Published in IET Intelligent Transport Systems
Received on 14th July 2009
Revised on 3rd February 2010
doi: 10.1049/iet-its.2009.0070

ISSN 1751-956X

Reinforcement learning-based multi-agent system for network traffic signal control

I. Arel[1] *C. Liu*[1] *T. Urbanik*[2] *A.G. Kohls*[2]

[1]*Department of Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, TN 37996-2100, USA*
[2]*Department of Civil and Environmental Engineering, The University of Tennessee, Knoxville, TN 37996-2100, USA*
E-mail: itamar@ieee.org

Abstract: A challenging application of artificial intelligence systems involves the scheduling of traffic signals in multi-intersection vehicular networks. This paper introduces a novel use of a multi-agent system and reinforcement learning (RL) framework to obtain an efficient traffic signal control policy. The latter is aimed at minimising the average delay, congestion and likelihood of intersection cross-blocking. A five-intersection traffic network has been studied in which each intersection is governed by an autonomous intelligent agent. Two types of agents, a central agent and an outbound agent, were employed. The outbound agents schedule traffic signals by following the longest-queue-first (LQF) algorithm, which has been proved to guarantee stability and fairness, and collaborate with the central agent by providing it local traffic statistics. The central agent learns a value function driven by its local and neighbours' traffic conditions. The novel methodology proposed here utilises the Q-Learning algorithm with a feedforward neural network for value function approximation. Experimental results clearly demonstrate the advantages of multi-agent RL-based control over LQF governed isolated single-intersection control, thus paving the way for efficient distributed traffic signal control in complex settings.

1 Introduction

A critical traffic engineering challenge is the scheduling and management of multi-intersection networks. Conventional deterministic traffic management systems fail to scale with respect to scheduling large networks of signals in urban settings, largely due to the lack of a long-term reward policy. In a heavy-loaded multi-intersection traffic network, congestion occurring in a single lane does not only impact upstream traffic, but also the other intersections. Thus, an efficient scheduling method that maximises a long-term reward and that can control a dynamic and complex traffic environment is highly desired.

In this paper, we present a solution to this problem by employing reinforcement learning (RL) – a machine learning framework which attempts to approximate an optimal decision-making policy. RL has been widely used as a practical computational tool to learn an optimal control policy [1]. Its applications include robotics, industrial manufacturing, and real-world traffic network management. We utilise a multi-agent setting, whereby RL is employed as means of controlling the different intersections in the network. Q Learning, a popular RL algorithm [2], is used to provide a control of the traffic signal scheduling.

In our study of a five-intersection, centrally connected traffic network, we have assigned an autonomous agent to each intersection and enabled the cooperation among agents, facilitated via the exchange of basic information. While outbound intersection agents are governed by the longest-queue-first (LQF) algorithm (Section 3.3), the critical intersection, the central one, is assigned a more advanced agent which can incorporate traffic statistics of its neighbours as part of its decision-making process. Subjected to the large dimension of the state space, a function approximation method is applied to store the value function – a fundamental construct in RL used to derive actions.

In summary, the proposed algorithm was developed considering the traffic schedule in an artificial intelligent

learning way, rather than the conventional traffic scheduling. The coordination factor among the intersection is an obscure object and has a nonlinear relation. The AI intelligent algorithm can compute the inner nonlinear relation which cannot be provided by the traditional approach. Simulation results clearly indicate that the proposed RL control scheme outperforms the LQF algorithm strategy by yielding a lower delay and cross-blocking frequency, particularly for medium and high traffic arrival rates.

The rest of the paper is structured as follows. Section 2 presents the basic principles of RL. In Section 3, a summary of work in traffic signal scheduling is provided. Section 4 describes the multi-intersection RL system model, including definitions pertaining to the state, action and reward function. Section 5 extends the algorithm by introducing a neural network-based function approximation module. Section 6 presents and discusses the simulation results, and in Section 7, the conclusions are drawn.

