of the planning period

4.1 Transfer signal status using (15);

4.2 Transfer queue status using (14);

4.3 Complete the state transition from i_t to i_{t+1}.

Step 5: Stopping Criteria

5.1 If $t < T$, then goes back to Step 1; Otherwise, stop.

In the next section, we discuss the application of the ADP control algorithm in numerical experiments.

4. Numerical experiments

Cai (2007) investigates the ADP signal controller for isolated intersection with two stages. The results from numerical experiments show that the ADP control algorithm using linear approximation function and perturbation training is as good as Robertson and Bretherton's heuristic using quadratic approximation function. Heydecker *et al.* (2007) extended the investigation to an isolated intersection with multiple stages. The study shows that the ADP controller, which produces acyclic signal timings, can reduce 48% vehicle delays from the best fixed-time plans. Both of the studies above use perturbation to update the linear approximation function online, and a resolution of 5-second per time step is adopted.

In this study, we first investigate the performance of ADP controller at a three-stage intersection, using a resolution of 0.5-second per time step. The ADP controller can be trained both by TD learning and perturbation. The results will be compared with the optimised fixed-time plans from TRANSYT 12.0, and with the ADP controller at the resolution of 5-second per time step.

We will further extend the investigation to a traffic network, where the ability of the ADP controller for distributive network control will be tested. The traffic network uses *cell transmission model* (CTM) to describe the traffic dynamics of the network. The performance of the ADP controller will be compared with optimised fixed-time plans and offset from TRANSTY. Only TD learning is used in this case.

The geometric layout and signal stage composition of the three-stage traffic intersection are shown in Fig.2. Traffic demand of the intersection is 432 v/h in link A and C, and 252 v/h in link B. The saturation flow rate at the stop line of each traffic link is 1440 v/h.

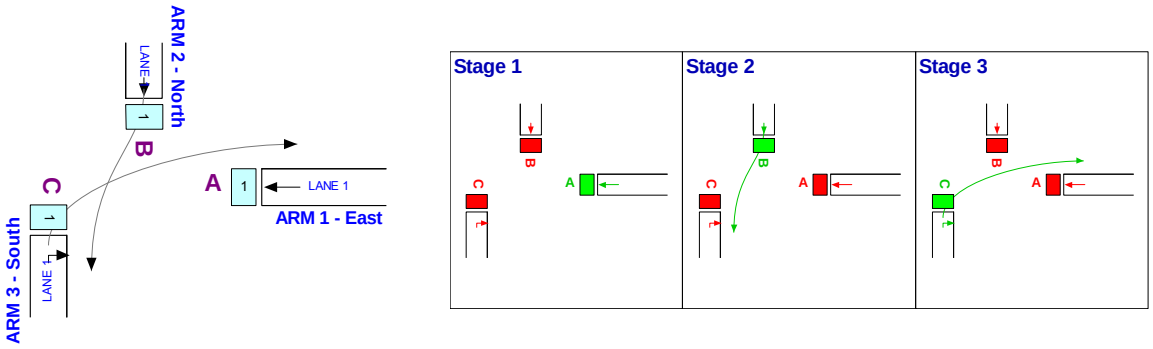

Fig. 2 Geometric layout and stage composition of the three-stage isolated traffic intersection

The control policy for the ADP controller is to evaluate whether to change signal indication at each time step. The option of "change" must reward better performance than "not change" and "change at any time later in the planning period." The planning period is 20-second long, with the first 10 seconds supplied with detected information and the second 10 seconds with predicted information. Only the first Δt seconds of the plan is implemented, and the system rolls forward to the next step.

At the three-stage intersection, we obtain 10 results from independent run of test for each control method. The control methods include ADP trained with TD learning, ADP trained with perturbation learning, the TRANSTY plans, and the ADP controller trained with perturbation learning but at resolution of 5-second per time step. The results are summarised in Table 1. Under the resolution of 0.5s, the ADP controllers reduce about 67% vehicle delays from the optimised fixed-time plans produced by TRANSYT. It is also worth noticing that the same ADP controller can reduce about 41% delays by operating at 0.5s instead of 5.0s per time step. The higher frequency of revising signal plans proves rewarding.

