# A intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems

Kit Yan Chan, *member IEEE*, Tharam S. Dillon, *Life Fellow IEEE,* Elizabeth Chang, *Senior member IEEE*

***Abstract* — On-road sensor systems installed on freeways are used to capture traffic flow data for short-term traffic flow predictors for traffic management, in order to reduce traffic congestion and improve vehicular mobility. This paper intends to tackle the impractical time-invariant assumptions which underlie the methods currently used to develop short-term traffic flow predictors: i) the characteristics of current data captured by on-road sensors are assumed to be time-invariant with respect to those of the historical data, which is used to developed short-term traffic flow predictors; and ii) the configuration of the on-road sensor systems is assumed to be time-invariant. In fact, both assumptions are impractical in the real world, as the current traffic flow characteristics can be very different from the historical ones, and also the on-road sensor systems are time-varying in nature due to damaged sensors or component wear. Therefore, misleading forecasting results are likely to be produced when short-term traffic flow predictors are designed using these two time-invariant assumptions. To tackle these time-invariant assumptions, an intelligent particle swarm optimization algorithm, namely IPSO, is proposed to develop short-term traffic flow predictors by integrating the mechanisms of particle swarm optimization, neural network and fuzzy inference system, in order to adapt to the time-varying traffic flow characteristics and the time-varying configurations of the on-road sensor systems. The proposed IPSO was applied to forecast traffic flow conditions on a section of freeway in Western Australia, whose traffic flow information can be captured on-line by the on-road sensor system. These results clearly demonstrate the effectiveness of using the proposed IPSO for real-time traffic flow forecasting based on traffic flow data captured by on-road sensor systems.**

***Index Terms* —** Neural networks, particle swarm optimization, fuzzy inference system, traffic flow forecasting, sensor systems, sensor data, time-varying systems, traffic contingency

## I. Introduction

**On-road** sensor systems installed on freeways provide traffic flow data for the development and implementation of short-term traffic flow predictors, which aim to forecast likely traffic flow conditions in the short-term, typically within ten minutes ahead [42]. This short-term forecasting information can be used to assist proactive traffic control centers to anticipate traffic congestion and improve the mobility of transportation [48]. To develop short-term traffic flow predictors, historical traffic flow data is first collected using the on-road sensor systems installed on a section of the freeway under investigation. Then, the short-term traffic flow predictors can be developed based on conventional statistical methods such as filtering techniques [37, 41], autoregressive integrated moving average (ARIMA) methods [50], statistical regression [43] and k-nearest-neighbor approaches [10], as found in the past literature. With the developed short-term traffic flow predictors, future traffic flow conditions can be forecast based on the current traffic flow conditions, which are captured on-line using the on-road sensor systems.

Even if traffic flow predictors developed by such statistical methods can obtain reasonable prediction accuracy for future traffic flow conditions, the predictors developed by these methods may not be able to address the strongly non-linear characteristics of short-term traffic flow. In order to address this issue, another recently used modeling or prediction approach, namely support vector regression [17-20], have been applied to develop short-term traffic flow predictors, which can obtain forecasting results with better accuracies than ARIMA. Also, a lot of research has been conducted by applying the artificial intelligent approach, namely neural networks (NNs) [9, 32-34] to develop short-term traffic flow predictors [11, 12, 25], which can obtain forecasting results with better accuracies than can statistical methods. Recent research involving NNs has been focusing mainly on enhancing the generalization capability of NNs by developing hybrid NNs, which incorporate other artificial intelligence techniques or statistical forecasting methods, such as fuzzy systems [15, 40, 44, 51], Kalman filter [37], fuzzy clustering method [45], and the autoregression moving average method [46] etc. These hybrid NNs can achieve more accurate predictions than those achieved by using NNs.

However, the development of all these approaches is subject to two common but impractical assumptions: i) that the characteristics of collected historical traffic flow data are time-invariant with respect to the current traffic flow characteristics. In fact, the time-varying aspects of traffic flow are unavoidable. Therefore, they are likely to produce misleading traffic flow forecasting on current road conditions, if the characteristics of the historical traffic flow data are very different from those of the newly acquired traffic flow data captured on-line by the on-road sensor systems; and ii) that the configuration of the on-road sensor system is time-invariant; the configurations of the traffic flow predictors are pre-defined based on this assumption. In fact, this is not always true and it is unwise to assume that all on-road sensors can work properly all of the time, as the performance of the on-road sensors may deteriorate over time, due to component wear. Also, some on-road sensors are likely to be damaged in very poor weather conditions such as rainstorm or flood. Therefore, misleading forecasting results are likely to be produced by the traffic flow predictors, which are designed under these two time-invariant assumptions.

To tackle these two time-invariant assumptions, a hybrid NN approach integrated with the particle swarm optimization (PSO) approach, namely intelligent PSO (IPSO), is proposed. The IPSO uses the particles in the swarm to represent the neural networks which are used to forecast the short-term traffic flow, without

Manuscript received December 12, 2011; revised April 16, 2012; revised July 4, 2012; accepted August 2, 2012. Date of publication xxxx; date of current version xxx.



K.Y. Chan is the corresponding author of the paper. He is with the Department of Electrical and Computer Engineering, Curtin University, Perth, Australia (Phone: 61-8- 9266 2948; fax: 61-8- 9266 2819; e-mail: kit.chan@curtin.edu.au). T.S. Dillon is with the Department of Computer Science and Computer Engineering, La Trobe University, Australia. E. Chang is with the School of Information Systems, Curtin University, Perth, Australia.

making these two time-invariant assumptions. The IPSO has the following three main features:

**1) Flexible neural network structure:** In the IPSO, each particle is represented by a three-layer neural network, where switches are configured between links of neural nodes, in order to determine both the optimal NN structures and the parameters which vary with respect to time [26]. Each particle consists of two parts: the integer string and the hierarchical string [47]. The integer string is used to represent the NN parameters. The hierarchical string is used to represent the NN structure. It is represented by the open/close actions of a number of switches which link the neural nodes. When the switch is opened, the link between the corresponding neural nodes exists. However, the link does not exist when the switch is closed. Based on this particle representation, both optimal NN structures and parameters can be adapted to newly-captured traffic flow data or time-varying configurations of on-road sensor systems. Also, the IPSO can automatically determine the optimal structures of NNs without involving trial and error methods. This is intended to overcome the limitations of the existing NN approaches [1, 11, 12, 25, 27] for traffic flow forecasting in which the NN structure has to be fixed and cannot be adapted with time.

**2) Active particle movement:** In IPSO, the particle movement of classical particle swarm optimization [23], is used, inspired by the social behaviours of animals. The particle movement is used to adapt to the optimal NN parameters and structures for short-term traffic flow forecasting, which is time-varying, since the particle movement can effectively tune the real-time adaptive controllers for many time-varying systems, including the Maglev transportation system [53] and generator system for power applications [28]. This mechanism can also be applied for neural network design effectively [29-31]. Also, Chan et al. [8] demonstrated that the particle movement can effectively adapt optimal structures and parameters of time-varying systems, where the parameters and the structures of the systems vary with time. Based on the particle movement, the IPSO intends to automatically and effectively tune both the parameters and the structures of the NNs, in order to obtain an optimal short-term traffic flow forecasting, which is time-varying.

3) **Further enhancement of particle movement:** In the classical PSO [23], the diversity of the solutions is likely to be lost when a solution with certain good quality is obtained. Hence, the classical PSO is likely to be trapped into this solution and no further progress in terms of better solutions can be made. To further assist the proposed IPSO to search for better solutions, activating components can be injected into the particles in order to increase the diversity of the particles [52]. In the IPSO, a mechanism based on a fuzzy intelligence system is designed, in order to maintain the diversity of particles by artificially injecting them with activating components. It monitors the traffic flow accuracies obtained by the IPSO and the changing rates of the traffic flow accuracies. When the traffic flow accuracy is low or the traffic flow accuracy decreases sharply, more activating components are injected into the particles. This is intended to prevent particles pre-maturely converging to solutions with poor traffic flow accuracies, and helps the IPSO to move the poor particles from a region with poor traffic flow accuracies to a better region.

