BUNCH BREAKER: A REINFORCEMENT LEARNING MULTI-AGENT FOR TRANSIT OPERATIONS CONTROL

Amer S. Shalaby, Kenny Ling
Department of Civil Engineering University of Toronto 35 St. George Street Toronto, Ontario, Canada, M5S 1A4, Telephone: (416) 978-5907 Fax: (416) 978-5054
E-mails: amer@ecf.utoronto.ca, kennyling@rogers.com

Abstract

Transit agencies often implement operations control strategies so as to mitigate the effects of transit vehicle bunching along transit routes. Typical control strategies include vehicle holding, expressing and short turning, which are usually implemented through manual means *via* field supervisors or central control centers. The objective of this study is to automate vehicle bunching control by means of multiple Reinforcement Learning (RL) agents that act on a series of successive signalized intersections. The multiple Reinforcement Learning agents developed in this study include the "bunch-splitting", "holding" and "expressing" agents which work cooperatively to break up a vehicle bunch if one is detected and to build a reasonable headway between the paired vehicles. Various elements of the RL agents such as the action set, the reward and the state space are set up under a traffic signal control context. Simulation results indicated that these multiple Reinforcement Learning agents could split up a streetcar bunch and prevent it from forming again with a high success rate.

Keywords: Reinforcement learning; Microsimulation; Modelling; Headway control; Public transport and intermodality
Topic Area: B1 Public Transport and Intermodality

1. Background and objectives

Bunching of transit vehicles is an undesirable operational problem that plagues many urban transit routes, particularly ones with high service frequencies (5-minute headways or less). It is usually caused by excessive delays experienced by a transit vehicle at a transit stop, with such delay increasing as the transit vehicle travels further downstream because of higher-than-average dwell times. At the same time, the next transit vehicle experiences less dwell times at such transit stops (due to less-than-average passengers waiting) and eventually it catches up with the previous transit vehicle, creating a "bunch" of the 2 vehicles which travel as such thereafter. Contributing factors that give rise to the bunching phenomenon include excessive traffic congestion, variable passenger demand, idiosyncratic driver behavior, transit vehicle failure, driver/passenger emergency, poor weather conditions, etc. Vehicle bunching causes excessive wait times for passengers, unreliable transit operations, uneven loading of transit vehicles, inefficient utilization of rolling stock, etc. As such, transit agencies often implement control strategies, usually through manual means *via* field supervisors or central control centers, in order to avoid the occurrence of vehicle bunching and/or to mitigate its effects if it occurs. Typical control strategies include bus holding and short turning (Abkowitz and Tozzi, 1987; Wilson et al., 1992).

In a previous study, Ling and Shalaby (2003) developed an innovative approach to headway control of a high frequency transit route (30 streetcars per hour) by means of adaptive traffic signal control. A Reinforcement Learning agent was developed to modify in real time the traffic signal timing so as to minimize headway variability. For example, if a streetcar is detected upstream of an agent-controlled intersection and it is arriving earlier than

the schedule headway of 120 seconds from the previous transit vehicle, the agent determines the optimal signal time setting to "delay" the second transit vehicle at that intersection such that the schedule headway is restored. The single agent was shown to be effective in reducing headway variability and reducing the incidents of bunching. However, it cannot deal with a bunch if it already occurred. This paper presents an extension to this work to deal with this particular problem. The objective is to develop multiple Reinforcement Learning agents to act on a series of successive signalized intersections to break up a vehicle bunch if one is detected and to build a reasonable headway between the paired vehicles.

2. Theory and methodology

Ling and Shalaby (2003) provide an introductory exposition of the Reinforcement Learning method, which is repeated here for the benefit of the reader. Reinforcement Learning is one of the most active research areas in the Machine Learning and Artificial Intelligence community (Sutton, 1998). This method has been applied quite extensively in Psychology, Neuroscience, Optimal Control and Artificial Intelligence. Under this approach, the learner or decision maker is called the agent. The agent interacts with its environment at each of a sequence of discrete time steps, t = 0, 1,2, 3… (see Figure 1) At each time step ***t***, the agent receives some representation of the environment state ***S(t)***. And according to the learned policy of the agent, a particular action ***A(t)*** is selected out of all actions available under that state. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken under those states. The policy is simply the "brain" of a Reinforcement Learning agent. One time step later, in part as a consequence of its action, the agent receives a numerical reward ***R(t)*** and finds itself in a new state ***S(t+1)***. The purpose or goal of the agent is formalized in terms of a special reward signal passing from the environment to the agent.

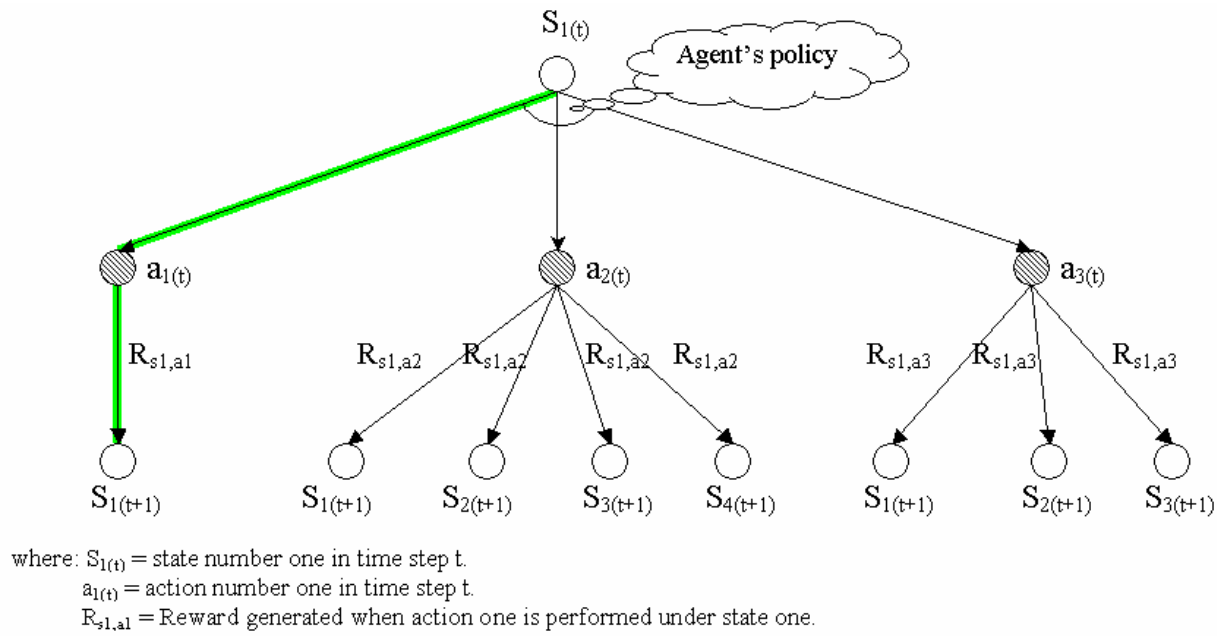

Figure 1. Relationship between states, actions and rewards

The goal of the agent can be achieved by maximizing the expected return, R_t, which is defined as some specific function of the reward sequence. For the simplest case, the return is the sum of the rewards:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \qquad \text{Eqn. (1)}$$

