# TEMPORAL DIFFERENCE LEARNING-BASED ADAPTIVE TRAFFIC SIGNAL CONTROL

*S. El-Tantawy is a PhD candidate in Civil Engineering Department, University of Toronto, ON, M5S 1A4, Canada (phone: 416-978-5049; fax: 416-978-5054; e-mail: samah.el.tantawy@utoronto.ca).*

*B. Abdulhai is Canada Research Chair in ITS and Director of Toronto ITS Centre and Testbed, Civil Engineering Department, University of Toronto, ON, M5S 1A4, Canada. (e-mail: baher.abdulhai@utoronto. ca).*

## ABSTRACT

Traffic congestion negatively affects our society, economy, and the environment. In most dense urban areas, transportation networks are operating near capacity for extended peak periods, due to the population growth and the limited resources for infrastructure expansions. As a result, numerous ITS strategies rapidly emerged to better utilize the existing infrastructure. Adaptive traffic signal control is a promising technique to alleviate traffic congestion. This paper focuses on the development of an adaptive traffic signal control method using Reinforcement Learning (RL) as one of the efficient approaches to solve such stochastic control problem. A generic RL engine is developed for acyclic-variable-phasing sequence adaptive traffic signal control and applied to an isolated traffic signal control case. Paramics, a microsimulation simulation platform, is used to evaluate several Temporal Difference (TD) learning methods on a major intersection in Downtown Toronto using actual observed traffic data. The performance of TD approaches is analyzed and contrasted to an optimized pre-timed control strategy as a bench mark.

*Keywords: Adaptive Signal Control, Reinforcement Learning, Temporal Difference. Q-Learning, SARSA, Eligibility Traces.*

## INTRODUCTION

Population is steadily increasing worldwide. Consequently the demand for mobility is increasing, traffic congestion is deteriorating, and undesirable changes in the environment are becoming major concerns. Until relatively recently, infrastructure improvements have been the primarily method to cope with congestion. However, tight constraints on financial resources and physical space have accentuated the consideration of a wider range of options. Therefore, the emphasis has shifted to improving the existing infrastructure by

optimizing the utilization of the available capacity. Intelligent Transportation Systems (ITS) have the potential to significantly alleviate traffic congestion by using innovative traffic signal control strategies. Pretimed and actuated traffic signal control systems are the most common control systems for isolated intersections. Adaptive traffic signal control on the other hand adjusts signal timing parameters in response to real-time traffic flow fluctuations; therefore, has a great potential to outperform both pretimed and actuated control (McShane *et al.*, 1998). Several adaptive signal control methods have been investigated in the literature, the majority of them are not self-learning. Due to the stochastic nature of the traffic system, a control strategy that is adaptive to the fluctuations in traffic conditions is paramount. Reinforcement Learning (RL) has shown great potential for self-learning traffic signal control in the stochastic traffic environment, the main advantage of which is the ability to perpetually learn and improve service over time (Abdulhai and Kattan, 2003). Although several studies investigated the potential of RL methods for adaptive signal control (Thorpe, 1997; Abdulhai *et al.*, 2003; Lu *et al.*, 2008), and more specifically Temporal Difference (TD) methods, an in-depth comparative analysis for different TD methods is still largely missing. Also, the above studies focused on fixed phasing sequence which limits the adaptability of the system to traffic flow fluctuations.

This paper starts with a brief description of the stochastic control problem and then TD learning methods are discussed. The state-of-the-art RL-based traffic signal control systems are then reviewed and the gaps in the literature are highlighted. A RL-based adaptive signal control system with variable phasing sequence is proposed to test different TD learning methods on a real-world multi-phase intersection in downtown Toronto. Finally, a discussion and analysis of the results is presented and contrasted to a traditional pretimed control strategy which is optimized offline.

# PROBELM DEFINITION: STOCHASTIC OPTIMAL TRAFFIC CONTROL

Stochastic optimal control problems (SOCP) are the control optimization problems in the presence of uncertainty. Markov Decision Process (MDP) represents the mathematical framework for SOCP. MDP is a discrete time stochastic control process. At each time step, the environment (process) is in some state s, and the agent (decision maker) chooses certain action *a* that will affect the environment. The environment responds at the next time step by stochastically moving into a new state s', and assigning the agent a corresponding reward r(s,a,s') for choosing action *a* at state *s*. The probability that the environment moves to a next state s' is influenced by the currently selected action a; which is given by the state transition function P(s,a,s'). Thus, the next state s' depends on the current state s and the agent's action a, but it is conditionally independent of all previous states and actions (Markov property). Due to the stochasticity associated with the traffic flow approaching signalized intersections, the adaptive signal control problem can been approached as an SOCP.

The goal of SOCP is to obtain a policy (mapping between states and actions) $\pi$, that maximizes the expected cumulative discounted reward at each state. The expected cumulative discounted reward at state s is defined as the *value function* of the state $V(s)$ or

state-action pair Q(s,a), also called Q-Value. For finite MDP (MDP with finite set of states (S) and actions (A)), V(s) and Q(s,a) are defined as follows;

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{n=0}^{\infty}\gamma^{k} r_{k+n+1}\,\middle|\, s_k = s\right\}$$

$$V^{\pi}(s)=\sum_{a\in A} P(s,a)\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma V^{\pi}(s')\right] \qquad (1)$$

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{n=0}^{\infty}\gamma^{k} r_{k+n+1}\,\middle|\, s_k = s, a_k = s\right\}$$

$$Q^{\pi}(s,a)=\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma\sum_{a'\in A} P(s',a')Q^{\pi}(s',a')\right] \qquad (2)$$

where $P(s,a)=\sum_{s'\in S} P(s,a,s')\,P(s')$, $\sum_{s'\in S} P(s')=1$ and $\gamma$ is the discounted factor. Equations 2 and 3 are defined as *Bellman* equations for discounted reward. They express the relationship between the value of a state and the value of its successor state.

The optimal policy, denoted $\pi^{*}$, is the policy that has the *optimal state-value function, $V^{*}$,* (optimal *Q-value function, $Q^{*}$)* denoted, and defined as follows:

$$V^{*}(s)=\max_{a\in A}\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma V^{*}(s')\right] \qquad (3)$$

$$\pi^{*}(s)\in arg\max_{a\in A}\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma V^{*}(s')\right] \qquad (4)$$

$$Q^{*}(s,a)=\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma\max_{a\in A} Q^{*}(s',a')\right] \qquad (5)$$

$$\pi^{*}(s)\in arg\max_{a\in A}\sum_{s'\in S} P(s,a,s')\left[r(s,a,s')+\gamma Q^{*}(s',a')\right] \qquad (6)$$

While equations 3, 5 are defined as Bellman optimality equations for state-value and action-state value functions, respectively, equations 4 and 6 represent the corresponding optimal policies. Dynamic programming (DP) is widely considered as the only feasible way of solving general SOCP (Bertsekas, 1976; Bertsekas, 2007).

Value iteration is one of the DP techniques that can be used to find the optimal policy (Sutton and Barto, 1998). The Value iteration method estimates the optimal state-value function by simply turning the Bellman optimality equation into an update rule of the value function as shown in Figure 1(a).

Although plausible method to tackle SOCP problems, DP suffers from the curse of dimensionality in which it requires enormous storage space to allow for representing large matrices of transition probabilities and rewards. In addition, DP requires pre-specified perfect values of the transition probabilities and rewards for every state–action pair which represents the theoretical model of the environment, yet the existence of which is questionable for traffic networks. Unfortunately, because of these limitations, using DP for real-time adaptive traffic signal control is impractical (Abdulhai and Kattan, 2003).