Comparisons were conducted based on the IPSO and the other existing methods to develop NNs for short-term traffic flow forecasting on the Mitchell freeway in Western Australia, where an on-road sensor system with many on-road sensors has been installed to capture traffic flow data at different locations on the freeway. The results show that better accuracies in short-term traffic flow forecasting can be obtained by using the IPSO compared with those obtained by other tested methods. The rest of the paper is organized as follows. Section II shows the configuration of the NN for short-term traffic flow forecasting. Section III discusses the mechanisms of the IPSO. Section IV presents and discusses the results obtained by IPSO and the other tested algorithms for forecasting short-term traffic flow conditions on the Mitchell freeway in Western Australia. Section V concludes the paper.

## II. NNs FOR SHORT-TERM TRAFFIC FLOW FORECASTING

To forecast future traffic flow conditions at location A illustrated in Figure 1, a short-term traffic flow predictor was developed based on traffic flow data collected by the $n$ on-road sensors ($D_1$, $D_2$, …and $D_n$). $D_i$ captures the traffic flow condition, $y_i(t)$, at time $t$ with a sampling time of $T_s$, where the traffic flow conditions can be reflected by average speed of vehicles. In general, the traffic flow on the freeway is smooth, if the average speed of the vehicles approximates the speed limit of the freeway and the density of vehicles is low.

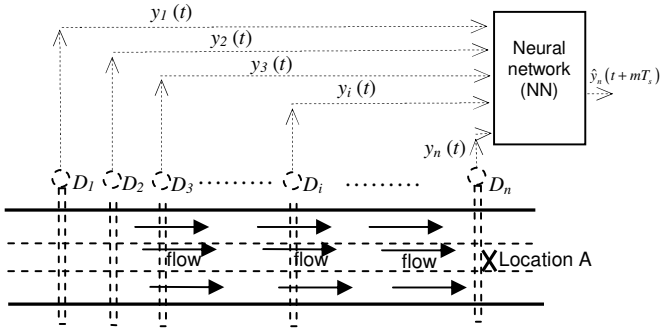



**Fig. 1** A NN for short-term traffic flow forecasting on the freeway

In this paper, a three-layer neural network is used to forecast future traffic flow conditions at location A, $\hat{y}_n(t+mT_s)$, with $m$ sampling time ahead, based on equation (1):

$$\hat{y}_n(t+mT_s) = \delta(s_0(t))\alpha_0(t) + \sum_{j=1}^{N_h}\left[\delta(s_j(t))\beta_j(t)\cdot \Psi\left(\delta(s_{j,0}(t))\gamma_{j,0}(t) + \sum_{k=1}^{n}\sum_{i=0}^{p}\delta(s_{j,i,k}(t))\gamma_{j,i,k}(t)\,y_k(t-iT_s)\right)\right]\Big|_{\bar{W}(t)} \quad (1)$$

where $\bar{W}(t)$ is the neural network parameters, denoted by equation (2) as follows:

$$\bar{W}(t) = \left[\alpha_0(t), \beta_j(t), \gamma_{j,0}(t), \gamma_{j,i,k}(t), s_0(t), s_j(t), s_{j,0}(t), s_{j,i,k}(t) \text{ with } i=0, 1, ..., p;\ j=1, 2, ..., N_h;\ k=1, 2, ..., n\right] \quad (2)$$

$y_k(t-i\cdot T_s)$ is the traffic flow condition captured by $D_k$ at time ($t$-$i$·$T_s$) with $i$=1, 2, …, $p$; $\alpha_0(t)$ denotes the weight of the bias of the output node at time $t$; $\beta_j(t)$ denotes the weight on the link from the $j$-th hidden node to the output node at time $t$; $\gamma_{j,0}(t)$ denotes the weight of the bias of the $j$-th hidden node at time $t$;

$\gamma_{j,i,k}(t)$ denotes the weight on the link from the input node for $y_k(t-i\cdot T_s)$ to the $j$-th hidden node; $\Psi(.)$ is the transfer function of the hidden node. Here, the sigmoid function is used, as it can achieve satisfactory results for traffic flow forecasting [1]; $s_0(t)$ denotes the parameter for the link switch from the bias node to the output node; $s_j(t)$ denotes the one from the $j$-th hidden node to the output node; $s_{j,0}(t)$ denotes the one from the bias node to the $j$-th hidden node; and $s_{j,i,k}(t)$ denotes the one from the input node, $y_k(t-iT_s)$, to the $j$-th hidden node; $\delta(\alpha)$ is a link switch between two nodes, which is defined by an unit step function as:

$$\delta(\alpha)=\begin{cases}0 \text{ if } \alpha<0\\ 1 \text{ if } \alpha\geq 0\end{cases}, \text{ where } \alpha\in \mathrm{R}. \quad (3)$$

The use of these link switches overcomes the limitations of the commonly used fixed-connected neural networks, where all nodes are restricted by a fixed configuration, and the structure of the neural network cannot be adapted with respect to newly-captured data [26].

As illustrated in Figure 2, the neural network parameters, $\bar{W}(t)$, are determined by two stages namely, off-line training and on-line adaption. Off-line training is defined for the time $t$ before $T_{off\text{-}line}$ (i.e. $t<T_{off\text{-}line}$) and on-line adaption is defined for the time $t$ after $T_{off\text{-}line}$ (i.e. $t>T_{off\text{-}line}$), where $T_{off\text{-}line}$ is the switching time between the off-line training and the on-line adaption. For off-line training, $\bar{W}(t)$ is pre-determined based on the past collected traffic flow data. For on-line adaption, $\bar{W}(t)$ is adapted with the traffic flow data which is captured on-line from the freeway. Detailed description for off-line training, on-line adaption and the error function for both states are given as following.

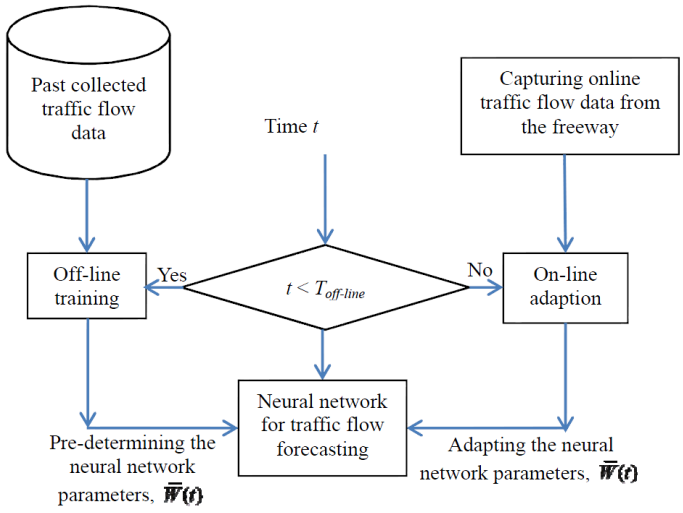



**Fig. 2** Illustration of pre-determining and adapting the neural network parameters

### A. *Off-line training*

For off-line training (i.e. $t<T_{off\text{-}line}$), $\bar{W}(t)$ is initialized by a set of collected $N_{hist}$ pieces of historical traffic flow data, $\Pi_{hist}(t)$, where $\Pi_{hist}(t)=\{\bar{Y}(t(i))$ with $i=1,2,..,N_{hist}$ and $t(i)<T_{off-line}\}$, and $\bar{Y}(t(i))$ is the $i$-th piece of historical traffic flow data collected at the time $t(i)$, which is given by:

$\bar{Y}(t(i))=[y_n(t(i)), y_1(t(i)-(j+m)\cdot T_s), y_2(t(i)-(j+m)\cdot T_s),$ ..., $y_{n-1}(t(i)-(j+m)\cdot T_s)$ with $j=0, 1, ..., p]$.