2 Basic principles of RL

RL is a field of study in machine learning where an agent, by interacting with and receiving feedback from its environment, attempts to learn an optimal action selection policy. RL algorithms typically learn and progress in an iterative manner. During each iteration, the agent observes its current environment, from which it infers the environment's state, then executes an action that leads the agent to the subsequent state. Next, the agent evaluates this action by the reward or penalty it is incurred and updates a value function, accordingly. The value function is the utility construct that it attempts to maximise (or minimise). A commonly used RL algorithm is Q_Learning [3]. The latter is a model-free RL algorithm, which assumes that the agent has no explicit knowledge of its environment's behaviour prior to interacting with it. Interaction with the environment is what offers the agent knowledge regarding both state transitions (as a function of actions taken) as well as their related long-term reward prospect. The goal of the agent is to maximise such long-term reward, by learning a good policy which is a mapping from perceived states to actions. In summary, RL methods provide a way to solve complex, real-world control problems, and one such challenge is traffic signal scheduling in multi-intersection settings.

3 Work in traffic signal scheduling

3.1 Adaptive signal control systems

In the field of adaptive signal control systems, well-known systems include SCOOT [4] and SCATS [5]. SCOOT is a centralised system that continuously measures traffic volumes and occupancies that serve to adjust signal timings with the primary objective of minimising the sum of the average queue in a specific area. A heuristic optimisation evaluates potential timing plans adjusting the signal timings. In larger networks, the system defines smaller regions for modelling and optimisation purposes. It uses in real time the same traffic flow modelling scheme used in TRANSYT (a steady-state model using platoon-dispersion equations). SCATS does not require modelling. It is an automated, real-time, traffic responsive signal control strategy using local controllers and regional computers for tactical and strategic control. Constant monitoring of traffic allows the system to choose the appropriate signal timings from a library, minimising vehicle stops with light demand, minimising delay with normal demand and maximising throughput with heavy demand. Systems such as SCOOT and SCATS suffer from inefficient handling of saturated conditions due to inadequate real-time adaptability [6].

Other approaches such as OPAC [7–9] and RHODES [7–9] calculate switching times by solving dynamic optimisation problems in a real-time manner. Both approaches do not consider explicitly cycle, split and offsets. To obtain the optimal switching times, a dynamic optimisation problem is solved in real-time employing dynamic traffic models and traffic measurements. The typical performance index to be minimised is the total intersection delay. Such systems suffer exponential complexities that diminish their chances of being deployed on a large scale [9].

Another real-time control strategy is TUC [10]. Based on a store-and-forward modelling of the urban network traffic and using the linear-quadratic regulatory theory, the design of TUC leads to a multivariate regulator for traffic-responsive coordinated network-wide signal control that is particularly suitable also for saturated traffic conditions. Real-time decisions in TUC cannot be taken more frequently than at the maximum employed signal cycle. The strategy will need to be redesigned in the case of modifications and expansions of the controlled network. TUC was compared with a fixed-time signal control producing reduction in total waiting time and total travel time in the system.

3.2 Multi-agent systems

On the basis of TUC system, but aiming to cope with large networks and to allow distributed reconfiguration, de Oliveira and Camponogara [11] proposes a framework for a network of distributed agents to control linear dynamic systems which are put together by interconnecting linear subsystems with local input constraints. The framework decomposes the optimisation problem arising from the model predictive control (MPC) approach into a network of coupled, but small subproblems to be solved by the agent network. Each agent senses and controls the variables of its subsystems, while communicating with agents in the vicinity to obtain neighbourhood variables and coordinate their actions. A problem decomposition and coordination protocol ensures convergence of the agents' iterates to a global optimum of

the MPC problem. The proposed approach achieved performance comparable to the TUC system.

A distributed and interactive network of agents to manage real-time traffic control was proposed in [12]. Each agent in the cooperative ensemble is able to dynamically determine the size of its cooperative zone. Therefore, a stochastic cooperative parameter update algorithm was designed improving the online learning and update process for the agent.

In [7], a collaborative RL (CRL) system using a local adaptive round robin phase switching model was employed at each signalised junction of a network. Each signalised junction collaborates with neighbouring agents in order to learn appropriate phase timing based on traffic patterns. Traffic patterns were of a steady and uniform nature. The approach was compared with a non-adaptive fixed-time system and to a saturation balancing algorithm producing reduction in average waiting time per arrived vehicle.

In [13], multi-agent RL was applied to schedule traffic signals at six intersections by constructing a vehicle-based model. The RL systems learn value functions estimating expected waiting times for cars given different settings of traffic lights. Selected settings of traffic lights results from combining the predicted waiting times of all cars involved. Results show that the proposed algorithm can outperform non-adaptable traffic light controllers.