Reinforcement Learning (RL) on the other hand is a class of algorithms that can generate the optimal policy for SOCP in which the agent learns the optimal state (state-action) value functions through the interaction with the environment; therefore, it does not require the transition probabilities and rewards for every state–action pair. RL endeavours not only achieving as much as DP does but also lessening the computation without always assuming

a perfect model of the environment (Sutton and Barto, 1998), which are all very useful for solving traffic SOCPs.

# THE REINFORCEMENT LEARNING APPROACH: TEMPORAL DIFFERENCE ALGORITHMS

Numerous RL algorithms have been investigated in the literature (Sutton and Barto, 1998; Gosavi, 2003); the most relevant algorithms to this study are the on-line value iteration with discounted reward algorithms which are called Temporal Difference (TD) learning algorithms. Similar to the DP value iteration algorithm (Figure 1(a)), TD methods estimate the optimal state (state-action) value functions. In Equation (3), the state value function (and similarly Equation (5) for state-action value function) is expressed as an average of a random variable $[r(s,a,s') + \gamma V^*(s')]$ which can be obtained from the interaction with the environment (e.g. within a traffic simulator in our case and in the field after initial deployment). One simple algorithm that can estimate the average of a random variable from its samples (that can be generated within simulator) is Robbins-Monro algorithm(Robbins and Monro, 1951). Hence, Robbins-Monro algorithm can be used to estimate the state value function as follows (Gosavi, 2003);

$$\mathrm{V}^k(\mathrm{s}) = \mathrm{V}^{k-1}(\mathrm{s}) + \alpha^k[\, r^k + \gamma V^{k-1}(\mathrm{s}') - V^{k-1}(\mathrm{s})] \quad (7)$$

where $\alpha^k$ is the step-size (or learning rate) at time k. At time k, the value of state s, $\mathrm{V}^k(\mathrm{s})$, is updated using the observed reward $r^k$ and the estimate $V^{k-1}(\mathrm{s}')$. Therefore, TD methods can learn directly from experience without a dynamic model of the environment (to generate transition probabilities). In addition, TD methods update the estimates of the state values immediately after visiting the state. The temporal difference notion emerged from the fact that the value function is estimated based on the *difference* between *temporally* successive estimations, which is called TD error, denoted $\mathrm{TD}^k$.

$$\mathrm{TD}^k(\mathrm{s}) = [\, r^k + \gamma V^{k-1}(\mathrm{s}') - V^{k-1}(\mathrm{s})] \quad (8)$$

Equation (7) represents the simplest form of TD known as TD(0). Two types of TD algorithms are presented in the literature, namely TD(0) and Eligibility Traces TD(λ).

## TD(0)

TD(0) , also known as 1-step TD resembles the on-line form of dynamic programming value iteration (Pendirth, 2000), in which the agent looks ahead one step in time and update the value functions based on the next reward $r^k$.

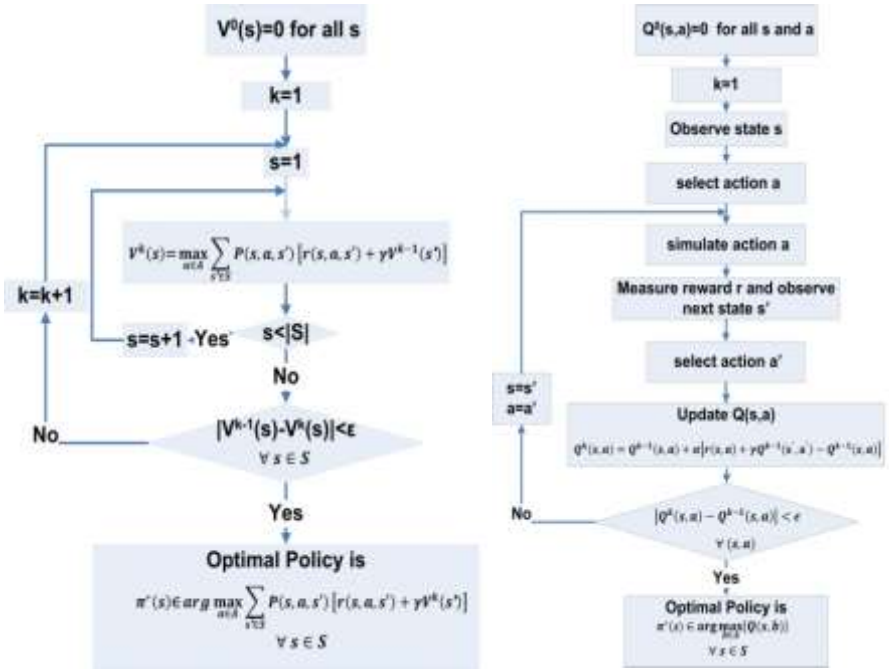
### *SARSA: On-Policy Algorithm*

The name SARSA emerged from the fact that the estimates of the state-action pair's values are conducted strictly based on experience which is represented by the quintuple (s,a,r,s',a'). The state-action value functions are updated using the results from executing actions determined by a policy that does not always select the greedy action (the action that has the highest value) (see Figure 1(b)). So, the SARSA algorithm is considered an on-policy TD

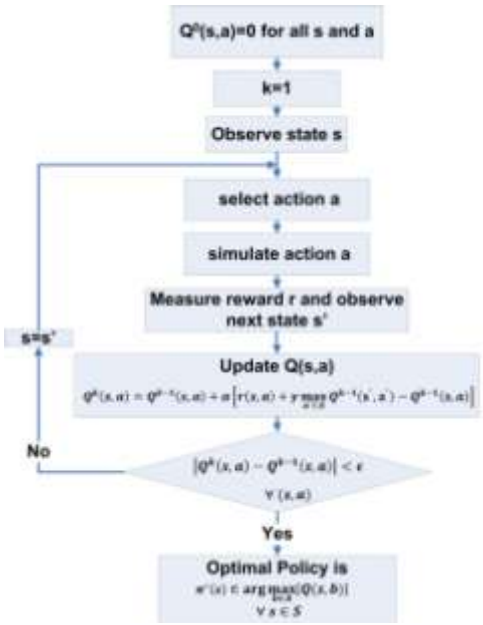
method since it is learning and improving the same policy that is followed in selecting the actions.

## *Q-Learning: Off-Policy Algorithm*

Q-Leaning updates the estimated value functions (Q-values) using greedy actions, independent of the policy being followed which involves exploratory actions. Therefore, Q-Learning is an off-policy because the agent attempts to improve a policy while following another one (see Figure 1(c)).




(a) (b)




(c)

Figure 1: Flowchart of (a) Value Iteration Dynamic Programming , (b) SARSA, and (c) Q-Learning

## Eligibility Traces (TD(λ))

In eligibility traces algorithms, the agent looks ahead all the way into and till the end of the learning horizon (T) and updates the value functions based on all future rewards. The TD error is then defined as follows (Sutton and Barto, 1998):

$$TD^k(s) = \alpha^k \left( (1-\lambda) \left( \sum_{n=1}^{T-t-1} \lambda^{n-1} R_k^{(n)} \right) + \lambda^{T-t-1} R_k \right) - V^k(s) \quad (9)$$

$$R_k^{(n)} = \sum_{l=0}^{n-1} \gamma^l \, r^{k+l+1} + \gamma^n V^k\left(s^{k+n}\right) \quad (10)$$

Where $R_k^{(n)}$ , the n-step return, is defined as the discounted reward of *n* future steps, each is weighted proportionally to $\lambda^{n-1}$, where $0 \leq \lambda \leq 1$ and $1-\lambda$ is a normalization factor ensures that the weighs sum to 1. This is known as forward view (theoretical view) of the TD(λ) algorithm. If λ = 0, $TD^k(s)$ reduces to Equation 8 and hence, the algorithm becomes the 1-step TD(0) which explains notion of TD(0).

