Neural Networks for Real-Time Traffic Signal Control

Dipti Srinivasan, *Senior Member, IEEE*, Min Chee Choy, and Ruey Long Cheu, *Member, IEEE*

***Abstract*—Real-time traffic signal control is an integral part of the urban traffic control system, and providing effective real-time traffic signal control for a large complex traffic network is an extremely challenging distributed control problem. This paper adopts the multiagent system approach to develop distributed unsupervised traffic responsive signal control models, where each agent in the system is a local traffic signal controller for one intersection in the traffic network. The first multiagent system is developed using hybrid computational intelligent techniques. Each agent employs a multistage online learning process to update and adapt its knowledge base and decision-making mechanism. The second multiagent system is developed by integrating the simultaneous perturbation stochastic approximation theorem in fuzzy neural networks (NN). The problem of real-time traffic signal control is especially challenging if the agents are used for an infinite horizon problem, where online learning has to take place continuously once the agent-based traffic signal controllers are implemented into the traffic network. A comprehensive simulation model of a section of the Central Business District of Singapore has been developed using PARAMICS microscopic simulation program. Simulation results show that the hybrid multiagent system provides significant improvement in traffic conditions when evaluated against an existing traffic signal control algorithm as well as the SPSA-NN-based multiagent system as the complexity of the simulation scenario increases. Using the hybrid NN-based multiagent system, the mean delay of each vehicle was reduced by 78% and the mean stoppage time, by 85% compared to the existing traffic signal control algorithm. The promising results demonstrate the efficacy of the hybrid NN-based multiagent system in solving large-scale traffic signal control problems in a distributed manner.**

***Index Terms*—Distributed control, hybrid model, neural control, online learning, traffic signal control.**

I. Introduction

THE INCREASE in urbanization and traffic congestion creates an urgent need to operate our transportation systems with maximum efficiency. As traffic volume continues to increase, the streets become more and more congested. One of the most cost-effective measures for dealing with this problem is traffic signal control. Traffic signal retiming and coordination of existing signals have been proven to bring about substantial reductions in traffic delay, considerable energy savings, and consequently, huge reduction in travel time and increased safety for the public.

Manuscript received September 20, 2004; revised April 21, 2005. This work was supported by the National University of Singapore under Research Grant R-263-000-226-112. The Associate Editor for this paper was H. Chen.

D. Srinivasan and M. C. Choy are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: dipti@nus.edu.sg; g0402564@nus.edu.sg).

R. L. Cheu is with the Department of Civil Engineering, National University of Singapore, Singapore 117576 (e-mail: cvecrl@nus.edu.sg).

Digital Object Identifier 10.1109/TITS.2006.874716

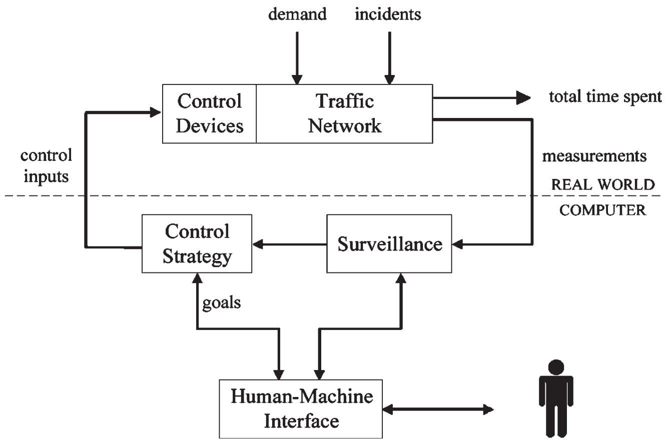


Fig. 1. Traffic control loop.

Control of traffic signals for efficient movement of traffic on urban streets constitutes a challenging part of an urban traffic control system (UTCS). Traffic signal control varies in complexity, from simple systems that use historical data to determine fixed timing plans, to adaptive signal control, which optimizes timing plans for a network of signals according to traffic conditions in real time. Some of the common traffic signal operations for controlling the traffic flow are cycle time adjustment; split adjustment, where "split" is defined as the fraction of the cycle time that is allocated to each phase for a set of traffic movements [1]; and offset adjustment, where "offset" is the time difference between the beginning of green phases for a continuous traffic movement at successive intersections that may give rise to a "green wave" along an arterial [1]. The basic elements of a traffic control loop are shown in Fig. 1.

The traffic flow behavior in the network depends on control inputs that are directly related to corresponding control devices, such as traffic lights, variable message signs, etc., and disturbances, whose values cannot be manipulated but may possibly be measurable (e.g., demand) or detectable (e.g., incident). The network's performance is measured via suitable indices such as the total time spent by all vehicles in the network over a time horizon, the total mean delay experienced by all vehicles in the network, and average vehicle speed. The function of the control strategy module is to specify the control inputs in real time based on available measurements (e.g., from loop detectors), estimations, or predictions so as to achieve the control objectives despite the influence of various disturbances.

For UTCS used for controlling traffic signals in a large-scale traffic network, it is crucial that the traffic signal control system has the capability to examine both the microscopic level of the situation (the traffic state of each intersection) as well as the macroscopic level of the situation (the overall traffic state of the traffic network). In addition, the traffic signal control system should be able to adjust various traffic signal control parameters (such as the green time, cycle length, etc.) in response to the varying traffic demand (as opposed to the fixed signal plans of some older systems). However, for a large-scale traffic management system, it may not only be difficult or impossible to tell whether the traffic network is flowing smoothly and assess its current state, but predicting the effects of modifying any of the traffic control parameters is a difficult task due to nonlinear and stochastic events in a traffic network. Additionally, given the complexity and dynamicity of the traffic signal control problem, the control system should be adaptive in nature so that its control function can be adjusted whenever necessary. In fact, such advanced traffic-responsive closed-loop systems and adaptive traffic signal systems are becoming increasingly critical for transportation agencies to meet their day-to-day operation and management needs [1]–[8]. It is anticipated that new technology, such as traffic-responsive closed-loop systems or adaptive traffic signal systems using advanced surveillance and traffic management centers, will become increasingly critical for city, region, and state organizations to meet future transportation needs.

The interdependency of each intersection on its neighbors makes it extremely difficult to set the signal parameter values for a large complex traffic network with multiple intersections. An attractive approach to deal with this complicated traffic signal control problem is the use of distributed control technique involving multiple intelligent agents. The primary objective of the multiagent system is to achieve coordinated traffic signal control so as to reduce the likelihood of traffic congestion. It is imperative that such a system performs real-time update of traffic signals in the traffic network based on the changes in the traffic volume. For a case where individual agents are controlling the traffic signals for an indefinite amount of time after they have been installed into the traffic network, the problem of real-time traffic signal control can be said to take the form of an infinite horizon distributed control problem. Hence, for effective traffic signal control, such controllers need to adapt themselves continuously.

Various computational intelligence-based approaches have been proposed for designing real-time traffic signal controllers, such as fuzzy sets [2], [3], genetic algorithm and reinforcement learning [4], and neural networks (NN) [5]–[7]. Most of these works are based on the distributed approach, where an agent is assigned to update the traffic signals of a single intersection based on the traffic flow in all the approaches of that intersection. As some of these models (e.g., [5] and [7]) have implemented and tested the controller on a simplified traffic network model consisting of a single intersection, the effectiveness of the proposed neural controller for controlling a large-scale traffic network with multiple intersections cannot be established. The method proposed in [6] involves the application of simultaneous perturbation stochastic approximation (SPSA) in modeling the weight update process of an NN. Although the SPSA algorithm is a viable option for online weight update as it presents some form of stochastic exploration and converges to a set of optimal values under certain conditions, the model proposed in [6] has some limitations concerning its robustness and responsiveness (more details will be given in Section IV).

The work presented by Choy *et al.* [8] introduced a hybrid multiagent system architecture for real-time system control. This paper presents an enhanced version of SPSA-NN-based multiagent system, which has been tested in more complex scenarios to determine its efficacy. These two multiagent systems as well as an existing traffic signal control algorithm Green Link Determining (GLIDE) are used to control the signalized intersections of a large simulated traffic network based on a section of the Central Business District (CBD) of Singapore. This paper seeks to demonstrate the efficacy of the hybrid multiagent system in solving the infinite horizon distributed control problem.

