Cooperative, Hybrid Agent Architecture for Real-Time Traffic Signal Control

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

***Abstract*—This paper presents a new hybrid, synergistic approach in applying computational intelligence concepts to implement a cooperative, hierarchical, multiagent system for real-time traffic signal control of a complex traffic network. The large-scale traffic signal control problem is divided into various subproblems, and each subproblem is handled by an intelligent agent with fuzzy neural decision-making module. The decisions made by lower-level agents are mediated by their respective higher-level agents. Through adopting a cooperative distributed problem solving approach, coordinated control by the agents is achieved. In order for the multiagent architecture to adapt itself continuously to the dynamically changing problem domain, a multistage online learning process for each agent is implemented involving reinforcement learning, learning rate and weight adjustment as well as dynamic update of fuzzy relations using evolutionary algorithm. The test bed used for this research is a section of the Central Business District of Singapore. The performance of the proposed multiagent architecture is evaluated against the set of signal plans used by the current real-time adaptive traffic control system. The multiagent architecture produces significant improvements in the conditions of the traffic network, reducing the total mean delay by 40% and total vehicle stoppage time by 50%.**

***Index Terms*—Cooperative systems, fuzzy neural networks, online learning, multiagent system, real-time traffic signal control.**

I. Introduction

CONTROL of traffic signals for efficient movement of traffic on urban streets constitutes a challenging part of an urban traffic control system (UTCS). Traffic responsive, closed-loop systems, or adaptive traffic signal systems are becoming increasingly critical for transportation agencies to meet their day-to-day operation and management needs [1]. For a large-scale UTCS, it may be difficult or impossible to tell whether the traffic network is flowing smoothly and can asses its current state. In addition, due to some of the nonlinear and stochastic traffic processes in a traffic network, predicting the effects of modifying any of the traffic control parameters is not easy. As such, besides the classical traffic control techniques [2], informed intervention is needed in the form of coordination between various controllers in the network to prevent the traffic network from degenerating into a pathological state where the vehicle progress slows or stops completely. Also, an additional challenge would be implementing adaptive traffic control in real-time, particularly for online signal optimization, as proposed by earlier researches in this field [3]. In view of these challenges, a multiagent architecture that incorporates computational intelligence techniques is implemented in this research to provide effective real-time traffic control. Various issues of intelligent cooperative problem solving in a multiagent system through effective coordination, communication, knowledge acquisition, decision-making, goal formulation, and online learning are investigated in this research.

Several researches have tried to realize distributed control, which acquires optimal signal control over a group of signals without supervision. Various types of multiagent systems have been applied in these works [4]–[7]. In some cases, the researches dealt with simplified traffic conditions, such as regularly located intersections [4] and inclusion of one-way roads [5] while others have used a section of the real-world scenario [8]. In [9], the agents are in the form of traffic fuzzy controller. In [6], genetic algorithm and reinforcement learning are applied in implementing the multiagent system, which is self-organizing.

Based on the studies, it has been found that more research needs to be done to implement a systematic, unsupervised, distributed control scheme for testing in a complex traffic network simulated for a real world scenario. The concept of cooperation among multiagents to solve a complex distributed problem has been applied in the traffic domain as seen in some of the mentioned works. However, much needs to be done to quantify the level of cooperation as well as to systemize the cooperation mechanism.

The main objectives of this research is to address this need and develop a new distributed, cooperative problem solving approach through the use of multiple interacting, autonomous agents to provide effective signal control strategies for real-time traffic management of an arterial network. This multiagent system is designed to leverage on the synergistic relationship between neural network and fuzzy logic systems [10], [11]. It also introduces an innovative multistage approach for online learning and self-organization of the agents via reinforcement learning [12], [13] and fuzzy rules adjustment by evolutionary algorithm [14], [15].

II. Hierarchical Multiagent Architecture

The multiagent architecture is designed in a hierarchical manner to provide different levels of control for the traffic network. The architecture consists of three layers. The lowest

Manuscript received July 15, 2002; revised May 1, 2003. This paper was recommended by Guest Editor H. Zhang.

M. C. Choy and D. Srinivasan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: engp1637@nus.edu.sg, dipti@nus.edu.sg).

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

Digital Object Identifier 10.1109/TSMCA.2003.817394

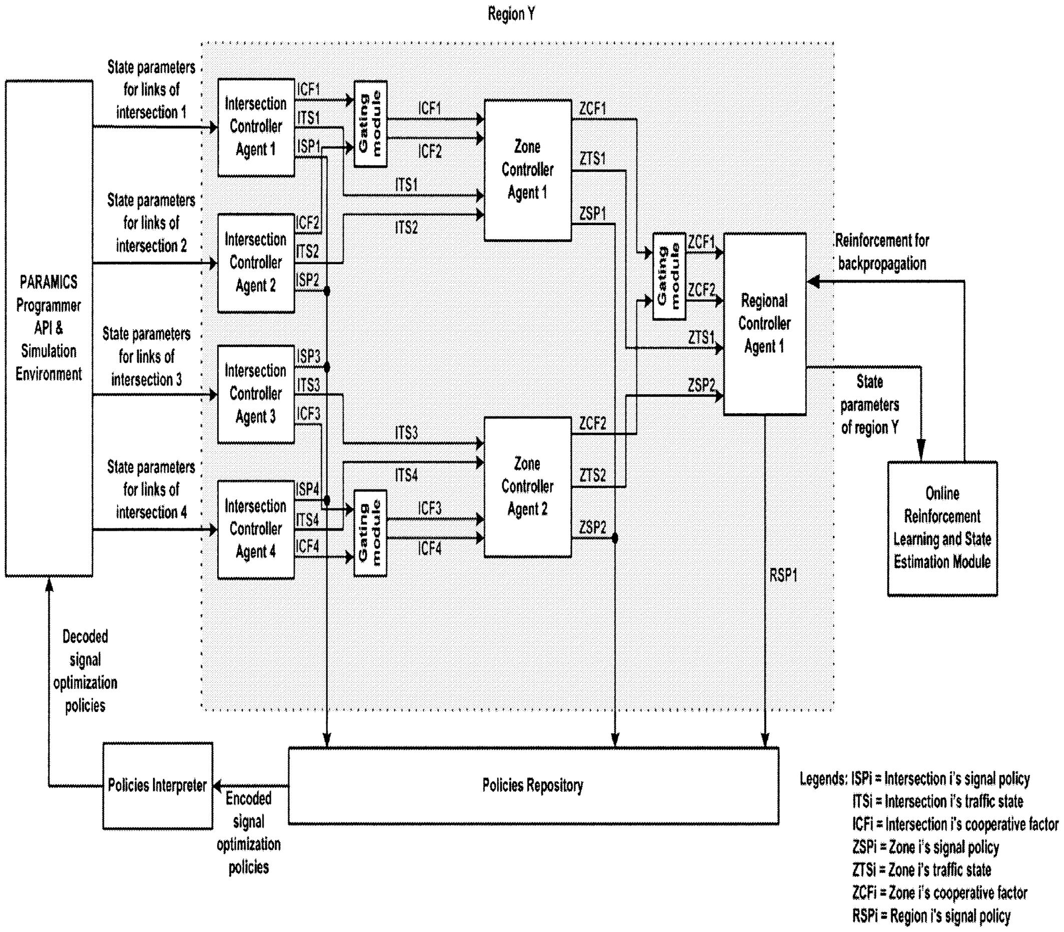

Region Y
PARAMICS Programmer API & Simulation Environment
State parameters for links of intersection 1
State parameters for links of intersection 2
State parameters for links of intersection 3
State parameters for links of intersection 4
Intersection Controller Agent 1
Intersection Controller Agent 2
Intersection Controller Agent 3
Intersection Controller Agent 4
Gating module
Zone Controller Agent 1
Zone Controller Agent 2
Regional Controller Agent 1
Reinforcement for backpropagation
State parameters of region Y
Online Reinforcement Learning and State Estimation Module
ICF1 ITS1 ISP1 ICF2 ITS2 ISP2 ISP3 ITS3 ICF3 ISP4 ITS4 ICF4
ZCF1 ZTS1 ZSP1 ZCF2 ZTS2 ZSP2 RSP1
Decoded signal optimization policies
Policies Interpreter
Encoded signal optimization policies
Policies Repository
Legends: ISPi = Intersection i's signal policy
ITSi = Intersection i's traffic state
ICFi = Intersection i's cooperative factor
ZSPi = Zone i's signal policy
ZTSi = Zone i's traffic state
ZCFi = Zone i's cooperative factor
RSPi = Region i's signal policy