In fact, the forward view is not directly implementable because it is acausal, i.e. values are updated based on future time steps. The backward view on the other hand provides an easy implementation of the theoretical view in a causal mechanism that approximates the forward view (Sutton and Barto, 1998); which can be achieved using eligibility traces algorithms. An eligibility trace is an additional variable for each state that keeps track of how often and how recently a state is visited. In other words, eligibility trace reflects how eligible a certain state for update whenever a reward occurs. At each time step, the eligibility trace for each state is updated as follows:

$$e^k(\mathrm{s}) = \begin{cases} \gamma\lambda e^{k-1}(\mathrm{s}) & if\ \mathrm{s} \neq s_k \\ \gamma\lambda e^{k-1}(\mathrm{s}) + 1 & if\ \mathrm{s} = s_k \end{cases} \quad (11)$$

where $\lambda$ is referred as the trace-decay parameter. The update is then performed on the value estimates of all the states as follows:

$$\mathrm{TD}^k(\mathrm{s}) = \alpha^k[\, r^k + \gamma V^{k-1}(\mathrm{s}') - V^{k-1}(\mathrm{s})]e^k(\mathrm{s}) \quad (12)$$

Implementation of equations (11) and (12) is equivalent to equation (9) (Sutton and Barto, 1998).

Eligibility traces thus aim for crediting states for rewards obtained later on. Eligibility traces allocates more credit for states visited recently than to states visited longer ago which matches biological brain strategies for deciding how recently received incentives should be used in combination with current incentives to determine actions (Jang *et al.*, 1997).

Combination of TD(0) methods (e.g. SARSA and Q-Learning) with eligibility traces results in SARSA(λ) and Q(λ) algorithms (Sutton and Barto, 1998).

### *SARSA (λ)*

Combining the typical SARSA algorithm (Figure 1(b)) and the eligibility traces concept, SARSA(λ) is produced (Figure 2(a)).

### *Q(λ)*

Since Q-learning learns about the greedy policy while it typically follows a policy involving exploratory actions, a subsequent experience can be used only as long as the greedy policy is being followed. Thus, unlike SARSA(λ), Watkins's Q(λ) does not look ahead all the way to the end of learning time; however, It only looks ahead as far as the next exploratory action. Particularly, all the eligibility traces are cut off to zero every time an exploratory action is taken (Figure 2(b)).

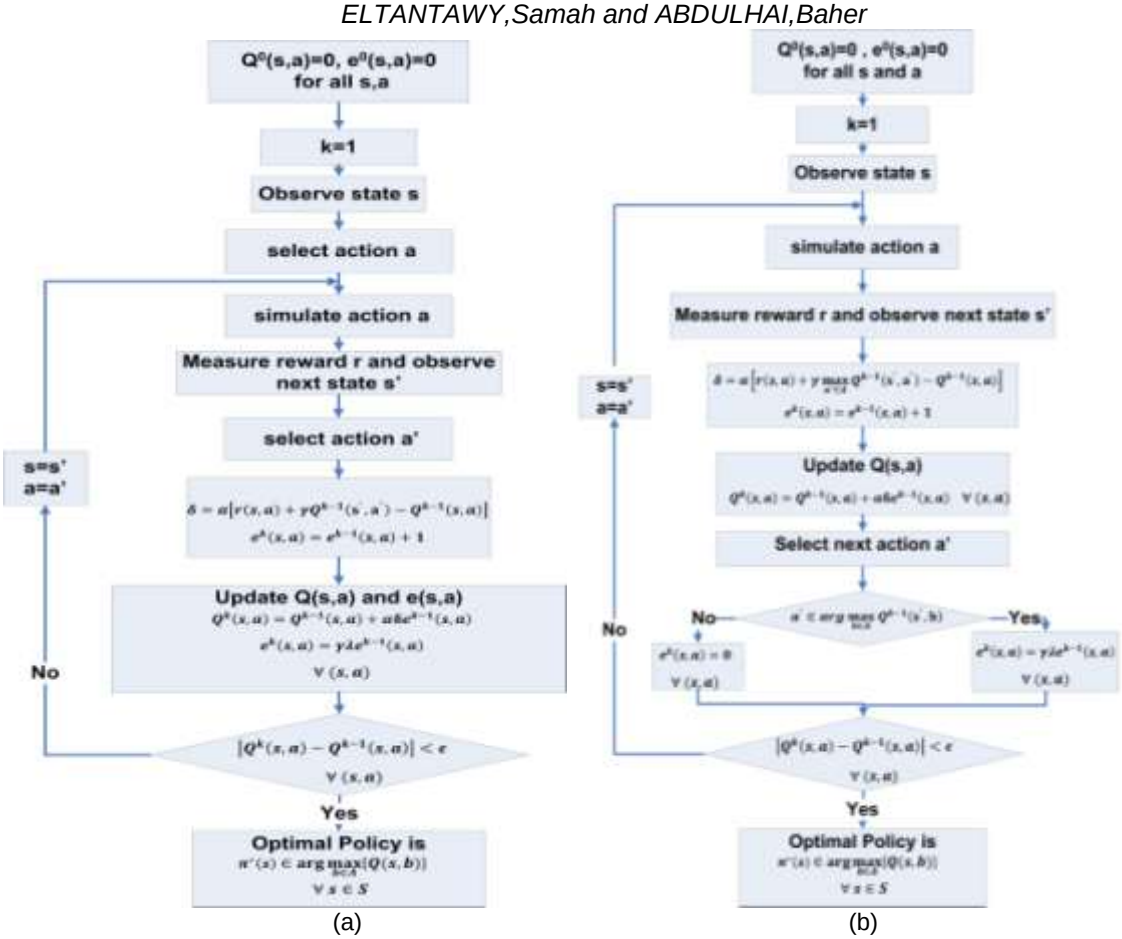



Figure 2 Eligibility Traces Algorithms (a) SARSA(λ), and (b) Watkin's Q(λ)

## Action Selection Strategy

One of the key aspects of RL is the trade-off between exploitation and exploration. To accumulate the maximum reward, the agent must exploit (or reinforce) the best experienced actions. However, it has to explore new actions to discover better selection of actions for the future. Also since the condition of convergence to optimal policy is to visit each state-action pair an infinite number of times (Watkins and Dayan, 1992), exploration allows visiting larger number of state-action pairs. To balance the exploration and exploitation in RL algorithms, algorithms such as $\epsilon$-greedy and softmax are typically used (Sutton and Barto, 1998).

### *$\varepsilon$ -greedy*

In each iteration, the $\varepsilon$-greedy method is used for action selection in which the agent selects the greedy action most of the time except for $\varepsilon$ amount of the time, it selects a random action uniformly (Sutton and Barto, 1998).
This research adopts and compares two $\varepsilon$ - greedy exploration methods; one with constant exploration rate ( $\varepsilon$ ) and another one with a gradually decreasing rate of exploration, so that at the beginning, it is mostly exploration, as the agent does not know much about the environment, and more exploitation takes place towards the end of the learning as the agent

converges to the optimal policy. Gradual shifting is required to ensure that the entire state space is covered during learning.

The gradually decreasing rate of exploration can be represented by an exponentially decreasing function as follows: $\epsilon = e^{-En}$, as suggested by (Jacob, 2005), where E is a constant and *n* is the iteration number (the "age" of the agent). Figure 3 demonstrates how the exploration rate depends on age for three values of *E:*0.05, 0.005 and 0.003. To maintain the balance between exploration and exploitation $\epsilon$ is typically gradually decreasing to a minimum value of 0.1.