Note that the magnitude of the future rewards depends on how the environment responds according to the future actions of the agent. Obviously, the magnitude of future rewards is fairly unpredictable and has a smaller relevance to the current action $\boldsymbol{a_t}$. As such, it is more appropriate to have the agent focus more on the immediate reward than future rewards. A discount factor is introduced into the above equation to take this concept into account:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \qquad \text{Eqn. (2)}$$

The discount rate, $\boldsymbol{\gamma}$, is a parameter that ranges from 0 to 1. For instance, a reward received $\boldsymbol{k}$ time steps in the future is worth only $\boldsymbol{\gamma^{k-1}}$ times what it would be worth if it were received immediately. If γ equals 0, the agent will only be concerned about the immediate reward and totally ignores future rewards. In such a case, the agent is said to be "myopic". And as γ approaches 1, the agent will give more weighting to the future rewards and becomes more "farsighted".

A state value function determines the long-term desirability of a specific state after taking into account the states that are likely to follow, and the rewards available in those states. Equation (3) expresses a recursive relationship between the value of a state, ***V(s)*** and the values of its successor states.

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s') \right] \qquad \text{Eqn. (3)}$$

$$P_{ss'}^{a} = transition\ probabilities = Prob\left\{ s_{t+1} = s' \mid s_t = s, a_t = a \right\}$$

where: $R_{ss'}^{a} = expected\ value\ of\ the\ next\ reward = E\left\{ r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s' \right\}$

$$\pi = policy\ of\ the\ agent$$

Solving a Reinforcement Learning problem means, roughly, finding a policy that achieves a lot of reward over the long run. A policy $\boldsymbol{\pi}$ is defined to be better than or equal to a policy $\boldsymbol{\pi'}$ if its expected return is greater than or equal to that of $\boldsymbol{\pi'}$ for all states. In other words, $\boldsymbol{\pi} \geq \boldsymbol{\pi'}$ if and only if $V^{\pi}(s) \geq V^{\pi'}(s)$. The policy that is better than all other policies is defined as the optimal policy π^*. So, the optimal state value function, V^* is defined as:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \text{Eqn (4)}$$

Notice that in Eqn (3), the determination of state value requires a complete set of state transition probabilities and expected rewards. Under a deterministic environment, the transition probabilities and expected reward values are fixed and can be obtained through a perfect model of the environment. However, under a stochastic environment like a transportation system, these values are variable and cannot be estimated. Thus, instead of estimating the state value function directly, we adopt the approach of Q-learning, which is having the agent to learn the action value, Q(s,a) directly. The action value, Q(s,a) is the reward received immediately upon executing action a from state s, plus the value of following the optimal policy thereafter.

$$Q(s,a) = r(s,a) + \gamma V^*(s') \qquad \text{Eqn (5)}$$

Now, the key problem is finding a reliable way to estimate values for *Q*, given only a sequence of immediate rewards *r* spread out over time. This can be accomplished through iterative approximation. Notice the close relationship between *Q* and V^*,

$$V^*(s) = \max_{a'} Q(s, a') \quad \text{Eqn (6)}$$

Rewriting Eqn (5) gives rise to,

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(s', a') \quad \text{Eqn (7)}$$

The right hand side of the equation is a *target* presumed to indicate a desirable direction in which to move, though it may be noisy. The iterative approximation of *Q(s, a)* can be illustrated by the following equation,

$$NewEstimate \leftarrow OldEstimate + StepSize\ [\ Target - OldEstimate] \quad \text{Eqn (8)}$$

Substituting Eqn (7) into the target of Eqn (8) gives,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\, r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad \text{Eqn (9)}$$

The Q-value, or Q(s, a), is normally referred to as the action value function. The discount factor γ controls the relative contribution of future rewards (obtained from the next state) to immediate rewards. As γ approaches 1, the future rewards will be given greater emphasis and the agent is more farsighted. The difference between the target and old estimates is in fact an error term in the estimate. Adjustment of the old estimate is implemented by taking a step toward the "Target". In fact, the value of step size, α is equivalent to the learning rate of the Q-learning agent. A large step size allows the agent to identify the action that happens to be the optimal one with a shorter training time. However, the tradeoff is that the state-action pair Q-values will fluctuate a lot and may encounter problems in reaching convergence.

Initially, the Reinforcement Learning agent has no knowledge on what actions to perform under different situations that help to achieve the goal. There is no teacher or supervisor to guide the behavior of the autonomous agent. When the agent is placed in a particular situation, it must carry out the best action according to its own experience.

The environment then responds to the agent's action by returning a reward or penalty and brings the agent to a new state. Through out this kind of interaction with the environment, the agent gradually learns how to differentiate between "good" and "bad" actions under different situations. The repetitive process of carrying out expected good actions and achieving positive feedback under a particular situation acts like a "positive reinforcement". The agent can use its experience to improve its performance over time.

A major feature of Reinforcement Learning is the trade-off between exploration and exploitation. In order to accumulate a lot of rewards in the long run, the agent must select an action that is estimated to have the highest "value" among the possible actions. In this sense, the agent exploits what it already knows in order to obtain more rewards. Note that the "value" of the selected action may not actually be the highest one since it is only an estimation made by the agent based on its current knowledge. Therefore, it is beneficial for the agent to try occasionally other possible actions so that previously missed out actions which might be good ones are explored. In other words, the agent is exploring in the hope of making a better action selection in the future.

3. The testbed

The 504 King streetcar route corridor in Downtown Toronto was selected as the testbed for this study. This is one of the most heavily used public transit routes in Toronto (nearly 50,000 passengers per day). The route is operated with a very high frequency of 30 streetcars per hour and consists of a total of 114 streetcar stops. About 100 of these stops are located

10-15 meters before the signalized intersection (i.e. near-sided transit stops). The end-to-end trip time is approximately 60 minutes.

