MACHINE LEARNING FOR OPTIMAL TRAFFIC CORRIDOR CONTROL: THEORETICAL FRAMEWORK AND EARLY RESULTS

Celine Jacob[*], Baher Abdulhai
ITS Centre and Test bed, Dept. of Civil Eng., Univ. of Toronto
35 St. George St., #105, Toronto, Ontario, M5S 1A4 Canada
jacobc@ecf.utoronto.ca, baher@ecf.utoronto.ca

Abstract

Advancements in Intelligent Transportation Systems and communication technology have the potential to considerably reduce delay and congestion through an array of network-wide traffic control and management strategies. Perhaps two of the most promising control tools for freeway corridors are traffic-responsive ramp metering and/or dynamic traffic diversion possibly using variable message signs (VMS). Technically, the use of these control methods separately might limit their potential usefulness. Therefore, integrated corridor control using ramp metering and VMS diversion simultaneously might be synergetic and beneficial. Administratively, freeways and adjacent arterials often fall under different jurisdictional authorities, for instance States or Provinces in North America administer freeways while cities/municipalities handle surface streets. Common lack of coordination amongst those authorities due to lack of means for information exchange and/or possible bureaucratic 'institutional grid-lock' could hinder the full potential of technically-possible integrated control. Therefore, fully automating corridor control could alleviate this problem. Motivated by the above, the aim of the research approach presented in this paper is to develop a self-learning adaptive integrated freeway-arterial corridor control for both recurring and non-recurring congestion. The paper introduces the use of Reinforcement Learning, an Artificial Intelligence method for machine learning, to provide a single integrated optimal control agent for a freeway-arterial corridor. Reinforcement Learning is an approach whereby the control agent directly learns optimal strategies via feedback reward signals from its environment. A simple but powerful reinforcement learning method known as Q-Learning is used. Early simulation results on travel time savings from a case study focused on a single corridor in Toronto are very encouraging and discussed in the paper.

Keywords: Traffic management; Integrated traffic control; Adaptive traffic control; Intelligent transportation systems; Artificial intelligence; Machine learning; Reinforcement learning; Q-learning
Topic Area: C Planning, Operation, Management and Control

1. Introduction

This project explores the application of Reinforcement Learning technique, a Machine Learning approach, to provide integrated automatic control using VMS and ramp metering on a freeway/ arterial corridor for both recurring and non-recurring congestion. In response to the recognized needs for an integrated, area-wide transportation management system, several research efforts have been carried out, both in the US and in other countries. Notable among

[*] Corresponding Author, Tel: (416) 946-7662, Fax: (416) 978-5054

them are the ones based on automatic control theory by Papageorgiou et al. (*1*, *2*) and knowledge-based system (KBS) by Ritchie et al. (*3, 4*). These approaches have their strengths and weaknesses. Control theory approaches are technically sound but computationally very demanding, which limits the size of the network that can be controlled. In addition, they rely on the presence of an accurate model of the controlled environment or the 'plant', the existence of which for traffic networks is debatable. Knowledge-based systems, although they are handy tools for packaging valuable human expertise, suffer from numerous limitations. They are based on heuristics and ad-hoc procedures for traffic control and hence optimality is either not sought or not guaranteed. Moreover, KBS are rigid as they, once developed, never learn further and hence do not automatically adapt to the temporal evolution of the controlled environment. This paper presents a new and potentially effective methodology for integrated traffic corridor control using Reinforcement Learning (RL) that has several advantages relative to the above methods. RL is closely related to Dynamic Programming, a common control theory tool, and hence inherits its appealing theoretical soundness, while offering several advantages. One of the main advantages of RL method is that it does not require a model of the environment (although can be used during initial stages of learning) for the control policy to be developed or applied. Moreover, RL is much less computationally demanding compared to dynamic programming. In addition RL models continuously learn from interacting with the environment and hence, continuously adapt to any changes in its environment over time, without human intervention.

Reinforcement Learning is an approach whereby the control agent directly learns to map sensed system states to optimal actions. In this research, a simple but powerful Reinforcement Learning- the Q-Learning approach of Watkins *(5, 6)* is selected to address the stochastic nature of traffic situation. Experiments with Q-learning agent have been done in the past with favorable results. Bradtke *(7)*, Littman and Boyan *(8)*, Crites and Barto *(9)*, Abdulhai et al. *(10, 11)* all presented encouraging results in applying reinforcement learning in dynamic fields, including traffic.

In this approach, the agent receives the information about the state of the system (state of traffic network can be estimated from measurements obtained from loop detectors, video cameras and so forth) and selects some action(s) based on certain policy (actions may be to divert traffic, alter on-ramp metering rate, or change the signal settings or possible combinations of all). On taking an action, the agent receives a reward (e.g. delay reduction). Based on the reward, it updates the value of the action for the given state. It has been proven that the Q-learning approach would tend to provide the best actions under all the states after it has come across those states a sufficient number of times. Once these values have been learned, the optimal control strategy from any state is the one with the highest value (known as Q-value).