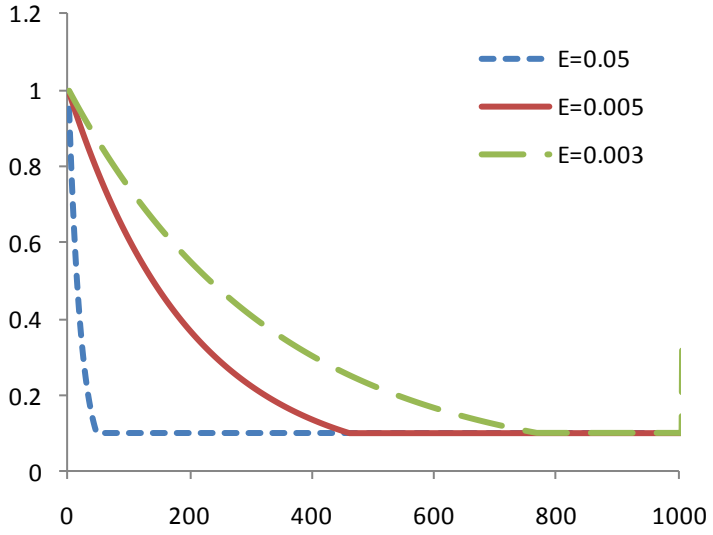



Figure 3 Exploration Rate with Iteration Number

## *Softmax*

One disadvantage of the $\varepsilon$ -greedy selection method is that it treats all exploratory actions equally, irrespective of the estimated value function of each action; which means that it is as likely to chose the worst action as it is to chose the next-to-best action.

To overcome this limitation that may adversely affect the real life applications, particularly where the worst action can be drastic, exploration can be concentrated on the most promising actions in terms of their estimated values. Thus, the selection of action probabilities can be represented by a proportional function to the estimated values, which is referred as softmax action selection strategy (Sutton and Bartoo, 1998). The most common softmax methods use Boltzman distribution and results in the following probability of action selection:

$$P_s(a) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{b \in A} e^{\frac{Q(s,b)}{\tau}}} \qquad (13)$$

The parameter $\tau$ is called the temperature. When $\tau$ is large, each action will have approximately the same probability to be selected (more exploration). When $\tau$ is small, actions will be selected proportionally to their estimated payoff (more exploitation).

*Mixed $\varepsilon$ -Greedy and Softmax ( $\varepsilon$ -Softmax)*

The choice rule suggested by (Wahba, 2008) follows a mixed of $\varepsilon$ -greedy and Softmax action-choice model, with a 1- $\varepsilon$ probability of exploitation and $\varepsilon$ probability of exploration: when exploiting,

$$P_s(a) = \begin{cases} 1 & Q(s,a) = \max_{a' \in A} Q(s,a') \\ 0 & otherwise \end{cases}$$

and when exploring

$$P_s(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in A} e^{Q(s,b)/\tau}} .$$

In this method, the greedy action is still given the highest selection probability, and all other actions are weighted according to their estimated values.

### Learning Rate

The learning rate (step-size), $\alpha$, controls how fast the estimates of value functions are modified. It is generally recommended to start with high learning rate to allow for fast modifications, and then using lowers rates as time progresses (Gosavi, 2003). By reducing $\alpha$ at an appropriate rate during learning, the convergence can be achieved even with nondeterministic setting. The step-size rule suggested by (Gosavi, 2003) has been adopted in this research as follows:

$$\alpha^k = \frac{1}{v^k(s,a)}$$

where $\alpha^k$ is the learning rate at time k and $v^k(s,a)$ is the number of visits to a particular state-action pair (s,a).

### Discount Rate

The discount rate $\gamma$ determines the present value of the future rewards, i.e., a reward that will be received k time steps in the future is worth only $\gamma^{k-1}$ times what it would have been worth it if the reward was received immediately. Thus, with increasing the value of $\gamma$ (i.e. approaching 1), the agent becomes more far-seeing because it accounts for future rewards more strongly (Sutton and Barto, 1998). In this research, $\gamma = 0.8$ is chosen.

## IMPLEMENTATION CASE STUDY: RL-BASED SINGLE AGENT ADAPTIVE SIGNAL CONTROL:

### The State of the Art

Abdulhai *et al.* (2003)and Lu *et al.* (2008) and, Thorpe (1997)(Thorpe, 1997) introduced the Q-Learning and SARSA for isolated adaptive traffic signal control. In (Bingham, 2001), a neuro-fuzzy traffic signal controller that uses RL for learning the neural network. Oliveira *et al*. (De Oliveira *et al.*, 2006) proposed an RL-based method that learns by detecting context changes in the traffic network. Although these studies contributed significantly to the

literature, the algorithms were tested on hypothetical simplified two-phase intersections. The following paragraphs summarize the major challenges and gaps in the existing literature and describe how the presented framework addresses these limitations/gaps.

- The studies that investigated single agent RL for traffic signal control (Thorpe, 1997; Bingham, 2001; Abdulhai *et al.*, 2003; De Oliveira *et al.*, 2006; Lu *et al.*, 2008) are designed to solve fixed phasing sequence intersections. Considering fixed phasing sequence signals can significantly reduce the dimension of action space and consequently shorten the computation time of the RL algorithm. However, these systems lack the flexibility to fully adapt to traffic flow fluctuations due to the phase sequence constraint. To address these limitations, the proposed single agent RL controller is designed to account for variable phasing sequence in which the control action is no longer an extension or a termination of the current phase as in the fixed phasing sequence approach; instead, the algorithm extends the current phase or switches to any other phase according to the fluctuations in traffic, possibly skipping unnecessary phases. Therefore, this algorithm is envisioned as an acyclic timing scheme with variable phasing sequence in which not only the cycle length is variable but also the phasing sequence is not predetermined. Although we opt to investigate this option, we are cognizant of potential driver confusion due to phase skipping; especially drivers are not used to such systems. However, the benefits can outweigh the disbenefits at least for the purpose of investigation.
- Different RL algorithms, and more specifically TD algorithms, have been quantitatively investigated separately in the literature without qualitatively discussing the appropriateness of each TD method in solving the traffic signal control problem. Also, $\varepsilon$-greedy action selection method is the most commonly used strategy with constant value of $\varepsilon$. In this paper, different action selection methods are investigated with different $\varepsilon$-profiles for $\varepsilon$-greedy method and more importantly, the performance of different TD algorithms in solving the adaptive traffic signal control problem are is investigated.

## System Design

A generic RL engine is developed using Java programming language in away so that different learning method and different action selection strategy can be generically tested. The interaction between the agent and the environment is represented in Figure 4.