3.3 Additional work in traffic signal scheduling

It is argued that the use of a model-based RL approach adds unnecessary complexities compared with using model-free Q_Learning RL. In [14], the model-free Q_Learning RL-based method was applied to derive an optimal and adaptive traffic control policy for an isolated, two-phase traffic signal. Performance was compared with that of a commonly used pre-timed signal controller, resulting in significantly lower delays with variable traffic flows.

In [15], RL and approximate dynamic programming was used to develop a self-sufficient adaptive traffic signal controller that substantially reduced vehicle delays when compared with fixed time control in an isolated intersection. Two learning techniques, temporal difference (TD) RL and perturbation learning were explored. The TD method constantly tracks the difference between current estimation and actual observation of state values and propagates the difference back to the functional parameter so as to update the approximation. The perturbation learning method directly estimates the gradients of the approximate function by giving a perturbing signal to the system state.

The problem of finding the optimal traffic signal timing plans has been solved as a decision-making problem for a controlled Markov process in [16]. The Markovian model developed as the system model for signal control incorporates Robertson's platoon dispersion traffic model between intersections and employs the value iteration algorithm to find the optimal decision for the controlled Markov process.

Three self-organising methods for traffic light control were proposed in [17]. The author defended schemes that are distributed and non-cyclic. The methods presented distinguished themselves because no direct communication between traffic lights was utilised, only local rules.

Finally, in previous work, the authors proposed an LQF traffic signal scheduling algorithm [18] for an isolated intersection. The LQF algorithm was designed for a signal control problem employing concepts drawn from the field of packet switching in computer networks. The novel method proposed utilised a maximal weight matching algorithm to minimise the queue sizes at each approach, yielding significantly lower average vehicle delay through the intersection. LQF has been proved to be stable and shown to yield strong performance attributes under various traffic scenarios. However, to schedule a multi-intersection network, where a phase scheduling decision at one intersection would largely impact the traffic conditions in its neighbouring intersections, is a more complex task. For such settings, LQF, as well as many other existing schemes, is inherently limited in the sense that it is unable to take into consideration neighbouring intersections conditions. It will be demonstrated that RL offers the capability to provide distributed control as needed for scheduling multiple intersections. At its core, the RL algorithm learns a nonlinear mapping between intersection elements, from which it can derive a high-performance policy for scheduling traffic signals at a network of intersections.

4 System model

4.1 Notation and terminology

We begin with defining the terms and notation used throughout the rest of the paper. Traffic throughput in our study is defined as the average number of vehicles per unit of time that successfully traverse the intersection. Traffic congestion, typically occurring in multi-intersection settings, is a condition in which a link is increasingly occupied by queued vehicles. Highly congested intersections often cause cross-blocking whereby vehicles moving upstream fail to cross an intersection due to lack of queuing positions at a designated link. Low traffic throughput and high congestion both lead to an increase in vehicle delay, a fundamental metric in evaluating traffic signal controller performance. In our study, as well as in the other multi-intersection scheduling schemes, the ultimate goal is to maximise traffic throughput, avoid traffic congestion and intersection cross-blocking and reduce overall vehicle delay.

doi: 10.1049/iet-its.2009.0070

4.2 Intersection network configuration

The network of intersections under study is illustrated in Fig. 1. This is a five-intersection traffic network, in which the intersection at the middle is referred to as the central intersection. The other four intersections are labelled as outbound intersections. In multi-agent systems, agent cooperation does not mean an agent can extract information from all other agents, but rather that agents are able to inquire about regional (local) information from their neighbours agents. Hence, we assume an intersection agent can only communicate with its immediate neighbours. Consequently, the network in Fig. 1 is a basic computational component for larger multi-intersection networks, without loss of generality.

Individually, a four-way intersection is the most common intersection in real life, and therefore being most suitable to be considered in our approach. Even though the capacity of intersections might diverge, the queues that impact the intersections are within a certain range from the intersection. Therefore, considering the same maximum capacity for intersections yields a more generic solution.

During each simulation time step, new vehicles are generated, as governed by a Poisson process, outside each outbound intersection. They are placed at the end of the queue of their respective destination lanes. No vehicles are generated at the central intersection. Based on our previous study in [18], eight-phase combination schemes are available to each intersection (Section 4.3.2). Vehicles in lanes which are notated by even numbers can either go straight or turn right. Vehicles at the odd lanes should turn left to their designated queues. All lanes can queue at most 40 vehicles. In general, vehicles cross either only one intersection or three intersections (outbound-central-outbound) prior to leaving the network. In our performance analysis, we collect statistics pertaining to vehicles which have crossed the central intersection.