Fig. 1. Schematic diagram of the multiagent architecture.

layer consists of intersection controller agents (ICA) that control individual, preassigned intersections in the traffic network. The middle layer consists of zone controller agents (ZCA) that controls several preassigned ICAs. The highest level consists of one regional controller agent (RCA) controls all the ZCAs. The three-layered multiagent architecture is shown in Fig. 1.

The problem of real-time network-wide signal control is divided into several subproblems, each with a different scale and magnitude. Individual agents from each layer of the multiagent architecture are tasked to manage the respective sub-problems according to their hierarchy. Each agent is a concurrent logical process capable of querying the environment (e.g., sensors and agents in the lower hierarchy within its control) and making decisions autonomously. The agents in each layer decide the policies (which comprise of signal timing plan adjustments and the direction of offset coordination) and levels of cooperation (defined by the cooperative factor) that they deem appropriate based on the conditions of the intersections, or set of intersections under their jurisdictions. Besides having higher-level traffic network parameters as inputs to their decision-making process, the higher-level agents obtain the cooperative factors recommended by their lower level agents as inputs too (Fig. 2 shows that the intersection cooperative factors recommended by the lower level ICAs are part of the inputs of a ZCA). Based on these inputs, the decision-making process of the higher level agents may present a set of higher-level policies that are different from those policies recommended by their lower level agents or they may choose to follow the lower level policies.

The policy repository is a dynamic database for storing all the policies recommended by the controller agents of all levels at the end of each evaluation period. The end of an evaluation period is indicated when all the intersections have finished their current signal phases. After each period, the previously recommended policies are updated with a new set of policies. The policy repository then performs arbitration and conflict resolution for the entire set of recommended polices. The arbitration process gives priority to higher-level policies. However, since one of the outputs, namely the cooperative factor, of the lower-level agents are part of the inputs of the higher-level agents (as mentioned earlier), these lower-level decisions affect directly the outcomes of the higher-level agent's decision-making process. As such, lower-level policies are not starved off by the arbitration process. Following the arbitration process, the set of chosen policies is sent to the policy interpreter. The function of the policy interpreter is to translate the chosen set of policies into actions, which may result in adjustment of the various signal-timing parameters such as phase-length, cycle-time, direction of offset coordination, for the affected intersections.

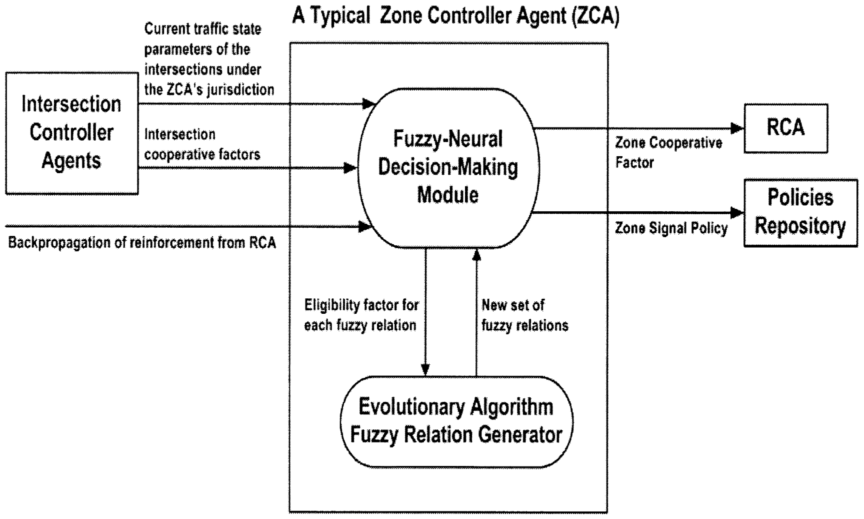

A Typical Zone Controller Agent (ZCA)
Intersection Controller Agents
Current traffic state parameters of the intersections under the ZCA's jurisdiction
Intersection cooperative factors
Backpropagation of reinforcement from RCA
Fuzzy-Neural Decision-Making Module
Zone Cooperative Factor
RCA
Zone Signal Policy
Policies Repository
Eligibility factor for each fuzzy relation
New set of fuzzy relations
Evolutionary Algorithm Fuzzy Relation Generator

Fig. 2. Schematic diagram of a zone controller agent.

III. Fuzzy-Neural Decision Making Module

Within each autonomous agent, the fuzzy-neural decision making (FNDM) module comprises of a knowledge base and an inference engine. As shown in Fig. 3, the FNDM module consists of two main functional blocks, namely, the signal policy inference engine for generating an appropriate policy (for this research, the policy refers to a traffic signal policy) and the cooperative factor inference engine for generating a suitable cooperative factor. A cooperative factor is one of the two outputs generated by a lower level agent which is in turn fed to its parent (higher-level) agent. It indicates the level of cooperation that the lower level agent deems appropriate for the current traffic conditions within its jurisdiction.

The architecture of the FNDM module follows the multilayer feed-forward neural network approach that mimics the fuzzy reasoning mechanism [16], [17]. As such, the fuzzy-neural architecture consists of five layers to represent, in between the layers, the fuzzification, implication, consequent and defuzzification processes of the fuzzy reasoning mechanism. Using this approach, the architecture provides means to justify the choices of signal policy and level of cooperation (defined by the cooperative factor).

Each ICA takes in the lane-specific occupancy, flow and rate of change of flow of the different intersection approaches as inputs. The occupancy, flow and rate of change of flow are based on the measurements by loop detectors (refer to Section V part B for more details on the loop detectors) during the phase when traffic lights are green. In order to quantify the traffic conditions of the intersections in a zone, the FNDM module of the ZCA takes in each intersection's representative occupancy, flow and rate of change of flow as its inputs. The fuzzy sets of *occupancy*, *flow* and *rate of change of traffic volume* have three linguistic labels, namely *high*, *medium* and *low* to describe the respective degrees of membership (Gaussian membership function). Besides these inputs, as mentioned in the previous section, the ZCA also takes in the intersection cooperative factors recommended by the respective ICAs under its jurisdiction (to reflect the level of cooperation each ICA sees fit for its own intersection, all of which are within the zone controlled by the ZCA). The antecedents of the fuzzy rules are defined by properly linking the nodes in the second layer to those in the third layer. The third layer fires each rule based on the T-norm fuzzy operation, implemented using the *MIN* operator.