The Q-learning model developed in this research was trained on a microscopic simulation model of a key corridor in Toronto using Paramics micro-simulation suite (Quadstone 1999) (*12*). Obviously, experimenting with the real corridor is not possible at this stage and hence a simulated virtual replica was used instead. Part of the waterfront network from the greater Toronto area, comprising of the Gardiner Expressway along with the Lake Shore Blvd., as an alternate route, is selected as the study corridor. ITS infrastructure, like variable message signs (VMS), and metered ramps were coded into the network. The corridor model is a comprehensive microscopic stochastic simulation model that has the potential to incorporate several control techniques including the subjects of this research. Individual vehicles are modeled in fine detail for the duration of their entire trip, providing traffic flow, travel time and congestion information, as well as enabling the modeling of the interface between drivers and ITS. For the purpose of traffic control, Paramics provides an application-programming interface (API) that allows advanced users to implement certain logic imposed by a particular control algorithm, such control policies and actions by the machine-learning agent. The agent, as mentioned, is a Q-learning algorithm that 'discovers' the optimal combined use of available control strategies like traffic diversion through VMS (variable message signs) and ramp metering. Once the simulation model is built, it can be used for learning the control decisions under various traffic states and situations, which cover expected real life situations. After the agent has learnt a reasonable initial optimal policy, it can refine this policy in real life after deployment using the same algorithm.

This paper elaborates the above. First, reinforcement learning and the Q- learning are briefly introduced. Control strategies under consideration are then presented. Description of the simulator used to build the Reinforcement-Learning agent is provided next, followed by the implementation details of the control strategies and Q-learning in the simulator. Finally, the learning and testing experiment, as well as results, conclusion and future work are discussed.

2. Reinforcement learning

Reinforcement learning addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its long-term goals. Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state (*13, 14, 15, 16*).

The agent exists in an environment that is described by a set of possible states *S*. It can perform any of a set of possible actions A. Each time it performs an action a_t in some state s_t, the agent receives a real-valued reward r_t that indicates the immediate value of this state-action transition. This produces a sequence of states s_i, actions a_i, and immediate rewards r_i, as shown in Figure 1.

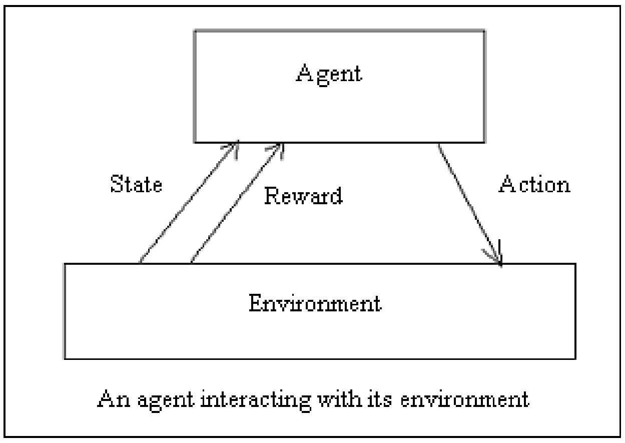

Figure1 Representation of an agent interacting with its environment.

The goal of the agent is to learn a control policy, π: S $\rightarrow$ A, i.e. a mapping from states to actions that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay as in Equation 1 below

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^4 r_4 + \ldots, \text{ where } 0 \leq \gamma < 1 \quad (1)$$

Where r0, r_1, r_2, … are the immediate reward, reward after one time step, two time step etc. and γ is the discount factor which makes the reward earned earlier more valuable than received later.

Q-Learning of Watkins is a simple but effective form of Reinforcement Learning algorithm that does not need a model of its environment and can be used on-line. Q-learning algorithms work by estimating the values of state-action pairs. The value Q(*s*, *a*) is defined to be the expected discounted sum of future payoffs obtained by taking action *a* from state *s* and following an optimal policy thereafter. Once these values have been learned, the optimal action from any state is the one with the highest Q-value.

After being initialized to arbitrary numbers, Q-values are estimated on the basis of experience as follows:

1. From the current state *s*, select an action *a* based on certain policy. This will cause a receipt of an immediate payoff *r*, and arrival at a next state *s'*.
2. Update Q(*s*, *a*) based upon this experience as follows:

Small changes in Q (*s*, *a*) = $r+\Upsilon \text{MAX}_b Q(s', b) - Q(s, a)$,

Where, Υ is the discount factor. The discount factor makes rewards earned earlier more valuable than those received later and its value lies between 0 and 1.

3. Go to 1 with s $\leftarrow$ *s'*.

This algorithm is guaranteed to converge to the correct Q-values with the probability one. If the environment is stationary, and depends on the current state and the action taken in it; i.e. *Markovian*, a lookup table can be used to store the updated Q-values as every state-action pair continues to be visited, and the learning rate is decreased appropriately over time.

Figure 2 shows an illustration of the Q-Learning approach as described by Mitchell (*15*).

Each grid square represents a distinct state, each arrow a distinct action. The immediate reward function, r (*s*, *a*) gives reward 100 for actions entering the goal state G, and zero otherwise. Value of Q (*s*, *a*) follows from r (*s*, *a*), and the discount factor $\gamma = 0.9$. An optimal policy, corresponding to actions with maximal Q values is shown.

For a stochastic environment such as traffic environment, a revised training rule as suggested by Watkins, which is sufficient to assure convergence is shown below.

$$Q_t(s, a) = (1-\alpha)\, Q_{t-1}(s, a) + \alpha(r+\Upsilon \text{MAX}_b Q_{t-1}(s', b) \quad (2)$$

Where $Q_t(s, a)$ is the revised estimate of the Q value in the state s on taking action a, $Q_{t-1}(s, a)$ is the previous estimate of the Q value for the same state action pair, and α is the learning rate in the interval [0,1]. The key idea in this revised rule is that the revisions to Q are made more gradually than the deterministic case. By reducing α at an appropriate rate during training, we can achieve convergence to the correct Q function.