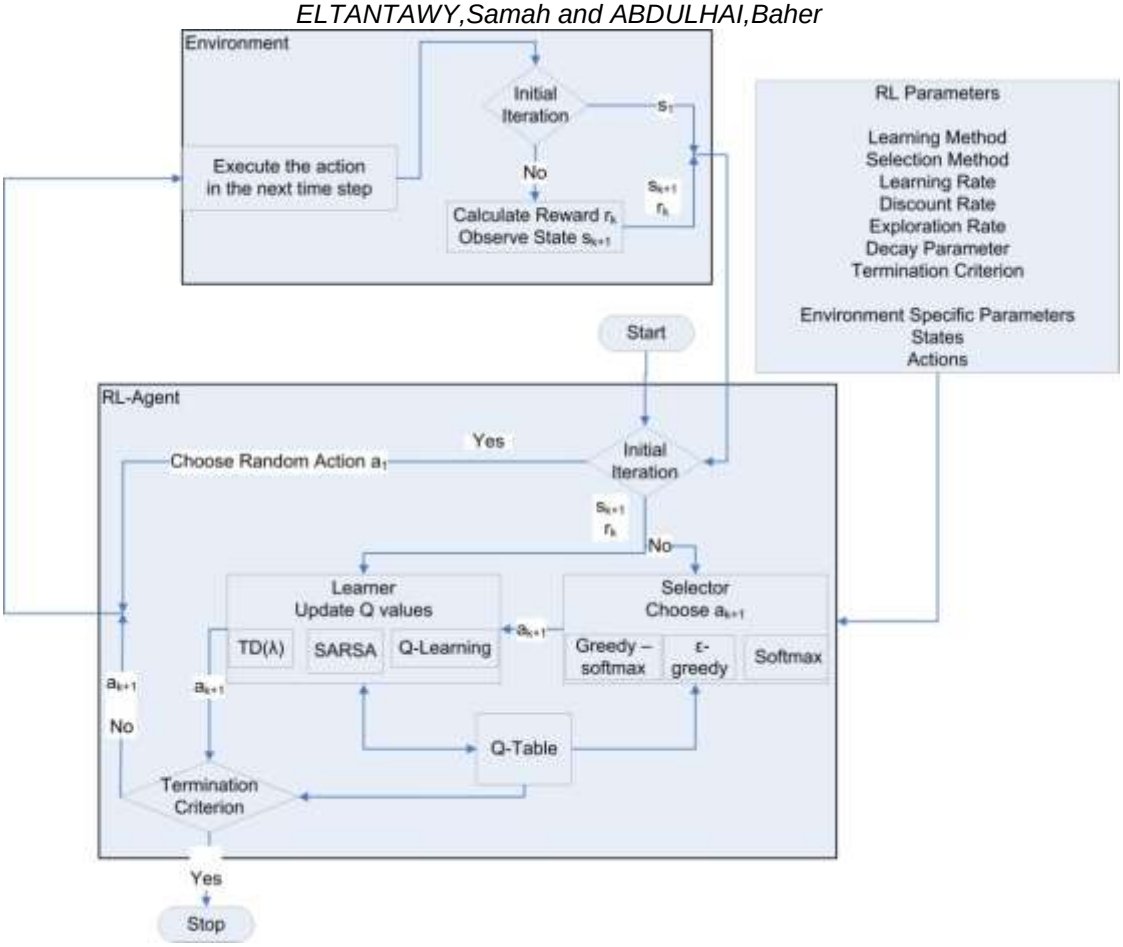



Figure 4 Agent-Environment Interaction

The design elements of the RL structure, i.e., the definitions of state, action, reward, and action selection strategies for adaptive traffic signal control are discussed next;

## *State*

The state is represented by a vector of N components that are the maximum queue length associated with each phase lane-group:

$$s_i = \max_{l \in L(i)} q_l \quad \forall\, i \in \{1,2,\dots,N\}$$

where $q_l$ is the number of queued vehicles in lane $l$. The maximum queue length is considered across all lanes $l$ that belong to the group (set) of lanes corresponding to phase i, $L(i)$. The vehicle is considered at a queue if its speed is below 5 kph.

## *Action*

As previously discussed, a variable phasing sequence is sought in this research and therefore the action is the phase that should be in effect next.

$$a = i\,, \qquad i \in \{1,2,\dots,N\}$$

It is worth noting that if the action is the same as the current green phase, then the green time for that phase will be extended by 1 sec (time interval). Otherwise, the green light will be

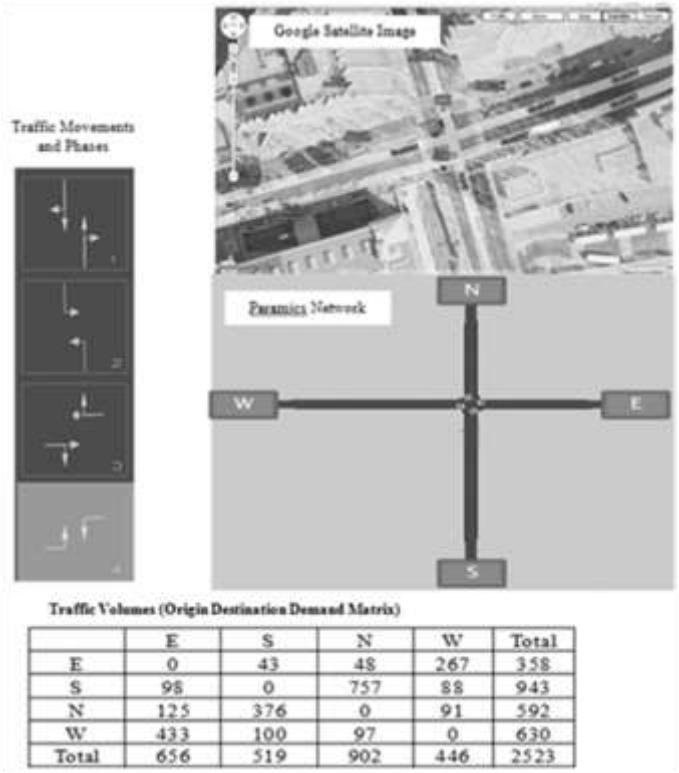
switched to phase *a* after accounting for the yellow, all red, and the minimum green times. Therefore, the decision point varies according to the sequence of actions taken.

### *Reward*

The immediate reward is defined as the reduction (saving) in the total cumulative delay, i.e., the difference between the total cumulative delays of two successive decision points. The total cumulative delay at time t is the summation of the cumulative delay, up to time t, of all the vehicles that are currently in the system. Vehicles leave the system once they clear the stop line at the intersection. If the reward has a positive value, this means that the delay is reduced by this value after executing the selected action. However, a negative reward value indicates that the action results in an increase in the total cumulative delay.
A typical reward function considers the delay experienced by the vehicles between two successive decision points (Abdulhai *et al.*, 2003; Lu *et al.*, 2008). This typical definition however does not consider how long the vehicles were delayed before the last decision point.

## TESTBED INTERSECTION

The agent is tested on a major intersection (4-approaches 3-lanes including an exclusive left turn lane) in Downtown Toronto in the heart of the financial district (Front and Bay Street, Figure 5). This intersection is chosen as an example of an important mutli-phase intersection. The morning rush hour observed traffic demand data for year 2006 (latest available) is attached to the figure in a form of an Origin-Destination (OD) matrix. Each of EB/WB and NB/SB has separate through and left-turn operations, resulting in four phase (movement) combinations as shown in Figure 5. The performance of the widely used fixed time control is used as a bench mark and is compared to the presented Acyclic Q-Learning control agent. The fixed time signal plan is optimized using Webster method (Webster, 1958). Paramics, a microscopic traffic simulator, is used to build the testbed intersection. The RL platform is developed as a standalone program that interacts with Paramics through the Application Programming Interface (API) functions in Paramics. While it is practically infeasible to continue the learning process indefinitely, a stopping criterion is specified to bring the Q-Learning to an end. In this implementation, the learning process is terminated after certain number of *one-hour* simulation runs. The state definition is discretized into four intervals ([0-1), [1-3), [3-6), and [6-10]) which results in 256 states.
Two demand levels are modelled in this experiment; one represents the actual observed demand from field data and the other represents a 50% increase in the demand level. The latter mimics a future forecasting scenario or a severely congested intersection. For each demand level, two demand profiles (i.e. the temporal arrival percentages) are considered; constant (uniform) profile in which the demand is spread uniformly across simulation horizon; and variable profile in which each movement has differently randomized arrival rates around its mean arrival rate (see Figure 6).