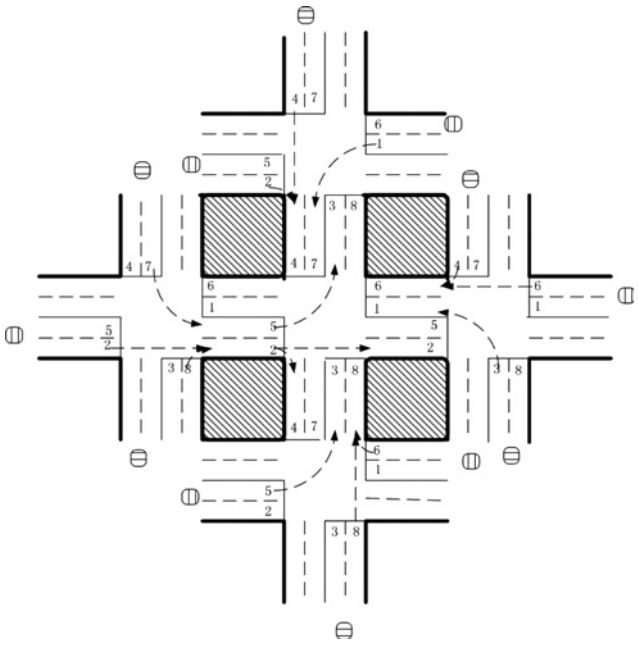

Figure 1 *Five-intersection, centrally connected vehicular traffic network studied*

Four outbound intersections operate based on local information while the central intersection hosts an RL-based agent that controls traffic signaling

4.3 Definition of the RL elements

An RL problem is defined once states, actions and rewards are clearly defined. To that end, we next describe these basic constructs in the context of our problem domain.

4.3.1 System states: At each simulation time step, the local state of an intersection is based on local traffic statistics. The state is represented by an eight-dimensional feature vector with each element representing the relative traffic flow at one of the lanes. The relative traffic flow is defined as the total delay of vehicles in a lane divided by the average delay at all lanes in the intersection. For the outbound intersection agent, only local traffic statistics is considered (as suggested by the LQF algorithm). However, the central intersection agent is assumed to have access to all states of its neighbouring intersections. Intuitively, such additional information allows the central intersection to better predict upstream traffic flow, thereby improving its signal scheduling behaviour. There is a nonlinear relation between the action selection and traffic statistics in the network. Given the large state space that is spanned under these assumptions, we employ a feedforward neural network providing nonlinear function approximation.

4.3.2 Action set: As indicated in our previous work [18], each intersection in Fig. 1 is labelled following the National Electrical Manufacturers Association convention, applying a two-ring structure. Each ring has four phases (1–4, 5–8). At any given moment, one phase from each ring will be displayed. Careful observation reveals that the maximal number of applicable, compatible and non-conflicting phase combinations is eight for each isolated intersection presented {(1,5), (1,6), (2,5), (2,6), (3,7), (3,8), (4,7), (4,8)}. In a multi-intersection network, each intersection agent individually selects an action among the available eight-phase combinations, according to its learned policy. It can be noted that real networks already work under the eight-phase combination schemes. In the proposed approach, the sequence constraints are relaxed and may affect only driver expectations, becoming a process of behavioural adaptation.

It should be noted that for a fixed time controller, all available phases should at least appear once within a cycle. This is not the case for actuated controllers where phases can be shortened or skipped, dependent on demand, and where minimum and maximum green times for each phase are imposed. Differently, the proposed algorithm decide on

actions to be taken at regular intervals of time (20 time units, for example), innovating the green time constraints concept. In other words, smaller (minimum) intervals can be set where a new action will take place or where the algorithm will continue to choose the same action until it does not yield the best reward. Therefore, the proposed algorithm deviates from the established notion of coordination (split, cycle and offset) and explores the concept that the proposed RL-based multi-agent system can potentially guarantee better overall network performance, minimising the average delay, congestion and likelihood of intersection cross-blocking.