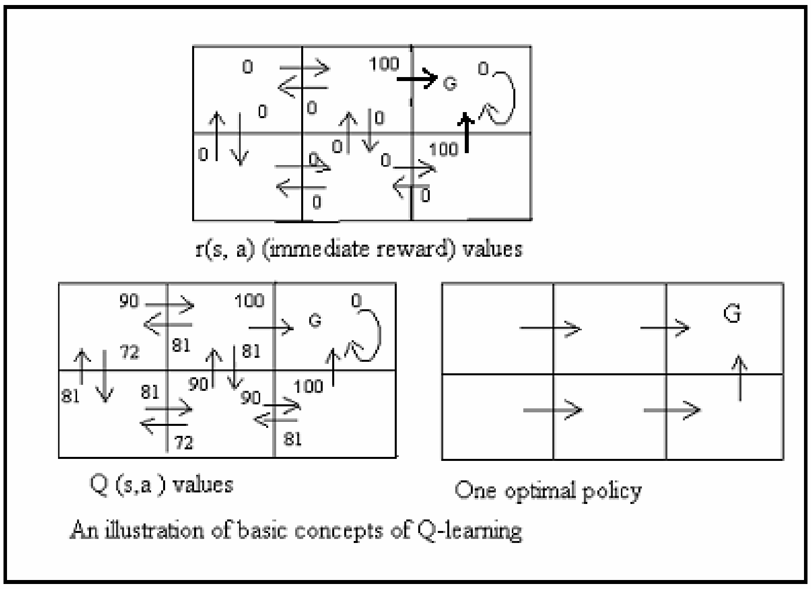
r(s, a) (immediate reward) values
Q (s,a) values
One optimal policy
An illustration of basic concepts of Q-learning
G

Figure 2 An example to illustrate the Q-learning.

2.1. Exploration and exploitation

One important decision in reinforcement learning is exploration vs. exploitation with respect to the policy to be adopted for selecting action in any state. One strategy would be to select the action *a* that maximizes the Q (*s*, *a*), thereby exploiting its current approximation Q'. This is a greedy action and it will lead to exploitation. It prohibits discovery of new potentially better actions though, i.e. limits exploration of new possibilities. Pure exploration on the other hand results in random behavior that lacks learning. Both exploitation and exploration are important and some balance between the two is necessary. A compromise action selection method is the ε- greedy selection, in which a greedy action will be taken most of the time, while a non-greedy exploratory action will be taken every now and then, with a small probability ε. Another method is to vary the action selection probabilities as a graded function of estimated action values. The best/greedy action is still given the highest selection priority, but all other actions are ranked and weighted according to their value estimates. These are called *softmax* action selection rules. The most common method uses a Gibbs, or Boltzmann distribution (*13*).

2.2. Generalization from examples

Estimating the Q value for unseen state action pairs requires generalization from those states that have already been visited. Function approximation techniques like Back-Propagation algorithm or Cerebeller Model Articulation Controller (CMAC) or a simple KNN (K nearest neighbor algorithm) could be used for the generalization.

3. The simulation model

Traffic simulation programs are useful tools for modeling and predicting traffic flow, evaluating traffic management strategies, and designing roadway facilities before field operations. This study employs a microscopic traffic simulation suite - Paramics, to provide the environment needed to implement the Reinforcement Learning algorithm and to train the agent in an offline mode. It is important to emphasize that, at least theoretically, RL agents do not need a model of the environment as they can interact with the real environment and learn directly from this interaction. At the early stages of learning though, an RL agent explores more than exploits and hence it is improper to do this directly with the real system to be controlled. A model can facilitate the early stages of learning until a deployment-worthy behavior is achieved, after which the agent can continue to refine its knowledge in the field.

Paramics, developed by Quadstone Limited, is a suite of high performance tools for microscopic simulation, consisting of Paramics *Modeller, Processor, Analyzer, Programmer, and Monitor*. (*12*).

The Modeller development cycle is summarized in the following seven steps:

1. Create a new Paramics network by embedding template road geometry file.
2. Build a road network by adding nodes, links and zones and coding detailed lane and junction descriptions.
3. Build demand matrices from origin/destination data including fixed demand data, such as bus routes.
4. Assign traffic using an appropriate dynamic assignment technique.
5. Collect and analyzing model results.
6. Calibrate base conditions by comparing model results to observed data.
7. Validate the base model against independent data.

The most valuable feature of Paramics is its Application Programming Interface (API). It enables advanced users to over-ride the default functions in Paramics, such as its car following, merging, vehicle release, route choice models. Users can also add their own APIs to implement many traffic control/management strategies within Paramics. Paramics *Modeller* is used in this research to develop the model of the environment while *Programmer* is used to develop the RL agent.

4. The study network and implementation of integrated control

For the purpose of applying reinforcement technique to the traffic management, a part of the calibrated waterfront corridor from the Greater Toronto area (built in the ITS Centre and Testbed, University of Toronto), comprised of the Gardiner Expressway along with the Lakeshore Boulevard as an alternate route, is selected. Figure 3 shows the small network used in this study.