Table 1 Performance results (vehicle seconds per second) of ADP controller and optimised fixed-time plans from tests at the three-stage isolated traffic intersection

	ADP_TD	ADP_Perturbation	TRANSYT plans	ADP_Perturbation
	0.5s	0.5s	0.5s	5.0s
1	4.38	4.36	15.03	7.51
2	4.69	4.67	13.67	9.10
3	5.03	5.09	13.78	8.62
4	4.27	4.34	12.08	7.40
5	4.63	4.74	13.71	7.81
6	5.15	5.20	14.08	8.83
7	4.05	4.02	13.20	6.68
8	4.45	4.35	14.13	7.32
9	5.11	5.19	15.06	8.36
10	4.46	4.61	14.74	7.83
Mean	4.62	4.66	13.95	7.94
SD	0.37	0.40	0.90	0.76

Table 2 The optimised fixed-time plans from TRANSYT 12.0 for the three-stage traffic intersection

Number of Stages	Starting time			Cycle time
	Stage 1	Stage 2	Stage 3	
3	55	101	9	120 seconds

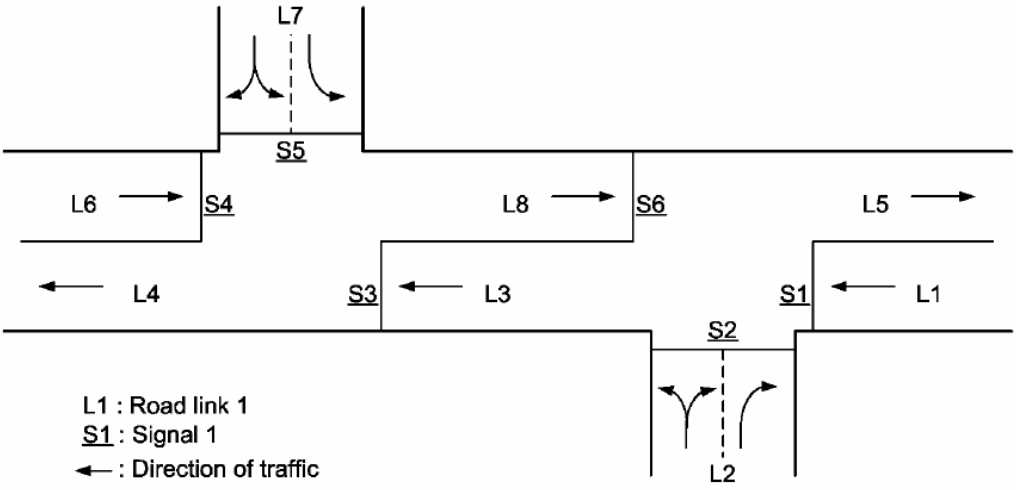

Fig. 3 The geometric layout of the example traffic network

The geometric layout of the traffic network is shown in Fig.3. The example network was originally designed by Wong *et al*. (2007) and uses the CTM to model traffic dynamics. The cells are homogenous in the network, with a capacity of 4 vehicles, except for the first cells of input links and the last cells of exit links, where the capacity is infinite to serve as the reservoir of traffic. The modelling details of the traffic links in the network is summarised in Table 3. There two intersections in the network, each having two stages and being governed by an independent ADP controller. The ADP controller uses the same control policy as for the isolated intersection presented earlier. Only TD learning is used to train the linear approximation function in this case. The resolution is 2-second per time increment, which is conforming to the resolution of the CTM in use. The planning period is 16-second long.

The traffic input and downstream distribution are summarised in Table 4. The saturation flow rate at the stopline of each link is 1440 v/h. An uneven flow pattern is seen in the network, with majority of traffic going from west to east, and a short link L8 in the middle. The coordination between signals S6 of intersection A and S4 of B are critical to the performance.