Nodes in the third layer define the degree of current traffic loading of the zone (i.e., *high*, *medium* and *low loads*) and the level of cooperation needed for the intersections within the zone (i.e., *high*, *medium and low degrees of cooperation*). The nodes in the fourth layer represent the various consequents that correspond to the fuzzy rules in the FNDM module. For the signal policy inference engine, the consequents consist of the various signal improvement/control policies. For the cooperation factor inference engine, the consequents consist of the various possible levels of cooperation. Since some of the fuzzy rules share the same consequent, the S-norm fuzzy operation is used to integrate the rules. For this research, the S-norm fuzzy operation is implemented using the *MAX* operator.

Finally, the fifth layer performs the defuzzification process in order to obtain crisp values correspond to the chosen signal policy and cooperative factor (i.e., outputs of the FNDM module for each agent). The architecture of the FNDM module for the ICA and RCA is largely similar to the one described for the ZCA. The main difference lies in the inputs and due to the hierarchical nature of the overall multiagent architecture. For example, for RCA, the inputs are the traffic parameters from all the zones in the region as well as the zonal cooperative factors recommended by the ZCAs while the outputs are the regional level signal policy and the regional level cooperative factor. An example of a rule in the FNDM is as follows:

IF {(*overall aggregate occupancy* is high) and (*overall aggregate flow* is high) and (*overall aggregate rate of change of traffic volume* is high)}
THEN {(*traffic loading* is high) and (*level of cooperation needed* is high)}

IV. Online Adaptation by the Agents

In this research, several techniques have been applied to facilitate online adaptation by the agents in the dynamically

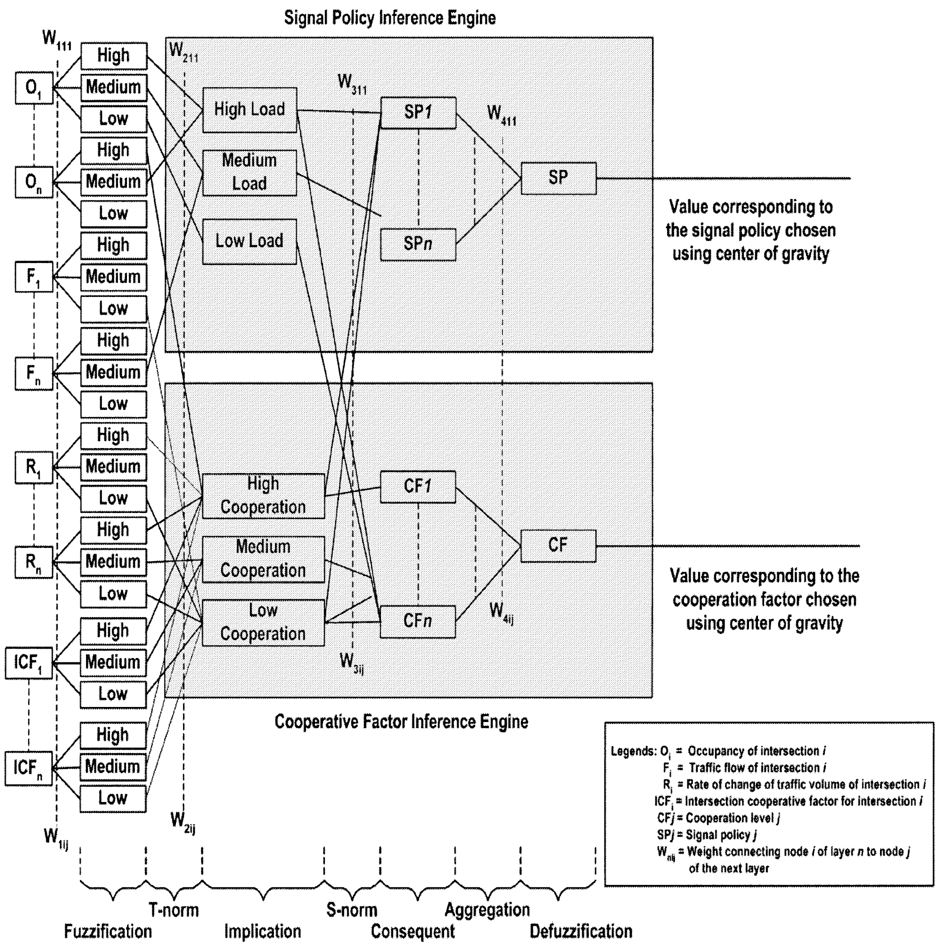

Signal Policy Inference Engine
W_{111}
W_{211}
W_{311}
W_{411}
O_1
O_n
F_1
F_n
R_1
R_n
ICF_1
ICF_n
High
Medium
Low
High Load
Medium Load
Low Load
SP1
SPn
SP
Value corresponding to the signal policy chosen using center of gravity
High Cooperation
Medium Cooperation
Low Cooperation
CF1
CFn
CF
W_{4ij}
W_{3ij}
Value corresponding to the cooperation factor chosen using center of gravity
Cooperative Factor Inference Engine
W_{1ij}
W_{2ij}
Legends: O_i = Occupancy of intersection i
F_i = Traffic flow of intersection i
R_i = Rate of change of traffic volume of intersection i
ICF_i = Intersection cooperative factor for intersection i
CFj = Cooperation level j
SPj = Signal policy j
W_{nij} = Weight connecting node i of layer n to node j of the next layer
T-norm
S-norm
Aggregation
Fuzzification
Implication
Consequent
Defuzzification

Fig. 3. Fuzzy-neural decision making module of a zone controller agent.

changing problem domain of a traffic network. Online adaptation mainly takes the form of a multistage online learning process as shown in Fig. 4. This process primarily consists of three subprocesses: reinforcement learning, weight adjustment and adjustment of fuzzy relations. Reinforcement learning is first performed. The reinforcement obtained from this process is backpropagated to the RCA and subsequently, to all the lower-level agents. Following which, each agent proceeds 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 agent received). When that is done, each agent adjusts the weights of the neurons in its FNDM module according to the topological weight update method. Finally, the reinforcement is used to update the fitness value of each neuron in the agent's FNDM module. 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) will be updated using the evolutionary algorithm fuzzy relation generator (EAFRG). Sections IV-A–IV-C will describe in details the various mechanisms of the multistage online learning process.

A. Online Reinforcement Learning Process

In view of the various advantages of reinforcement learning [12], [13], this concept is adopted in the multiagent architecture to develop an unsupervised, online learning mechanism based on feedbacks from the environment (whose state is in turn affected by the policies recommended by the multiagent architecture). Fig. 1 shows that the online reinforcement-learning (ORL) module is part of the multiagent architecture. The role of the ORL module is to generate reinforcements which are to be backpropagated to the agents to facilitate dynamic adjustment of their rule-bases represented by their FNDM modules. The ORL module is implemented using similar fuzzy-neural concepts that are applied in developing the FNDM module (refer to Fig. 5). The ORL module is designed to generate reinforcement based on comparison of the current estimated state of the zone with the previous state. The reinforcement signal R that is generated from the ORL module can be derived as follows:

$$R = \lambda\,(S_c - S_p) - (S_c - S_b) \tag{1}$$

where λ is the state change sensitivity constant (determined empirically), S_c is the current state value, S_b is the best state value, and S_p is the previous state value. For there to be a positive reinforcement, it is necessary that $S_c > S_p$ and $\lambda\{S_c - S_p\} > (S_c - S_b)$. Using the backpropagation technique, the change of weight ΔW_{ij} from neuron i to the activated output neuron j is as follows:

$$\Delta W_{ij} = -\eta\left(\frac{\partial R}{\partial W_{ij}}\right) = \eta S_j Y_i \tag{2}$$