The paper is arranged as follows: Section II describes the modeling of the traffic signal control problem using a form of a directed graph. Section III describes the performance measures that are used to evaluate the traffic signal control models developed in this paper. Section IV presents the SPSA-NN-based multiagent system, whereas Section V presents the hybrid NN-based multiagent system. Section VI describes the experimental setup. Finally, Section VII presents the simulation results and, Section VIII concludes the findings of this paper.

II. Modeling the Traffic Signal Control Problem

The problem of traffic signal control for a large traffic network can be divided into subproblems, where each subproblem is handled by a local traffic signal controller or an agent. For such a distributed approach, each agent will generate its own control variables based on the local information it receives. Based on the approach in [9], the following are defined:

G	$= (N, L)$, directed graph with a set N of n nodes and a set L of l links describing the traffic network;
A_i	agent acting at node $i \epsilon N$;
T	total number of temporal stages at time t;
$I_i(T)$	local input information vector of A_i at stage T;
$c_i(T)$	vector of control parameters of A_i at stage T;
$c_i(T)^o$	optimal vector of control parameters of A_i at stage T;
$u_i(T)$	$= F_{\mathrm{iT}}[I_i(T), c_i(T)]$, control function of A_i at stage T;
$u_i(T)^o$	$= F^o_{\mathrm{iT}}[I_i(T), c_i(T)^o]$, optimal control function of A_i at stage T;
$s(T)$	state vector of the traffic network at stage T;
C	cost function for the entire traffic network.

The distributed control problem of the traffic network is therefore finding the set of control function $u_i(T)$ for each A_i, $i \in N$ that minimizes the cost function C, where C is a function of the states of the traffic network at different temporal stages.

The distributed control problem can be easily rephrased based on this definition, depending on the nature of the control techniques and the application domain. For example, if the agents are implemented using only NNs, then $c_i(T)$ will represent the weight and bias vector of the NN and $u_i(T)$ will be the neural control function modeled by the NN. Due to the difficulty in obtaining the optimal solution for the distributed control problem in an analytical manner, an alternative approach involving an approximated optimal solution can be used. The optimal control function $u_i(T)^o$ for each agent will be used as an approximation to the optimal control function for the traffic network. In the case where all the A_is of the traffic network are designed for continuous traffic signal control, the optimization problem becomes that of an infinite horizon one. A reasonable approximation of such a problem has been presented in [9] based on [10] in the form of the "receding-horizon limited memory" where the requirement for infinite memory storage capacity can be overlooked.

For the receding-horizon limited-memory optimization problem, at stage $T = 0$, one has to solve the distributed control problem for the stage $\{0, \ldots, M\}$, $M > 0$. The approximated optimal control variables generated by the optimal control functions $u_i(0)^o$, $i \in N$ are applied at stage $T = 0$. For the next stage $T = 1$, the distributed control problem is restated again for stage $\{1, \ldots, M + 1\}$. Once again, the approximated optimal control variables generated by the control functions $u_i(1)^o$, $i \in N$ are applied at $T = 1$. Two important issues to be addressed here are the approximation ability of the techniques that are used to implement the A_i and the ability of each A_i to derive a good approximation of the optimal solution in a timely manner. The first issue involves the appropriate usage of various relevant techniques that have good approximating capabilities while facing an ill-defined problem with a high level of associated uncertainty and diverse input data. In this case, well-known techniques in the field of computational intelligence, such as NNs, provide possible solutions, whereas some classical traffic signal control techniques may have severe limitations. The second issue, however, may not be easily solved even if the first issue has been resolved. The difficulty in obtaining a reasonably good approximated optimal solution for stage $\{0, \ldots, M\}$ at stage $T = 0$ (or any other future state) is due to the limited number of stage M in which each A_i can be considered at any particular stage T due to finite memory storage space as well as the computational speed of the processor. Some of the existing algorithms such as the parameters update algorithms for connectionist networks are mainly designed for finite horizon learning processes, and hence, adjustments need to be made in order for them to deal with the infinite horizon problem. Possible adjustments include the adoption of hybrid techniques that can possibly leverage on the strength of the individual techniques so as to overcome their individual shortcomings.

Based on the limitations of earlier research works, it is evident that more needs be done to implement a systematic unsupervised distributed control scheme for testing in a complex traffic network simulated based on a real-world scenario. This paper presents a hybrid multiagent system that integrates the advantages offered by various computational intelligence techniques to implement a new distributed control architecture as described in the following sections.

III. Performance Measures

The performance of the multiagent systems developed in this paper is evaluated using two measures, namely 1) mean delay of vehicles and 2) mean stoppage time of vehicles. The microscopic traffic simulation platform of PARAMICS has been used to take detailed measurements of various parameters associated with each vehicle that enters and leaves the traffic network. As such, during the course of the simulations, the delay faced by each vehicle entering and leaving the network was being stored in memory, and the mean delay of vehicles was calculated as

$$T_{\mathrm{MD}} = \frac{T_D}{T_V} \tag{1}$$

where T_{MD} is the mean delay of vehicles, T_D is the total delay, and T_V is the total number of vehicles that entered and left the traffic network during the time the measurement was taken. Similarly, the total stoppage time for each vehicle was also stored in memory to facilitate the calculation of the mean stoppage time, which is expressed as

$$T_{\mathrm{MST}} = \frac{T_{\mathrm{ST}}}{T_V} \tag{2}$$

where T_{MST} is the mean stoppage time for the vehicles and T_{ST} is the total amount of stoppage time faced by all vehicles that entered and left the traffic network during the time the measurement was taken.

The current vehicle mean speed (equivalent to the instantaneous speed of a vehicle) is defined as the average speed of all the vehicles that are currently in the traffic network. These two performance measures are reflective of the overall traffic condition in the network. For an overcongested traffic network, the total mean delay of the vehicles will be high and the current mean speed of the vehicles that are in the traffic network will be low.

IV. SPSA-NN Model

Stochastic optimization is the process of finding a minimum point θ^* of a real-value function $C(\theta)$ in the presence of noise. Stochastic approximation is a popular technique of stochastic optimization and is often applied when the gradient $g(\theta)$ of the cost function $C(\theta)$ is not readily available. The relations (e.g., convergence properties) between some of the online learning techniques such as Q-learning and stochastic approximation theory have been established in [11] and [12]. Spall [13] introduces a new form of stochastic approximation theory termed as SPSA and establishes the conditions in which SPSA converges and becomes asymptotically normally distributed.

The approximating properties of neural networks with a single hidden layer for solving the Distributed Control Problem have been established in [15]–[17].

A. Using SPSA to Update the Neurons' Weights

Given the favorable properties of SPSA mentioned previously, it has been used to update the weights of an NN [6], [18], [19] in order to find the minimum point of the loss function. The distributed control problem can be redefined as finding the set of NN weight parameters $w_i(T)$ for each A_i, $i \in N$ that minimizes the approximated cost function $\hat{C}$, where $\hat{C}$ is a function of $w_i(T)$.

For clarity and without the loss of generality, some of the notations used in the earlier section can be redefined as

w_T	estimate of θ at stage T for each A_i, $i \epsilon N$;
$\hat{g}_T(w_T)$	estimate of $g(\theta)$ at stage T for each A_i, $i \epsilon N$.

The Robbin–Monro stochastic approximation (RMSA) [14] can then be written as

$$w_{T+1} = w_T - \alpha_T \hat{g}_T(w_T) \tag{3}$$

where α_T is the gain sequence that must satisfy the following well-known convergence conditions:

$$\begin{aligned}&\alpha_T > 0 \quad \forall T; \quad \alpha_T \to 0 \text{ as } T \to \infty\\ &\sum_{T=0}^{\infty} \alpha_T = \infty\\ &\sum_{T=0}^{\infty} \alpha_T^2 < \infty.\end{aligned} \tag{4}$$

Based on the preceding conditions, Spall [13] developed SPSA using the idea of "simultaneous perturbation" to estimate the gradient. The formal proof of convergence of SPSA and the asymptotic normality of w_T can be found in [13]. Some of the expressions of the SPSA algorithm are given as follows to facilitate the understanding of how SPSA can be applied in an NN.