Currently, unconditional priority for streetcars is implemented at most signalized intersections along the King Streetcar Route. Once detected by an upstream detector, each streetcar is granted signal priority (green extension or red truncation) until it clears the intersection (i.e. passes a downstream detector) or a maximum extension is reached. Because most transit stops are near-sided, streetcars have to stop for loading/unloading passengers before clearing the intersection. The streetcars travel along the left lane of King St., and when stopped for loading/unloading passengers, traffic in the right lane is required to stop by law.

Although transit speed has improved following implementation of the above priority scheme, the high frequency route (30 streetcars per hour) has been experiencing poor reliability in terms of deviation from the planned headway. In addition, the magnitude of headway deviation tends to increase as streetcars travel further downstream. Thus, the problem of "streetcar bunching" still exists.

The microscopic traffic simulation software Paramics (Quadstone, 2000) was used to model the entire King St. corridor including its crossroads (Lee, 2001). The geometric layout of the road network was built according to the Toronto Centerline data. The traffic demand volumes were encoded into the network using data from the Transportation Tomorrow Survey conducted in 1996 in the Greater Toronto Area. The developed network has been tested in a previous research project (Shalaby et al., 2003) and results indicate its high accuracy in replicating the real traffic and transit operations taking into account the interaction between traffic signals, streetcars and the general traffic. A plug-in program that contains the source code of the Reinforcement Learning algorithm was imported into Paramics as an Application Programming Interface (API) to test the performance of the proposed algorithm.

4. Application of multiple q-learning agents

In a previous study (Ling & Shalaby, 2003) the single Q-learning agent demonstrated its capability of reducing streetcar's headway deviation. The improvement in the streetcar's reliability performance is most obvious for cases with mild degree of headway deviation. However, streetcars with extremely high headway deviation cannot be handled properly by implementing the agent on one single signalized intersection. Consider, for example, two streetcars with headway of 10-20 seconds between them arriving together at the Q-agent controlled intersection. Ideally, the agent should build up the gap between these two "bunching" streetcars and recover to the scheduled headway. Unfortunately, this can hardly be done properly by making use of only one Q-learning agent controlled intersection, because the phase length is under constraints of certain practical maximum and minimum values.

On the other hand, if the Q-learning agent can be implemented over several consecutive signalized intersections, we can expect that the gap between the "streetcars bunch" builds up slowly as it travels through these intersections until a reasonable headway can be achieved. This study develops three different types of Q-learning agents, namely the "bunch-splitting" agent, the "holding" agent and the "expressing" agent. Each agent has its own state space, action scheme, action domain, reward scheme and local level objective. Despite the fact that these multiple agents have slightly different objectives, the combinatorial effect of those actions carried out by the multiple agents ensures the ultimate goal of mitigating the bunching problem.

Selection of location

Previous literature [Rossetti et al.,1998; Strathman et al.,1993] has pointed out that the fluctuating demand at transit stops is one of the major factors that contribute to the deterioration of performance and reliability for a transit route. It also implies that the smaller

the number of transit stops in between consecutive Q-learning agent controlled signalized intersections, the smaller the uncertainties (randomness factor) being introduced into the problem. Therefore, multiple consecutive signalized intersections with a minimum number of transit stops in between them is more suitable to be selected as a testbed for the multiple Q-learning agents. In this study, the number of signalized intersections required for the multi Q-learning agent is arbitrarily set to three. The optimal number of signalized intersection to implement the algorithms will be of great interest in future research.

According to the simulated King St route, the following three signalized intersections were found to match the above selection criteria:

- King & Bay
- King & Yonge
- King & Church

Therefore, the above intersections were selected as the testbed for the multiple Q-learning agents of this study. Since the agents applied to these signalized intersections work together to achieve a common goal, the intersections of Bay, Yonge and Church will henceforth be named as the 1st, 2nd and 3rd intersection, respectively.

The state space

It is crucial for the state variables to capture all the necessary information relevant to the situation and be useful in aiding the agents to make the best decision. In developing he single Q-Learning agent (Ling and Shalaby, 2003), it was found that a state space based on the following pieces of information was quite effective: the current status of the signal phase and the transit vehicle's current headway. As such, the same state space definition was used for each of the three agents of this study.

It is noteworthy that the training time of the agent increases exponentially with the increase in the total number of states. For the agent to learn at an acceptable rate, a discrete state space is adopted as opposed to a continuous one. The number of data types fed to the agent determines the dimensions of the state space matrix. For the "bunch-splitting" agent, the signal cycle is divided into 20 intervals and the gap time among the bunch is categorized into 6 ranges. A two-dimensional matrix (a total of 120 states) is formed as shown in Table 1.

Table 1. State space scheme of the "Bunch-Splitting" agent

		EW phase (elapsed time in the current phase)									
		0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	24-29	29-35
		Green	Green	Green	Green	Green	Green	Green	Green	Green	Amber & All-red
Gap time within the bunch	0-4 s	1	7	13	19	25	31	37	43	49	55
	4-8 s	2	8	14	20	26	32	38	44	50	56
	8-12 s	3	9	15	21	27	33	39	45	51	57
	12-16 s	4	10	16	22	28	34	40	46	52	58
	16-20 s	5	11	17	23	29	35	41	47	53	59
	20+ s	6	12	18	24	30	36	42	48	54	60
		NS phase (elapsed time in the current phase)									
		35-38	38-41	41-44	44-47	47-50	50-53	53-56	56-59	59-64	64-70
		Green	Green	Green	Green	Green	Green	Green	Green	Green	Amber & All-red
Gap time within the bunch	0-4 s	61	67	73	79	85	91	97	103	109	115
	4-8 s	62	68	74	80	86	92	98	104	110	116
	8-12 s	63	69	75	81	87	93	99	105	111	117
	12-16 s	64	70	76	82	88	94	100	106	112	118
	16-20 s	65	71	77	83	89	95	101	107	113	119
	20+ s	66	72	78	84	90	96	102	108	114	120

For example, if the signal is running at 17 seconds into the East-West Green phase and the gap time among the bunch is 15 seconds at the moment when the streetcar bunch is detected within the zone of detection, the algorithm will inform the agent a state number of 34. A flowchart that clearly shows the flow of information in the state number determination

process is depicted in Figure 2. For the "holding" agent and the "expressing" agent, the state space numbering scheme also utilize the elapsed time of the current signal phase and the headway of the current streetcar. The only difference is that the headway information is categorized into increments of 7 seconds (from 0-63+ seconds) and 12 seconds (from 0-120+ seconds) for the "holding" and "expressing" agents, respectively.