4.3.3 Reward function: For both types of intersection agents, a reward is provided to an intersection agent after executing a given action. The reward ranges from −1 to 1, where positive reward values are received if the current delay is lower than that of the previous time step. On the other hand, the agent is subject to a penalty (negative value) if an increased average delay is observed. In [19], a weighted exit traffic flows metric was used to quantify the reward. In our study, the local reward is directly affected by vehicle volume and vehicular delay at an intersection, which is defined as

$$r = \frac{D_{\text{last}} - D_{\text{current}}}{\max[D_{\text{last}}, D_{\text{current}}]} \tag{1}$$

where D_{last} and D_{current} are the previous and current intersection total delay, respectively. However, considering the fact that the behaviour of the central intersection agent impacts those of the other intersection in the network, we propose to incorporate delay information from the other networks into the reward measure of the central intersection. This is achieved by defining r to be a weighted sum (at a ratio of 1:4) of the central intersection delay and that of the other intersections.

In a real-life application, vehicle delay could be estimated through the application of technological advancements in the field of vehicular sensors and traffic controllers. Advance detector (upstream detectors) actuations can be used to track vehicular arrivals at each intersection approach over time. Phase change data and saturation headway data can be used to estimate the number of departures from the stop bar over time. The two flow profiles can then be combined to estimate the queue accumulation on the intersection approach. The time in queue can be used to estimate delay for each approach and consequently for the entire intersection.

5 System scheduling algorithm

5.1 The Q-Learning scheduling algorithm

The most important and popular model-free RL algorithm, Q_Learning [3], is utilised to quantify the preference and effectiveness of selecting an action given a perceived state. Following every selection of an action, the corresponding Q value is updated as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2}$$

where a_t is the action executed while in state s_t leading to the subsequent state s_{t+1}, and yielding a reward r_{t+1}, α is the step-size parameter used in the incremental method described above and changes from time step to time step and γ is the discount rate for the rewards.

With the continuous task, the reward to be maximised could easily go to infinite. The discount rate γ determines the present value of the future rewards. A reward received at time step in the future is worth γ^{n-1} after n time steps. In vehicular traffic network, an intersection agent is charged with selecting a phase combination for a given traffic condition and updating the value function approximation using the resulting reward. In time, as the agent becomes proficient, the value function approximation error is expected to become marginal.

5.2 Q Learning with function approximation

Ideally, the estimation of the value function can be represented in a tabular form, for which an optimal policy can be obtained. However, most real-world problems have large or infinite state spaces, with the problem at hand being no different. This can be addressed by utilising function approximation techniques, so as to approximate the true value function [3]. Commonly used function approximation methods included neural networks, cerebellar model articulator controllers [20] and radial basis functions [21]. In our study, a feedforward neural network, trained using the back-propagation algorithm, is used to provide an approximation to the state–action value function.

The neural network used has 40 input nodes (representing the 5 state vectors), 25 hidden units and an output corresponding to each of the actions. At the beginning of a time step, state information is collected, and for each of the eight possible actions, a feedforward process is applied to the neural network. This results in state–action value estimates at the output of the neural network. The action that corresponds to the highest state–action value is selected with high probability, while with the complementing likelihood an exploratory action is taken. At the subsequent time step, the same process is repeated, with the reward signal being received, allowing for an estimation error of $\alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$ to guide the process of learning the state–action values. The goal of this learning process is to obtain a set of weights (network parameters) that facilitate accurate state–action value estimation.

doi: 10.1049/iet-its.2009.0070

Since the weights of the neural network are initially randomly assigned, the system is expected to be exposed to sufficient experience so as to gain proficiency. Therefore a large exploration rate is necessary at the very beginning as means of promoting exposure to more states. The exploration rate is reduced with the agent's proficiency, to a point where very little exploration is required.

5.3 Convergence for Q Learning by function approximation

Although Q_Learning in a tabular form has been proved to converge deterministically to an optimal policy [22], Q_Learning with function approximation can only guaranteed to reach a suboptimal policy [23], primarily because of the limitations of the function approximation module. Having said that, several techniques can be effectively utilised to improve the overall performance of neural networks in the context of value function approximations, with the following techniques used in the setup described here.

1. *Data standardisation*: Large variance in input signal magnitudes would belittle the contributions of inputs with relatively small dynamic range. To address this problem, the input values should be normalised within a specific range. For each element in state vector, we impose a dynamic range of $[-1, 1]$.

2. *Activation function*: Instead of using the common sigmoid function, an anti-symmetric sigmoid function which is centred at zero is taken as the activation function.

3. *Learning rate adaptation*: The learning rate controls how fast the Q value would be updated. If the learning rate is smaller than the second-order derivative of the delta error, a local minimum can be found. A more efficient way to achieve this goal is to use the Boltzmann learning rate [24], which gradually decreases in time, proportional to the reduction in mean error. In our simulations, the learning rate is discounted by a fixed value and is lower-bounded by 0.01.