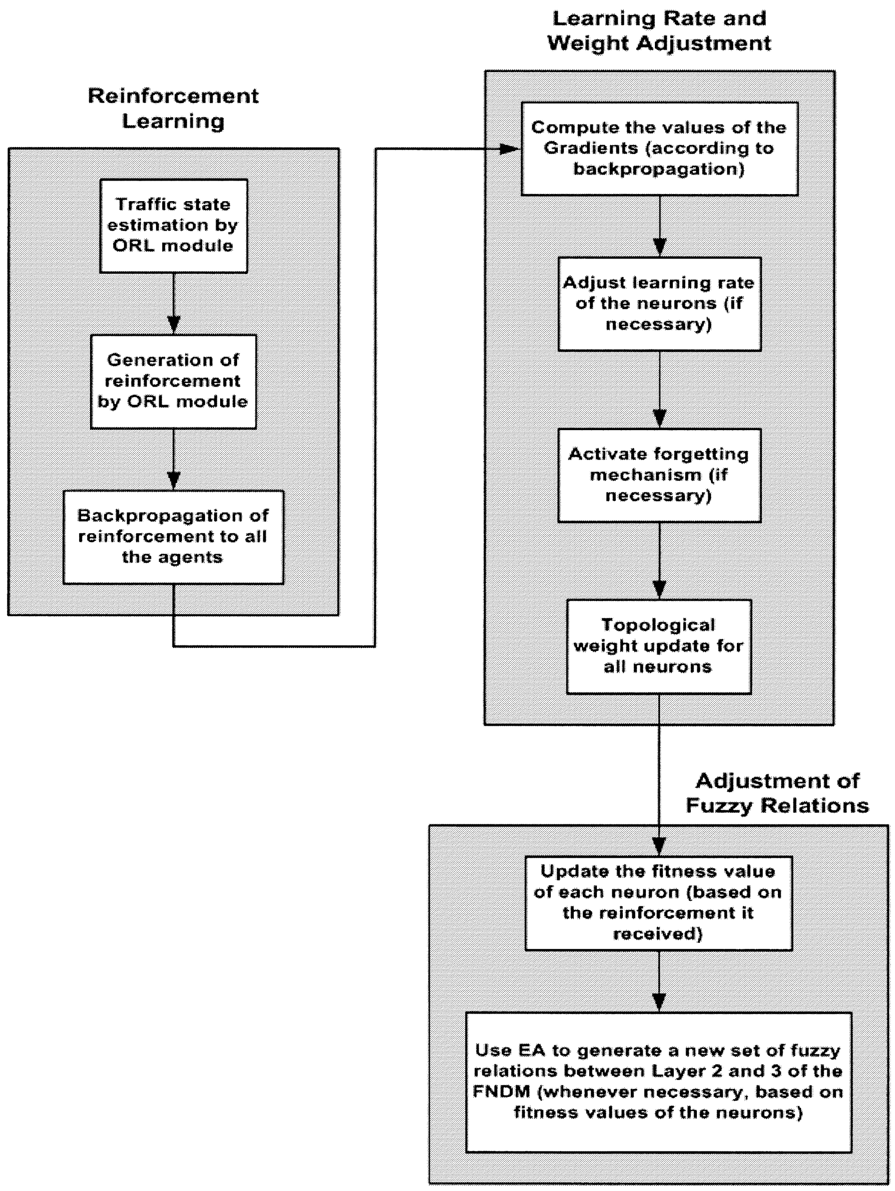

Reinforcement Learning
Traffic state estimation by ORL module
Generation of reinforcement by ORL module
Backpropagation of reinforcement to all the agents
Learning Rate and Weight Adjustment
Compute the values of the Gradients (according to backpropagation)
Adjust learning rate of the neurons (if necessary)
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 it received)
Use EA to generate a new set of fuzzy relations between Layer 2 and 3 of the FNDM (whenever necessary, based on fitness values of the neurons)

Fig. 4. Multistage online learning process for each agent.

where η is the learning rate, S_j is the gradient for the output neuron j, Y_i is the output of neuron i. Note that

$$S_j = R_j \phi_j^l (V_j) \tag{3}$$

$$V_j = \sum_{i=1}^{m} (W_{ij} Y_j) \tag{4}$$

in which m is the number of inputs for neuron j, R_j is the backpropagated reinforcement value at output node j and ϕ_j is the transfer function for neuron j. The superscript l in ϕ_j denotes the first order derivative of ϕ_j. For hidden layers of the neural network, the local gradient S_i for hidden neuron i with k neurons on its right is defined as follows:

$$S_i = \phi_i^l (V_i) \sum_{j=1}^{k} (S_j W_{ij}) \tag{5}$$

where

$$V_i = \sum_{n=1}^{p} (W_{nj} Y_n) \tag{6}$$

in which p is the number of inputs for neuron i. Hence, if a fuzzy relation represented by a neuron is appropriate for a particular traffic condition, a positive reinforcement in the form of a positive R will be received and vice versa. Upon receiving a reinforcement, each neuron in the FNDM module of an agent can proceed to adjust its learning rate and weight. The details of this process will be described in Section IV-B.

B. Learning Rate and Weight Adjustment Process

The weight of each neuron in the FNDM module of an agent can be adjusted dynamically either by topological weight update or by activating the forgetting mechanism. The learning rate of each neuron can also be adjusted dynamically according

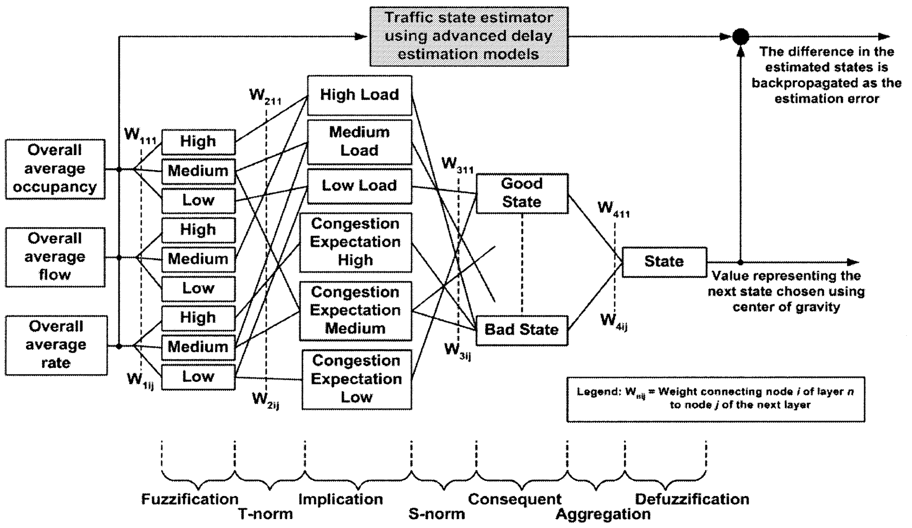

Traffic state estimator using advanced delay estimation models
The difference in the estimated states is backpropagated as the estimation error
Overall average occupancy
Overall average flow
Overall average rate
High
Medium
Low
High Load
Medium Load
Low Load
Congestion Expectation High
Congestion Expectation Medium
Congestion Expectation Low
Good State
Bad State
State
Value representing the next state chosen using center of gravity
Legend: W_nij = Weight connecting node i of layer n to node j of the next layer
Fuzzification
T-norm
Implication
S-norm
Consequent
Aggregation
Defuzzification

Fig. 5. ORL module.

to some well-known methods [18], [19]. Sections IV-B.I–III describe each of these techniques in more details.