Table 3 Modelling details of the traffic links in the example network

Intersection	Link	Total no. of Cells	Signal	Signal stage	Signal in Cell no.	Remarks
A	L1	5	S1	A1	5	Input link
	L2	5	S2	A2	5	Input link
	L5	3	-	-	-	Exit link
	L8	1	S6	A1	1	Short link
B	L3	1	S3	B1	1	Short link
	L4	3	-	-	-	Exit link
	L6	5	S4	B1	5	Input link
	L7	5	S5	B2	5	Input link

Table 4 Traffic input and downstream distribution in the example network

Link	L1	L2		L6	L7	
Flow rate	350 (v/h)	382 (v/h)		440 (v/h)	382 (v/h)	
Downstream	L3	L3	L5	L8	L4	L8
Turning ratio	100%	25%	75%	100%	25%	75%

The ADP controller's performance indices obtained in a single run of test are shown in Table 5. The ADP controllers located in the two intersections maintain queues low in all input links, as indicated by the mean queue length of the concerning links. The temporal high demand from links L6 and L7 are quickly facilitated by the coordination between the two intersections, as indicated by the maximum queue length and the mode queue length of the respective links. A sample of coordination is visualised in Fig.4. Controller at intersection A trends to use the holding capacity of L8 to accommodate the first few arrivals from upstream until fully occupied (e.g. time 4551 to 4561), and then switch S6 to green to dissipate queues at saturation flow rate (1 v/2s) until flow rate drops (e.g. time 4562 to 4578). This means that controller at A usually maximises flow until upstream queues are cleared, at which point the incoming flow rate to L8 converges to arriving rate. In the mean time, using the holding capacity of L8 to accommodate the first few arriving vehicles gives the controlller at A opportunites to clear local queues in L2 (time 4551 to 4558, and 4581 to 4592). Controller at intersection B coordinates flows from L1 and L2 in a similar manner, despite the lower demand from east to west. Overall, the average vehicle delays over 10 tests runs is 9.00 vehicle seconds per second, with a standard deviation of 0.38.

The corresponding results of the TRANSYT plan in a single test run are shown in Table 6, and details of the signal plans in Table 7. It is clear that the signal plans optimised by TRANSYT does not fit well in the CTM traffic model, which is under the influence of stochastic traffic from the input links, whereas the TRANSTY model assumes cyclic traffic flow profiles. The TRANSYT plans leave substantial queues in links L2 and L7, whiles the mean queue lengths in L1 and L6 are still greater than those obtained with the ADP controllers.

The comparison suggests that the ADP controllers can be adopted for distributive control in network, and are far more advantageous in a stochastic traffic environment.

Table 5 Performance result of a single test run of 4-hour simulated time using ADP_TD controllers for distributive network control

	Intersection A			Intersection B		
Link	L1	L2	L8	L3	L6	L7
Mean queue	0.78	1.76	1.93	0.84	1.35	2.32
SE	0.01	0.02	0.02	0.01	0.02	0.03
Median	0	1	2	0.75	1	2
Mode	0	1	3	0	0	0
Maximum	7	11	4	4	11	12

Table 6 The performance results of a single test run of 4-hour simulated time using optimised fixed-time plans from TRANSYT 12.0 for network control

	Intersection A			Intersection B		
Link	L1	L2	L8	L3	L6	L7
Mean queue	0.86	70.14	2.26	0.81	2.05	34.36
SE	0.01	0.42	0.02	0.01	0.02	0.27
Median	0	67	3	0.25	1.25	27.67
Mode	0	66	4	0	0	21
Maximum	7	132	4	4	11	81.67

Table 7 The optimised TRANSTY 12.0 plans for network control

Intersections	Number of Stages	Stage 1	Stage 2	Stage 3	Stage 4	Cycle time
A	4	119 (A1)	49 (A2)	73 (A1)	107 (A2)	120 seconds
B	4	2 (B1)	35 (B2)	67 (B1)	97 (B2)	120 seconds