4. *Momentum*: During the learning process, should the error result in negligible changes to the state-action value estimates, an agent would assume it found the optimal policy. In all other cases, a momentum element is used to modified the weight changes such that

$$\varpi^{k+1} = \varpi^{k} + (1 - c^{k})\Delta\varpi^{k}_{bp} + c^{k}(\varpi^{k} - \varpi^{k-1}) \qquad (3)$$

where ϖ^{k} is the weight at time step k, $\Delta\varpi^{k}_{bp}$ the error gradient and c^{k} the learning rate.

5. *Weight decay*: Because of the random initial value of weights when constructing the neural network, a weight decay scheme has similar impact to that of pruning methods and often leads to better performance. In our Q_Learning realisation, a simple weight decay approach was taken, such that $\varpi_{\text{new}} = (1 - \varepsilon)\varpi_{\text{old}}$.

6 Simulation results

6.1 Simulation setup and parameters

All simulations were executed in the Matlab environment using a discrete event environment. A one time unit refers to one time step in discrete simulation environment. One discrete simulation time step is defined as one discrete second. Intersection agents took actions once every 20 time units, and no more than one vehicle was allowed to traverse the intersection at each time unit. If a light signal transitions from red to green, there is a 2 unit delay for red-clear. Traffic arrivals followed the Poisson distribution with average arrival rate ranging from 0.1 to 1.0. As mentioned above, to evaluate the performance, we only collected statistics pertaining to vehicles which passed the central intersection. All simulations pertained to the five-intersection network described in Section 4.2. The duration of each simulation run was 20 000 time units. The weight decay factor was 0.05, and the Q_Learning discount factor (γ) was chosen to be 0.95.

6.2 Results and discussion

Promising results were reported by the authors in their previous work [18] pertaining to the case of a single intersection operating under the LQF scheme. Additional work on traffic signal scheduling (Section 3) would usually be compared with a fixed time strategy, what can be considered not to be a fair comparison to adaptive, responsive or intelligent systems, due to its static nature. Differently, the LQF algorithm was compared not only to an optimised fixed time strategy but to a vehicle-actuated controller as well, yielding best results at higher relative traffic loads. Appropriately, the comparison of the RL system with the novel LQF system is then intrinsically valid. The primary goal of the simulations presented next were to contrast results drawn from a five-intersection traffic network performing solely under the novel LQF algorithm, with the results obtained using the CRL framework proposed. Therefore the RL-based agent was compared with that running the LQF algorithm, in which every agent only considers its own local traffic volume and thus controls its traffic signals in isolation. Prior to measuring and evaluating the performance of the central agent, the latter was given the opportunity to run for 10 000 steps so as to learn the environment with a decreasing exploration rate. Following the learning phase, the state–action value function updating scheme was allowed to continue with an exploration rate of 0.02. It should be noted that the outbound agents only apply the LQF algorithm without RL and therefore not being required to go through the learning phase.

An initial goal was to investigate the mean delay experienced by vehicles traversing the central intersection under either scheme. Fig. 2 depicts the mean delay per vehicle, averaged over ten simulation runs. As expected,

with an increase in traffic volume, we observe an increase in average delays for both scheduling methods. When arrival rates are lower than 0.45, the LQF scheduling algorithm performs slightly better than the multi-agent Q_Learning system. This is because at low arrival rates, a relatively small number of vehicles flow into the network and thus lengths of queues are typically small. Under the LQF scheduling scheme, all intersection agents simply count the number of vehicles and select the maximal phase combination. However, in the RL agent scheme, the central intersection agent does not only take the queue lengths into consideration, but also the waiting times, as reflected by the unique reward function defined.

Upon issuing an action, the central intersection agent receives a positive reward if the total intersection delay was reduced with respect to the previous interval. If the opposite holds, the agent is provided with a negative reward (i.e. penalty). Therefore at low arrival rates, the negligible discrepancy in the average delay times somewhat limits performance. The latter has a turning point observed at arrival rates of around 0.45. At higher arrival rates, which correspond to more challenging traffic scenarios, the RL based scheme improves significantly over the localised scheme. In a congested traffic network, almost all queues are fully occupied (relative to the prescribed limit). As an extreme case, if the traffic arrival rate is 100%, the queue length combinations would be close to identical and thus LQF would take any of the eight actions with equal probability.