Traffic Volumes (Origin Destination Demand Matrix)

| | E | S | N | W | Total |
|---|---|---|---|---|---|
| E | 0 | 43 | 48 | 267 | 358 |
| S | 98 | 0 | 757 | 88 | 943 |
| N | 125 | 376 | 0 | 91 | 592 |
| W | 433 | 100 | 97 | 0 | 630 |
| Total | 656 | 519 | 902 | 446 | 2523 |

Figure 5 Testbed Intersection

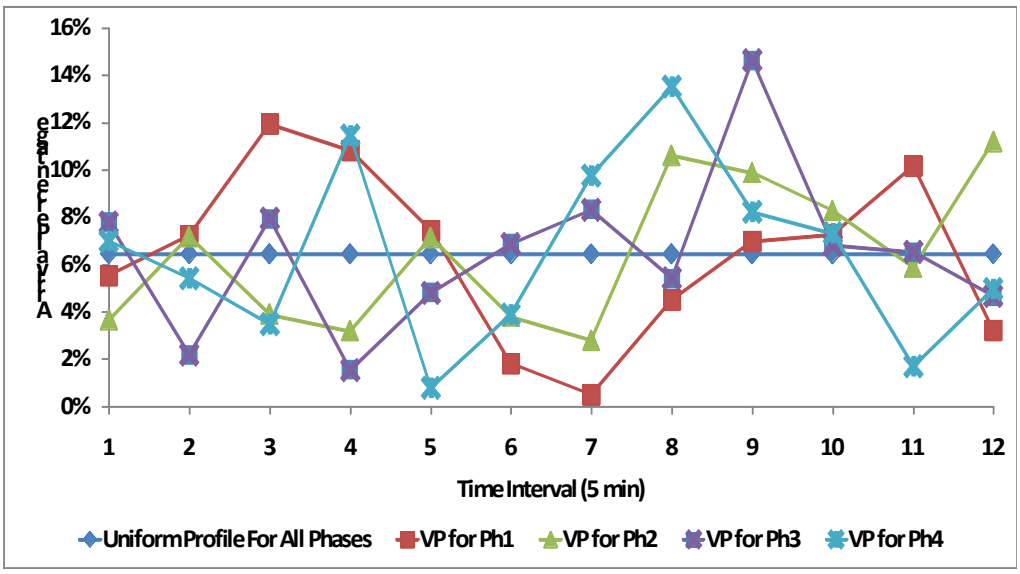



Figure 6 Demand Profile Distribution

# EXPERIMANTAL RESULTS AND ANALYSIS

## Action Selection Strategy

The experiments shown below are conducted using the Q-Learning algorithm to investigate the effect of different action selection strategies in the performance of the adaptive signal control system.

### *$\varepsilon$ -Greedy*

Two profiles of $\varepsilon$ are conducted; constant value of 0.1 and the exponential function discussed previously ($e^{-En}$) where three values of E are tested as show in Figure 3 (0.05, 0.005, and 0.003). In this experiment, the learning process is terminated after 1000 simulation runs. As can be shown in Table 1 , there is no significant difference in the average delay values after the convergence for different $\varepsilon$ - distributions. However, the exponentially decreasing $\varepsilon$ distribution captures more state-action space in terms of the percentage of the state-action pairs visited (from the entire state-action space) during the learning time. For more gradually decreasing exponential $\varepsilon$ (the higher the value of E), it is found that more state-action space is covered. It is worth noting that under the actual demand level, the maximum number of pairs that can visited during the learning time is 76% state-action pairs. This is primarily due to the light demand, so states that represent the situation of having long queue lengths (e.g. between 6 and 10) for more than two approaches do not occur. But for the anticipated higher demand case, almost the entire state-space can be covered with exponential decreasing of $\varepsilon$ and with E=0.003. A possible reason for the insignificant difference of the average delay values at the end of learning (Table 1.) is that the % of state-action pairs that are visited are corresponding to the states that are occurred most of the time. Hence, it can also be concluded that it is sufficient to visit the state-action spaces corresponding to the important recurring. Table 1 includes the results of experiments conducted for the variable demand profiles.

Table 1 Average delay for different $\varepsilon$ - profiles and demand levels

| Demand Level | $\varepsilon$ - Profile | Convergence Time (simulation runs) | % State-action pairs visited | % States visited more than 10% of the time | % States tried with all the actions | Average delay (sec/veh) |
|---|---|---|---|---|---|---|
| Actual | Constant (0.1) | 10 | 46% | 40% | 40% | 7.863 |
| | Exp(-0.05*n) | 45 | 46% | 38% | 40% | 7.842 |
| | Exp(-0.005* n) | 450 | 68% | 47% | 64% | 7.722 |
| | Exp(-0.003* n) | 2300 | 76% | 50% | 74% | 7.619 |
| High | Constant (0.1) | 20 | 81% | 75% | 76% | 12.822 |
| | Exp(-0.05*n) | 45 | 86% | 74% | 77% | 12.815 |
| | Exp(-0.005* n) | 450 | 94% | 85% | 93% | 12.745 |
| | Exp(-0.003* n) | 770 | 99% | 89% | 98% | 12.736 |

### *$\varepsilon$ -Greedy, softmax, and $\varepsilon$ -Softmax*

Since the Q-Learning agent can explore the entire state-action space under the 50%increase in demand with variable profile, three action selections methods are compared under this level and profile of demand; $\varepsilon$ -Greedy, softmax, and $\varepsilon$ -Softmax. Since the function $\epsilon = e^{-En}$ with E=0.005 resulted in a good compromise between the % of state-action pairs visited, the average delay value, and the convergence speed, this function is utilized for $\varepsilon$ and $\tau$. As shown in Figure 7, after converges, softmax and $\varepsilon$ -Greedy algorithms resulted in the same average delay. Although softmax explores the actions more intelligently, a possible reason for this conclusion is that it exploits less aggressively than $\varepsilon$ -greedy. Also by using the softmax selection method, the exploration rate does not only depend on the value of $\tau$ (like $\varepsilon$ -greedy method), but also depends on the magnitude of the Q-Values, so it is harder to find a parameter setting for $\tau$ as cited in Sutton and Barto (1998) "*Most people find it*

*easier to set the $\varepsilon$ parameter with confidence; setting $\tau$ requires knowledge of the likely action values and of powers of e*." $\varepsilon$ -Softmax, on the other hand, synergizes the effect of both methods and converges to a lower value of average delay with a higher convergence speed.

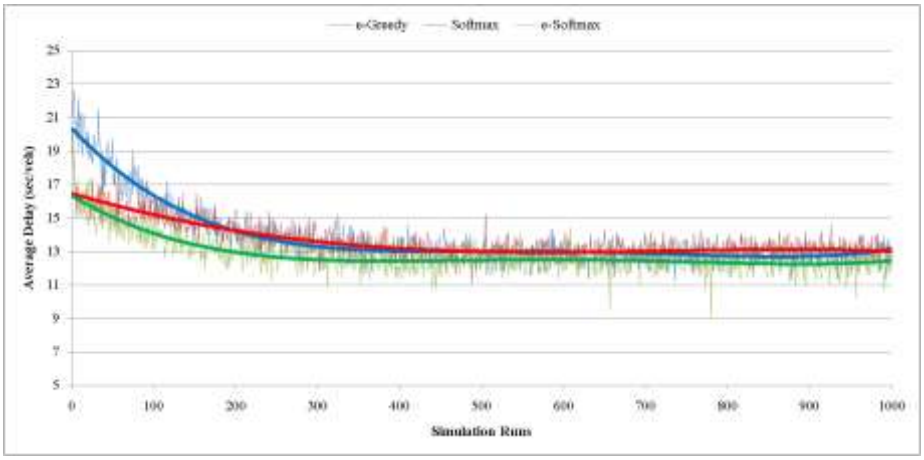



Figure 7 Average delay for different action selection methods

## Q-Learning Vs. Pretimed

The optimal state-action values (Q values) obtained by the Q-Learning using the ϵ-Softmax strategy ,where both ϵ and $\tau$ equals exp(-0.005* n), are used to test the following scenarios; actual demand level, 50% increase in demand and 100% increase in demand. Also, each demand level is tested under uniform profile and variable profile. The average cumulative delay per vehicle (average total delay), the average cumulative delay per queued vehicle (average stopped delay), and the total vehicle delay are calculated for the above test scenarios. The average of 10 one-hour simulation replications is reported in Table 2. The results are compared with the pretimed signal plan that is designed for the actual demand case. Using Webster's method the pretimed plan is defined as follows: - cycle length = 65 sec , - green times = 22 sec, 6 sec,14 sec, and 5 sec for phase 1, 2, 3 and 4, respectively.