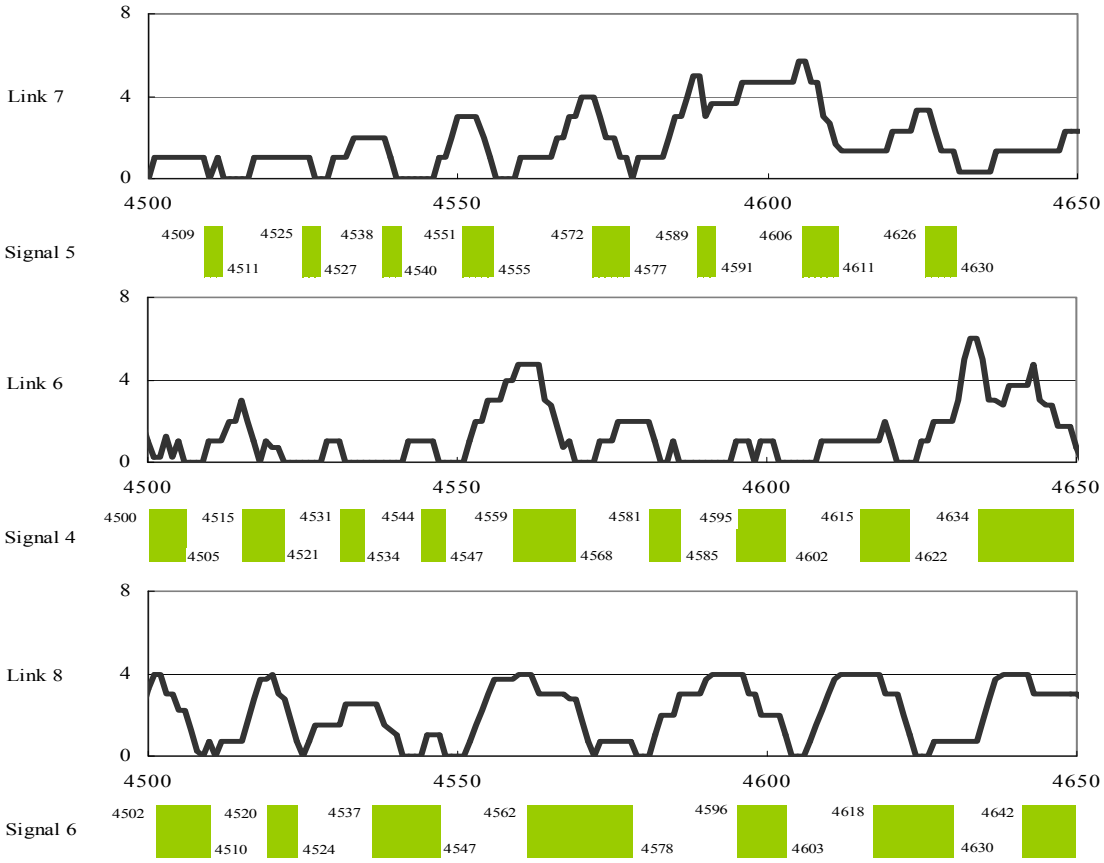

Fig. 4 Signal coordination between intersection A and B;
Links L7 and L6, signals S5 and S4, belong to intersection B, L8 and S6 to A.

The evolutions of functional parameters r for each link of the traffic network are shown in Fig. 5. The functional parameters are updated by using online TD learning. Because that we use a constant rate $\eta_t = 0.001$, the values of r do not converge. For links of intersection A, i.e. L1, L2 and L8, the parameters come close to certain values since the transient-state of the simulation is passed. On the other hand, for links of intersection B, i.e. L3, L6 and L7, the parameters show greater oscillations. In general, the evolution of parameters reflects the increasing monotonicity of the true value function. The controller also penalises queue in red signal, as r^+ is constantly greater than r^- in all cases.

5. Conclusion

This study develops a novel adaptive traffic signal controller by using approximate dynamic programming (ADP). The ADP approach preserves the fundaments of the original dynamic programming (DP) problem, and exhibits significant implication to real time signal operation. The key feature of the ADP approach is to replace the true value function of the DP with an approximation function. The approximation function may start with an arbitrary initialisation. Machine learning techniques for online operation are then employed to update the approximation function progressively. In this study, we use a linear approximation function, which can be trained either by temporal difference (TD) learning or perturbation learning. Numerical experiments have shown that the ADP controller reduces substantial vehicle delays from the best fixed-time plans both at a multi-stage isolated intersection and in a traffic network. A comparison with the optimised signal plans from TRANSYT in the network operation also reveals that the ADP controller is far more advantageous in managing stochastic traffic arrivals than the fixed-time plans. Due to the limited scope of this study, we are unable to establish a direct comparison with any existing adaptive signal controller. A field test is strongly recommended to evaluate the merit of the ADP controllers, and their actual implication to practice.

On the approximation function, the formulation is not limited to linear functions only. By using a general approximation tool, such as the multi-layer perceptron (MLP), the approximation function may take a range of non-linear forms. There is also a rich quantity of learning techniques that can be explored to train the function parameters online.