This portion of the real network has one variable message sign (VMS) upstream of the Gardiner-Lakeshore bifurcation, and one metered on-ramp downstream of the bifurcation, and both were coded in the network using the Paramics *modeller*. Only eastbound traffic was considered for the study. At any traffic state, an integrated control agent can choose any of the following control actions:

- Divert traffic from the Gardiner to the Lakeshore, by means of the VMS.
- Limit access to the freeway by changing the metering rate at the onramp.
- Alter the signal timing plans along the Lakeshore to accommodate changes in traffic patterns due to the above two.

In order to facilitate the above control actions in the Paramics, *Paramics Programmer* is used to provide an Application Programming Interface (API) to the modeller. This research focuses on integrating diversion and ramp metering only and excludes signal control at this stage. This simplifies the investigation. In addition, signals along the Lakeshore are controlled by a SCOOT system adopted by the City of Toronto, which is traffic responsive, and hence changes to signal plans by the agent is not exercised. Backup SCOOT timing plans were obtained from the City and coded in the model. The implementation of re-routing through VMS and the ramp metering using the *API* is explained next.

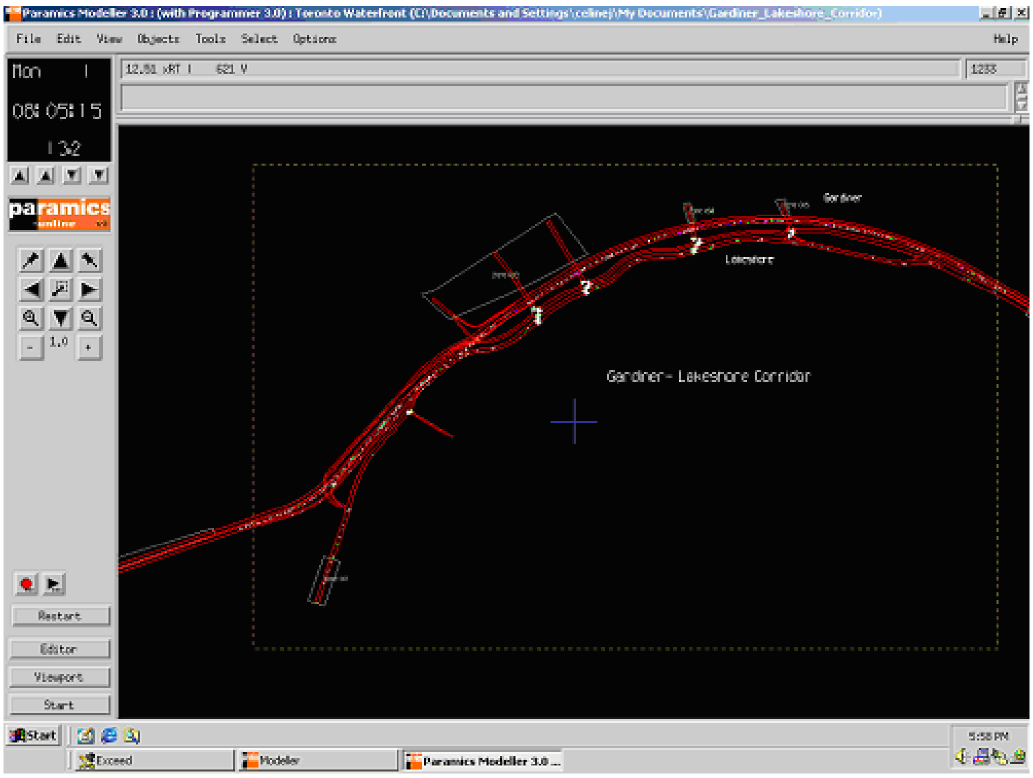
Paramics Modeller 3.0 : (with Programmer 3.0) : Toronto Waterfront (C:\Documents and Settings\celine\My Documents\Gardiner_Lakeshore_Corridor)
File
Edit
View
Objects
Tools
Select
Options
Help
Mon
08:05:15
132
12.81 xRT
621 V
1233
paramics
1.0
Gardiner
Lakeshore
Gardiner- Lakeshore Corridor
Restart
Editor
Viewport
Start
Start
Exceed
Modeller
Paramics Modeller 3.0 ...
5:58 PM

Figure 3 The study network.

4.1. Dynamic rerouting of traffic through VMS

Dynamic rerouting of traffic from the Gardiner Expressway to the Lakeshore parallel arterial is implemented in the field using a single Variable Message Sign. When there is an incident or recurrent congestion on the freeway, drivers are informed by displaying pertinent information on the VMS panel critically located upstream of the Gardiner-Lakeshore bifurcation. As a result, some drivers would divert, more so from familiar drivers as opposed to unfamiliar drivers who would tend to stick to the freeway. In order to simulate this effect in *Paramics*, drivers or Driver/Vehicle Units (DVU's) are advised on network congestion through an artificial VMS sign and provided with details of available alternate route choices to avoid the congested area. VMS logic in the model effects diversion by influencing drivers' route choice behavior while considering the DVU's reaction to the displayed information. Drivers' reaction is affected by several factors related to drivers' understanding, perception, and trust in the VMS beacon message. As suggested in the user guide of Paramics programmer (*12*), factors to consider may include:

- Driver Aggression (an aggressive driver takes chances and reroutes often);
- Driver Awareness (unaware drivers remain on main and more familiar routes);
- Driver Patient (how much delay will a driver tolerate);
- Variations in individual drivers perception of travel cost; and

- Driver trust in VMS message.