Table 2 Average Total Delay and Average Stopped Delay per Vehicle for Different Demand Levels and Profiles

| | | **Q-Learning** | | **Pretimed** | | **% Saving** | |
|---|---|---|---|---|---|---|---|
| | **Level** / **Profile** | Unifrom | Variable | Unifrom | Variable | Unifrom | Variable |
| **Total Vehicle Delay (sec)** | Actual | 19126 | 20789 | 28896 | 34623 | 34 | 40 |
| | + 50% | 38427 | 48289 | 48903 | 65017 | 21 | 26 |
| | +100% | 80718 | 86269 | 94263 | 95891 | 14 | 10 |
| **Average Total Delay (sec/veh)** | Actual | 8 | 8 | 12 | 14 | 34 | 40 |
| | + 50% | 11 | 14 | 14 | 19 | 22 | 26 |
| | +100% | 19 | 20 | 22 | 23 | 14 | 13 |
| **Average Stopped Delay (sec/veh)** | Actual | 12 | 13 | 19 | 21 | 37 | 41 |
| | + 50% | 15 | 17 | 21 | 26 | 29 | 33 |
| | +100% | 21 | 22 | 29 | 29 | 26 | 26 |

As shown in Table 2, regardless of the demand level or demand profile, the Q-Learning-based acyclic method with variable phasing sequence control consistently outperforms the pretimed signal control. As intuitively anticipated, the intersection delay increased proportionally to the demand level. The effectiveness of the Q-Learning-based signal control is more vivid in the variable profile case compared to the uniform profile. As shown in Table 2, in the higher demand levels, however, the effectiveness of the Q-Learning-based signal control gradually diminishes. A possible reason is that the green time is almost always max-out due to the increase of the demand level. Adaptive control in general works better than pretimed control if demand is highly stochastic. Under very heavy traffic conditions, traffic flows are steadily heavy from all approaches and hence pretimed control is less inefficient. Also, it can be shown from the results that the agent performs well even under 100% more demand which the agent was not trained for. This reflects the ability of the RL agent (when the entire state-action space is visited) to operate under abnormal traffic conditions (e.g. unusual surges in demand after concerts, games, or during emergency evacuation).

Figure 8 represents an example for the green time allocated for phases 1 using the acyclic variable phasing sequence Q-Learning approach compared to the pretimed signal plan for the observed demand with variable profile scenario. It is clearly shown from Figure 8 that the acyclic approach green splits adapt to the demand profile. On the other hand, the fixed plan assigns a constant green time for each phase based on the flow per hour regardless of the demand variability within that hour.

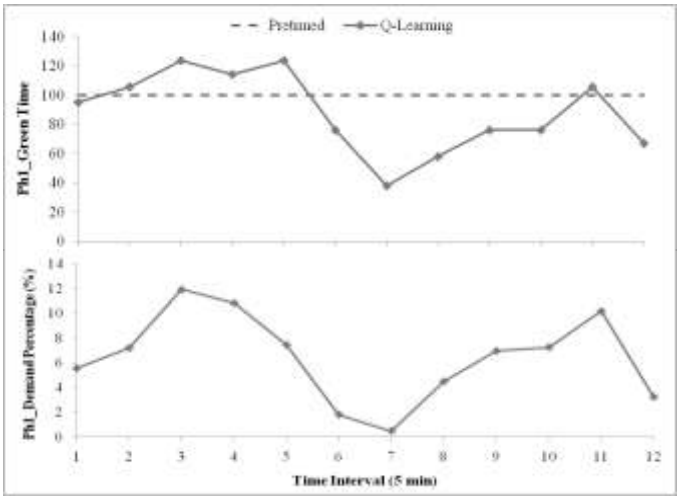


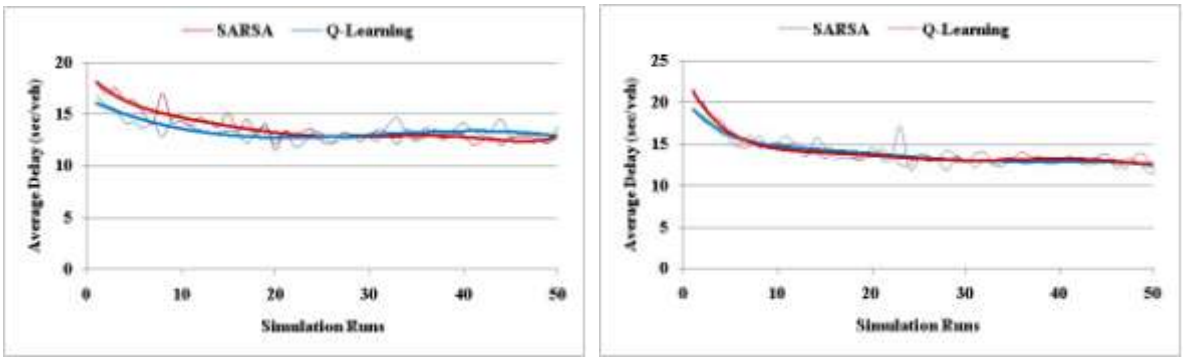

Figure 8 Allocated Green Time and Demand Arrival Percentages

## TD Learning Method

### *Q-Learning Vs. SARSA*

Q-Learning and SARSA are tested with $\varepsilon$-greedy action ($\varepsilon$ =0.1 ) selection method. In this experiment, the learning process is terminated after 50 simulation runs. As shown in Figure 9 (a), although both Q-Learning and SARSA converge to the same average delay value, the Q-Learning converges to the optimal average delay faster than SARSA. Early convergence of the Q-Learning agent is attributed to the fact that the agent is trained using an off-policy

method in which the agent learns actions that are not necessarily performed during the learning process which enabled early convergence. In other words, the agent is gaining useful experience even while exploring actions that may later turn out to be non-optimal. Although Q-Learning converges in less number of iterations, it takes longer time per iteration due to the search for the greedy action, especially during early learning stages compared to SARSA.




(a) (b)

Figure 9 Average Delay using SARSA and Q-Learning (a) constant $\varepsilon$ ,and (b) exponentially decreasing $\varepsilon$

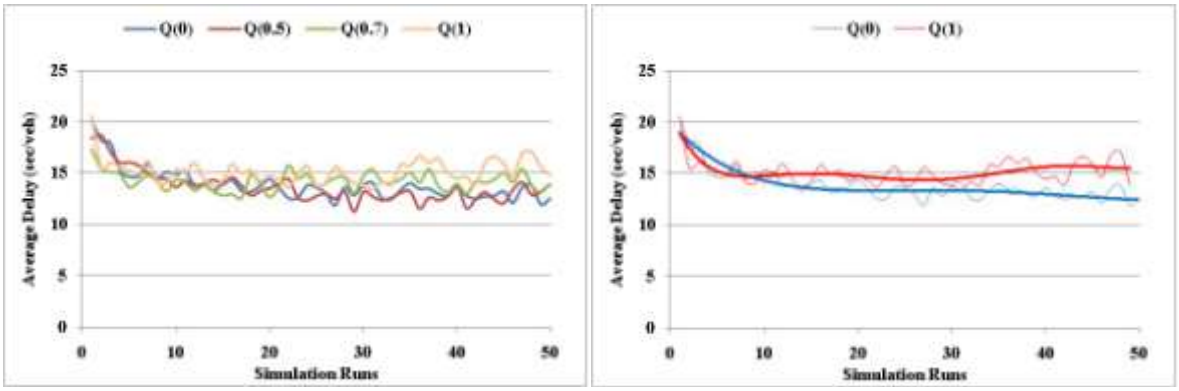
Q-Learning and SARSA are tested also with $\varepsilon$ -Softmax where E=0.05. As shown in Figure 9 (b) , Q-Learning and SARSA converges to the optimal policy with the same convergence speed. A possible explanation for this observation is the asymptotically decreasing of $\varepsilon$ and $\tau$ as cited in (Sutton and Barto, 1998) *"Of course, if $\varepsilon$ were gradually reduced, then both methods would asymptotically converge to the optimal policy"*.