Let $\Delta_T \in R^P$ be a vector of p mutually independent mean-zero random variables $\{\Delta_{\mathrm{T1}}, \Delta_{\mathrm{T2}}, \ldots, \Delta_{\mathrm{TP}}\}$ satisfying the various conditions given in [13] (an example would be a symmetrical Bernoulli distribution) at stage T. Let c_T be a positive scalar. The noisy measurement of the loss function $L(\theta)$ is given as

$$\begin{aligned}y_T^{(+)} &= L(w_T + c_T\Delta_T) + \varepsilon_T^{(+)}\\ y_T^{(-)} &= L(w_T - c_T\Delta_T) + \varepsilon_T^{(-)}\end{aligned} \tag{5}$$

where $c_T\Delta_T$ is the stochastic perturbation that is applied to w_T during stage T and $\varepsilon_T^{(+)}$ and $\varepsilon_T^{(-)}$ represent measurement noise terms that must satisfy the following:

$$E\left(\varepsilon_T^{(+)} - \varepsilon_T^{(-)} | \{w_1, w_2, \ldots, w_T\}\right) = 0 \; a.s., \qquad \forall T. \tag{6}$$

The estimate of $g(\theta)$ can thus be written as

$$\hat{g}_T(w_T) = \begin{bmatrix} \frac{y_T^{(+)} - y_T^{(-)}}{2c_T\Delta_{\mathrm{T1}}} \\ \vdots \\ \frac{y_T^{(+)} - y_T^{(-)}}{2c_T\Delta_{\mathrm{TP}}} \end{bmatrix}. \tag{7}$$

Defining the error term as

$$e_T(w_T) = \hat{g}_T(w_T) - E\left(\hat{g}_T(w_T) | w_T\right) \tag{8}$$

(3) can be rewritten in a more generalized form as

$$w_{T+1} = w_T - \alpha_T \left[\hat{g}_T(w_T) + b_T(w_T) + e_T(w_T)\right] \tag{9}$$

where $b_T(w_T)$ is the bias in $\hat{g}_T(w_T)$ $b_T(w_T) \to O(c_T^2)(c_T \to 0)$ or near unbiasness if the loss function $L(\theta)$ is sufficiently smooth, the measurement noise satisfies (6), and the conditions for Δ_T are met.

The strong convergence of (9) to θ^* has been proven in [13] subject to stated assumptions. As such, the iterative forms presented in (3) and (9) can be used to model the iterative weight update process in an NN controller.

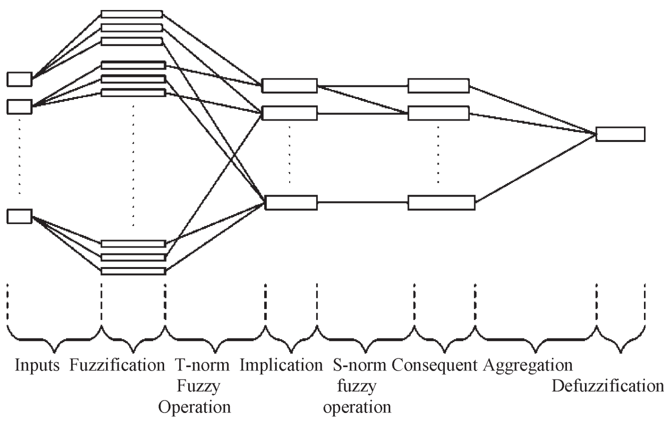


Fig. 2. Five-layered fuzzy NN.

In [6], the SPSA-NN model was applied to control traffic signals for a large traffic network with multiple intersections, using a three-layer NN and taking relevant traffic variables as inputs. This model, however, had two shortcomings. First, the approach involved the use of heuristics to manually identify the general traffic patterns (morning peaks and evening peaks) and the assignment of time periods for each pattern. The robustness of the system may come into question if the fluctuations of the traffic volume in the traffic network are not periodic. Second, an NN is assigned to each time period, and the weights of the NN are updated only during the duration of the time period. This implies that the weight update is done only on a daily basis whenever the same traffic pattern and time period arises. As such, the traffic controllers may not be able to respond well to changes in the traffic network within the same time period. Although the results in [6] show that the SPSA-NN is able to converge to a set of optimal signal plans, the NN has to be reupdated time and again to take into consideration changes in the long-term dynamics of the traffic network even after the convergence. In addition, no formal proof of convergence to a set of globally optimal solutions is presented in [6] or in [18] and [19]. Nevertheless, given that the research work in [6] yields good results for traffic signal control of a large traffic network and coupled with the favorable characteristics of the SPSA algorithm, an alternative SPSA-NN that overcomes the aforementioned limitations has been implemented in this paper and its performance compared to that of the hybrid NN-based multiagent system.

B. Structure of an Agent Using SPSA-NN

As mentioned, the modified SPSA-NN model used in this paper strives to avoid the two limitations of [6]. The structure of A_i is formulated in such a way that each A_i can act as an independent intelligent agent once it has been implemented. A five-layer fuzzy NN in Fig. 2 is used as the basic building block for each component in A_i.

In the fuzzy NN shown in Fig. 2, the operators between the fuzzification layer (second layer) and the implication layer (third layer) are taken to be T-norm, whereas the operator used between the implication layer (third layer) and the consequent layer (fourth layer) is S-norm. Singleton membership

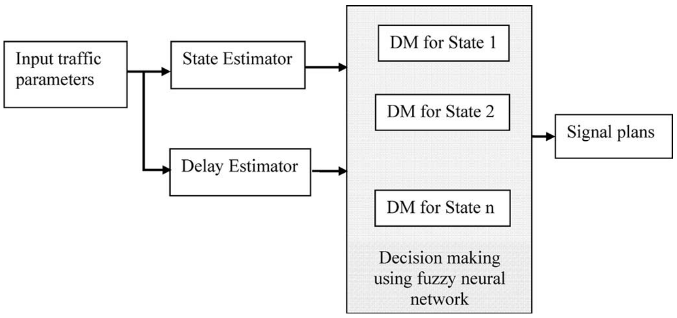


Fig. 3. Structure of a A_i for the SPSA-NN-based multiagent system.

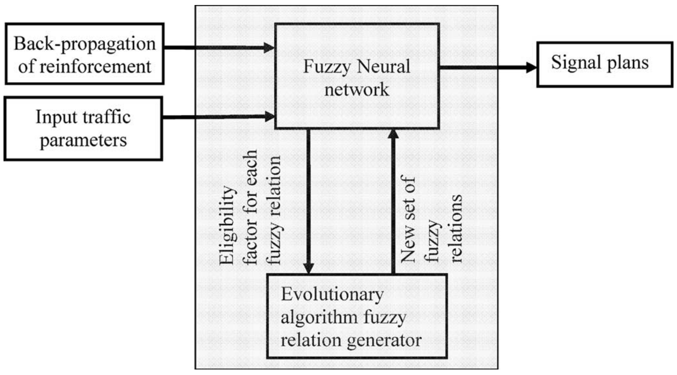

Back-propagation of reinforcement
Input traffic parameters
Fuzzy Neural network
Signal plans
Eligibility factor for each fuzzy relation
New set of fuzzy relations
Evolutionary algorithm fuzzy relation generator

Fig. 4. Structure of an agent A_i for the hybrid multiagent system.

functions are used for the fuzzy output terms to reduce the computational complexity in the defuzzification process. The structure of an agent A_i is shown in Fig. 3.

The agent A_i takes in traffic parameters as its inputs and generates a set of signal plans as the output via a two-stage process. The state estimation stage generates the estimated current state of the traffic network. Each agent A_i also contains several fuzzy NN-based decision makers (DMs), one for each state. The number of DMs used in the decision-making stage depends on the complexity of the problem and is determined based on the estimated current state of the traffic network. The decision-making and learning processes of each A_i are enabled by the NNs as follows: At stage T, the weights belonging to the state estimator are randomly perturbed by $c_T \Delta_T$ using the stochastic perturbation shown in (5). The state estimator estimates the current state of the intersection based on the input parameters. A DM is chosen to generate the signal plans for the intersection based on this estimate. The weights for the DM are also perturbed by $c_T \Delta_T$ using (5). The signal plans are then implemented into the simulated traffic network, and the average delay is calculated for all the vehicles passing through that intersection after the implementation of the new signal plan. If the weights of the state estimator (SE) have not been updated in the previous round (stage $T-1$), they are now updated using (9) based on the scaled difference between the delay during stage T and stage $T-1$; otherwise, the average delay is stored. The gradient can be estimated using (7).