All of the above factors could induce a change in a particular route's perceived cost (based on the information supplied by the VMS sign) to individual drivers; ultimately affecting the proportion of driver who will adjust their route choice versus those who will not. In this research, a control point is defined in the network at which the DVU's access the information provided by the VMS sign. This control point is assumed to influence the DVU's heading for a particular range of zones (referred to as the control points 'capture zones'). A DVU approaching a VMS control point will assess its environment and decide whether or not it should follow the information provided and 'accept' VMS guidance.

A fair number of trial runs are required to set the VMS parameters for a desired rerouting effect. Also the actual effect of the VMS could be judged only from the experience gathered from the actual location. Literature has indicated that the response rate of VMS is highly unpredictable and as such it is difficult to decide upon the proper percentage that re-routes and hence difficult to set the VMS parameters accordingly. To keep the investigation manageable, an arbitrary percentage of 75% of vehicles have been assumed to reroute in response to information. This is the best doable until drivers behavior can actually be measured in the future, possibly using a driver simulator. In ongoing research, we also attempt to treat the diversion percentage as a stochastic variable that depends on the above factors.

In the test network, the VMS is provided on the Gardiner, just before the off-ramp, to Lakeshore. Vehicles, upon reaching this control point, will decide whether to follow the default routing or to divert. If the decision is to divert, then it will follow the alternate path specified. This functionality could be achieved by specifying the required code in the *paramics*' control functions vis a vis *vehicle_link action* and *routing_decision.*

The VMS logic in the simulated network is implemented using a Paramics VMS plug in using the API, which takes the following inputs.

- The number of VMS in the network – *vms control points*
- The links on which the VMS is located - *the vms control links*
- The destination zones for the diverted traffic – *capture zones*
- The links along the freeway (Gardiner in this case) route – *delay links*
- The links along the alternative route (Lakeshore in this case) - *alternative links*
- Links other than the above two used for the total delay calculation (e.g. links on the ramp) – *other links*
- The alternate route for the familiar drivers for each control points- *familiar links*
- The alternate route for the unfamiliar drivers for each control points- *unfamiliar links*

The function *vehicle_link action* will be called for every vehicle per time step. Here each vehicle will be tested for the following conditions.

- if it is on the vms control link,
- if the vehicles destination zone is in the capture zones of the vms control point.
- if the vehicle is not under vms control.
- if the delay is greater than the minimum specified for diversion to activate

If the answer is yes, then this particular vehicle will be added to the counter for the vehicles to be diverted and the decision to divert will be taken so as to have 75% diversion on the familiar vehicle's type. Once it is marked for diversion, it will be diverted along the alternate path specified using the function *routing_decision* to specify the next link corresponding to the alternate path. As a result the vehicle will be following the specified path instead of the default route assigned by the paramics.

4.2. Ramp metering

Ramp metering, or ramp control, has been considered to be a very important component of corridor traffic control. Ramp metering is the application of control devices, such as metering signals to limit the number of vehicles entering a freeway (*17, 18*). In this research however, ramp metering is used simultaneously with VMS control using the same agent.

The Q-learning algorithm tries to find the best metering rate under different traffic and congestion scenarios. In the Q-learning process, number of metering rates varying from no metering (signal will always be green, useful when there is no congestion) to a minimum rate of 180veh/hr (1 veh/20sec) has been considered. The metering rate on the ramp signal is dynamically changed based on the action selected by the Q-learning algorithm. This is achieved by implementing the required code in the API function using *call back functions* and *set functions* for the signal. A green time of 2sec. is provided in each cycle with the red time varying from 0 to 18 secs., depending on the action selected by the agent. This will allow 1 car per green. In reality when the metering rate on the ramp is very low, a long queue will be formed on the entrance to the ramp. Drivers approaching the ramp from a distance are likely to respond the visible queues and avoid the ramp. In order to simulate this effect in the Paramics, a detector is provided at the entrance to the ramp and whenever the queue reaches the main line, the vehicles on Lakeshore are directed away from the ramp onto alternate routes along Lakeshore itself.

5. Elements of the Q-learning control agent

5.1. States

At any time, the state of the network is based on direct traffic measurements such as speed, volume and occupancy, which can be obtained from the loop detectors provided on the road. Artificial loop detectors were created in the simulation model at the same locations they exist in the real network. System states in this research are defined on the basis of average speeds on the competing routes, presence or absence of incidents and the status of VMS.

As the aim of this study is to provide control measures in case of recurrent congestion due to lack of capacity or non-recurrent congestion due to incident-related loss of capacity, various incident states must also be considered. An incident scenario is defined by (a) the location of the incident, (b) the incident severity, and (c) the estimated duration of the incident. The route on which the incident took place describes the incident location. The severity can be expressed as the number of lanes blocked by the incident. The incident states corresponding to single lane blocking and double lane blocking was considered. The incident duration is to be described in a slightly different manner. We only describe whether an existing incident is terminating in the current time step or continues to the next time step.

In summary, the continuous variables used to represent system states were discretized into a 6 dimensional matrix as follows:

- Average speed on the Gardiner – 9 discrete intervals.
- Average speed on the Lakeshore - 7 discrete intervals.
- State of the ramp-signal -10 values ranging from a red time of 0-18.sec, with a green time of 2.sec.
- VMS states - value 0 for VMS not activated and 1 for VMS activated.