Learning Rate Adaptation: According to Fig. 4, the learning rate of each neuron is adjusted first before the topological weight update. This is done dynamically according to the sign of the backpropagated gradient. The adjustments are made according to the guidelines mentioned in [18], [19]. As such, the weigh update equation for neuron j at the nth iteration is as shown below.

$$W_{ij}(n) = W_{ij}(n-1) + \lambda_j(n) h_{jk}(n) Y_i S_j \tag{7}$$

where, W_{ij} is the synaptic weight joining node i to j, h_{jk} is the topological neighborhood of the activated neuron k with respect to neuron j, λ_j is the dynamic learning rate of neuron j, Y_i is the output of neuron i and S_j is the gradient of neuron j.

Topological Weight Update: Unlike the conventional backpropagation method, not all neurons in the FNDM module have their weights updated during the backward pass. Based on the neurobiological evidence for lateral interaction among a set of excited neurons, it is clear that a firing neuron tends to excite neurons within its immediate neighborhood more than the neurons away from it. This observation has also been made in the researches on self-organizing maps (SOM) [20].

Let h_{ji} denote the topological neighborhood centered on the winning neuron i and encompassing a set of excited neurons in which a typical one is denoted by j. Let d_{ji} denote the lateral distance between the winning neuron i and the excited neuron j. Fig. 6 depicts the h_{ji} of the winning neuron i (which is taken to be the middle one for this example) as a function of the d_{ji}. Unlike SOM, the topological network, h_{ji}, is only symmetrical for the neuron in the middle of the layer due to the nature of the fuzzy reasoning mechanism and the position of each neuron. As shown in the figure, the function for h_{ji} is chosen for convenience and other functions such as the Gaussian function can be used instead. It should be noted that the amplitude of h_{ji} decreases with increasing d_{ji}. This is a necessary condition for convergence. However, since the learning process of the FNDM module continues indefinitely in the dynamic traffic network,

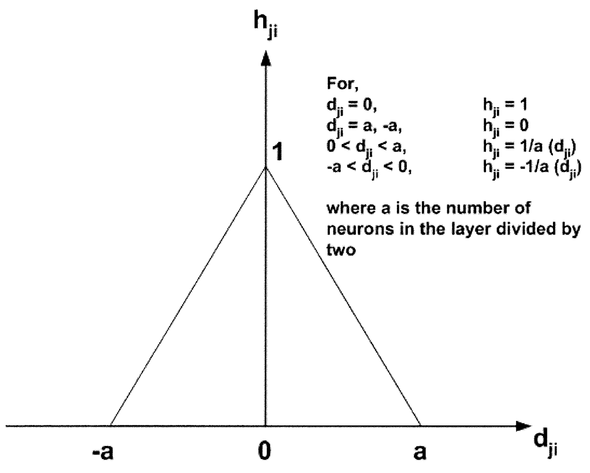


Fig. 6. Topological neighborhood h_{ji} as a function of d_{ji}.

the size of the topological neighborhood does not shrink with time.

Hence, using this concept, only weights belonging to neurons within the topological neighborhood of the winning/activated neuron are updated and the process of backpropagation can be accelerated. The winning neuron in the case of the FNDM module is decided by the center of area defuzzification process (fourth layer), S-norm fuzzy operator (third layer) and T-norm fuzzy operator (second layer).

Forgetting Mechanism: Finally, weight adjustment can also be accomplished by using the forgetting mechanism. A forgetting mechanism is implemented in the FNDM of all agents and ORL module to affect the weight adjustment process. The principle behind the forgetting mechanism is to enable the decision module to search through the solution space in an explorative manner rather than a purely exploitative manner [21] in order to reduce the number of instances whereby the search is trapped in a local minima. This is similar to the concept of simulated annealing whose objective is to find the global minimum of a cost function that characterizes large and complex systems. In doing so, simulated annealing proposes that instead of going downhill all the while to favor low energy ordered states, it is good to go downhill most of the time. In other words, an uphill search is needed at certain times. Results have shown in [21] that this new approach provides a robust framework for reinforcement learning in a changing problem domain where the improvised algorithm with the forgetting mechanism outperformed conventional the Q-learning approach. For this research, a variation of the forgetting mechanism is used.

The following equation shows the additional weight adjustment (besides the one using backpropagation) that is implemented in the fuzzy-neural networks:

$$W_{ij}^{(k+1)} = \mu W_{ij}^{(k)} + K \tag{8}$$

where, W_{ij} is the synaptic weight between node i and node j, μ is the forgetting term and its value is $0 < \mu < 1$, and K is a positive constant to be determined empirically. Using this approach, the search for the optimal solution does not get stuck in a local minima since the transition out of it is always possible.

Following the learning rate and weight adjustment process, the last stage of the multistage online learning process involves using evolutionary algorithm for adjustment of fuzzy relation according to the fitness values of individual neurons in the FNDM module. The following section will describe this stage in details.

C. Evolutionary Algorithm Fuzzy Relations Generator

Fuzzy rules lack precision and their membership functions need to be updated regularly in order for the rules to be valid in a dynamically changing problem domain. Invalid rules may even need to be discarded and new rules generated to replace them. Getting training data for rules optimization problems can be time consuming and it may not even be feasible to do so in certain problem domains due to issue of validity and accuracy. In this paper, the fuzzy rules adjustment process using evolutionary algorithm is 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 network evolves. The chromosome that is used by the EAFRG determines the way nodes in layer 2 of the FNDM module (antecedents) are linked to the implication nodes in layer 3.

Obtaining a suitable fitness function to evaluate the generated fuzzy relations is not an easy task since an ideal fitness function should be able to produce a fuzzy relation that results in a generally satisfying (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 (i.e., valid according to the current context of the problem state) 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 defined as follows:

$$F = \frac{E}{|diff(cur\_chromo, gen\_chromo)| + 1} + \gamma G \tag{9}$$

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 can be seen, E and G have counter balancing influence on each other. A relation may be generally satisfying, having a high G value, but due to the changing system dynamics, it may not be eligible.

Hence, a low E value will result. Adding them up will produce the overall fitness F of the relation. E is further defined as follows:

$$T(t) = DT(t) + (1 - D)R(t-1)A(t-1) \tag{10}$$

where, μ is the eligibility sensitivity factor (determined empirically) and T denotes the eligibility trace computed as follows:

$$E = \mu T \tag{11}$$

where t denotes time, D is the decay constant (determined empirically), R is the reinforcement, A is the activation value which is zero (0) if the rule is not activated and one (1) if activated. The function $\mathrm{diff}()$ denotes taking the difference between two chromosome vectors. In this case, the first chromosome vector is cur_chromo, the current chromosome used by the FNDM module and the second chromosome vector is gen_chromo, the chromosome generated by EAFRG. For this research, G is defined as follows:

$$G = \sum_i \sum_j \mathrm{Corr}(i,j) \tag{12}$$

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(0) if node i and j are not linked, and $\mathrm{Corr(i,j)}$ denotes the correlation between i and j.

For each update to obtain the best chromosome (representing a set of fuzzy relations between layer 2 and layer 3), the configuration used is as follows:

population size = 100;
number of epochs = 50;
probability of mutation = 0.07;
probability of crossover = 0.4.;
number of members in the pool of elites = 80.

As can be seen, the best 80 members of a previous population are carried forward in the new population before the processes of mutation and crossover repeat. Tests have been carried out to optimize these parameters according to the requirements of the system. Overall, the computational speed of the EAFRG does not hinder the real-time performance of the controller agent since the chromosome is a binary vector and the population size is relatively small.