The delay estimator is used for computing the average delay for each intersection during each stage T. The computation of the average delay is based on well-known delay equations [1], and the cost or error function of the SE at stage T is computed as

$$\begin{aligned}\text{Error}_{\text{SE}} = M(&\text{ave. delay at stage } T \\ &- \text{ave. delay at stage } T-1). \qquad (10)\end{aligned}$$

The cost (error) function of the DM at stage T is calculated as

$$\begin{aligned}\text{Error}_{\text{DM}} = M(&\text{ave. delay at stage } T \\ &- \text{ave. delay at stage } n) \quad n < T \qquad (11)\end{aligned}$$

where M is a constant that takes the value $0 < M < 1$ and stage n is the stage prior to stage T in which the chosen DM at stage T was previously selected.

V. Hybrid Multiagent System

The hybrid NN-based multiagent system has been designed to solve the distributed control problem directly, as opposed to the indirect method used for the SPSA-NN-based model presented previously. In this hybrid system, each agent A_i consists of a five-layered fuzzy NN that facilitates its decision-making process. The cost function C is a function of $s(T)$, which is the state of the traffic network at stage T. The structure of each A_i in the hybrid NN-based multiagent is as shown in Fig. 4.

A multistage online learning process has been implemented for real-time update of weights and connections in the NN. This online learning process overcomes the problems of lack of stochastic exploration and the possibility of getting stuck in local minima, which are often associated with performing online learning for an infinite horizon control problem [20], [21]. The learning process is a three-stage process that involves reinforcement learning, weight adjustment, and adjustment of fuzzy relations, as shown in Fig. 5.

The first step in this online learning process involves reinforcement learning. The reinforcement obtained from this process is backpropagated to each A_i, following which, all A_i proceed to adjust the learning rate for each neuron and activate the forgetting mechanism if necessary as determined by the value of the reinforcement that the A_i received. Next, each A_i adjusts the weights of the neurons according to the topological weight update method. Finally, the reinforcement is used to update the fitness value of each neuron in the A_i's NN. If the fitness values of the neurons fall below some prespecified values, the fuzzy relations (represented by how the outputs of a layer of neurons are connected to the inputs of the next layer of neurons) are updated using the evolutionary algorithm fuzzy relation generator (EAFRG). A detailed description of the multistage online learning process can be found in [8].

In this dynamically changing problem domain, the fuzzy rules and their membership functions need to be regularly updated to remain valid. In this work, fuzzy rules adjustment process is also performed online throughout the running of the simulation in order to accommodate possible fluctuations of the system dynamics. The evolutionary algorithm fuzzy relation generator (EAFRG) is used to generate new fuzzy relations based on the reinforcements received by the agents, thus changing the knowledge representation for each agent as the traffic

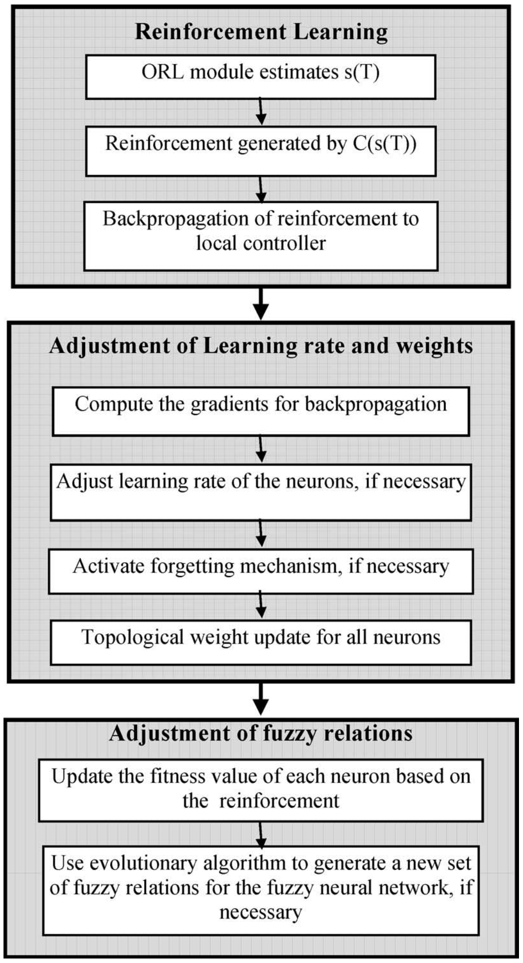
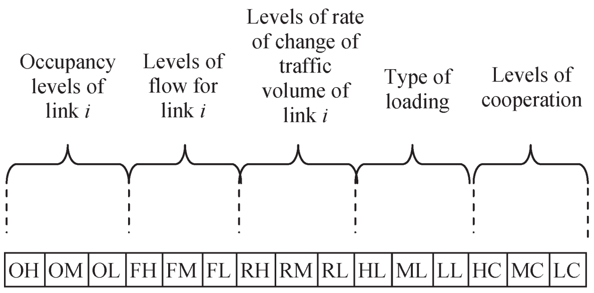

Reinforcement Learning
ORL module estimates s(T)
Reinforcement generated by C(s(T))
Backpropagation of reinforcement to local controller
Adjustment of Learning rate and weights
Compute the gradients for backpropagation
Adjust learning rate of the neurons, if necessary
Activate forgetting mechanism, if necessary
Topological weight update for all neurons
Adjustment of fuzzy relations
Update the fitness value of each neuron based on the reinforcement
Use evolutionary algorithm to generate a new set of fuzzy relations for the fuzzy neural network, if necessary

Fig. 5. Multistage online learning process.


Occupancy levels of link *i*
Levels of flow for link *i*
Levels of rate of change of traffic volume of link *i*
Type of loading
Levels of cooperation
OH OM OL FH FM FL RH RM RL HL ML LL HC MC LC

Legend:

OH, OM, OL	High, medium and low occupancy for link *i*, respectively
FH, FM, FL	High, medium and low traffic flow for link *i*, respectively
RH, RM, RL	High, medium and low rate of change of traffic volume for link *i*, respectively
HL, ML, LL	High, medium and low load, respectively
HC, MC, LC	High, medium and low cooperation, respectively

Fig. 6. Chromosome structure.

network evolves. The chromosome that is used in an EAFRG for an agent is as shown in Fig. 6.

Each chromosome determines the way nodes in layer 2 of the fuzzy NN (Fig. 2) are linked to the implication nodes in layer 3 for each link. Individual alleles in each chromosome have binary values. The first nine alleles represent the neurons in layer 2, whereas the last six alleles represent the neurons in layer 3. Only alleles representing different layers and having a binary value of one (1) are connected to one another. Fig. 7 shows a possible mapping of a chromosome to antecedent–implication relations. In Fig. 7, we can derive the following antecedent–implication relations:

Each chromosome corresponds to several antecedent-implication relations for a single link. The mapping of the chromosome for link i is as shown below.

OH	OM	OL	FH	FM	FL	RH	RM	RL	HL	ML	LL	HC	MC	LC
0	1	0	0	0	1	1	0	0	0	1	0	0	0	1

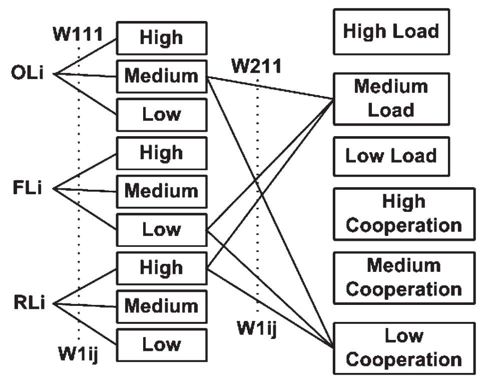


Fig. 7. Mapping of a chromosome to antecedent–implication relations.

$$\begin{aligned}&\{\text{occupancy is Medium}\}\&\{\text{flow is Low}\}\\&\quad\&\{\text{rate is High}\} => \{\text{load is Medium}\}\\&\{\text{occupancy is Medium}\}\&\{\text{flow is Low}\}\\&\quad\&\{\text{rate is High}\} => \{\text{cooperation is Low}\}\end{aligned}$$

Obtaining a suitable fitness function to evaluate the generated fuzzy relations is not an easy task because an ideal fitness function should be able to produce a fuzzy relation that results in a generally satisfying (i.e., how much/little deviations the fuzzy relation possesses in comparison with some well-known guidelines or rule-of-the-thumb for the chosen problem domain) as well as contextually valid/eligible fuzzy rule that can accommodate exceptions in the current problem state to a reasonable extent. As such, the fitness function should take into consideration the current eligibility of the fuzzy relation as well as the degree to which the rule is generally satisfying.