The next performance metric to be considered was cross-blocking, being computed by the time units when vehicles can not cross the intersection in green phase due to the fully occupied desired lane, divided by 20 time units. In Fig. 3, cross-blocking comparative results are shown. The proposed algorithm exhibits superiority over the LQF algorithm. While vehicles can move across the intersection

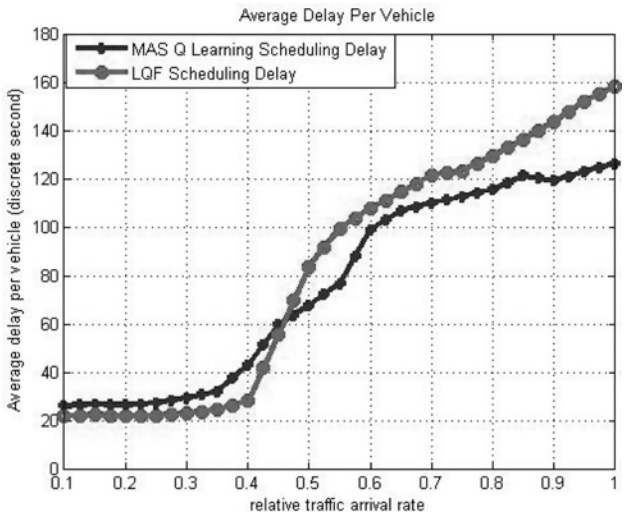

Figure 2 *Average delay per vehicle for the five-intersection network discussed*

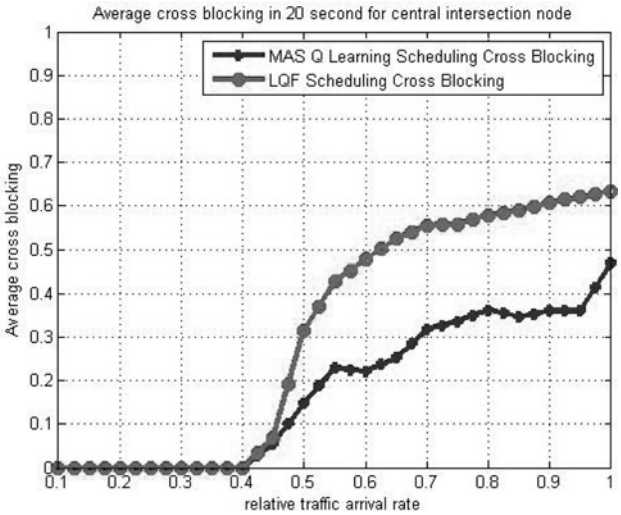

Figure 3 *Average cross-blocking in 20 time units for the central intersection in the network studied*

smoothly, as more and more vehicles flow into the traffic network, the frequency at which vehicles are blocked and are forced to come to a halt under a green light is increasing. At medium to high arrival rates, the cross-blocking frequency under the proposed algorithm only accounts for half of that observed under the LQF controller.

7 Conclusions

This paper introduced a multi-agent system and RL-based framework for scheduling traffic signals at intersection networks. At the core of the proposed system is an approximate optimal controller, which aims to maximise an expected reward construct that directly relates to minimising queuing delays across an intersection. The proposed methodology is distributed and inherently scalable where the latter pertains to the ability to map to larger intersection networks. Therefore the algorithm can be readily applied to more complex intersections without substantial changes to its core structure. The proposed approach was developed while viewing the traffic scheduling problem as an artificial intelligence task, representing a shift from conventional traffic scheduling problem formulations. Careful integration of the adaptive RL system with static LQF based controllers was studied as both a developmental process and a gauging mechanism for indicating the degree of improvement that can be expected. Performance improvement was observed with respect to both average vehicle delay as well as cross-blocking likelihood, particularly in the context of high traffic scenarios.