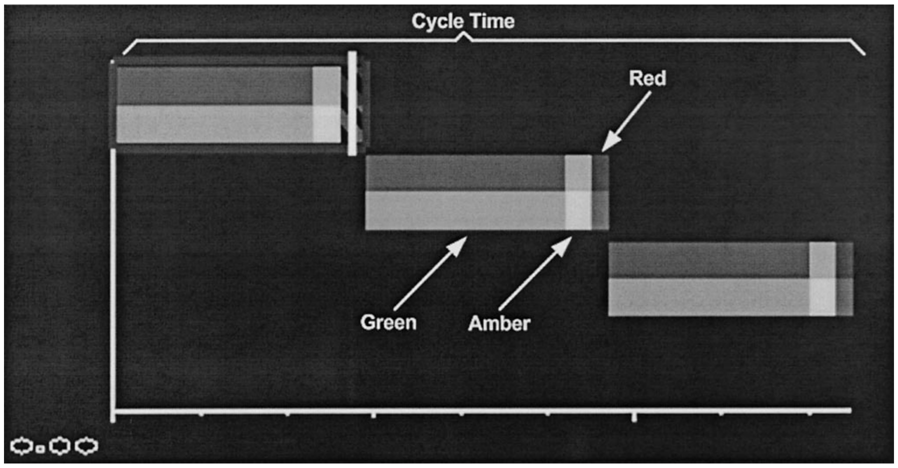


Fig. 7. Breakdown of a three-phased cycle.

V. Simulations and Results

Sections V-A–C describe the application domain of the multiagent architecture in details. In particular, the problem of controlling traffic signals within a complex traffic network with multiple intersections will be discussed. This is followed by a detailed description of the simulation platform and an analysis of the simulation results.

A. Traffic Signal Control Problem

Traffic signal operations at the intersections of an arterial network are one of the ways in which traffic conditions within the network can be influenced and controlled. For this research, the signal control policies formulated by the agents involved implementing the three types of control actions. They are namely cycle time adjustment (see Fig. 7), split adjustment (where split is the fraction of the cycle time that is given as the green phase for a set of traffic movements), 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). For a large, complex traffic network with multiple intersections, setting the values of these traffic signal parameters for each intersection in the network in a traffic-responsive manner is an extremely difficult task especially with the interdependency of each intersection and its neighbors. Hence, due to the complicated nature of the traffic signal control problem, the multiagent architecture is applied with the objective of achieving coordinated traffic signal control for a complex traffic network so as to reduce the likelihood of traffic congestion.

B. Simulation Platform

The multiagent architecture is implemented as a multithreading Java application. To ensure that agents satisfy the real-time considerations and deadlocks do not exist during the simulation, features for synchronizing access to critical sections such as monitors are implemented in the multiagent architecture. In addition, the priorities of the multiple threads are systematically assigned.

The traffic network used to evaluate the performance of the proposed multiagent architecture is based on a section of the central business district (CBD) area of Singapore. The real-world network is modeled in PARAMICS modeler using a total of 330 links and 130 nodes. This section represents one of the busiest regions of the road network where frequent traffic congestion is common during peak hours. The network's traffic operations were simulated using Version 4.0 of PARAMICS [22]. The necessary data for simulation was obtained from the Land Transport Authority (LTA) of Singapore. Inductive loop detectors were coded in the simulated network at stop lines of the intersection approaches, similar to the real-world installations.

Using the PARAMICS application programming interface (API), information such as lane occupancy, flow and rate of change of flow is extracted in real-time from the loop detectors. The flow, lane occupancy and rate of change of flow were measured at the green phase only (including the amber time). This information would be fed into the respective ICAs at the end of each signal phase. The agents' outputs in the form of traffic signal control policies (as mentioned in the previous section) are implemented in the simulation via the API to effect the latest signal adjustments. The sampling rate for the agents implemented in Java and the PARAMICS API has to be coordinated in order to make sure that the agents make timely responses to the dynamically changing traffic network.

The actual intersections modeled in the simulated network are shown in Fig. 8 including the 25 intersections to be controlled by the multiagent approach (each circle in Fig. 8 denotes an intersection that is controlled by an ICA) presented in this paper. For this research, each ZCA controls five prespecified ICAs. The jurisdiction of each ZCA is fixed throughout the simulation. Coordinated offset adjustment is implemented together with the changes in the phase length (and hence cycle length) of the traffic signal at the end of each evaluation period, if necessary.

Tests have been conducted to evaluate the performance of the traffic network with and without the multiagent architecture. The signal settings used for benchmarking were the actual signal plans implemented by LTA's Green Link Determining (GLIDE) signal control system. GLIDE is the local name of Sydney Coordinated Adaptive Traffic System (SCATS) and it is the state-of-the-art adaptive traffic signal control system [23] which is currently used in over 70 urban traffic centers in

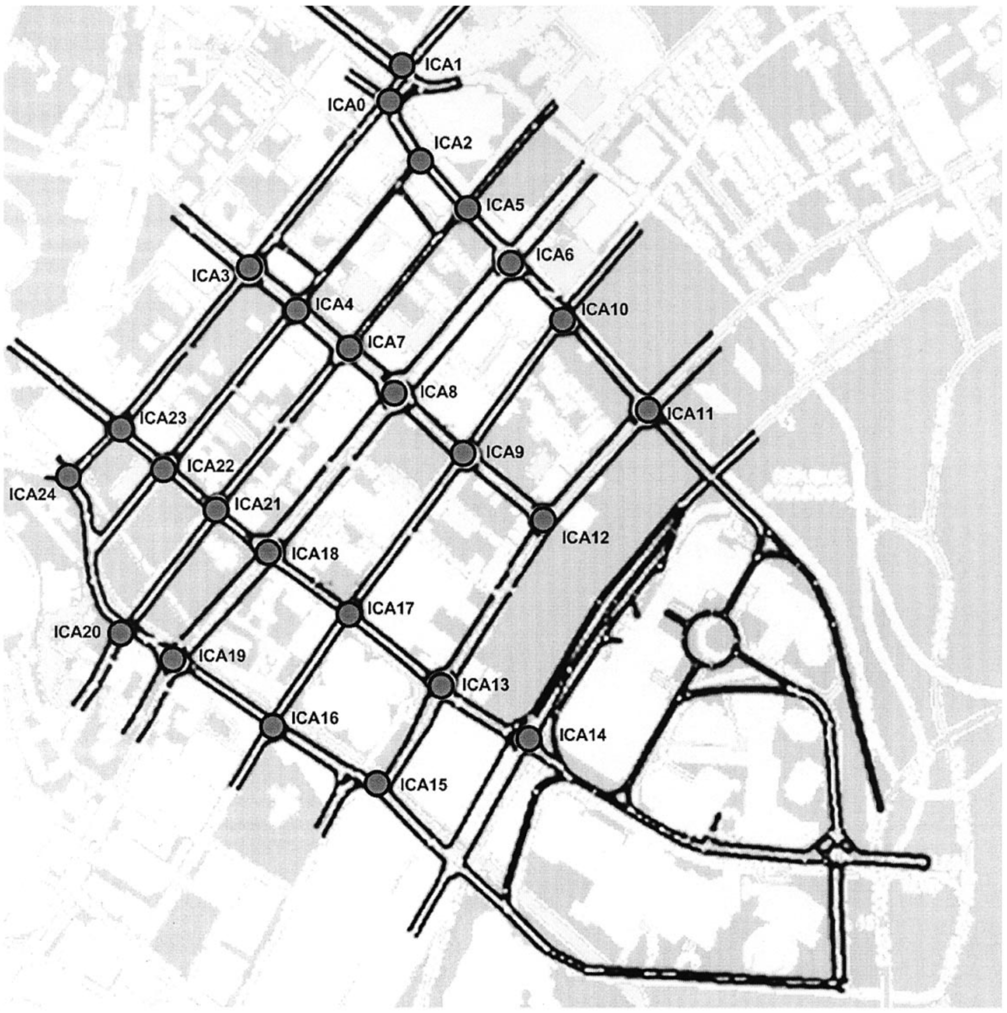


Fig. 8. ICAs in the traffic network.