These agents have the capability of learning the passenger service time at the near-sided transit stop located within the detection zone. Under such a high frequency transit route, passengers are assumed to arrive at the transit stop randomly and uniformly. For instance, a streetcar with a long headway (late run) can expect to have a longer service time at the stop and *vice versa*. After an adequate number of revisits under a certain state, the agent starts to learn the expected service time at the stop under a particular headway. Even if there is a change in passenger demand profile, the agent is still able to capture its effect and readjust itself because of the life-time updating process of Q-values.

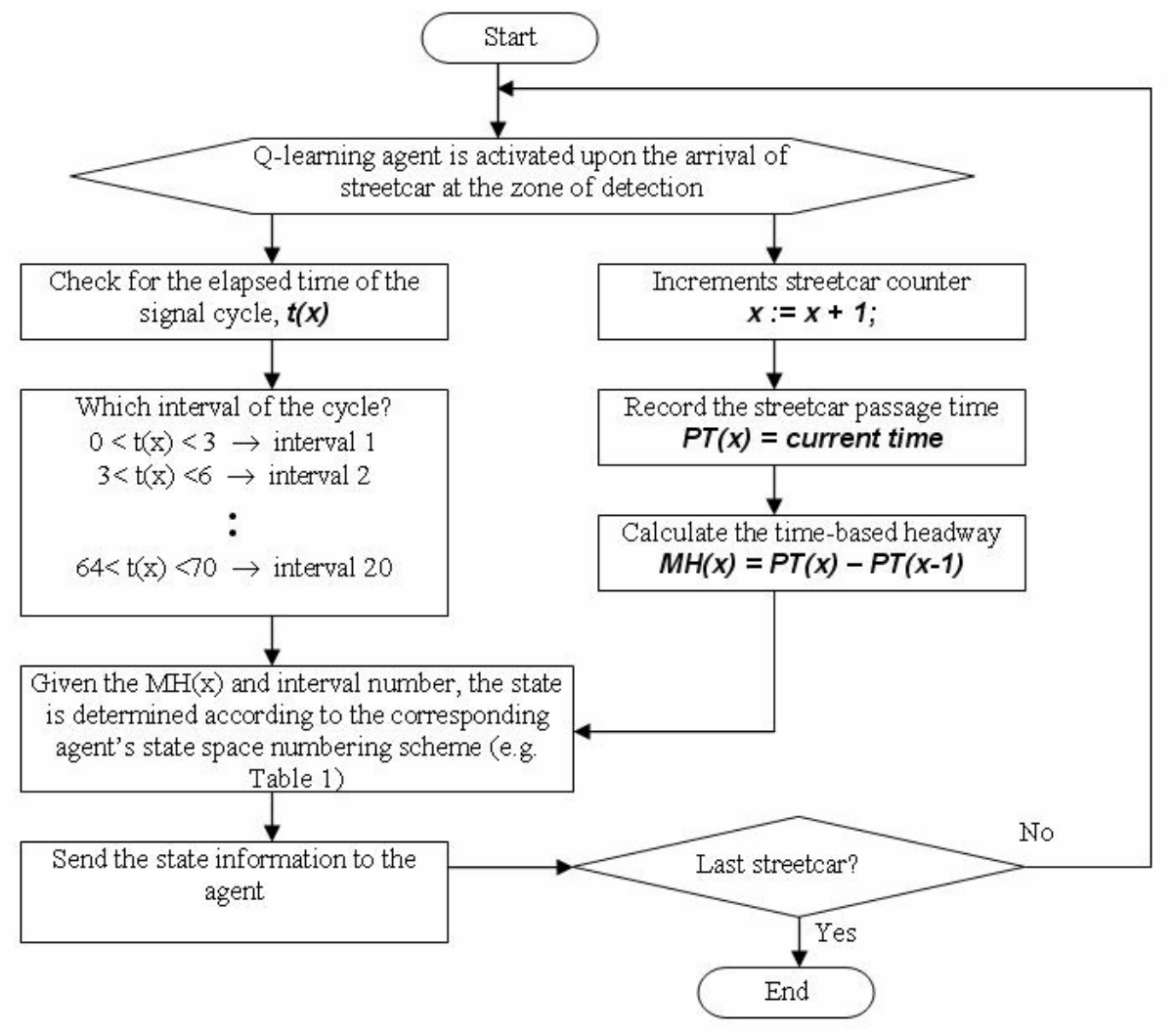

Figure 2. State space information flow diagram

The actions

Under the context of Reinforcement Learning, the agent is supposed to be trained until it can select consistently the best action among all available ones under a specific state most of the time. The actions performed should be able to alter the state of the environment and a reward is granted after the action is carried out.

A signal cycle can roughly be separated into the transit corridor split (EW-bound) and the crossroad split (NS-bound). Each split can further be divided into two portions, which are displayed green time and clearance time (amber & all-red). The agent is allowed to modify the displayed green time portion of any split but not the clearance time. The clearance time will remain fixed at the existing value for traffic safety reasons.

In this study, the agent will only be activated whenever a streetcar is detected within the detection zone. Once the agent is activated, it is expected to modify the current signal timing accordingly. If the agent is activated while at the displayed green time portion of the cycle, the choice of action under different states is simply a duration period (in units of seconds) applied to the currently running phase from this moment on (Figure 3). On the other hand, if it is activated while at the clearance time portion, the duration period determined will be applied to the following phase.