For $t<T_{off\text{-}line}$, the initial neural network parameters, $\bar{W}(t)$, are initialized by minimizing the following error function (4), which is an average of the absolute errors between real observations and estimates:

$$\min e_{hist}^{MAE}(\bar{W}(t),\Pi_{hist}(t))=\frac{1}{N_{hist}}\sum_{i=1}^{N_{hist}}\left|\frac{y_n(t(i))-\hat{y}_n(t(i))}{y_n(t(i))+\partial y_n}\right|_{\bar{W}(t),\Pi_{hist}(t)} \quad (4)$$

where $y_n(t(i))$ is the traffic flow condition collected by the on-road sensor $D_n$ at the past time $t(i)$; $\hat{y}_n(t(i))$ is estimated based on equation (1) with respect to $\bar{W}(t)$; and $\partial y_n$ is a very small value in order to avoid the denominator to be zero when $y_n(t(i))$ is equal to zero, but $\partial y_n$ is small enough that it does not affect the calculation for the mean absolute error.

### B. *On-line adaption*

For on-line adaption (i.e. $t>T_{off\text{-}line}$), $\bar{W}(t)$ varies by adapting $m$ pieces of new traffic flow data, $\Pi_{adapt}(t)$, which are newly captured, where $\Pi_{adapt}(t)=\{\bar{Y}(t-i\cdot T_s)$ with $i=0,1,...,m$ and $t>T_{off-line}\}$, and $\bar{Y}(t-i\cdot T_s)=[y_n(t-i\cdot T_s), y_1(t-(j+m+i)\cdot T_s),$ $y_2(t-(j+m+i)\cdot T_s)$, ..., $y_{n-1}(t-(j+m+i)\cdot T_s)$ with $j=0, 1, ..., p]$.

For $t>T_{off\text{-}line}$, $\bar{W}(t)$ is determined by minimizing the following error functions (5):

$$\min e_{adapt}^{MAE}(\bar{W}(t),\Pi_{adapt}(t))=\frac{1}{m}\sum_{i=1}^{m}\left|\frac{y_n(t-i\cdot T_s)-\hat{y}_n(t-i\cdot T_s)}{y_n(t-i\cdot T_s)+\partial y_n}\right|_{\bar{W}(t),\Pi_{adapt}(t)} \quad (5)$$

where $y_n(t-i\cdot T_s)$ is the traffic flow condition captured by the on-road sensor $D_n$ at time $(t-i\cdot T_s)$; $\hat{y}_n(t-i\cdot T_s)$ is estimated based on equation (1) with respect to $\bar{W}(t)$; and $\partial y_n$ is a very small value in order to avoid the denominator to be zero when $\hat{y}_n(t-i\cdot T_s)$ is equal to zero, but $\partial y_n$ is small enough that it does not affect the calculation for the mean absolute error.

### C. *Error function for both off-line training and on-line adaption*

For all $t$, a time-varying error function, $J(t)$, is formulated by combining the error functions (4) and (5). $J(t)$ is denoted by:

$$\min J(t)=\min\begin{cases}e_{hist}^{MAE}(\bar{W}(t),\Pi_{hist}(t)) \text{ for } t<T_{off-line}\\ e_{adapt}^{MAE}(\bar{W}(t),\Pi_{adapt}(t)) \text{ for } t>T_{off-line}\end{cases} \quad (6)$$

To solve the optimization problem (6), classical tools for nonlinear programming such as nonlinear branch-and-bound, sequential linearization, and Lagrangian relaxation methods can be used. However, their common shortcoming is that they cannot cope with significantly nonlinear functions, and time-varying

functions which the error function, $J(t)$, may involve. The nonlinear characteristics of traffic flow are caused by the drivers' behaviours or reaction times regarding current traffic flow [4]. For example, different drivers have different reaction times when having to apply their brakes to stop the vehicle when they encounter an obstacle in front. Also, they have different behaviours, when using their accelerators to control their car speeds, in order to match the current traffic flow conditions. Apart from this nonlinear characteristic, the time-varying characteristic also exists in traffic flow data. It is caused by uncontrollable sequences of events for drivers using the roads, and contingent incidents on the roads. Also, the configurations of the on-road sensor systems are time-varying, due to component wear and unexpected damages of on-road sensors.

Therefore, we propose to use the particle swarm optimization, to solve these optimization problems, as this method can obtain satisfactory solutions to such problems, which are nonlinear, and time-varying [38]. In the following section, an algorithm based on particle swarm optimization, namely intelligent particle swarm optimization IPSO, is developed, in order to determine the neural network parameters, $\bar{W}(t)$.

## III. INTELLIGENT PARTICLE SWARM OPTIMIZATION

In the IPSO, a random initial swarm consisting of $N_s$ particles is first created, where the position of the $l_1$-th particle at time *t*, $P_{l_1}(t)$, is used to represent the neural network parameters, $\bar{W}(t)$, formulated in equation (2). $P_{l_1}(t)$ is denoted as follows:

$$P_{l_1}(t) = \left(p_{l_1,1}(t), p_{l_1,2}(t), ..., p_{l_1,n_p}(t)\right) = \bar{W}(t) \quad (7)$$
$$= \left[\alpha_0(t), \beta_j(t), \gamma_{j,0}(t), \gamma_{j,i,k}(t), s_0(t), s_j(t), s_{j,0}(t), s_{j,i,k}(t)\right.$$
$$\left. \text{with } i=0, 1, ..., p;\ j=1, 2, ..., N_h;\ k=1, 2, ..., n\right]$$

in which $P_{l_1}(t)$ consists of $n_p$ elements, $p_{l_1,l_2}(t)$ with $l_2$=1,2,…, $n_p$; $n_p = (1+p)\cdot N_h \cdot n$; and $p_{l_1,l_2}(t)$ is randomly generated within the domain $p_{l_1,l_2}(t) \in \{p_{\min} ... p_{\max}\} = \{-1...1\}$. Then, each particle, $P_{l_1}(t)$, is evaluated based on the error function (6), and the error obtained by $P_{l_1}(t)$ is denoted by $J_{l_1}(t)$. $J^{\text{best}}(t)$ represents the smallest error obtained by the best particle in the swarm, which is is denoted by

$$J^{\text{best}}(t) = \min_{\forall l_3 \in [1,..,N_s]} J_{l_3}(t), \quad (8)$$

where $J^{\text{best}}(t) = J_{l_3}(t) < J_{l_4}(t)$ for $l_3 \neq l_4$. The best particle in swarm is denoted by $P_{\text{best}}(t)$, which can obtain the smallest error, $J^{\text{best}}(t)$, in the swarm.