The fitness function F for the EAFRG is therefore defined as

$$F = \frac{E}{|\text{diff}(\text{cur\_chromo}, \text{gen\_chromo})| + 1} + \gamma G \qquad (12)$$

where E is the current eligibility of the antecedent–implication relation, G is the measure of whether the relation is generally satisfying, and γ is the sensitivity factor for G (determined empirically). Hence, as shown, E and G have counterbalancing influence on each other. A relation may be generally satisfying and have a high G value but may not be eligible due to the changing system dynamics, causing a low E value as a result.

E is further defined as

$$E = \mu E_T \tag{13}$$

where μ is the eligibility sensitivity factor, which is determined empirically. E_T is the eligibility trace at stage T and is computed as

$$E_T(T) = (1 - D)R(T - 1)A(T - 1) + E_T(T)D \tag{14}$$

where D is the decay constant (determined empirically), R is the reinforcement, and A is the activation value, which is zero if the rule is not activated and one (1) if activated. The function diff() denotes taking the difference between two chromosome vectors. In this case, the first chromosome vector is cur_chromo, which is the current chromosome used by the fuzzy NN, and the second chromosome vector is gen_chromo, which is the chromosome generated by EAFRG. For this research, G is defined as

$$G = \sum_i \sum_j \mathrm{Corr}(i, j) \tag{15}$$

where i denotes a node in the antecedent (second layer) and j denotes a node in the implication (third layer) such that the ij relation is zero if nodes i and j are not linked, and $\mathrm{Corr}(i, j)$ denotes the correlation between i and j.

Extensive simulations and tests were carried out to optimize these parameters according to the requirements of the system. A population size of 100, 7% probability of mutation, and 40% probability of crossover were found to be optimal. The algorithm was run for 50 generations. It was observed that the computational speed of the EAFRG does not hinder the real-time performance of the controller agent due to the small size of the population and the binary nature of the chromosome.

VI. Simulation Setup

The multiagent systems presented in the previous sections were tested in a simulated real-world scenario in order to effectively evaluate their performances when applied to a large complicated and realistic problem. This section describes in detail how a real-world traffic network was modeled using a microscopic simulation program and presents the results as compared to those obtained from the GLIDE benchmark.

A. Modeling of the Traffic Network in PARAMICS

The traffic network used to evaluate the performance of the proposed multiagent architecture was based on a section of the CBD area of Singapore. This section represents one of the busiest regions of the road network where frequent traffic jams are common during peak hours and the size of the traffic network is significantly bigger compared to those used in previous research works [2]–[7]. The traffic network was modeled using Version 4.0 of PARAMICS [23]. The use of PARAMICS Modeler version 4.0 provides the advantages of microscopic simulation whereby various microscopic details of the traffic network such as driver behaviors and incidents can be modeled to reasonable accuracy to ensure a close resemblance to the real-world scenario. Details regarding the simulated traffic network model are given in Table I.

TABLE I
Details of the Simulation Network

Total number of nodes	130
Total number of links	330
Total number of zones	23
Total number of loop detectors used	132
Types of vehicles in the traffic network	15

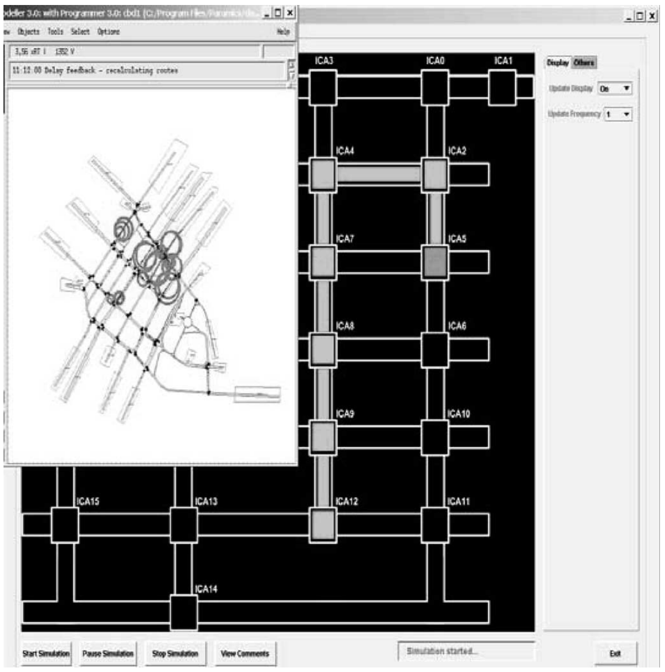
Important information such as lane occupancy, flow, and rate of change of flow is obtained via loop detectors in real time while the simulation is running. These loop detectors are coded in the simulated network at stop lines of the intersection approaches, as in the real-world installations. Using the PARAMICS application programming interface (API), information such as occupancy and flow was extracted from the loop detectors, and the rate of change of flow was calculated. The flow, occupancy, and rate of change of flow were measured at the green phase only (including the amber time). In addition, signal control directives (i.e., agents output) were implemented in the simulation via the PARAMICS API to effect the latest signal optimization actions. Policies recommended from the various levels of controller agents were translated into signal control directives, and changes were made to the signal plans in real time using the API. The total number of signalized intersections in the simulated traffic network is 25. As such, for each multiagent system, 25 agents were implemented to control the traffic signals of each intersection. Given that PARAMICS is a microscopic simulator and has the ability to model individual vehicles, it is able to accurately depict various traffic behaviors such as congestion formation and dispersion.

Simulation results under three types of scenarios of different time horizons are presented to evaluate the performance of different neural-network-based multiagent systems. Three-hour and 6-h simulations are used to test the response of the proposed control models before they are used for the infinite horizon problem. Due to time constraint and practical reasons, the infinite horizon problem is approximated using 24-h simulation runs. The number of peak periods for simulation runs for the three simulation scenarios is 1, 2, and 8, respectively, where each peak period lasts for 1 h.

B. Implementation of the Multiagent Systems

The multiagent systems are implemented using Java and its multithreading technology. During the running of the simulation in PARAMICS Modeler, the multiple threads/processes in Java representing the agents are running concurrently. The sampling rate for the agents can be adjusted in order to make sure that the agents make timely responses to the dynamically changing traffic network. For this paper, the agents for the multiagent systems are tuned to sample the traffic network for the traffic parameters once every 10 s (simulation clock time).

Fig. 8 shows the concurrent running of the traffic network and the multiagent systems in two windows, namely 1) the foreground depicting the simulated traffic network in PARAMICS Modeler and 2) the background, which is the Java graphical


11:12:00 Delay feedback - recalculating routes

Simulation started...

Fig. 8. Concurrent running of PARAMICS and the multiagent system.

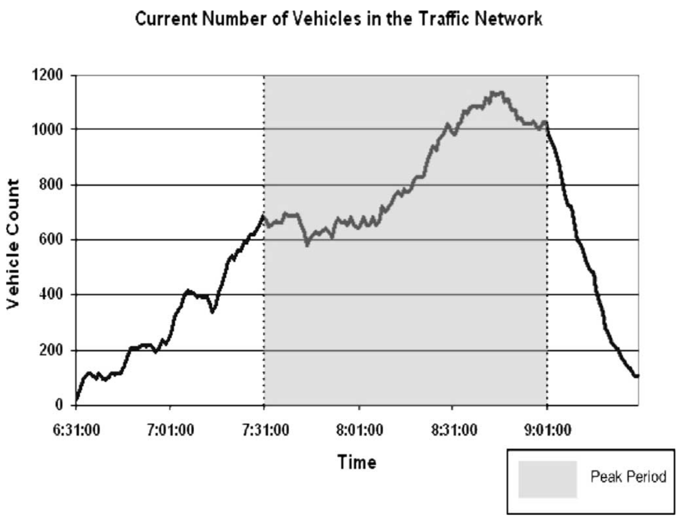


Fig. 9. Number of vehicles for a typical scenario with one morning peak.

user interface (GUI) for the multiagent system. The inset shows the map of the Suntec City area and the 25 intersections, which are being controlled by the agents.