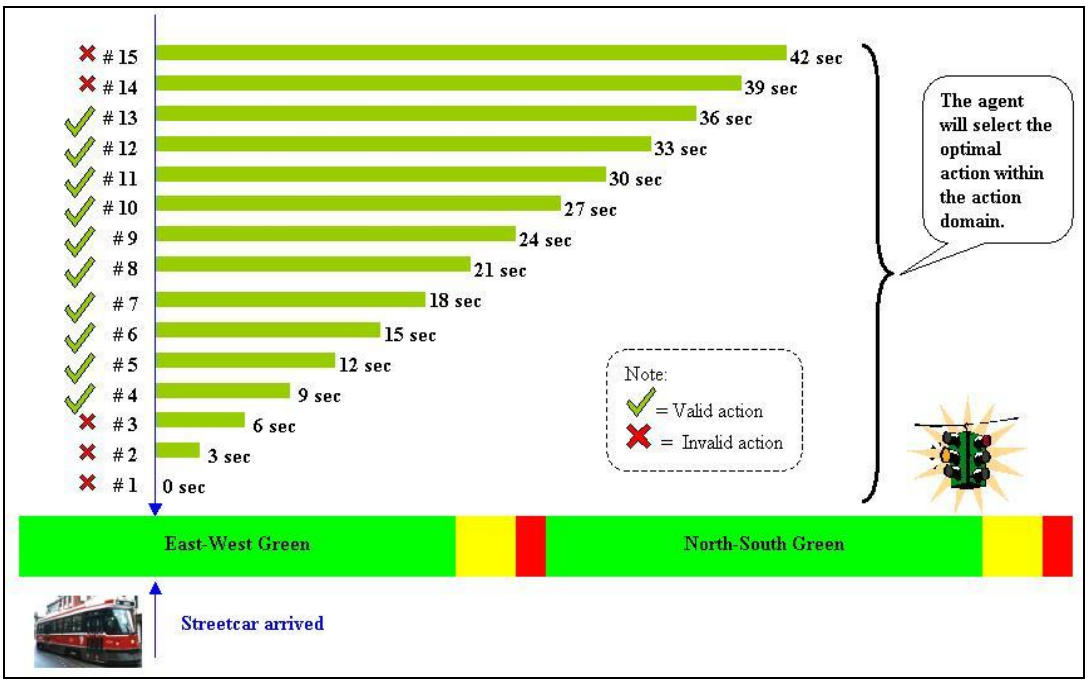

Figure 3. Action set of the reinforcement learning agent

Practical signal plan constraints & action domain

Although the Q-learning agent can learn the optimal policy to minimize the headway deviation, it may run into the risk of giving a too long or too short green time & red time for the transit corridor. Therefore, we have to impose some constraints on the choices of actions by specifying the maximum and minimum green time. Specifically, the domain of actions should be different under different elapsed times of the current signal phase running. Although each agent has a slightly different action domain, the basic idea is the same. The agent is set up in such a way that the proposed action will always check against the actions domain table before a final decision is made. For instance, the “tick” or “cross” mark besides each action in Figure 3 informs the agent whether the action selected is valid or not at that instant.

The rewards

As mentioned earlier, the agent always attempts to maximize its reward. If the agent is desired to perform some operation, a corresponding reward has to be provided in such a way that in maximizing it the agent will achieve the desired goal. The reward of the "bunch-splitting" agent is defined as the gap time between this pair of bunching streetcars measured at the downstream detector of the corresponding signalized intersection. As mentioned earlier, the agent achieves its goal by accumulating a lot of rewards in the long run. Thus, if the agent can continuously select those actions which lead to a large gap time within the streetcar bunch after the modified signal plan is carried out, the objective of splitting the bunch is achieved.

The reward of the "expressing" agent is the negative of the time required to travel through the road section between the upstream and downstream detectors. Since the agent always favors actions that give rise to a good reward (in this case, a less negative number), the actions which lead to the shortest travel time has the largest chance of being selected. By the same token, the reward for the "holding" agent can almost be defined as the opposite of the "expressing" agent. The reward is equivalent to the time required to travel the road segment between the upstream and downstream detectors. Through maximizing the reward, the "holding" agent is likely to select the actions which can effectively stop the 2nd streetcar from catching up with the first one again.

These agents will learn from interaction with the environment and find the best way to accomplish the task. It is thus critical that the rewards be set up to indicate what we want to accomplish.

The Q-values updating process

The Q-value updating process is shown in a flowchart (Figure 4). In this study, an **ε**-greedy policy is employed to guide the behavior of the agent. Generally speaking, the agent selects an action that has maximum estimated Q-values most of the time, but with a probability **ε** it selects an action at random. Suppose **ε** is 0.15, then the agent will select an action with the highest estimated Q-value (under the encountered state) with an 85% chance. In other words, at around 15 % of the time, the agent will select an action randomly.

This **ε**-greedy policy is highly important during the training process of the Q-learning agent. In the initial stages of training, we want the agent to explore the environment and try out as many different actions as possible. Thus, the **ε** value should be given a higher value. Throughout this kind of interaction with the environment, the agent slowly builds up the knowledge required to differentiate between good and bad actions. After enough exploration is carried out, the estimated Q-value matrix is expected to capture the long-term desirability of each state-action pair (the true "Q-value") fairly well. From that moment, the **ε** value will be set to a smaller value aiming to have the agent exploits what it already knows in order to obtain more rewards. However, the low **ε** value will still enable the agent to continue exploring various actions occasionally, which is a useful process if the operating conditions in the field (traffic levels, demand patterns, etc.) change over time.