The position of the particle, $p_{l_1,l_2}(t)$, is updated based on equation (9) at time *t*:

$$p_{l_1,l_2}(t) = p_{l_1,l_2}(t-T_s) + v_{l_1,l_2}(t) \quad (9)$$

where $p_{l_1,l_2}(t-T_s)$ is the position of the particle at time, $(t-T_s)$, and $v_{l_1,l_2}(t)$ is the velocity of the particle at time *t*. When the classical PSO [14] is used, the velocity, $v_{l_1,l_2}(t)$, of the $l_2$-th element on the $l_1$-th particle at time *t* is given by:

$$v_{l_1,l_2}(t) = \omega(t)\cdot v_{l_1,l_2}(t-T_s) + \phi_1 \cdot r_1 \left(pbest_{l_1,l_2} - p_{l_1,l_2}(t-T_s)\right) + \phi_2 \cdot r_2 \left(gbest_{l_2} - p_{l_1,l_2}(t-T_s)\right), \quad (10)$$

where $pbest_{l_1} = \left[pbest_{l_1,1}, pbest_{l_1,2}, ..., pbest_{l_1,n_p}\right]$ is the best position of the $l_1$-th particle moved so far, and $gbest = \left[gbest_1, gbest_2, ..., gbest_{n_p}\right]$ is the position of the best particle among all the particles; $r_1$ and $r_2$ return a uniform random number between the range of [0, 1]; $\phi_1$ and $\phi_2$ are acceleration constants; and $\omega(t)$ is the intelligent inertia weight, which is a constant between the range of [0.1, 1.1] for all *t* [13].

Initial studies of PSO show that the value of $\omega(t)$ is important to ensure convergent behavior [13]. When $\omega(t)$ >1, $v_{l_1,l_2}(t)$ increases with respect to time that cause divergent behavior. When $\omega(t)$ <0, particles decelerate until their velocities reach zero. It has been shown by [5] that a particular set for $\omega(t)$, $\phi_1$ and $\phi_2$, can ensure that PSO can be rapidly converged, if the following condition is satisfactory:

$$\omega(t) > 0.5\cdot(\phi_1 + \phi_2) - 1 \quad (11)$$

As the error function (6) has a time-varying characteristic, the landscape and the optima of the error function varies with respect to time. Therefore, it is not effective to use the classical PSO to solve the time-varying error function (6), as the swarm parameters such as inertia weight in the classical PSO is pre-defined based on the past experience. Even thought the pre-defined parameters work very well on forecasting future traffic flow conditions which have similar characteristic to past traffic flow conditions used for training, they may not work effectively on forecasting future traffic flow conditions which have different characteristic to those past conditions. The pre-defined parameters, which are time-invariant, may not be appropriate to optimize the time-varying error function (6), which varies with respect to time. To enhance the performance of the classical PSO, the IPSO is proposed by adapting the swarm parameters to the time-varying error function.

### A. Mechanisms of the IPSO

For IPSO, the velocity of the element at time *t* is given by:

$$v_{l_1,l_2}(t) = \omega(t)\cdot\left(\left(1-\beta(t)\right)\cdot v_{l_1,l_2}(t-T_s) + \beta(t)\cdot\tilde{v}_{l_1,l_2}(t-T_s)\right) + \phi_1 \cdot r_1 \left(pbest_{l_2} - p_{l_1,l_2}(t-T_s)\right) + \phi_2 \cdot r_2 \left(gbest_{l_2} - p_{l_1,l_2}(t-T_s)\right) \quad (12)$$

where the two components, $\omega(t)$ and random velocity component, $\tilde{v}_{l_1,l_2}(t-T_s)$, are used in order to obtain better solutions for the time-varying error function (6).

$\omega(t)$ is used to create a dynamical balance between the exploration and exploitation characteristics of the IPSO. $\omega(t)$ changes dynamically by monitoring the error obtained by the particles at time *t*, in order to dynamically adjust the search capability between the exploration and exploitation. A large $\omega(t)$ facilitates an exploration which induces the particles to leave their current regions and pushes the particles to search in the other regions. A small $\omega(t)$ facilitates exploitation, which refines the

best solution of the particles by exploiting a small vicinity around this best solution.

To solve the time-varying error function (6), a large $\omega(t)$ is required, when the error obtained by the IPSO is high, or the error obtained by the IPSO increases. The current traffic flow conditions are very different to those of the previous. Thus, a large $\omega(t)$ is necessary to push the particles to explore the searching regions which have not been explored, and are likely to produce better solutions with lower forecasting errors. The reason is that refining the particles in a searching area with large errors does not effectively obtain a significant improvement. However, when the error is small, or decreases, a small $\omega(t)$ is used. The current traffic flow conditions are not significantly different to those of the previous. Small $\omega(t)$ intends to let the particles exploit a small vicinity by refining the positions of the particles, as the solutions located by the particles are good enough to produce a better. It refines solution with a smaller forecasting error.

To further help the particles to search for good solutions with small errors, a random velocity component, $\tilde{v}_{l_1,l_2}(t-T_s)$, is injected into the $l_2$-th element on the $l_1$-th particle. $\tilde{v}_{l_1,l_2}(t-T_s)$ is determined based on equation (13),

$$\tilde{v}_{i,j}(t-T_s) = \lambda \cdot r_3 \cdot (p_{\max} - p_{\min}), \quad (13)$$

which is randomly generated and bound with $\lambda$ of the range of the particle element, and $r_3 \in [0 \quad 1]$ is an uniform random number. More random components are introduced into the particle element, when large $\lambda$ is used. It intends to adapt traffic conditions which have large time-varying dynamics. Small $\lambda$ intends to adapt those with small time-varying dynamics. $\lambda = 0.25$ is used in this research. Thus, 0.25 of the range of the particle element is used. The resulting velocity, $v_{l_1,l_2}(t)$, in equation (10) is determined based on the weighted sum of $v_{l_1,l_2}(t-T_s)$, and $\tilde{v}_{l_1,l_2}(t-T_s)$, where the intelligence weight factor, namely $\beta(t)$, is introduced. $\beta(t)$ controls the amount of random velocity component, which is injected into the regular velocity component, in order to help the particles to escape the poor solutions with large errors. A large $\beta(t)$ is required, when the forecasting error obtained by the IPSO is large, or the forecasting error obtained by the IPSO increases. Using large $\beta(t)$, the population diversity of the particles increases, and it allows the particles to have a greater chance of exploring searching areas which have better solutions. A small $\beta(t)$ is used, when the forecasting error obtained by the IPSO is small, or the forecasting error obtained by the IPSO decreases. It allows the particles to refine their positions, in order to obtain a better solution by exploring search areas which are good enough to perform fine-tuning.

However, determining the appropriate $\omega(t)$ and $\beta(t)$ for IPSO is not an easy task, as the landscape of the error function (6) is related to the traffic flow characteristics which are highly nonlinear, complicated and time-varying. It is impractical and almost impossible to mathematically model the search process of the IPSO, in order to determine the appropriate $\omega(t)$ and $\beta(t)$, which vary with respect to the time-varying traffic flow characteristics. As both $\omega(t)$ and $\beta(t)$ can be adjusted based on the linguistic understanding for minimizing the errors of traffic flow forecasting, the fuzzy inference system is used. It has rich literature in adjusting process parameters for performing process optimization. This linguistic understanding helps in the design of a fuzzy inference system, in order to dynamically vary both $\omega(t)$ and $\beta(t)$. Determining both $\omega(t)$ and $\beta(t)$ based on the fuzzy inference system is discussed in the following sub-section III.B, and the operations of the IPSO are summarized as follows:

Step 1: $t \leftarrow 0$.

Step 2: Initialize $N_S$ particles $P_1(t)$, $P_2(t)$, ... , $P_{N_s}(t)$ based on (7).

Step 3: Evaluate the error $J_{l_1}(t)$ of each particle, $P_{l_1}(t)$, with $l_1 = 1$, 2,, …, $N_s$ based on the time-varying error function (6).

Step 4: Return the best particle, $P_{\text{best}}(t)$, with the smallest error, $J^{\text{best}}(t)$, formulated by equation (8) as the outcome of the IPSO at time $t$.

Step 5: Increment $t$ by a sampling time, $T_s$, i.e. $t \leftarrow t+T_s$.

Step 6: Determine intelligence inertia weight, $\omega(t)$, and intelligence weight factor, $\beta(t)$, based on the fuzzy inference system discussed in Section III.B.