## *Eligibility traces*

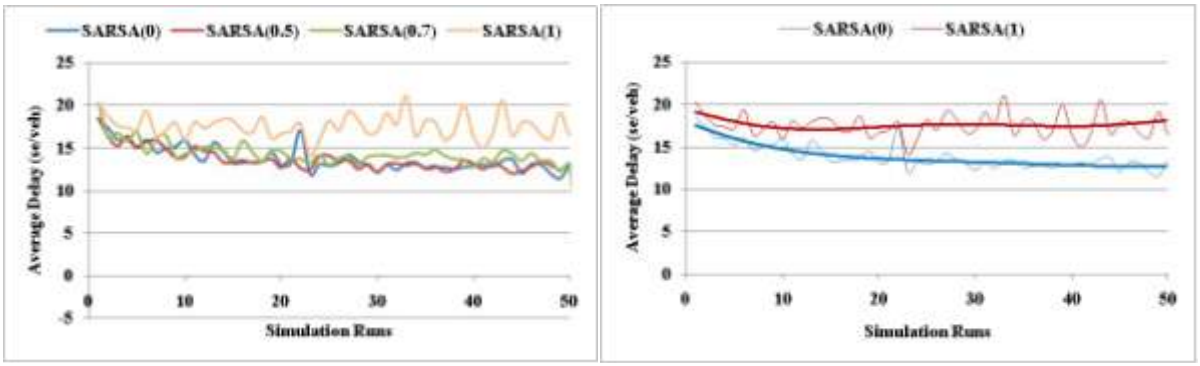
Eligibility traces allocates more credit for states visited recently than to states visited longer ago which results in faster learning as shown in Figure 10. The higher the value of λ the faster the convergence occurs but to a higher average delay value. This possibly means that for high values of λ, the algorithm converges to a local minimum. This observation is more vivid in Watkins's Q(λ).In general, Watkins's Q(λ) has better performance than SARSA(λ) for any value of λ.

The performance of the TD(λ) algorithms with low values of λ is almost identical which could be due to the nature of the problem since the control task, in general, is a continuing task (i.e., not an episodic) with discounted reward in which looking ahead to future steps is less important compared to the episodic task. To the best of the authors' knowledge, no study in the literature compared the effect of TD methods for such problems; however, most of the studies in the literature (de Queiroz *et al.*, 2006; Karaoglu *et al.*, 2006; Leng *et al.*, 2009) compared the algorithms for episodic tasks in which there is an initial state and the target is to reach to a final state with undiscounted reward.
Also, the significant effect of TD(λ) appears particularly when rewards are delayed by many steps (Kaelbling *et al.*, 1996) which could not be adopted in ourcase since the reward definition is the difference in the cumulative delay between two successive decision points.

It should be noted that the online performance of TD(λ) where λ >0 ,in general, is slower than TD(0) because the agent updates the eligibility traces , in each iteration, for each state-action pair.




(a)




(b)

Figure 10 Average Delay using (a) Watkins's Q(λ) , and (b) SARSA (λ)

# CONCLUSIONS

In this paper, the literature on the use of reinforcement learning for traffic control is reviewed and the gaps are highlighted. An RL-based acyclic variable-phasing signal control system is presented using several Temporal Difference (TD) RL methods. The agent's goal is to minimize the cumulative delay at the intersection. The different control systems are tested on a simulation of a real multi-phase intersection in downtown Toronto. The RL methods are compared to the Webster-based pretimed signal control as a bench mark. The results showed that the Q-Learning approach consistently outperforms the pretimed signal approach, in terms of the savings in the cumulative delay per vehicle regardless of the demand level up to 40%. The effectiveness of the acyclic Q-learning approach is more vivid in case of variable demand profiles compared to uniform profile cases; which reflects its adaptability to fluctuation in traffic conditions. The study shows that the combination of $\varepsilon$-greedy and softmax action selection methods achieves the best performance. Also, it is found that Q-Learning performs at least as SARSA for different action selection strategies. On the other hand, eligibility traces provide an important mechanism that can improve the learning speed for TD(λ). However,according to the proposed design of RL-based adaptive

signal control problem, the higher the value of λ the faster the convergence occurs but to a higher average delay value.

# REFERENCES

Abdulhai, B. and L. Kattan (2003). "Reinforcement Learning: Introduction to Theory and Potential for Transport Applications." *Canadian Journal of Civil Engineering* 30(6): 981-991.

Abdulhai, B., R. Pringle and G. J. Karakoulas (2003). "Reinforcement Learning for True Adaptive Traffic Signal Control." *Journal of Transportation Engineering* 129(3): 278-285.

Bertsekas, D. P. (1976). *Dynamic Programming and Stochastic Control*, Academic Pr.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control*, Athena Scientific

Bingham, E. (2001). "Reinforcement Learning in Neurofuzzy Traffic Signal Control." *European Journal of Operational Research* 131(2): 232-241.

De Oliveira, D., A. L. C. Bazzan, B. C. da Silva, E. W. Basso, L. Nunes, R. Rossetti, E. de Oliveira, R. da Silva and L. Lamb (2006). Reinforcement Learning-Based Control of Traffic Lights in Non-Stationary Environments: A Case Study in a Microscopic Simulator. *Proc. of EUMAS06, pp.31-42, 2006, Citeseer*.

de Queiroz, M. S., R. C. de Berr do and A. de Pádua Braga (2006). "Reinforcement Learning of a Simple Control Task Using the Spike Response Model." *Neurocomputing* 70(1-3): 14-20.

Jacob, C. (2005). Optimal, Integrated and Adaptive Traffic Corridor Control: A Machine Learning Approach. *Department of Civil Engineering*. Toronto, University of Toronto.

Jang, J. S. R., C. T. Sun and E. Mizutani (1997). *Neuro-Fuzzy and Soft Computing*, Prentice Hall Upper Saddle River.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). "Reinforcement Learning: A Survey." *Journal of Artificial Intelligence* 4(1): 237-285.

Karaoglu, N., F. Van de Maele, F. Temmermans and P. Vanhee (2006). "Learning in Multi-Agent Systems: An Analysis of Learning Algorithms for the Gridworld Problem."

Leng, J., C. Fyfe and L. C. Jain (2009). "Experimental Analysis on Sarsa ( ) and Q ( ) with Different Eligibility Traces Strategies." *Journal of Intelligent and Fuzzy Systems* 20(1): 73-82.

Lu, S., X. Liu and S. Dai (2008). Incremental Multistep Q-Learning for Adaptive Traffic Signal Control Based on Delay Minimization Strategy. *Proceedings of the 7th World Congress on Intelligent Control and Automation June 25 - 27, 2008, Chongqing, China*.

McShane, W. R., R. P. Roess and E. S. Prassas (1998). *Traffic Engineering*, Prentice Hall.

Robbins, H. and S. Monro (1951). "A Stochastic Approximation Method." *The Annals of Mathematical Statistics* 22(3): 400-407.

Sutton, R. S. and A. G. Barto (1998). "Introduction to Reinforcement Learning." *MIT Press, Cambridge Mass.*

Thorpe, T. (1997). "Vehicle Traffic Light Control Using Sarsa." *Master's Project Rep., Computer Science Department, Colorado State University, Fort Collins, Colorado*.

Wahba, M. M. (2008). Milatras: Microsimulation Learning-Based Approach to Transit Assignment *Department of Civil Engineering*. Toronto, University of Toronto.

Watkins, C. and P. Dayan (1992). "Q-Learning." *Machine learning* 8(3): 279-292.
Webster, F. V. (1958). *Traffic Signal Settings*, HMSO.