15 countries world-wide). As such, for simulation scenarios without the multiagent architecture, the signal plans selected and executed by GLIDE were implemented in the coded network at the respective intersections as the traffic loading at each intersection changes with time. The traffic loading was derived from GLIDE's traffic count from the loop detectors.

C. Simulation Scenarios and Results

The following two sets of simulation scenarios are investigated in this paper.

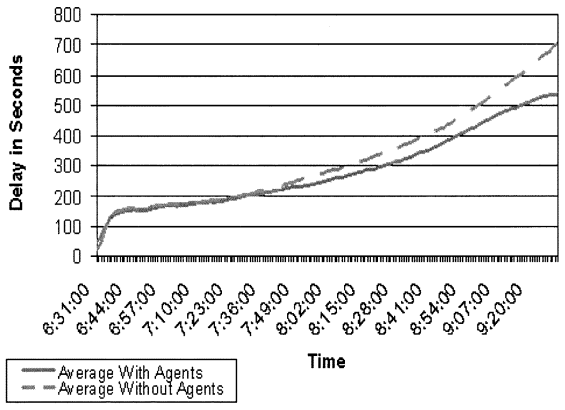
Scenario 1: It involves a total simulation time of 3 h with a single peak-within-peak period between 7:30 a.m. and 8:30 a.m.

Scenario 2: It involves a total simulation time of 6 h with two peak-within-peak periods introduced between 7:30 a.m. and 8:30 a.m., and between 10:30 a.m. and 11:30 a.m. to evaluate the learning process of the agents.

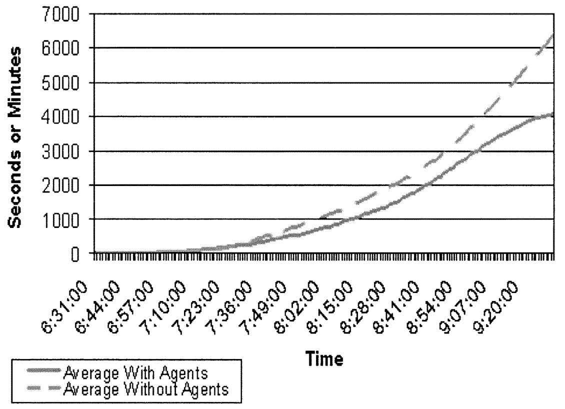
Six repeated simulation runs with different random number seeds were conducted for each scenario. The average delay per vehicle, and total stoppage time computed from all the vehicles were used as the performance criteria.

The plots given in Fig. 9 show the overall network performance measured in terms of the performance criteria mentioned earlier for the first scenario involving a single peak-within-peak period. It can be seen that the overall traffic network performance is better when the multiagent architecture is implemented with the network. Through using the agents, the total vehicle mean delay has decreased by approximately 15% [Fig. 9(a)] and the total all stoppage time for vehicles has reduced by approximately 30% [Fig. 9(b)] during the peak hour at 0830 hrs.

From the plots given in Fig. 10, it can be seen that the overall network performance is better when the multiagent architecture is implemented for scenario 2. Through using the agents, the average delay per vehicle is reduced by approximately 40% and the total stoppage time for vehicles is reduced by approximately 50% at the end of the simulation run. It can also be seen from Fig. 10 that the traffic network without the agents begins to degenerate into a pathological state after the second peak period, as suggested by the positive gradients and the first order derivatives of the curves for average delay and total stoppage time. Compared to the performance of the traffic network with agents during the first three hours of simulation (scenario 1), the performance of the traffic network has gained significant improvement for the next three hours with respect to the fixed time signal plans as shown in the plots for scenario 2. This indicates that to a certain extent,



(a)



(b)

Fig. 9. Results of simulation runs with and without agents for scenario 1.

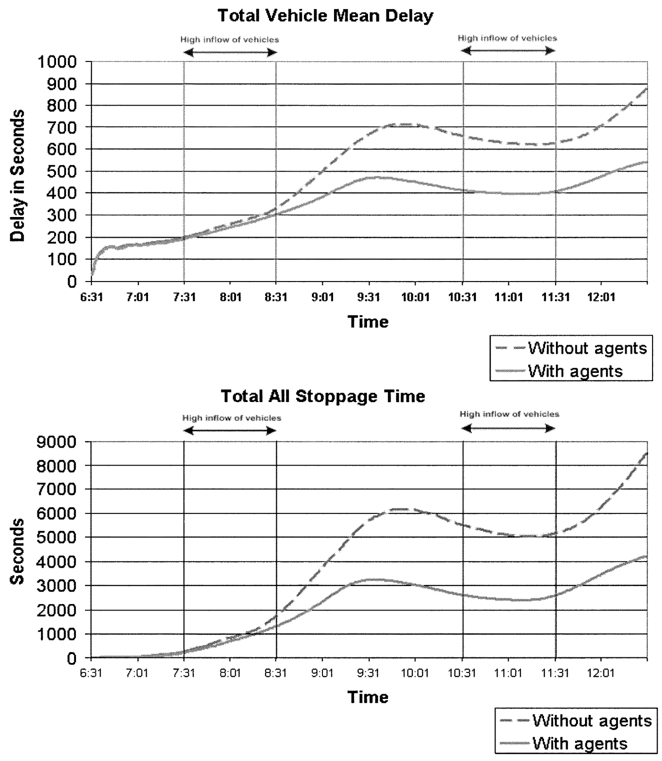


Fig. 10. Results of simulation runs with and without agents for scenario 2.

the multiagent architecture has adapted itself according to the changing dynamics of the traffic network.

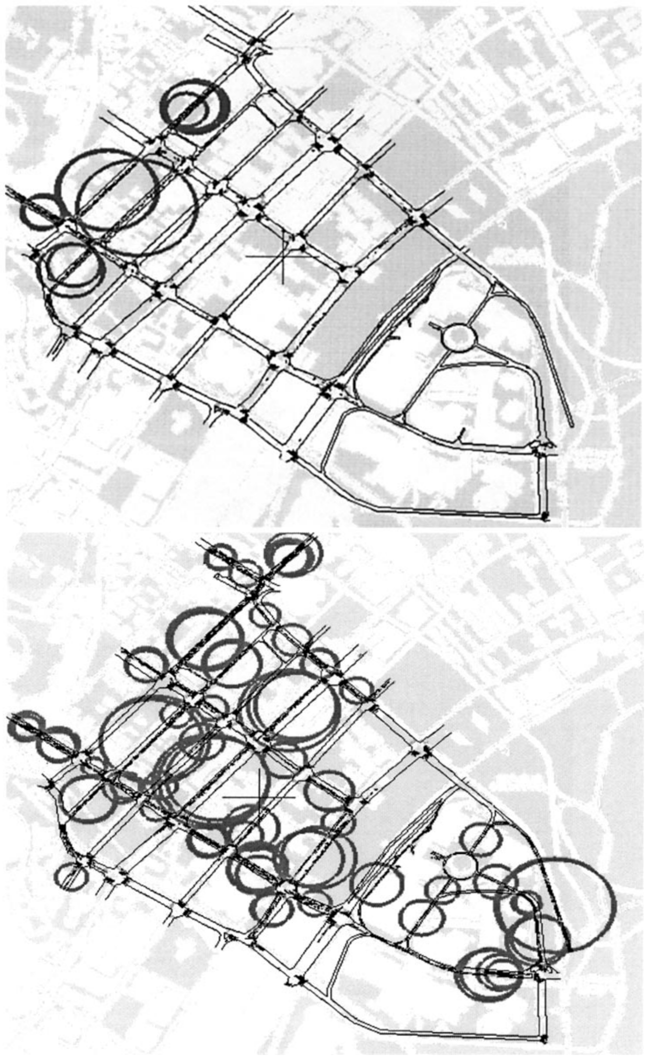


Fig. 11. Hotspots at 0930 h.