Step 7: Generate the random velocity component, $\tilde{v}_{l_1,l_2}(t-T_s)$, based on equation (13).

Step 8: Update the velocity, $v_{l_1,l_2}(t)$, of the $l_1$-th element of the $l_2$-th particle based on (12).

Step 9: Update each particle element $p_{l_1,l_2}(t)$ based on (9).

Step 10: Go to Step 3 if termination condition is not reached. Otherwise terminate.

### *B. Fuzzy inference system*

The fuzzy inference system consists of two inputs and two outputs. The two outputs are the intelligence inertia weight, $\omega(t)$, and the intelligence component factor, $\beta(t)$, which are used to governed the velocity of the particle element formulated in equation (12). These two outputs are adapted with respect to the two inputs, i) the smallest error, $J^{\text{best}}(t)$, and ii) the change of the smallest errors, $\partial J^{\text{best}}(t)$, which represents the mean difference between the smallest errors obtained by the best particles at time $t$ and time $(t-n_w \cdot T_s)$. $\partial J^{\text{best}}(t)$ is denoted by:

$$\partial J^{\text{best}}(t) = \left(J^{\text{best}}(t) - J^{\text{best}}(t-T_s)\right) \quad (14)$$

When the error obtained by the IPSO is high, or the error obtained by the IPSO increases, a larger $\omega(t)$ is required, in order to give the particles more energy to escape poor solutions with greater errors. Therefore, $\omega(t)$ is set to large, if $J^{\text{best}}(t)$ is large and $\partial J^{\text{best}}(t)$ is large. However, when the error is small, or the error decreases, a large $\omega(t)$ is not needed. Smaller $\omega(t)$ is required, in order to fine tune the particles to better solutions with smaller errors. Therefore, $\omega(t)$ is set to small, if $J^{\text{best}}(t)$ is small and $\partial J^{\text{best}}(t)$ is small. Based on this linguistic understanding

regarding $\omega(t)$, the following two basic principles, **P1** and **P2**, for determining $\omega(t)$ are used:

**P1**: If $J^{\text{best}}(t)$ is large or $\partial J^{\text{best}}(t)$ is large, $\omega(t)$ is large.

**P2**: If $J^{\text{best}}(t)$ is small or $\partial J^{\text{best}}(t)$ is small, $\omega(t)$ is small.

When $J^{\text{best}}(t)$ is large or $\partial J^{\text{best}}(t)$ is large, large $\beta(t)$ is required, in order to inject more random velocity components into the particles. With large $\beta(t)$, the IPSO intends to move the poor particles with large errors to good solutions with small errors. When $J^{\text{best}}(t)$ is small or $\partial J^{\text{best}}(t)$ is small, a small $\beta(t)$, is needed in order to refine the particles of the IPSO to slightly better solutions. Based on this linguistic understanding regarding $\beta(t)$, the following two basic principles, **P3** and **P4**, for determining $\beta(t)$ are used:

**P3**: If $J^{\text{best}}(t)$ is large or $\partial J^{\text{best}}(t)$ is large, $\beta(t)$ is large.

**P4**: If $J^{\text{best}}(t)$ is small or $\partial J^{\text{best}}(t)$ is small, $\beta(t)$ is small.

Using the four basic principles, **P1**, **P2**, **P3** and **P4**, nine fuzzy rules embedded in the fuzzy inference system are developed:

Rule $i$: **If** $J^{\text{best}}(t) = M_j^{J^{\text{best}}}$ **AND** $\partial J^{\text{best}}(t) = M_k^{\partial J^{\text{best}}}$,
**THEN** $\omega(t) = \sigma_{\bar{p}(i)}(t)$ **AND** $\beta(t) = \chi_{\bar{p}(i)}(t)$

where $i$=1, 2…, 9; $M_1^{J^{\text{best}}}$, $M_2^{J^{\text{best}}}$, and $M_3^{J^{\text{best}}}$ represent the memberships of 'Small', 'Medium', and 'Large' with respect to $J^{\text{best}}(t)$ respectively; $M_1^{\partial J^{\text{best}}}$, $M_2^{\partial J^{\text{best}}}$, and $M_3^{\partial J^{\text{best}}}$ represent the memberships of 'Small', 'Medium', and 'Large' with respect to $\partial J^{\text{best}}(t)$ respectively; and $\bar{p}(i) = \left[1 + \lfloor i/3 \rfloor, 1 + i - \lfloor i/3 \rfloor \cdot 3\right]$ is an array regarding the position of the fuzzy rule table, of which $\lfloor x \rfloor$ is defined by the largest integer not greater than $x$.

As recommended by Eberhart and Y. Shi [13], the range of the inertia weight, $\omega(t)$, of the PSO is within the range between 0.1 and 1.1. Hence, $\sigma_{\bar{p}(i)}(t)$ which intends to determine $\omega(t)$, is divided into three levels, 'Small', 'Medium' and 'Large', where 'Small' is within the range between 0.1 and 0.3; 'Medium' is within the range between 0.3 and 0.8; and 'Large' is within the range between 0.8 and 1.1. Also, $\chi_{\bar{p}(i)}(t)$ which intends to determine $\beta(t)$, is divided into three levels, 'Small', 'Medium' and 'Large', where $\beta(t)$ is within the range between 0 and 1. 'Small' is within the range between 0 and 0.33. 'Medium' is within the range between 0.33 and 0.66. 'Large' is within the range between 0.66 and 1.0. Figures 3(a) and 3(b) illustrate the fuzzy rule tables for $\sigma_{\bar{p}(i)}(t)$ and $\chi_{\bar{p}(i)}(t)$ respectively, which have been implemented in this research.

$\sigma_{\bar{p}(i)}(t) =$

| $\partial J^{best}(t)$ \ $J^{best}(t)$ | $M_1^{J^{best}(t)}$ | $M_2^{J^{best}(t)}$ | $M_3^{J^{best}(t)}$ |
|---|---|---|---|
| $M_1^{\partial J^{best}(t)}$ | 0.1 | 0.15 | 0.2 |
| $M_2^{\partial J^{best}(t)}$ | 0.3 | 0.5 | 0.8 |
| $M_3^{\partial J^{best}(t)}$ | 1.0 | 1.05 | 1.1 |

**Fig. 3(a)** Fuzzy rule table for $\sigma_{\bar{p}(i)}(t)$

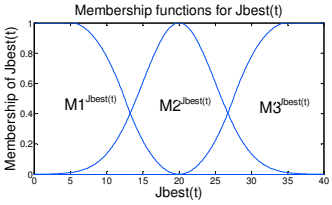
$\chi_{\bar{p}(i)}(t) =$

| $\partial J^{best}(t)$ \ $J^{best}(t)$ | $M_1^{J^{best}(t)}$ | $M_2^{J^{best}(t)}$ | $M_3^{J^{best}(t)}$ |
|---|---|---|---|
| $M_1^{\partial J^{best}(t)}$ | 0.1 | 0.13 | 0.22 |
| $M_2^{\partial J^{best}(t)}$ | 0.28 | 0.5 | 0.72 |
| $M_3^{\partial J^{best}(t)}$ | 0.81 | 0.86 | 0.9 |

**Fig. 3(b)** Fuzzy rule table for $\chi_{\bar{p}(i)}$

The three fuzzy membership functions of $J^{\text{best}}(t)$ are denoted by $\mu^1_{J^{\text{best}}}(J^{\text{best}}(t))$, $\mu^2_{J^{\text{best}}}(J^{\text{best}}(t))$ and $\mu^3_{J^{\text{best}}}(J^{\text{best}}(t))$ with respect to $M_1^{J^{\text{best}}}$, $M_2^{J^{\text{best}}}$ and $M_3^{J^{\text{best}}}$ respectively, where $M_1^{J^{\text{best}}}$ $M_2^{J^{\text{best}}}$ and $M_3^{J^{\text{best}}}$ represents the 'Small', 'Medium' and 'Large' memberships for the errors respectively. The error dominates the 'Small' membership, $M_1^{J^{\text{best}}}$, when it is around 5%. When the errors are around 20% and 35%, they dominates the 'Medium' membership, $M_2^{J^{\text{best}}}$, and the 'Large' membership, $M_3^{J^{\text{best}}}$ respectively. $M_1^{J^{\text{best}}}$, $M_2^{J^{\text{best}}}$, and $M_3^{\partial J^{\text{best}}}$ are represented by the Gaussian form as illustrated in Figure 4(a).