In future work, the authors intend to extend the initial results obtained to include additional performance metrics, such as probability of stopping and vehicle velocity jitter. Moreover, the basic five-intersection network considered here will be expanded to include larger traffic networks and more extensive collaboration among agents.

doi: 10.1049/iet-its.2009.0070

8 References

[1] KAELBLING L.P., LITTMAN M.L.: 'Reinforcement learning: a survey', *J. Artif. Intell. Res.*, 1996, **4**, pp. 279–284

[2] WATKINS C.J.C.H.: 'Learning from delayed rewards'. PhD thesis, Cambridge University, Cambridge, UK, 1989

[3] SUTTON R., BARTO A.: 'Reinforcement learning: an introduction' (MIT Press, 1998)

[4] JAYAKRISHNAN R., MATTINGLY S., MCNALLY M.: 'Performance study of SCOOT traffic control system with non-ideal detectorization: field operational test in the city of Anaheim'. 80th Ann. Meeting of the Transportation Research Board, Washington, DC, 2001

[5] WOLSHON B., TAYLOR W.: 'Analysis of intersection delay under real-time adaptive signal control', *Transp. Res. Part C*, 1999, **7**, pp. 53–72

[6] FEHON P.K.: 'Adaptive traffic signals are we missing the boat?'. ITE District 6 Ann. Meeting, DKS Associates, 2004

[7] SALKHAM A., CUNNINGHAM R., GARG A., CAHILL V.: 'A collaborative reinforcement learning approach to urban traffic control optimization'. Proc. 2008 IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology, Sydney, Australia, December 2008, pp. 560–566

[8] PAPAGEORGIOU M., DIAKAKI C., DINOPOULOU V., KOTSIALOS A., WANG Y.: 'Review of road traffic control strategies', *Proc. IEEE*, 2003, **91**, (12), pp. 2043–2067

[9] PAPAGEORGIOU M., BEN-AKIVA M., BOTTOM J., BOVY P., HOOGENDOORN S., HOUNSELL N., KOTSIALOS A., MCDONALD M.: 'ITS and traffic management', in BARNHART C., LAPORTE G. (EDS.): 'Handbook in operations research & management science: transportation' (Elsevier, North Holland), vol. 14, pp. 715–774

[10] DIAKAKI C., PAPAGEORGIOU M., ABOUDOLAS K.: 'A multivariable regulator approach to traffic-responsive network-wide signal control', *Control Eng. Pract.*, 2002, **10**, pp. 183–195

[11] DE OLIVEIRA L.-B., CAMPONOGARA E.: 'Multi-agent model predictive control of signaling split in urban traffic networks', *Transp. Res.*, 2010, **18C**, (1), pp. 120–139

[12] SRINIVASAN D., CHOY M.C.: 'Cooperative multi-agent system for coordinated traffic signal control', *Intell. Transp. Syst., IEE Proc.*, 2006, **153**, (1), pp. 41–50

[13] WIERING M.: 'Multi-agent reinforcement learning for traffic light control'. Proc. 17th Int. Conf. on Machine Learning, 2000

[14] ABDULHAI B.: 'Reinforcement learning for the true adaptive traffic signal control', *J. Transp. Eng.*, 2003, **129**, (3), pp. 278–285

[15] CAI C., WONG C.K., HEYDECKER B.G.: 'Adaptive traffic signal control using approximate dynamic programming', *Transp. Res. Part C*, 2009, **17**, (5), pp. 456–474

[16] YU X.-H., RECKER W.W.: 'Stochastic adaptive control model for traffic signal systems', *Transp. Res.*, 2006, **14C**, (4), pp. 263–282

[17] GERSHENSON C.: 'Self-organizing traffic lights', *Complex Syst.*, 2005, **16**, (1), pp. 29–53

[18] WUNDERLICH R., LIU C., ELHANANY I., URBANIK T.: 'A novel signal scheduling algorithm with quality of service provisioning for an isolate intersection', *IEEE Trans. Intell. Transp. Syst.*, 2008, **9**, (3), pp. 536–547

[19] JACOB C., ABDULHAI B.: 'Automated adaptive traffic corridor control using reinforcement learning', *Transp. Res. Rec.: J. Transp. Res. Board*, 2006, **1959**, pp. 1–8

[20] ALBUS J.S.: 'A theory of cerebellar function', *Math. Biosci.*, 1971, **2**, pp. 25–61

[21] BROOMHEAD D.S., LOWE D.: 'Multivariable functional interpolation and adaptive networks', *Complex Syst.*, 1998, **2**, pp. 321–355

[22] WALKINS C.J.C.H., DAYAN P.: 'Q-learning', *Mach. Learn.*, 1992, **8**, pp. 270–300

[23] HAYKIN S.: 'Neural networks: a comprehensive foundation' (Prentice-Hall, 1998, 2nd edn.), pp. 670–700

[24] YEGNANARAYANA B.: 'Artificial neural networks' (PHI Learning Pvt. Ltd, 2004), pp. 190–192