- Incident states for Gardiner – value 0 for no incident, 1 for incident on the current and next state, 2 for the incident on the current state but not on the next state.
- Incident states for Lakeshore similar to that of the Gardiner.

5.2. Actions

As described earlier, the set of actions available to the agent in any particular state are a combination of VMS and ramp signal settings. The total number of actions possible is defined to be 20. With 10 actions comprising of various ramp metering rates with VMS activated and another 10 actions with ramp metering alone while the VMS is off.

5.3. Reward

The reward can be either positive or negative based on whether a benefit (e.g. delay reduction) or penalty (e.g. delay) is accrued. As described by Papageorgiou (*18*), time spent in the system is minimized if early exit flows are maximized. As a consequence, any set of control measures or control strategy that can manage to increase the early exit flows of the network, will lead to a corresponding decrease of the total time spent. Consider a discrete time representation of traffic variables with discrete time index k= 0, 1, 2, 3… and time interval T. Our aim is to apply control measures so as to minimize the total time spent T_s in the network over a time horizon K, i.e.

$$Ts = T\sum_{k=1}^{k=K} N(k) \qquad (3)$$

Where N (k) is the total number of vehicles in the network at time k.

The aim is to minimize the total time spent by all the vehicles in the system; the reward function to be maximized by the agent during learning is taken as the negative of T_s.

5.4. Function approximation and generalization

Even after sufficient training runs with all the situations possible, still the agent may sometimes come across some new state, which it has never visited before. In such cases, the model has to select a similar nearby state, which has already been visited, and uses it as the basis for generalization. This could be achieved by using function approximation algorithms, such as Neural Network such as the Cerebellar Model Articulation Controller (CMAC) or a simple nearest neighbor (NN) algorithm. In this research to date, function approximation is not included; instead a simple look-up table is implemented to store the Q values for simplicity. However, we are currently experimenting with incorporating a CMAC but results are not yet available.

5.5. Parameters

For this study the following parameters were selected for the Q-learning, based on typical values for the literature:

Exploration threshold for the ε- greedy algorithm = 0.7

Learning rate β =0.2

Discount rate γ = 0.3.

Different values of these parameters however should be experimented with and possibly optimized. Since we at the proof-of-concept stage still, we defer formal optimization of parameter selection to a later stage and use typical default values as a starting point.

6. Implementation

The corridor network is simulated with the peak period demand (8-9am), which is considered the worst scenario, i.e. each simulation was carried out for 1hr. In each simulation run, the first 15 minutes of the simulation is considered as a warm up period and the incident is injected at the end of this time. At the very first simulation run, the Q values for all state action pairs were initialized to zero. The Q- learning agent activated at the occurrence of the incident, receives the state values as described earlier for the 1 minute duration and selects the action based on the ε- greedy policy and implements this action for the next 1minute period. After the end of this 1-minute, the reward (in this case it is a penalty or negative reward) is obtained and the Q-value for the state action pair is updated. This is continued until the end of the simulation representing one iteration. Numerous iterations were required for convergence to occur as presented later in the paper.

7. Measurements of effectiveness (MOEs)

In order to evaluate the feasibility of the approach, the result obtained from the learning agent has been compared with the result obtained from the base scenario - without the ramp and vms control. The tests were carried out after a sufficient number of iterations with various state values. For this purpose, the paramics simulations were run for the same network (with the same incident state) twice, once with the learning agent active and then without using the learning agent. In the test runs, the action selected will be the one with maximum Q-value.

As the objective is to assess the overall effectiveness of the control measures, the characteristics that relate to all vehicles vis a vis the average travel time, total distance, mean speed and stop time, were used for evaluation, i.e.:

- % Reduction in average travel time.
- % Increase in distance.
- % Increase in mean speed.
- % Reduction in stop time.

8. The overall Q-learning plug In

The Q-learning algorithm is coded as a Paramics plug in using the API. The necessary code to obtain the states, taking action according to the policy, implementing the actions like VMS and ramp metering, obtaining the reward and updating Q value, were all codes as various functions. During each run, the state action visited, the number of visits and the updated Q-value for that state-action pair are collected as output. During the test runs, the *MOEs* are collected for evaluation.

9. Results and analysis

Three cases of incident scenarios were considered for the learning of the agent. The cases in the order of learning done are as follows:

Case A – a single-lane-blocking incident on the Gardiner for 10 minutes duration.

Case B – double-lane-blocking incident on the Gardiner for 15 minutes duration.

Case C – a single lane-blocking incident on the Gardiner and a double lane-blocking incident on the Lakeshore for 15 minutes. The numbers of iterations per scenario are:

Iteration	1- 1549	case A
Iteration	1550- 3129	case B
Iteration	3130- 4560	case C
Iteration	4561- 5338	case B

An independent learning with case C for 2856 iterations was also performed to verify the effect of changing incident attributes on the learning. As mentioned earlier, each iteration is an hour long run and the incident with the specified duration was injected after 15secs from the start of the simulation. Iterations were repeated again and again until the number of visits to each of the states was sufficiently large so as to achieve convergence of the Q values.

Figure 4, shows an example plot of the changes in one of the Q values from an initial value of zero as the number of visits to that particular state-action pair increase. It can be seen that the Q-values stabilize after some 30 visits to that state-action pair.