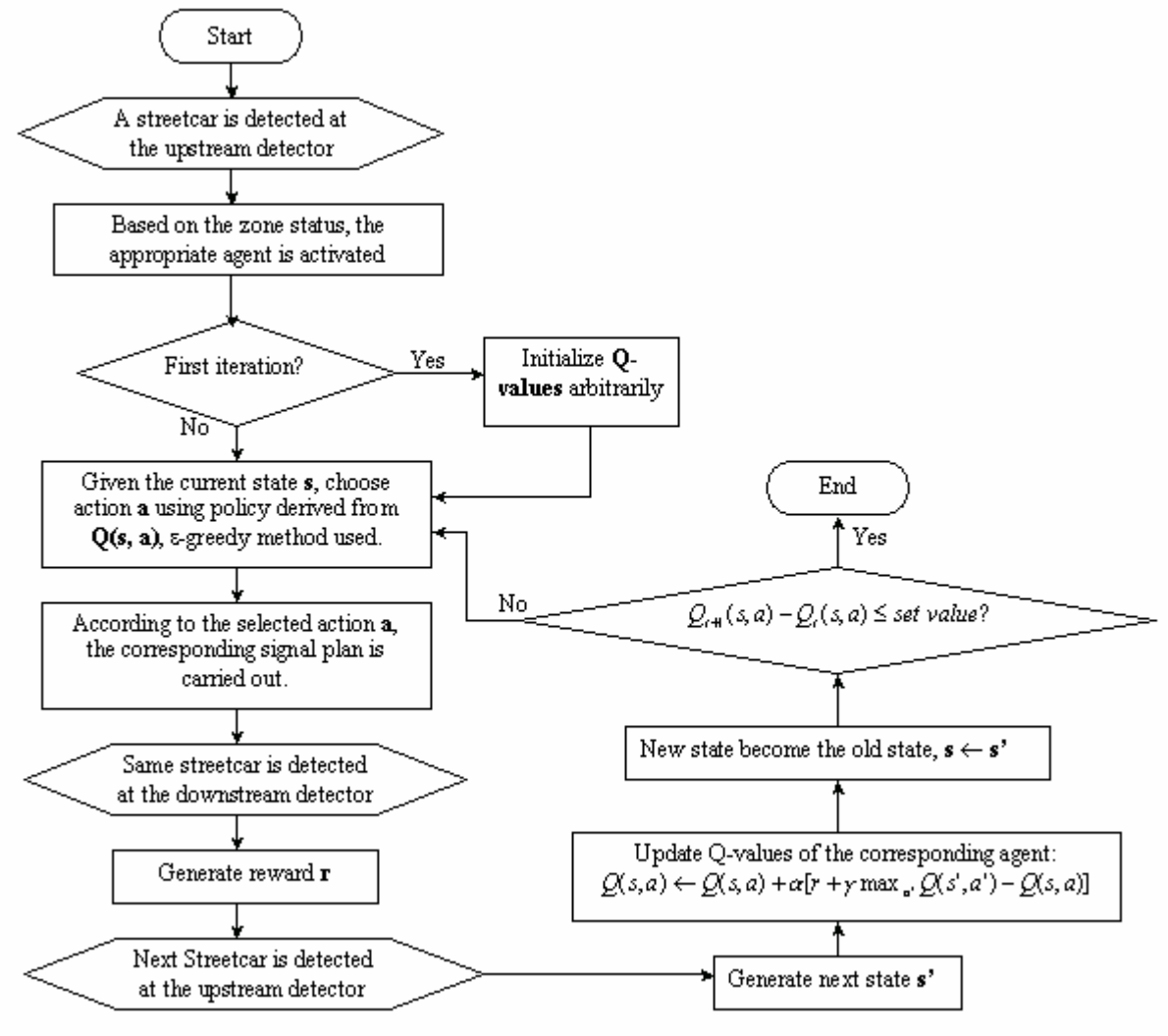

Figure 4. The Q-values updating process

A unified view

As mentioned earlier, three consecutive signalized intersections were used to implement the multiple Q-learning agents. Each intersection has an upstream detector located around 100 m upstream of the stop line and a downstream detector located just downstream of the stop line. The roadway in between the upstream and downstream detectors is considered as a detection zone. The zone can be in one of the following three states:

- **Non-active:** a situation when no streetcar is located within the zone
- **Standby:** a situation when one streetcar is located within the zone
- **Active**: a situation when two streetcars are located within the zone

A flowchart diagram outlining the general operation procedure of the multiple Q-learning agents is shown in Figure 5. For instance, if the zone is active at the first signalized intersection, the "bunch-splitting" agent will be activated. This "bunch-splitting" agent's major task is to split the bunch by modifying the signal timing accordingly. On the contrary, if the detection zone's state is "standby", the "headway-control" agent defined in the previous study (Ling & Shalaby, 2003) will be activated to minimize headway deviation for this streetcar.

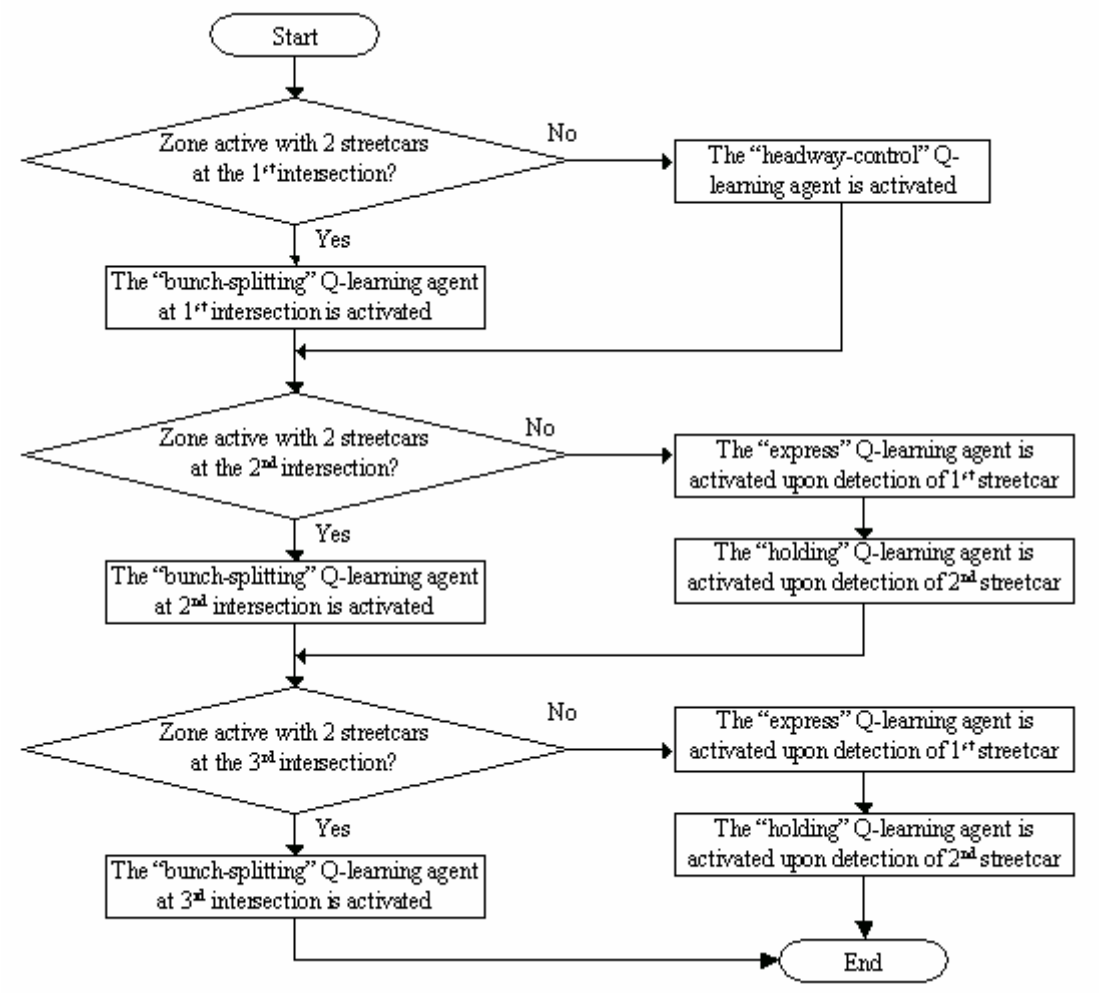

Figure 5. The operations of the multiple Q-learning agents

The zone status in the second signalized intersection provides an indication of whether the first intersection's agent is effective in splitting a streetcar bunch encountered on that intersection. A check on the zone status will be carried out. If the zone switches from a state of non-active to standby, it clearly indicates that the upstream agent was able to split the bunch. In this situation, the agent of this intersection is supposed to further widen the gap between this streetcar and the following one. Strictly speaking, this target is achieved by making use of two agents with different objectives. One agent is used to "express" the current streetcar and the other agent is used to "hold" the following streetcar. Through the combination of expressing the first and delaying the second streetcar, the gap between this pair of streetcars can be widened effectively.