Also, the three fuzzy membership functions of $\partial J^{\text{best}}(t)$ are denoted by $\mu^1_{\partial J^{\text{best}}}(\partial J^{\text{best}}(t))$, $\mu^2_{\partial J^{\text{best}}}(\partial J^{\text{best}}(t))$ and $\mu^3_{\partial J^{\text{best}}}(\partial J^{\text{best}}(t))$ with respect to $M_1^{\partial J^{\text{best}}}$, $M_2^{\partial J^{\text{best}}}$ and $M_3^{\partial J^{\text{best}}}$ respectively, where $M_1^{\partial J^{\text{best}}}$, $M_2^{\partial J^{\text{best}}}$ and $M_3^{\partial J^{\text{best}}}$ represent the 'Small', 'Medium' and 'Large' memberships for the changes of errors respectively. The change of error dominates the 'Small' membership, $M_1^{\partial J^{\text{best}}}$, when the change of error is around 4%. When the changes of errors are around 10% and 18%, they dominate the 'Medium' membership, $M_2^{\partial J^{\text{best}}}$, and the 'Large' membership, $M_3^{\partial J^{\text{best}}}$ respectively. $M_1^{\partial J^{\text{best}}}$, $M_2^{\partial J^{\text{best}}}$, and $M_3^{\partial J^{\text{best}}}$ are represented by the Gaussian form as illustrated in Figures 4 (b).



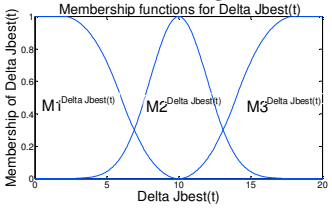




**Fig. 4(a)** Membership functions of $J^{\text{best}}(t)$ **Fig. 4(b)** Membership functions $\partial J^{\text{best}}(t)$

The values of $\omega(t)$ and $\beta(t)$ are given by taking the weighted average with respect to the membership functions:

$$\omega(t) = \sum_{i=1}^{9} m_i(t)\,\sigma_{\bar{p}(i)}(t), \tag{15}$$

and

$$\beta(t) = \sum_{i=1}^{9} m_i(t)\,\chi_{\bar{p}(i)}(t) \tag{16}$$

respectively, where

$$m_i(t)=\frac{\mu^i_{J^{\text{best}}}\left(J^{\text{best}}(t)\right)\times\mu^i_{\partial J^{\text{best}}}\left(\partial J^{\text{best}}(t)\right)}{\sum_{j=1}^{3}\sum_{k=1}^{3}\mu^j_{J^{\text{best}}}\left(J^{\text{best}}(t)\right)\times\mu^k_{\partial J^{\text{best}}}\left(\partial J^{\text{best}}(t)\right)}. \quad (17)$$

## IV. TRAFFIC FLOW FORECASTING ON A FREEWAY

In this section, the effectiveness of the IPSO for developing NNs for short-term traffic flow forecasting is evaluated based on the traffic flow data collected from a freeway in Western Australia.

First, comparisons between the IPSO and the other algorithms are utilized based on the traffic flow data captured by the on-road sensors, where a traffic jam occurred and none of the on-road sensors used for capturing the data is damaged. The objective is to evaluate the adaptive capability of the IPSO switching from different traffic flow conditions, where all on-road sensors are able to work normally. Second, the performance of the IPSO is further evaluated based on the contingent cases, where some of the on-road sensors used for capturing the traffic flow conditions are damaged unexpectedly. The objective is to evaluate the adaptive capability of the IPSO from a proper on-road sensor configuration to a damaged on-road sensor configuration.

### A. Traffic flow data

The traffic flow data was collected by fourteen on-road sensors ($D_1$ to $D_{14}$) installed along the Mitchell Freeway, Western Australia, which are illustrated in Figure 5. Three on-road sensors were installed near the off-ramp, near the on-ramp, as well as between the off-ramp and on-ramp, for the Reid Highway ($D_1$ to $D_3$), Karrinyup Road ($D_6$ to $D_8$), Cedric Street ($D_9$ to $D_{11}$) and Hutton Street ($D_{12}$ to $D_{14}$), respectively. On Erindale Road, two on-road sensors ($D_4$ and $D_5$) were installed near the off-ramp and near the on-ramp. The sampling time for all on-road sensors was half minute (or 30 seconds). The distance between Reid Highway and Hutton Street is about 7 kilometers, where the speed limit for these sections of the Mitchell Freeway is 100 km/hour. The NNs was developed to forecast future traffic flow conditions with five sampling times ahead.

The traffic flow data were collected over the 2-hour peak traffic period (7.30 – 9.30 am) on the Monday, which is the busiest business day of the week. All these traffic flow data were collected from the six weeks including weeks 6, 7, 8, 9, 11 and 12 in 2009. The proposed algorithm was evaluated by two different sets of training data, namely large training set and small training set, which involve larger and smaller numbers of training data respectively.

For the large training set, more training data was used for developing the NNs. The data collected from the first five weeks, weeks 6 to 9 and week 11, was used for training the NNs. Hence, 1200 pieces of data were used for training. For the small training set, less training data was used for developing the NNs. The data collected from weeks 6 to 9 was used as the training data. Hence, there were 960 pieces of training data. The second sub-set of traffic flow data, namely test data, collected from week 12, was used to evaluate the generalization capability of the developed NNs. Hence, 240 pieces of traffic flow data were used to validate the developed NNs.

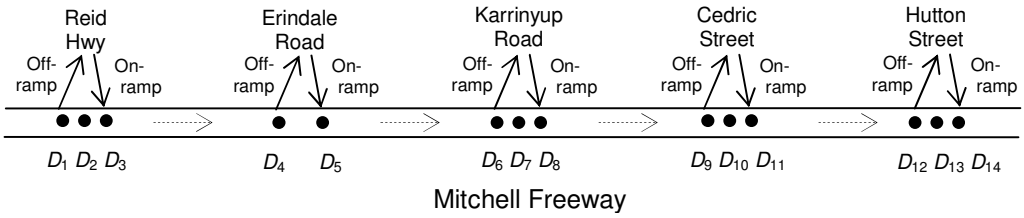



**Fig. 5** On-road sensor configuration on the Mitchell Freeway

### B. Tested algorithms

The following parameters were used in the IPSO: both the acceleration constants $\phi_1$ and $\phi_2$ were set at 2.05, and the maximum velocity $v_{\max}$ was 0.2, which can be found in reference [22, 23]; the number of particles in the swarm was 50; The specific number of hidden nodes, $N_h$, used depends on the particular problem. However, no well-defined algorithm exists for determining the optimal $N_h$. As determining the optimal number of hidden nodes is not the goal of this research, we used the one recommended by Mirchandani and Cao [36], where $N_h \approx \log_2(T)$ with $T$ be the number of training data. For both large and small training sets, $N_h = 10$ were used, where $T$ are equal to 1200 and 960 for large training set and for small training set respectively, and $10 \approx \log_2(1200) \approx \log_2(960)$.