Tables 1 to 3 summarize the MOEs of the control agent compared to the no control case under different incident conditions. All three cases have shown considerable improvement (values shown in brackets) under the reinforcement-learning agent. The extent of improvement is of course more pronounced for severer incidents on the freeway (Case B) where the control measures become more effective. For case A, the incident effect was minor and for case C, both routes had incidents. As a result, improvements obtained were smaller compared to Case B.

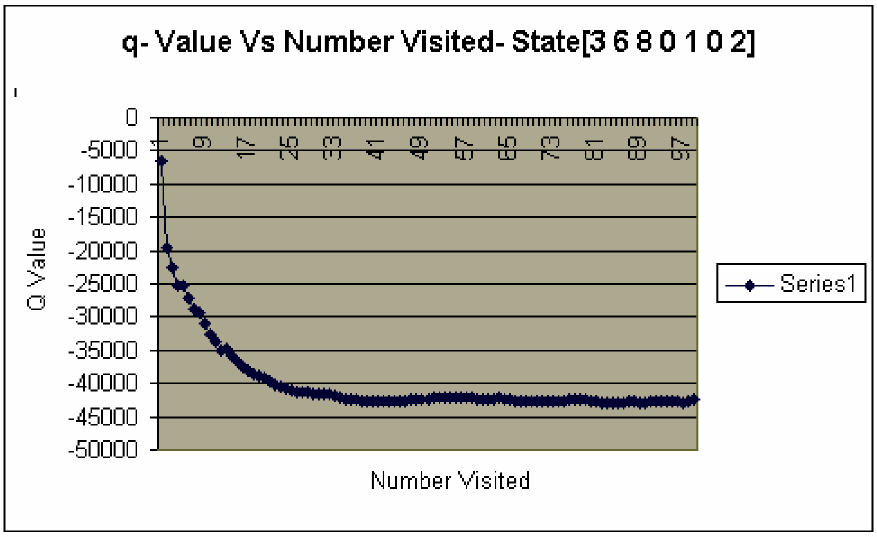

Figure 4 Q-value Vs Number of visits to the state after 5338 iterations (2356 iterations with Case B).

TABLE 1 Results of Simulation with Case A

	Mean travel time in sec.	Total vehicle distance in m	Mean speed in mps	Stop time in sec.
Without Q-Learning	653	38439313	28.5	3877
With Q-Learning	562	41722062	33.5	3246
% Improvement	(13.9%)	(8.5%)	(17.5%)	(16.3%)

Table 2 Results of Simulation with Case B

	Without Q--Learning	With Q-Learning with 1150 iterations	With Q-Learning with 1580 iterations	With Q-Learning with 2030 iterations	With Q-Learning with 2357 iterations
Mean travel time in sec.	826	730.2 (11.6%)	680 (17.7%)	685 (17.1%)	688.3 (16.7%)
Total veh. distance	33422294	37165343 (11.2%)	38014065 (13.7%)	37927553 (13.5%)	38144350 (14.1%)
Mean speed (mps)	22.0	25.3 (15%)	27.2 (23.6%)	27.2 (23.6%)	26.7 (21.4%)
Stop time in sec.	5495	4449.6 (19.0%)	3920 (28.6%)	4225 (23.1%)	4072.8 (25.8%)

Table 3 Results of Simulation with Case C

	Without Q-Learning	With Q learning with 1430 iterations	With Q learning with 1930 iterations	With Q learning with 2450 iterations
Mean travel time in sec.	647	548.8 (15.2%)	569.3 (12.05%)	568.2 (12.2%)
Total veh. distance	38355591	41224561 (7.5%)	40169416 (4.7%)	40852001 (6.5%)
Mean speed (mps)	28.4	34.2 (20.42%)	32.7 (10.5%)	33.2 (16.9%)
Stop time in sec.	3832.4	3212 (16.2%)	3422.7 (10.7%)	3234.3 (15.6%)

In the above simulations with Case C, the learning was done as a continuation of Case A and Case B. In order to test the effect of incident locations on the result, a fresh simulation was run with Case C alone. The result of this is shown in Table 4. Comparing the numbers to those in table 3, mean travel times after approximately 2000 iterations in both cases were almost similar at 569.3 for the first case and 568.1 for the second case. Similarly, mean travel times after approximately 2500 iterations were 568 and 562 respectively. Other MOEs are also similar. This strongly indicates that separate learning for different incident scenarios is *not* required.

Table 4 Result of Independent Simulation Runs with Case C

	Without Q learning	With Q learning with 511 iterations	With Q learning with 1511 iterations	With Q learning with 2011 iterations	With Q learning with 2856 iterations
Mean travel time in sec.	647	571.8 (11.6%)	582.7 (9.9%)	568.1 (12.1%)	562 (13.1%)
Total veh. distance	38355591	41136266 (7.2%)	39969401 (4.2%)	41051432 (7.0%)	41181035 (7.4%)
Mean speed (mps)	28.4	32.9 (15.8%)	32.3 (13.7%)	32.9 (15.8%)	33.2 (16.9%)
Stop time in sec.	3832.4	3259.6 (14.9%)	3445.9 (10.1%)	3316.6 (13.4%)	3318.9 (13.4%)

10. Provisions for real time implementation

The results of the study indicate that reinforcement learning is promising for multi-jurisdictional freeway-arterial corridor control, especially under incidents. Once the agent is trained on a simulated replica of the real network, it should be ready for field where the selection and implementation of actions can be executed in real time. The measurements for state estimation, i.e. the average speed on each route, as well as VMS status, and ramp status are readily available at typical control centres. Incident information can be obtained either from an Automatic Incident Detection system or provided manually by the operator. The total time spent by all vehicles in the system can be estimated from volume counts from loop detectors at all entry and exit points. Once deployed, the agents should be able to continue learning to refine the Q-values further with real situations, and hence operates and adapts maintenance-free.