Conversely, if the zone (of the 2nd intersection) switches from a state of inactive to active, it illustrates that either the first agent failed to split the bunch or the bunch formed again within the zone of the 2nd signalized intersection. This triggers the "bunch-splitting" Q-learning agent at the 2nd signalized intersection. Similar to the first intersection, this agent aims to split the bunch through modifying the signal-timing plan at the current intersection.

5. Results and discussion

The multiple Q-learning agents are designed exclusively to tackle the problem of "bunching". Only those streetcars that arrive in bunches can be considered for the learning

experience for the multiple Q-learning agents. To facilitate the training process of these multiple Q-learning agents, the agents should be exposed to an environment with enough cases of "bunching". The problem here is how many cases are considered enough for the purpose of training the agents? Suppose there are 120 distinct states and for each state the agent has on average a choice of 9 valid actions, it results in 1080 state-action pairs. In other words, a single visit to each state-action pair requires at least 1080 cases of bunching. It requires an unacceptable large amount of training time if streetcar bunches occurs at a rate equals to the existing field condition. To cope with this problem, streetcar bunches were created artificially and released into the simulated network every 5 minutes. The frequent occurrence of these artificially created streetcar bunches can speed up the training process for these multiple agents significantly.

The training of these multiple Q-learning agents was started with an all-zero initialized Q-value matrix with a discount factor of 0.1 and a learning rate of 0.3. The Q-values od each agent were monitored throughout the training period. Training is considered complete if these Q-values show signs of convergence. After around 15,000 iterations, convergence for most Q(s,a) values had been reached and training was completed. Simulation was run for the period from 6:30 am to 11:30 am using the King Street network in Paramics.

Gap time within the streetcar bunch

The performance of the multi Q-learning agents in dealing with the problem of "streetcar bunching" can be measured by tracking the gap time within the streetcar bunch over those agent-controlled signalized intersections. During the simulation period, each occurrence of bunching will trigger the Q-learning agents to split the bunch. At the same time, the time-based headway will be calculated at the detector located just downstream of the stop line after the modified signal plan is carried out.

The multi Q-learning agents have been trained under the simulated environment with over 5000 incidents of streetcar bunching. Note that the simulation time required for the training of the multi-agent is much more than that of the single-agent. The main reason is that the multi-agent only performs the updating process of Q-values whenever a bunch is encountered. This is in contrast to the single agent case where the updating process is carried out whenever any streetcar is detected.

The gap time within the streetcar bunch measured at various locations with multiple Q-learning agents implemented are plotted in Figure 6. The bar chart shows clearly that the gap time within the streetcar bunch increases substantially after the activation of the first agent responsible for splitting the bunch. The gap time continues to increase slowly with the cooperation of the "expressing" agent and "holding" agent at the 2nd and 3rd signalized intersection. However, the headway increases only up to about 65 sec at the 3rd downstream intersection, which is still relatively far from the desired 120-sec line headway. This is mainly due to the agent settings and constraints imposed in this study. For example, if the number of agent-controlled intersections is increased beyond 3, it is expected that the headway can be restored to a value close to the line headway. The number of additional agent-controlled intersections can be minimized if the constraints on the green time length of the "holding" and "expressing" agents are relaxed.