In order to better illustrate the improvements of the traffic network with the implementation of the multiagent architecture, screenshots of the network hotspots at 9:30 a.m. are taken and presented in Fig. 11. The PARAMICS modeling environment is preset to denote 13 stopped or queued vehicles with a hotspot or red circle. As can be seen from the Fig. 11, the traffic network evolves into a pathological state with over saturation at 0930 h without the multiagent architecture. The numbers of congested links are well over forties in number. Using agents, congestions have been confined to the upper section of the traffic network and the number of congested links is reduced to less than ten. Finally, it has to be noted that the performance of the multiagent architecture is limited by the overall capacity of the simulated traffic network.

VI. Conclusion

In this paper, a novel distributed, unsupervised coordinated traffic responsive strategy is implemented using a hybrid, cooperative multiagent architecture. In order to achieve effective unsupervised control in a dynamically changing environment, the multiagent architecture is designed using a variety of innovative applications involving concepts and theories from neural network, evolutionary algorithm, fuzzy logic, and reinforcement learning. The uniqueness of this multiagent architecture lies in the synergistic integration of these computational intelligence techniques, the multistage online learning process, as well as the well-defined cooperative mechanisms in the architecture that enable agents to adapt and make effective control policies in the dynamic traffic network. The performance of the multiagent architecture has been evaluated using a simulated model of a real world traffic network. Repeated rounds of simulation produce results which show that the presence of the multiagent architecture has generated significant improvement in the conditions of the traffic network when compared with signal plans implemented by the current real-time, responsive and adaptive signal control system. The tests have also shown that the multiagent architecture is capable of making timely decisions in real-time even with the employment of various artificial intelligence techniques, which are more computationally intensive than other approaches.

Acknowledgment

The authors would like to thank the Land Transportation Authority of Singapore (LTA) for providing data necessary for the simulation modeling and F. Logi for his advice.

References

[1] M. Tsavachidis, M. Hoops, and H. Keller, “Coordinated traffic management in the greater area of munich,” in *Proc. Int. Conf. Applications Advanced Technologies Transportation Engineering*, 1998, pp. 25–32.
[2] N. J. Garber and L. A. Hoel, *Traffic and Highway Engineering*, 2nd ed. Boston, MA: PWS-Kent, 1997, pp. 281–329.
[3] D. A. Haver and D. J. Tarnoff, “Future directions for traffic management systems,” *IEEE Trans. Veh. Technol.*, vol. 40, pp. 4–10, Feb. 1991.
[4] S. Chiu and S. Chand, “Self-organizing traffic control via fuzzy logic,” in *Proc. IEEE 32nd Conf. Decision Control*, 1993, pp. 1987–1902.
[5] G. Nakamiti and F. Gomide, “Fuzzy sets in distributed traffic control,” in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, 1996, pp. 1617–1623.
[6] S. Mikami and Y. Kakazu, “Genetic reinforcement learning for cooperative traffic signal control,” in *Proc. IEEE 1st Conf. Evolutionary Computation*, vol. 1, 1994, pp. 223–228.
[7] V. Manikonda, R. Levy, S. Satapathy, J. D. Lovell, C. P. Chang, and A. Teittinen, “Autonomous agents for traffic simulation and control,” in *National Research Council Transportation Research Board. Meeting 80th*, Washington, D.C., 2001.
[8] S. Takahashi, S. Nakamura, H. Kazama, and H. Fujikura, “Adaptive traffic signal control for the fluctuations of the flow using a genetic algorithm,” in *Proc. 8th Int. Conf. Urban Transport Environment 21st Century*, 2002, pp. 239–247.
[9] J. H. Lee and H. Lee-Kwang, “Distributed and cooperative fuzzy controllers for traffic intersections group,” *IEEE Trans. Syst., Man, Cybern. C*, vol. 29, pp. 263–271, May 1999.
[10] R. A. Francelin and F. A. C. Gomide, “A neural network for fuzzy decision making problems,” in *Proc. IEEE 2nd Int. Conf. Fuzzy Systems*, vol. 1, San Francisco, CA, 1993, pp. 655–660.
[11] K. Bogenberger, H. Keller, and S. Vukanovic, “A neuro-fuzzy algorithm for coordinated traffic responsive ramp metering,” in *Proc. IEEE Intelligent Transportation Systems*, 1994, pp. 94–99.
[12] R. Jaksa, P. Majernik, and P. Sincak, *Reinforcement Learning Based on Back Propagation for Mobile Robot Navigation*. Kosice: Computational Intelligence Group Dept. Cybern. Artif. Intell., Technical University.
[13] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*: MIT Press, 1998.
[14] D. A. Linkens and H. O. Nyongesa, “Genetics algorithms for fuzzy control,” in *Proc. Institute Elect. Engg. Control Theory Application*, 1995, pp. 161–185.
[15] M. Mohammadian and R. J. Stonier, “Generating fuzzy rules by genetic algorithms,” in *Proc. 3rd IEEE Int. Workshop Robot Human Communication*, 1994, pp. 362–367.
[16] S. Horikawa, T. Furahashi, and Y. Uchikawa, “On fuzzy modeling using fuzzy neural networks with back propagation algorithm,” *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, Sept. 1992.
[17] Y. H. Lin and G. A. Cunningham, “A new approach to fuzzy-neural system modeling,” *IEEE Trans. Fuzzy Syst.*, pp. 190–198, 1995.
[18] R. A. Jacobs, “Increased rates of convergence through learning rate adaptation,” *Neural Networks*, vol. 1, 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, pp. 226–245, 1991.
[20] T. Kohonen, *Self-Organizing Maps*, 2 ed. Berlin, Germany: Spring-Verlag, 1997b.
[21] G. Yan, F. Yang, T. Hickey, and M. Goldstein, “Coordination of exploration and exploitation in a dynamic environment,” in *Proc. International Joint Conf. Neural Networks*, 2001, pp. 1014–1018.
[22] *Paramics Modeller v4.0 User Guide and Reference Manual*, Quadstone Ltd, Edinburgh, U.K, 2002.
[23] P. B. Wolshon and W. C. Taylor, “Analysis of intersection delay under real-time adaptive signal control,” *Trans. Res. C*, vol. 7C, no. 1, pp. 53–72, Feb. 1999.

Min Chee Choy (S’03) was born in 1976. He received the B.Eng. degree in electrical and computer engineering from the National University of Singapore, Singapore in 2001. He is currently pursueing the Ph.D. degree in application of computational intelligence techniques for real-time control of traffic signals.

His main research interests are applications of distributed computational techniques and online learning.

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

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

Ruey Long Cheu (M’01) received the B.Eng. and M.Eng. degrees from the National University of Singapore, Sinapore and the Ph.D. degree from University of California, Irvine.

He is currently an Associate Professor in the Department of Civil Engineering and head of the Intelligent Transportation and Vehicle Systems Laboratory, the National University of Singapore, Singapore. His research interests are in intelligent transportation systems with emphasis on the applications of artificial intelligence and emerging computing techniques in transportation.

Dr. Cheu is a member of the U.S. Transportation Research Board Committee on Artificial Intelligence.