The coordinated offset adjustment allows users to monitor the dynamic changes of the signal policies that are taking place in real time. As shown, the green wave starts from node ICA5 in the Java GUI, representing the critical intersection for offset adjustment and follows through several successive green nodes. These nodes correspond to the congested intersections as shown in the PARAMICS Modeler GUI in the foreground.

The traffic signal plans generated by each agent are first deposited in the signal plans repository before they are decoded by the signal plans interpreter and implemented into the traffic network. Each agent has eight different types of signal plans that it can use to control the traffic signals at its intersection. These plans are designed to cater for different amounts of traffic loading at the intersection. It is up to each agent to choose an appropriate signal plan based on its own perception of the traffic loading of its intersection.

C. Benchmarking

It is difficult to find a good benchmark for this large-scale traffic signal control problem for several reasons. Existing algorithms or control methodologies have been developed for controlling the traffic networks of other cities [5]–[7] with different traffic patterns, and hence, the results obtained from those works cannot be applied directly on this problem. Moreover, some of the algorithms presented in the literature [5], [7] are developed for simplified scenarios and hence are not suitable for benchmarking. Last, commercial traffic signal control programs that are known to have worked well are not easily available due to proprietary reasons.

In view of these issues, the signal settings used for benchmarking are derived from the actual signal plans implemented by Land Transportation Authority (LTA)'s GLIDE traffic signal control system. GLIDE is the local name of Sydney Coordinated Adaptive Traffic System (SCATS) and is one of the state-of-the-art adaptive traffic signal control systems [24] widely used in many countries. As such, for simulation scenarios without the local neural-network-based controllers, the signal plans selected and executed by GLIDE are implemented in the traffic network at the respective intersections as the traffic loading at each intersection changes with time. The information obtained from GLIDE includes the position of the loop detectors at the various approaches (or links), the number of signal phases, traffic movements of each signal phases, and the number of lanes for each approach (or link). Based on this information, a signalized intersection (termed as "node" in PARAMICS) can be configured so that it represents the actual intersection. However, these data are insufficient to fully configure the signalized intersection because it does not have the values for the various traffic signal control parameters such as green time, cycle length, etc. Additionally, there is no information on the number of vehicles that pass through each approach during the day. Hence, two more sets of data need to be collected and interpreted to obtain information pertaining to these aspects. The number of vehicles that pass through the various approaches of an intersection is recorded via the loop detectors implemented at the intersection in the CBD. This information is collated and presented on a daily basis. From this information, an estimation of the number of vehicles passing through the various approaches can be derived, and the vehicle flow can thus be coded into the simulated traffic network in PARAMICS.

VII. Results and Discussions

Three types of simulations as previously described are used to evaluate the performance of the multiagent systems and the GLIDE benchmark. The typical scenario with a single morning peak is used to test the response for the multiagent systems when they are implemented to provide real-time traffic signal

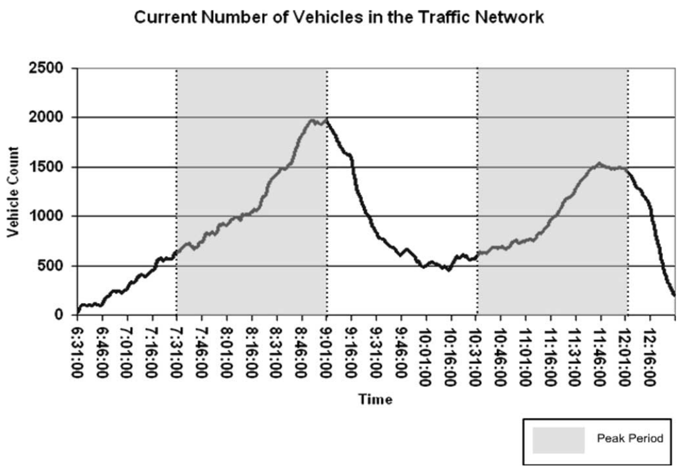


Fig. 10. Number of vehicles for a short extreme scenario with two peaks.

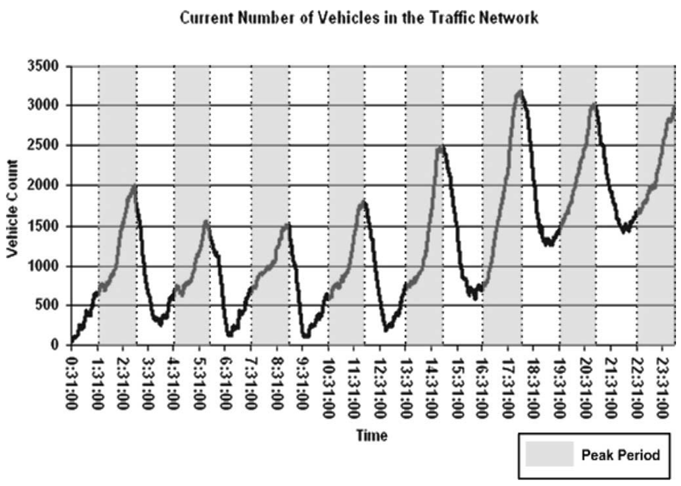


Fig. 11. Number of vehicles for an extreme scenario with multiple peaks.

control of a large traffic network for a short duration. The short extreme scenario is designed to increase the level of difficulty of a short simulation run by having two peak periods closely packed together. These two short scenarios are basically used to verify the efficacy of the multiagent systems before they are tested for the infinite horizon problem. The long extreme scenario with multiple peaks lasting 24 h simulates a fictitious scenario, where numerous peak periods are cascaded closely together within a 24-h period. This fictitious scenario greatly increases the level of difficulty of the dynamic problem given that the multiagent systems will now have to deal with more frequent fluctuations of traffic volume. With that, it can be said that the 24-h simulation (extreme) is an approximation of an infinite horizon problem.

Traffic volume will increase substantially during these peak periods. Even though the increase in demand during the different peak periods is largely similar, the number of vehicles that are actually released into the traffic network varies according to the random seeds that are set before the simulations. Considering that PARAMICS is able to model various characteristics (such as gap acceptance, lane changing, driver aggression, car following, etc.) of vehicles on an individual basis, the outcome of each simulation run varies with the use of different random seeds. The following figures show how the current number of vehicles in the traffic network typically changes for the three different types of simulations. Note that in Figs. 9–11, a gray-shaded region denotes a single-peak period.

A. Three-Hour Simulations

For the typical scenario with morning peak (3 h), 15 separate simulation runs using different random seeds were carried out for each control technique (SPSA-NN, hybrid NN-based multiagent system, and GLIDE). Because the variances of the outcomes of the simulations are small, the average values are taken to be reasonable representation of a typical outcome. The control schemes and the GLIDE benchmark are evaluated using all two performance measures, namely 1) total mean delay of vehicles and 2) current mean speed of vehicle. Table II shows the respective average values of the total mean delay of vehicles and the current vehicle mean speed at the end of the peak period (i.e., at 0900 h) for all these control techniques. The results show that the SPSA-NN-based multiagent system as well as the hybrid NN-based multiagent system outperforms the GLIDE signal plans at the end of the 3-h simulation runs.

TABLE II
TOTAL MEAN DELAY AND MEAN SPEED FOR TYPICAL SCENARIO WITH MORNING PEAK (3 h)

Control technique	Total Mean Delay (sec per vehicle)	Current Vehicle Mean Speed (mph)
	1st Peak Period	1st Peak Period
SPSA-NN	420	7
Hybrid NN	460	5
GLIDE	470	5

B. Six-Hour Simulations

The 6-h short extreme scenario is similar to the 3-h simulations, wherein the total mean delay of vehicles and current vehicle mean speed are used to evaluate the performances of the techniques involved. The results for a 6-h simulation with two peak periods are shown in Table III. Like in the previous case, results shown in Table III were obtained by taking the average of 15 separate simulation runs using different random seeds for each control method. It can be observed that the performance levels of the multiagent systems at the end of the second peak period are rather similar compared to their respective performance levels at the end of the first peak period. The GLIDE benchmark, on the other hand, shows a sign of degradation. However, it has been verified experimentally (via observing PARAMICS Modeler) that all three techniques managed to ease the traffic congestion at the end of the 6-h simulations even though the traffic network is somewhat more loaded when the GLIDE benchmark is being implemented. It is also observed that the current mean speed of vehicles recovers at the fastest rate at the end of the second peak period when the hybrid NN-based multiagent system is implemented into the traffic network.