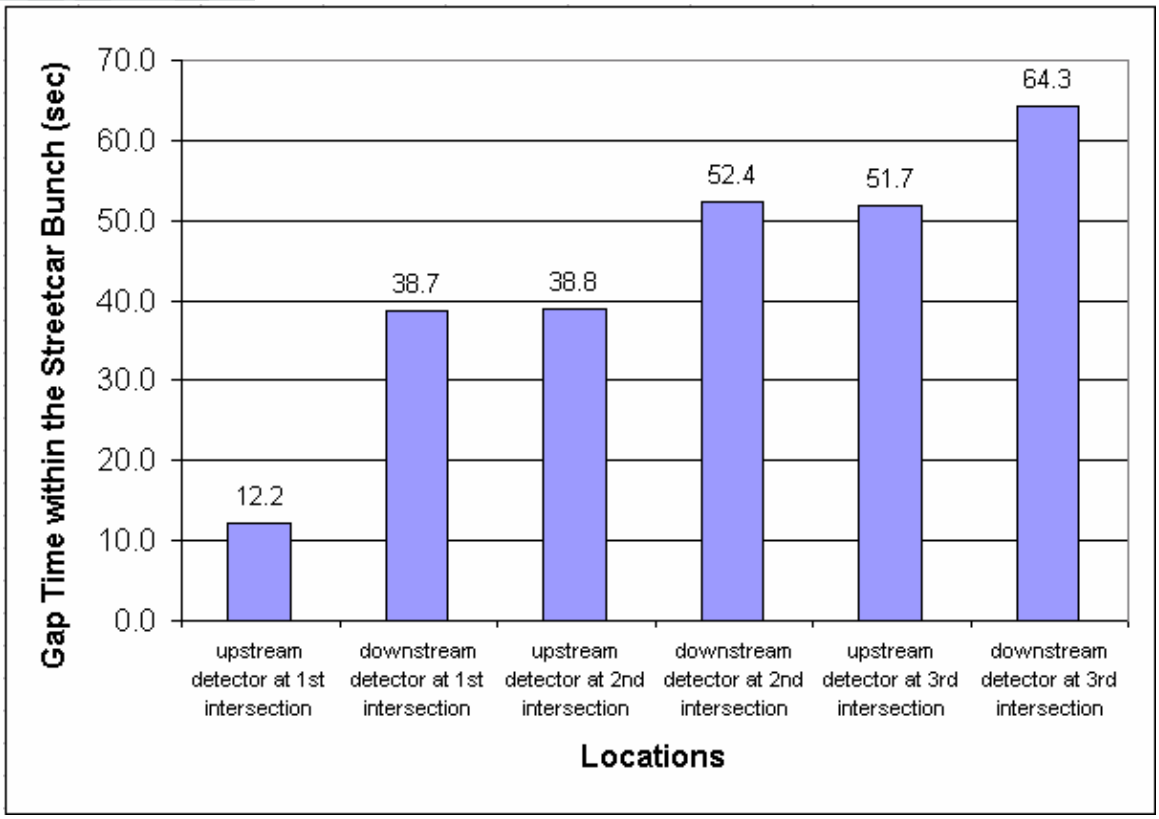

Figure 6 -- Variation of gap time within the streetcar bunch

Bunching reduction

In addition to measuring the gap time within the streetcar bunch along the three signalized intersections, another way of evaluating the performance of the multi Q-learning is to calculate the reduction in the cases of bunching. In this study, whenever two streetcars occupied the same detection zone at the same time, we define that a bunch is formed. On a time-based headway basis, any streetcar with less than 20 seconds of headway is also defined as a "bunch". By recording the total number of bunching cases before and after the activation of the 1st, 2nd and 3rd Q-learning agents, a bar chart like Figure 7 can be plotted. From the bar chart, the total number of streetcar bunching drops dramatically from 27 to 10 after the activation of the first agent. As the streetcars proceed further downstream, the 2nd agent and the 3rd agent can prevent the bunch from forming again. Moreover, these agents can further lower the cases of bunching down to only 3 cases. Overall, this is equivalent to an 88.9 % reduction of the total number of bunching occurred.

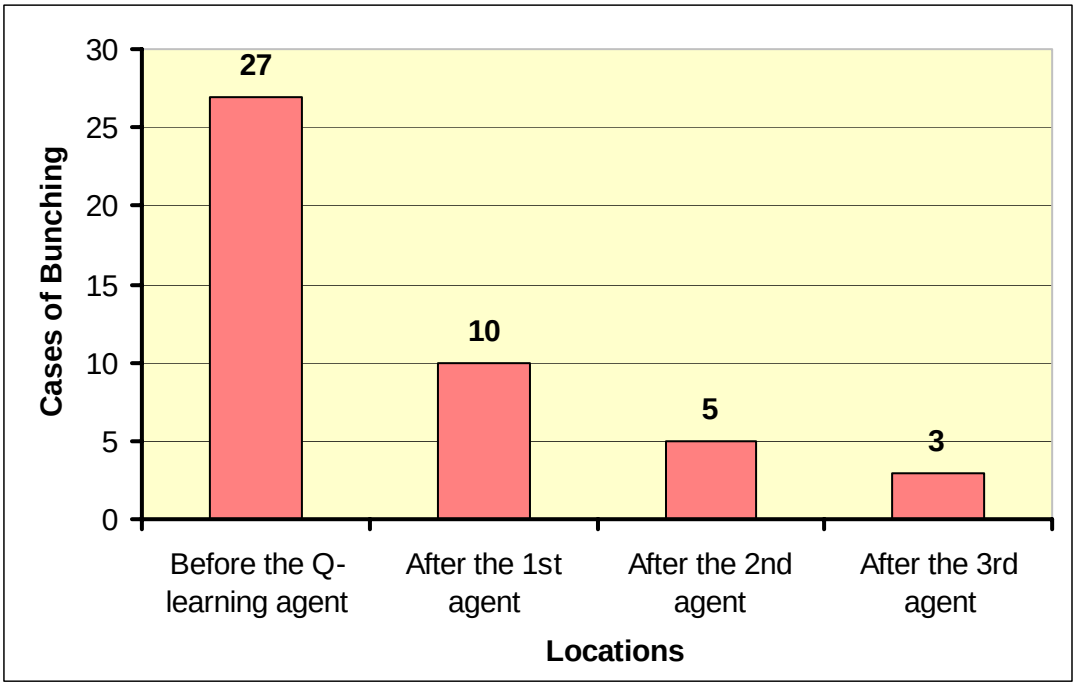

Figure 7 -- Cases of streetcar bunching observed at various signalized intersections

6. Conclusions and recommendations

The concept of Reinforcement Learning (an Artificial Intelligence approach) was successfully applied to control transit vehicle bunching through the adaptive modification of signal timing plan. Different elements under the context of Reinforcement Learning were defined and set up with respect to traffic signal control operation. The “bunch-splitting” agent was highly effective in splitting a streetcar bunch by modifying the signal-timing plan appropriately. The gap time of the two closely spaced streetcars was widened further through the cooperation of the “expressing” and “holding” agents. Upon the detection of a streetcar bunch, the multiple Q-learning agents were shown to be able to split up the bunch and slowly build up a gap between the closely spaced streetcars. The cases of bunching were reduced by 88.9 % with the aid of multiple Q-learning agents implemented over three consecutive signalized intersections.

Although the approach and the results of this study are very promising and encouraging, several limitations need to be addressed. In particular, further research is required to determine the optimal number of Reinforcement Learning agents and the best settings and constraints of each agent. Moreover, an expansion of state space information to capture general vehicular traffic conditions on the major and minor road may further improve the performance of the agents. It is also recommended to further investigate the optimal location of the agent-controlled intersections along the transit corridor in order to achieve optimal transit performance.

Ackowledgements

This research was funded by GEOIDE (Geomatics for Informed Decisions) and CITO (Communications and Information Technology Ontario)

References

Abkowitz, M., Tozzi, J., 1987. Research contributions to managing transit service reliability. Journal of Advanced Transportation, vol. 21, Spring.

Lee, J., 2001. Microsimulation Modeling and Advancement of Transit Priority Options at Major Arterials. M.A.Sc. Thesis, Department of Civil Engineering, University of Toronto.

Ling, K., Shalaby, A., 2003. Automated Transit Headway Control via Adaptive Signal Priority, in press, Special Issue on Modelling for Transit Operations and Service Planning, Journal of Advanced Transportation..

Quadstone, 2000. Paramics Modeler ver 3.0 User Guide and Reference Manual.

Quadstone, 2000. Paramics Programmer ver 3.0 User Guide and Reference Manual.

Rossetti, M.D., Turitto, T., 1998. Comparing static and dynamic threshold based control strategies. Transportation Research, part A, 32(8), 607-620.

Shalaby, A., Abdulhai, B. and Lee, J., 2003. Assessment of Streetcar Transit Priority Options Using Microsimulation Modelling, in press, Special Issue on Innovations in Transportation Engineering, Canadian Journal of Civil Engineering, 30(6), 1-10.

Strathman, J.G., Hopper, J.R., 1993. Empirical analysis of bus transit on-time performance. Transportation Research, Part A, 27A(2), 93-100.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

Wilson, N.H.M., Macchi, R.A., Fellows, R.E. and Deckoff, A.A., 1992. Improving service on the MBTA Green Line through better operations control. Transportation Research Record 1361, 296-304.