The termination conditions can be divided into two: 1) if $t<T_{off\text{-}line}$, the IPSO terminates, until the training errors are below 10%; 2) if $t>T_{off\text{-}line}$, the IPSO terminates, until the on-road sensors stop capturing the traffic flow data for the IPSO. The performance of the IPSO is compared with the following population-based stochastic algorithms:

a) Classical PSO [22] namely CPSO: the mechanisms and the parameters used in the CPSO are identical to those of the IPSO, except that IPSO utilizes equation (12) to update the velocity of each element of the particle, and CPSO utilizes equation (10) to update the velocity.
b) Advanced particle swarm optimization [28], namely APSO, is identical to the CPSO except that APSO utilizes another mechanism for updating the positions of the particles. It includes the worst particle, in order to update the velocity of each element of the particle, while both CPSO and IPSO include only the best particle. We compare the IPSO with APSO, because APSO can obtain very good results when adapting the parameters of the fuzzy neural networks for on-line control of power converter [28].
c) Improved genetic algorithm [26] namely IGA: which has been used to optimize the NNs, where the structure of the NNs is identical to the one implemented by IPSO. IGA utilizes two

genetic operations, namely improved mutation and improved crossover, which outperforms several genetic algorithms developed for optimizing the generalization capability of the NNs [26].

Apart from those algorithms, two recent evolutionary algorithms [49] which have been developed for generating NNs for traffic flow forecasting, was used for comparing with the IPSO. The two evolutionary algorithms are described as following:

d) First version of Vlahogianni's genetic algorithm namely VGA-I: The NNs are trained with Levenberg-Marquardt algorithm [3, 16] using the training data. The genetic algorithm was used for determining the parameters of the Levenberg-Marquardt algorithm, namely step size for training the NNs, training momentum, and the number of hidden nodes of the NNs.

e) Second version of Vlahogianni's genetic algorithm namely VGA-II: The NNs are trained with Levenberg-Marquardt algorithm (LM) [16] using the training data, where the adaptive step size was used for training the NNs. The genetic algorithm was used for determining the optimal number of hidden nodes of the NNs.

Both VGA-I and VGA-II only use the training data to develop the NNs for traffic flow forecasting. The parameters and the configurations of the NNs are kept unchanged with respect to the newly captured data after $t>T_{off\text{-}line}$. Hence, the developed NN does not involve adaption with newly captured data. By comparing both VGA-I and VGA-II and the other algorithms involved adaption, the differences between the algorithms involved no adaption and those involved adaption can be indicated.

The PSO parameters including acceleration constants and inertia weight used in CPSO and APSO are the same as those used in [13] and [28] respectively. The GA parameters including mutation rate and crossover rate used in IGA, VGA-I and VGA-II are same as those used in [26]. To make a fair comparison, the computational effects used in all these population-based stochastic algorithms, IPSO, CPSO, APSO, IGA, VGA-I and VGA-II are identical. The number of iterations was pre-defined as 200 and the population size was 50, when $t<T_{off\text{-}line}$.

*C. Simulation results*

After consulting with the engineering personnel from Mainroad, Western Australia (which supporting this research), the on-road sensors installed at some particular locations are not likely to work properly in very poor weather conditions such as rainstorms or floods. Hence, the following two contingencies were considered:

Contingency 1: The three on-road sensors installed at Hutton Street cannot work properly and no signal can be provided by the three on-road sensors, from 8.20 am to 10.00 am.

Contingency 2: The three on-road sensors installed at Karrinyup Road cannot work properly and they can generate only a random white noise signal of a value between zero to a hundred, from 9.00 am to 10.00 am.

Using these two contingencies, the following 4 Cases were built:

Case 1: traffic flow forecasting without any damaged on-road sensors;

Case 2: traffic flow forecasting under contingency 1 considered only;

Case 3: traffic flow forecasting under contingency 2 considered only; and

Case 4: traffic flow forecasting under both contingency 1 and contingency 2 considered.

Mean absolute error was used to evaluate the generalization capabilities of the NNs.

$$MAE = \frac{1}{m}\frac{\sum_{i=1}^{m}\left|y_{14}\left(i\cdot T_s\right)-\hat{y}_{14}\left(i\cdot T_s\right)\right|}{y_{14}\left(i\cdot T_s\right)}\cdot 100\%, \qquad (18)$$

$y_{14}\left(i\cdot T_s\right)$ is the $i$-th test data captured at the on-ramp of Hutton Street, at time $\left(i\cdot T_s\right)$; all $y_{14}\left(i\cdot T_s\right)$ are larger than zero; $\hat{y}_{14}\left(i\cdot T_s\right)$ is the estimate of traffic flow condition at the on-road sensor, $D_{14}$, which is forecast by the NNs, and the traffic flow data were captured between time, $\left(i-m-p\right)\cdot T_s$, to time, $\left(i-m\right)\cdot T_s$, with $p=10$ and $m=5$.

All these algorithms were implemented using Matlab 7.7 in a PC which has a CPU of Intel(R) Core(TM)2 Duo 2.66GHz and a memory of 7.99GB. As the algorithms are stochastic algorithms, different results can be obtained with different runs. Therefore, CPSO, APSO, IPSO, CGA, VGA-I and VGA-II, were run for 30 times, and the results of the 30 runs were recorded. Table 1 shows the results obtained by the algorithms for large training set. The average of MAE and variance of MAE among the 30 runs of the algorithms are shown. For case 1, the table shows that the averages of mean test errors obtained by VGA-I and VGA-II, which involved no adaption of newly captured traffic flow data, were poorer than those of CPSO, APSO, IPSO and CGA, which involved adaption. For cases 2, 3 and 4, similar results can be illustrated that the results obtained by CPSO, APSO, IPSO and CGA are better than those obtained by VGA-I and VGA-II. These results clearly demonstrate the effectiveness of the algorithms involved adaption of newly captured traffic flow data. Also, among the four algorithms involved adaption, the results show that the results obtained by the IPSO is generally better than the other algorithms, CPSO, APSO, and CGA.

In addition, the *t*-test [7] was used to evaluate the significance of the hypothesis that the sample means of the test errors obtained by the proposed IPSO are smaller than those obtained by the other algorithms (CPSO, APSO, IPSO, CGA, VGA-I and VGA-II). The *t*-values between IPSO and the other algorithms are also shown in Table 1. Based on the *t*-distribution table, if the *t*-value is higher than 1.699, the significance is 95% confidence, which means that the test errors obtained by the IPSO are smaller than those trained by the other algorithm with 95% confidence level. The *t*-value can be determined by:

$$t\text{-value} = \frac{\mu_2-\mu_1}{\sqrt{\sigma_2^2/N_2+\sigma_1^2/N_1}},$$

where $\mu_1$ is the mean test error obtained by the IPSO and $\mu_2$ is the one for the other compared algorithm; $\sigma_1^2$ is the variance of test errors obtained by the IPSO and $\sigma_2^2$ is the one for the other compared algorithm; $N_1$ and $N_2$ are the number of tests performed by IPSO and the other compared algorithm, respectively. In general, the results indicate that the significances of differences between IPSO to the non-adaptive algorithms (i.e. VGA-I and VGA-II) are large than those between IPSO to the

adaptive algorithms (i.e. CPSO, APSO and CGA). These finding illustrates the effectiveness of the adaptive algorithm. In general, IPSO can obtain significantly better results than those obtained by all other tested algorithms. Hence, the effectiveness of the IPSO can be further demonstrated.

Table 2 shows the results obtained by the algorithms which used small training set. In general, similar results can be found as those used large training set. The results obtained by the algorithms involved adaption are generally better than those involved no adaption. Also, the *t*-test shows that IPSO can obtain significantly better results than those obtained by the other tested algorithms.