TABLE III
TOTAL MEAN DELAY FOR SHORT EXTREME SCENARIO WITH TWO PEAKS

Control technique	Total Mean Delay (sec per vehicle)		Current Vehicle Mean Speed (mph)	
	1st Peak Period	2nd Peak Period	1st Peak Period	2nd Peak Period
SPSA-NN	420	510	5	5
Hybrid NN	400	470	7	8
GLIDE	500	650	5	3

TABLE IV
TOTAL MEAN DELAY FOR THE LONG EXTREME SCENARIO WITH MULTIPLE PEAKS (24 h)

Control technique	Total Mean Delay (sec per vehicle)							
	1st Peak Period	2nd Peak Period	3rd Peak Period	4th Peak Period	5th Peak Period	6th Peak Period	7th Peak Period	8th Peak Period
SPSA-NN	600	750	600	550	500	500	500	850
Hybrid NN	400	450	450	450	450	500	650	700
GLIDE	400	500	600	650	800	1500	2300	3200

TABLE V
CURRENT VEHICLE MEAN SPEED FOR THE LONG EXTREME SCENARIO WITH MULTIPLE PEAKS (24 h)

Control technique	Current Vehicle Mean Speed (mph)							
	1st Peak Period	2nd Peak Period	3rd Peak Period	4th Peak Period	5th Peak Period	6th Peak Period	7th Peak Period	8th Peak Period
SPSA-NN	5	10	15	7.5	4	5	4	0
Hybrid NN	7	10	14	7.5	6	4	4	4
GLIDE	7	5	5	5	0	0	0	0

C. Long Extreme Scenario With Multiple Peaks (24 h)

Based on the successful results of the 3-h and 6-h simulations, the control techniques are tested in simulation runs that have significantly longer durations. The long extreme scenario with multiple peaks is a hypothetical scenario designed to test the robustness of the different techniques under extreme conditions. Six separate runs are carried out using different random seeds for each of the three control techniques. Once again, it has been observed that the variances of all the simulation runs that are performed for a single control technique are small. Hence, the mean of these values provides a good representation of a typical outcome for that particular control technique.

The simulation results are summarized in Tables IV and V. It is evident that the GLIDE benchmark as well as the SPSA-NN-based multiagent system fails to provide effective real-time traffic signal control for the 24-h simulation period. It can be observed that the total mean delay of vehicles begins to increase steadily during the fifth peak period when the GLIDE benchmark is being implemented. When the SPSA-NN-based multiagent system is implemented, the average value of the total mean delay starts to increase after the seventh peak period (at around 2100 h). A similar degradation in performance is also reflected in the current vehicle mean speed.

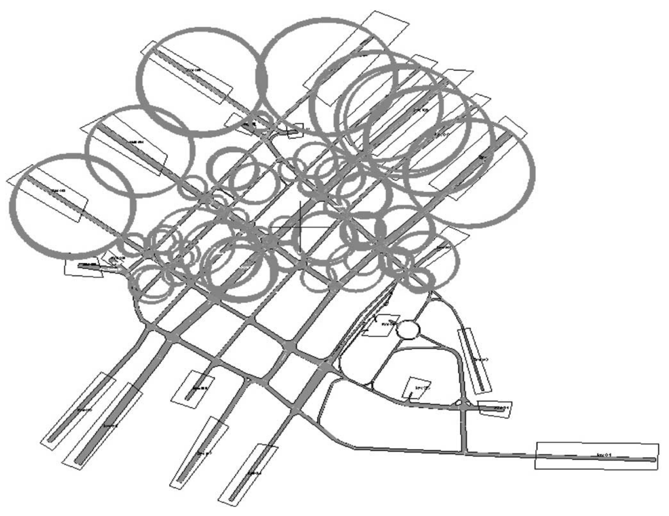


Fig. 12. Traffic network controlled by SPSA-NN-based multiagent system after 24 h.

In Table V, it is shown that the current vehicle mean speed comes to zero at the end of the fifth peak period when the GLIDE benchmark is implemented. When SPSA-NN-based multiagent system is implemented, the current vehicle mean speed drops to almost zero after the seventh peak period. On the other hand, the hybrid NN-based multiagent system demonstrates its capability to provide effective real-time traffic signal control even under this long extreme scenario. The average value of the total mean delay of vehicles as well as the current vehicle mean speed is maintained at a reasonable level throughout the duration of the simulations.

It should be noted that the main reason for the overall drop in performance for this long extreme scenario is the large number of vehicle injection during the eight peak periods. The frequency with which the traffic volume fluctuates is much higher (as shown in Fig. 11) in this fictitious extreme scenario compared to the former two cases, requiring the control techniques to quickly adapt to cope with these frequent fluctuations in traffic volume. Given this difficulty as well as the existence of other constraints such as the actual capacity of the traffic network, it is largely expected that the overall performance of all three techniques will drop and the total mean delay will increase.

In order to better illustrate the conditions of the traffic network at the end of the 24-h simulation runs, two two-dimensional screenshots of the traffic network are taken from the PARAMICS Modeler when the traffic network is controlled by the SPSA-NN-based multiagent system (Fig. 12) and hybrid NN-based multiagent system (Fig. 13). The two screenshots are captured at 0030 h at the end of the 24 h.

The PARAMICS modeling environment was preset to denote 13 stopped or queued vehicles with a hotspot or red circle. As shown in Fig. 12, the traffic network evolves into a pathological state with oversaturation at 0030 h with the SPSA-NN-based multiagent system due to the steady increase of mean delay and mean stoppage time after 17 h. The number of congested links is well over 30. Using the hybrid NN-based multiagent system, congestions are confined to the upper section of the traffic network (refer to Fig. 13), and the number of congested links is reduced to less than ten.

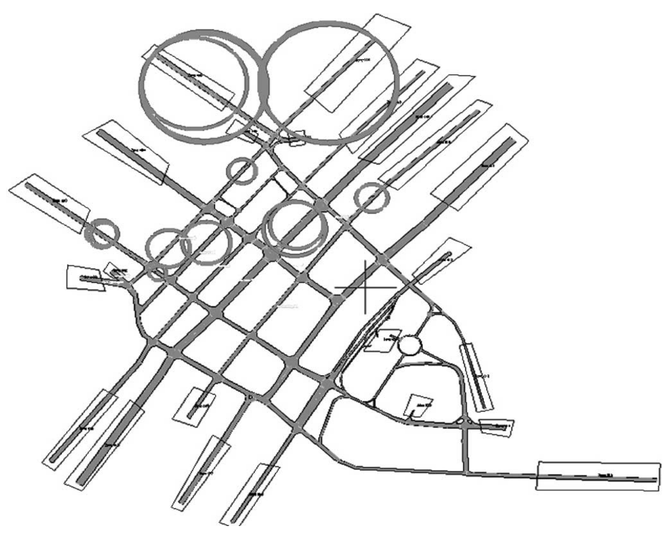


Fig. 13. Traffic network controlled by hybrid NN-based multiagent system after 24 h.

D. Continuous Online Learning Capabilities of the NN Models

For the approximated version of the infinite horizon problem, it is very important that the control functions of individual agents can produce good approximates of the optimal control function each time the distributed control problem is restated. Hence, the control algorithm as well as the whole multiagent system needs to have a high level of adaptability. This implies that the learning period for the multiagent systems will have to vary from time to time depending on the nature of the dynamic problem.

From the simulation results, it is observed that in comparison to GLIDE, the overall percentage reduction in mean delay was 8%, 44%, and 50% for 3-h, 6-h, and 24-h simulations, respectively, when SPSA-NN was used, whereas with the hybrid NN-based multiagent system, the percentage reduction in mean delay was 23%, 36%, and 78%, respectively.