11. Conclusions and future research

In this study, the application of Reinforcement Learning or more specifically Q-Learning was examined for the integrated control of freeway/arterial traffic corridor using VMS and ramp metering. The agent was built around a microscopic model of a corridor in Toronto using *Paramics*. The Gardiner/ Lakeshore part of the Waterfront network near the Humber River was used as the study area. ITS components such as VMS and ramp metering were added into the network using the *Paramics Modeller*. The major part of the work involved the development of the Q-learning agent as a set of API functions.

On applying the Q- learning agent and based on selected MOEs, all the three incident scenarios analyzed have shown considerable improvement (about 9-17% for case A, 14-25% for Case B, and 8-17% for Case C) over the base scenario with no control. The evaluation results demonstrate the potential for the Q-learning approach in providing area-wide traffic control, particularly under incidents. Ongoing research by the authors is examining an expanded network involving more corridors, VMS, ramp meters, as well as signal controls, for which multiple reinforcement learning agents are used.

The experience gained from this study will be utilized for future research. The following refinements and extensions will be considered:

- Inclusion of function approximation for generalizing the Q values and also to deal with the enormous state action space required for a larger network.
- Refinement of the parameters used in the Q-learning and Paramics, using optimization techniques.
- Application to a larger network with more number of VMS, ramps, and intersections, possibly involving a multi-agent reinforcement learning approach.
- Testing other exploration strategies.
- Gaining further insights into driver response to information. The proportion of the traffic, which will be diverted by the VMS, depends on driver factors, such as VMS trust, familiarity, aggression, awareness, patience, perceived cost and so forth. This can be achieved using a driver simulator.

References

Papageorgiou, M., 1995. An integrated control approach for traffic corridors. Transportation Research-part C 3, 19-30.

Kotsialos, A., Papageorgiou, M., Mangeas, M. and Haj-Salem, H., 2002. Coordinated and integrated control of motorway networks via non-linear optimal control. Transportation Research-part C 10, 65-84.

Logi, F. and Ritchie, S. G., 2001. Development and evaluation of a knowledge-based system for traffic congestion management and control. Transportation Research- part C 9, 433-459.

Logi, F., Rindt, C.R., McNally, M.G., Ritchie, S.G., 2001. TRICEPS-CARTESIUS: An ATMS testbed for evaluation of inter-jurisdictional traffic management strategies. Transportation Research Record 1748, TRB, National Research Council, Washington, D.C., pp. 125-131.

Watkins, C. J. C. H., 1989 Learning from Delayed Rewards. Ph.D. Thesis. King's College, University of Cambridge, Cambridge, UK.

Watkins, C. J. C. H., and P. Dayan, 1992. Q-Learning. Machine Learning, 8, 279-292.

S.J Bradtke, M.O. Duff, 1995. Reinforcement Learning Methods for Continuous-Time Markov decision problems. In: G.Tesauro, D.S. Touretzky and T.K.Leen, eds., Advances in Neural Information Processing Systems, Vol.8, MIT Press, Cambridge, MA.

Littman, M., Boyan, J., 1993. A Distributed Reinforcement Learning Scheme for Network Routing. A technical Report CMU-CS-93-165, School of Computer Science, Carnegie Mellon University.

Crites, R.H., Barto, A.G., 1996. Improving Elevator Performance Using Reinforcement Learning. Advances in Neural Information Processing Systems 8. MIT Press, Cambridge MA.

Abdulhai, B., Pringle, R., Karakoulas, G.J., 2001. Reinforcement Learning for ITS: Introduction and a Case Study on Adaptive Traffic Signal Control. Paper No. 01-2694.Transportation Research Board, 80th Annual Meeting.

Abdulhai, B., Pringle, R., and Karakoulas, G.J., 2003. Reinforcement Learning for True Adaptive Traffic Signal Control. ASCE Journal of Transportation Engineering, 29 (3) 278-285.

Quadstone Ltd., 2000. Modeller and Programmer V3.0 User Guide and Manual, Edinburgh, UK

Sutton, R.S., and Barto, A.G., 1998. Reinforcement Learning-An Introduction. MIT Press, Cambridge, Massachusetts.

Abdulhai, B., and Kattan, L., 2003. Reinforcement Learning Crystallized: Introduction to Theory and Potential for Transport Applications, Accepted, CSCE Journal (forthcoming).

Mitchell, T., 1997. Machine Learning. McGraw-Hill, New York.

Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4, 237-285.

Zhang, M., Kim, T., Nie, X., Jin, W., Chu, D.L., 2001. Will Recker. Evaluation of On-ramp Control Algorithms, California PATH Research Report. UCB-ITS-PRR-2001-36.

Papageorgiou, M. and Kotsialos, A., 2001. Freeway Ramp Metering: An overview, Dynamic Systems and Simulation Laboratory, IEEE Intelligent Transportation Systems Conference Proceedings.