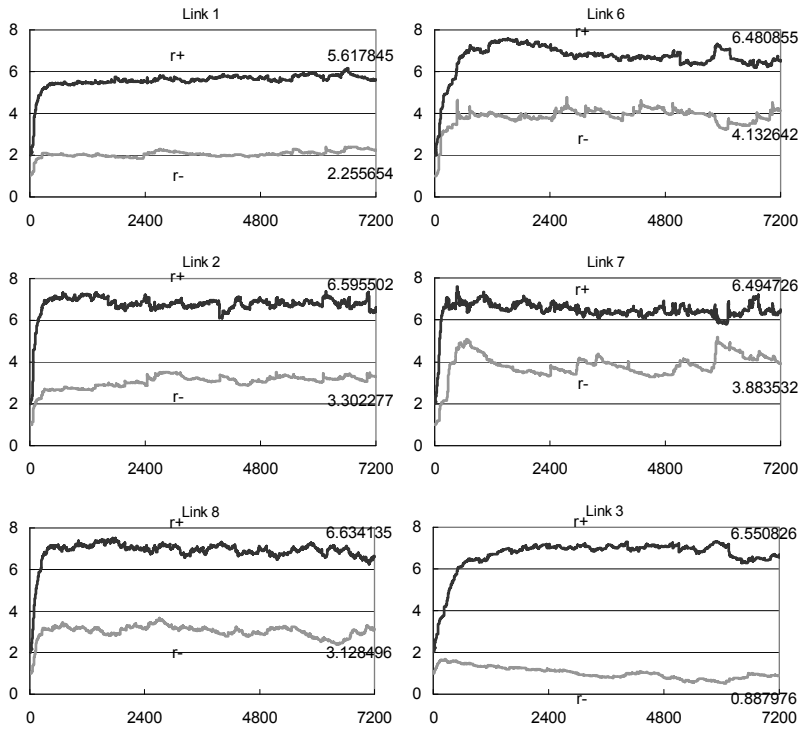

Fig. 5 Evolutions of functional parameters in traffic network control

Acknowledgement

The first author would like to thank the Rees Jeffreys Road Fund for financial support for the work presented here, and Dr. Wong Chi Kwong from Hong Kong City University for the contribution on the traffic network design and modelling.

References

Bellman, R., 1957. Dynamic programming, Princeton: Princeton University Press.

Betsekas, D.P., Tsitsiklis, J.N., 1995. Neuro-Dynamic programming, Belmont, MA: Athenas Scientific.

Cai, C., 2007. An approximate dynamic programming strategy for responsive traffic signal control, *Proceedings of 2007 IEEE international Symposium on Approximate Dynamic Programming and Reinforcement Learning,* Hawaii, U.S., pp.303-310.

Heydecker, B.G., Cai, C., Wong, C.K., 2007. Adaptive dynamic control for road traffic signals, *Proceedings of 2007 IEEE International Conference on Networking, Sensing and Control*, London, United Kingdom, pp.193-198.

Papadaki, K., Powell, W. B., 2002. A monotone adaptive dynamic programming algorithm for a stochastic batch service problem, *European Journal of Operational Research*, **142**(1), pp.108-127.

Papadaki, K., Powell, W. B., 2003. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem, *Naval Research Logistics*, **50**(7), pp. 742-769.

Robertson, D.I., Bertherton, R.D., 1974. Optimum control of an intersection for any known sequence of vehicular arrivals, Proceedings of the 2nd IFAC-IFIP-IFORS Symposium on Traffic Control and Transportation system, Monte Carlo.

Sutton, R.S., 1988. Learning to predict by the methods of temporal differences, Machine Learning, **3**, pp. 9-44.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press. Tsitsiklis, J.N., Van Roy, B., 1997. An analysis of temporal difference learning with function approximation, *IEEE Transactions on Automatic Control*, **42** (5), pp. 674–690.

Wong, C.K., Wong, S.C., Lo, H.K., 2007. Reserve capacity of a signal-controlled network considering the effect of physical queuing. In: Allsop, R.E., Bell, M.G.H., Heydecker, B.G. (Eds.), Transportation and Traffic Theory 2007, Elsevier Ltd, pp. 533-553.