These results indicate that the hybrid NN-based multiagent system is able to effectively control the traffic signals in a dynamically changing environment even as the complexity of the problem increases. This indicates that the hybrid NN-based multiagent system is able to adjust its weights parameters effectively throughout the duration of the simulation so that the signal plans it generates can accommodate the periodic as well as random fluctuations of traffic volumes.

The differences in the performance of the hybrid NN-based multiagent system and the SPSA-NN-based multiagent system maybe due to their individual weight update algorithms. For the SPSA-NN-based multiagent system, the weight update algorithm follows the form of (9), and the gain sequence (or learning rate) has to satisfy the classical stochastic approximation conditions defined in (4). Choosing an appropriate form for the gain sequence is not a trivial matter as it would affect the performance of the NN in the long run. This is because the SPSA algorithm converges under conditions stated in [6], and when SPSA is applied to adjust the neural weights, the convergence property may result in premature convergence of the neural weights. The weight update algorithm for the hybrid NN-based multiagent system, however, is performed at various stages, each involving tuning the weight parameters, learning rate, and the neural connection in response to the changes in the environment. Once the hybrid NN becomes a good approximate of the optimal control function, the neural weights will be updated by a smaller amount (or possibly not updated at all). If, on the other hand, the external random dynamics produce a significant change in the existing dynamics of the environment (e.g., random injection of large amount of vehicles into the traffic network), negative reinforcements will be received and weights are updated again as the control functions do not continue to be a good approximate of the optimal control functions anymore.

E. Cooperation and Learning Mechanisms of the Multiagent Systems

The SPSA-NN-based multiagent system is a nonhierarchical multiagent system with fixed cooperative zones. Hence, communication-based cooperation mechanisms (CCMs) are only present within each cooperative zone and are essentially in the form of lateral communication channels. Such a design results in a less complex multiagent system and eases the process of implementation and offline optimization of the various control parameters. However, the tradeoff is that each agent within the SPSA-NN-based multiagent system will not have the chance to obtain the overall perspective of the traffic network (i.e., the regional perspective), and consequently, it can only learn based on the perspectives it derived from its own cooperative zone. At the same time, its collaborative actions are limited to within its cooperative zone only. These limitations contribute toward the factors affecting the performance of the SPSA-NN-based multiagent system in the 24-h extreme scenario with multiple peaks.

The agents in the hybrid NN-based multiagent system, on the other hand, use mainly vertical-based communication channels given the nature of its structure. Individual agents relay their cooperative perspectives to their parent agents, which in turn serve as arbitrators for them. Each agent also uses a multistage online learning process. As a result of the hierarchical structure and the multistage online learning process, this multiagent system has a more complex design. Additionally, the process of implementation as well as offline optimization of its control parameters is also more time consuming. In this system, each agent has the opportunity to gain the overall perspective of the traffic network despite the fact that its cooperative zone is fixed in size. The regional scope of the collaborative actions proves to be advantageous in the 24-h extreme simulation scenario, where traffic flow fluctuates frequently over a long period of time.

VIII. Conclusion

A hybrid NN-based multiagent system involving a novel multistage online learning process has been successfully tested together with an enhanced SPSA-NN-based multiagent system in highly complex traffic simulation scenarios. Simulation results using actual data showed that both multiagent systems achieved an overall better performance compared to GLIDE for the three simulation scenarios, the hybrid NN-based multiagent system demonstrating clearly superior performance for the 24-h extreme scenario. The results suggest that the hybrid NN-based multiagent system can provide effective control of the large-scale traffic network even as the complexity of the simulation increases. This research work has extended the application of hybrid computational intelligence techniques to a large-scale real-world application using an approximated version of an infinite horizon problem. For such applications, the concept of effective continuous learning is of utmost importance given the undesirability of having to retune the controllers from time to time.

ACKNOWLEDGMENT

The authors would like to thank the Land Transportation Authority of Singapore for providing data necessary for the simulation modeling.

REFERENCES

[1] N. J. Garber and L. A. Hoel, *Traffic and Highway Engineering*, 2nd ed. Boston, MA: PWS-Kent, 1997, pp. 281–329.
[2] S. Chiu and S. Chand, "Self-organizing traffic control via fuzzy logic," in *Proc. 32nd IEEE Conf. Decision Control*, 1993, pp. 1987–1902.
[3] G. Nakamiti and F. Gomide, "Fuzzy sets in distributed traffic control," in *Proc. 5th IEEE Int. Conf. Fuzzy Syst.*, 1996, pp. 1617–1623.
[4] S. Mikami and Y. Kakazu, "Genetic reinforcement learning for cooperative traffic signal control," in *Proc. 1st IEEE Conf. Evol. Comput.*, 1994, vol. 1, pp. 223–228.
[5] W. Wei and Y. Zhang, "FL-FN based traffic signal control," in *Proc. FUZZ-IEEE*, May 12–17, 2002, vol. 1, pp. 296–300.
[6] J. C. Spall and D. C. Chin, "Traffic-responsive signal timing for system-wide traffic control," *Transp. Res.—C*, vol. 5, no. 3/4, pp. 153–163, 1997.
[7] E. Bingham, "Reinforcement learning in neural fuzzy traffic signal control," *Eur. J. Oper. Res.*, vol. 131, no. 2, pp. 232–241, Jun. 2001.
[8] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Cooperative, hybrid agent architecture for real-time traffic control," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 5, pp. 597–607, Sep. 2003.
[9] M. Baglietto, T. Parisini, and R. Zoppoli, "Distributed-information neural control: The case of dynamic routing in traffic networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 3, pp. 485–502, May 2001.
[10] T. Parisini, M. Sanguineti, and R. Zoppoli, "Nonlinear stabilization by receding-horizon neural regulators," *Int. J. Control*, vol. 70, no. 3, pp. 341–362, 1998.
[11] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Comput.*, vol. 6, no. 6, pp. 1185–1201, Nov. 1994.
[12] M. Littman and C. Szepesvari, "A generalized reinforcement learning model: Convergence and applications," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 310–318.
[13] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1994.
[14] A. V. Wouwer, C. Renotte, and M. Remy, "On the use of simultaneous perturbation stochastic approximation for neural network training," in *Proc. Amer. Control Conf.*, 1999, pp. 388–392.
[15] E. Gomez-Ramirez, P. L. Najim, and E. Ikonen, "Stochastic learning control for nonlinear systems," in *Proc. IJCNN*, May 2002, vol. 1, pp. 171–176.
[16] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, no. 22, pp. 382–386, 1951.
[17] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[18] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.
[19] Z. Luo, "On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks," *Neural Comput.*, vol. 3, no. 2, pp. 226–245, 1991.
[20] T. Kohonen, *Self-Organizing Maps*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997.
[21] G. Yan, F. Yang, T. Hickey, and M. Goldstein, "Coordination of exploration and exploitation in a dynamic environment," in *Proc. IJCNN*, 2001, pp. 1014–1018.
[22] *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. New York: IOP, 1997.
[23] *PARAMICS Modeller v4.0 User Guide and Reference Manual*, Quadstone Ltd., Edinburgh, U.K., 2002.
[24] P. B. Wolshon and W. C. Taylor, "Analysis of intersection delay under real-time adaptive signal control," *Transp. Res., Part C Emerg. Technol.*, vol. 7C, no. 1, pp. 53–72, Feb. 1999.

Dipti Srinivasan (M'89–SM'02) received the Ph.D. degree in engineering from National University of Singapore, Singapore, in 1994.

She worked as a Postdoctoral Researcher at the University of California, Berkeley, from 1994 to 1995 before joining the National University of Singapore, where she is currently an Associate Professor in the Department of Electrical and Computer Engineering. Her research interest is the application of soft computing techniques to engineering optimization and control problems.

Min Chee Choy is currently working toward the Ph.D. degree at the National University of Singapore, Singapore.

His research interests include distributed object computing, hybrid computational intelligence techniques, online-reinforcement learning, and intelligent transportation systems.

Ruey Long Cheu (M'01) received the B.Eng. and M.Eng. degrees from the National University of Singapore (NUS), Singapore, in 1987 and 1990 respectively, and the Ph.D. degree from the University of California, Irvine, in 1994.

He is currently an Associate Professor in the Department of Civil Engineering, NUS. His research interests are intelligent transportation systems with emphasis on the applications of artificial intelligence and emerging computing techniques in transportation.