Table 1 Results obtained by the algorithms using large training set

| | | IGA | CPSO | APSO | VGA-I | VGA-II | IPSO |
|---|---|---|---|---|---|---|---|
| **Case 1** | Mean errors | 6.54 | 6.06 | 5.90 | 7.90 | 7.71 | 5.21 |
| | Variance of errors | 1.10 | 1.65 | 2.22 | 0.99 | 0.99 | 2.68 |
| | T-values | 3.75 | 2.24 | 1.71 | 7.70 | 7.16 | Nil |
| **Case 2** | Mean errors | 7.55 | 7.21 | 6.69 | 8.91 | 8.31 | 6.28 |
| | Variance of errors | 1.65 | 2.28 | 2.51 | 1.07 | 1.04 | 2.38 |
| | T-value | 3.46 | 2.36 | 1.01 | 7.75 | 6.02 | Nil |
| **Case 3** | Mean errors | 10.15 | 9.42 | 9.06 | 13.84 | 13.37 | 8.13 |
| | Variance of errors | 1.69 | 3.74 | 3.45 | 0.81 | 1.02 | 3.35 |
| | T-values | 4.93 | 2.65 | 1.95 | 15.33 | 13.73 | Nil |
| **Case 4** | Mean errors | 11.57 | 10.81 | 10.52 | 14.24 | 14.99 | 9.52 |
| | Variance of errors | 2.68 | 3.64 | 3.97 | 1.14 | 1.342 | 3.66 |
| | T-values | 4.46 | 2.61 | 1.98 | 11.80 | 13.39 | Nil |

Table 2 Results obtained by the algorithms using small training set

| | | IGA | CPSO | APSO | VGA-I | VGA-II | IPSO |
|---|---|---|---|---|---|---|---|
| **Case 1** | Mean errors | 7.27 | 7.03 | 6.66 | 8.49 | 7.91 | 5.85 |
| | Variance of errors | 1.16 | 1.76 | 2.44 | 1.03 | 1.04 | 2.71 |
| | T-values | 3.96 | 3.06 | 1.97 | 7.48 | 5.82 | Nil |
| **Case 2** | Mean errors | 7.85 | 7.69 | 7.32 | 9.54 | 9.18 | 6.80 |
| | Variance of errors | 1.71 | 2.44 | 2.63 | 1.18 | 1.21 | 2.56 |
| | T-value | 2.76 | 2.17 | 1.23 | 7.77 | 6.69 | Nil |
| **Case 3** | Mean errors | 10.47 | 9.98 | 9.90 | 14.08 | 13.93 | 8.98 |
| | Variance of errors | 1.76 | 3.81 | 3.70 | 1.00 | 1.27 | 3.27 |
| | T-values | 3.65 | 2.07 | 1.92 | 13.53 | 12.74 | Nil |
| **Case 4** | Mean errors | 12.17 | 11.87 | 11.24 | 14.75 | 14.93 | 10.32 |
| | Variance of errors | 2.85 | 3.78 | 4.04 | 1.29 | 1.48 | 3.78 |
| | T-values | 3.94 | 3.09 | 1.80 | 10.78 | 11.01 | Nil |

Also, Figure 6 shows the computational time used by each algorithm for each iteration in order to adapt the NN parameters. It shows that the computational time taken for each iteration when using the population-based stochastic methods (CPSO, APSO, IPSO and IGA), is about 1 to 2.5 seconds, while the computational time taken by the adapt-LM is less than 0.5 seconds. Therefore, the computational time taken by the adapt-LM is less than that taken when using the population-based stochastic methods. Even if the computational time taken by the adapt-LM is much less than that of the population-based stochastic methods, the accuracies in term of traffic flow forecasting conditions achieved by the adapt-LM are poorer than those obtained by the population-based stochastic methods. As accuracy in traffic flow forecasting is important, one may still use the population-based stochastic methods to conduct traffic flow forecasting. Figure 6 also shows that the computational time taken for all population-based stochastic methods, including the proposed IPSO is much less than the sampling time of 30 seconds. Therefore, IPSO is implementable, in order to adapt traffic flow data which is captured with the sample time of 30 seconds. The sample time of IPSO can be set smaller, if the computational time taken for the IPSO for one iteration is smaller. To achieve this, the following two approaches can be used: a) the IPSO can be implemented on a more powerful microprocessor, in order to reduce the computational time; b) the IPSO is currently implemented by Matlab, which is a high level programming language. If the IPSO is implemented by a lower level programming language, the computational time required by the IPSO is smaller, and thus, shorter sampling times can be used.

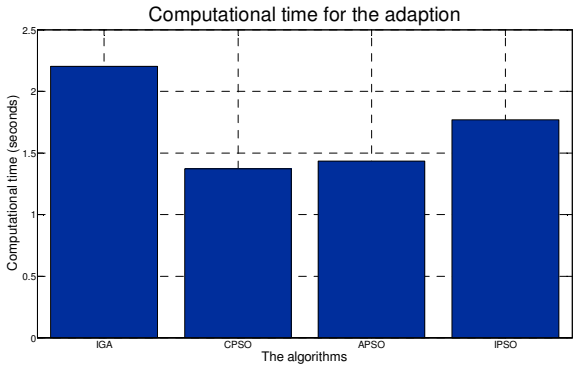



**Fig. 6** Computational time used on each algorithm

For Case 1, the simulation results for Monday (week 12) in terms of average speeds of vehicles are depicted in Figure 7, when no contingency has occurred. It shows that from the 40th sample to the 150th sample (about 7.51-8.45 am), the average speeds of vehicles were about 50km/hours, which is far below the speed limit. Hence, the traffic flow was congested and there was a traffic jam during that time. From the 170th sample to the 210th sample (about 8.55-9.15 am), the average speeds of vehicles are about 90 km/hr, which are nearer the speed limit. Hence, the traffic flow was smoother and there was no traffic jam during that period of time. The data being investigated contains traffic jam conditions. Figures 7 shows two results: the upper one shows the forecasting obtained by the NN without adaption, which is developed based on non-adapt-LM; and the lower one shows the forecasting obtained by the adaptive NN, which was generated by the IPSO. By comparing the results obtained by the IPSO and the non-adapt-LM, we can see that the better average speed estimates are produced by the IPSO rather than by the non-adapt-LM. In general, the results forecast by IPSO are more accurate than those forecast by non-adapt-LM. Therefore, the IPSO can obtain more accurate results for forecasting traffic flow conditions.

Figures 8, 9 and 10 show the results for Case 2, Case 3 and Case 4, respectively, which involve the contingencies. For Case 2, the simulation results are depicted in Figure 8, where the on-road sensors, $D_{12}$, $D_{13}$ and $D_{14}$, were damaged at 8.20 (at the 100th sample time). It can be clearly observed that relatively larger errors between the actual traffic flow and the estimated traffic flow were produced, when the non-adapt-LM was used to perform the forecasting. Hence, the non-adapt-LM performs poorly and only unacceptable estimates are generated. When the adaptive NN developed by the IPSO was used, relatively smaller errors occurred. For Case 3, the on-road sensors, $D_6$, $D_7$ and $D_8$, were damaged at 9.00 (at the 160th sample time). Figure 9 shows similar results in that relatively smaller errors occurred, when the NNs developed by the IPSO were used to perform the forecasting. Case 4 is the situation where there was the most serious damage, since the on-road sensors, $D_{12}$, $D_{13}$ and $D_{14}$, as well as the on-road sensors, $D_6$, $D_7$ and $D_8$, were damaged at 8.20 (at the 100th sample time), as well as at 9.00 (at the 160th sample time), respectively. It can be observed from Figure 10 that a significantly greater forecasting error was produced by non-adapt-LM, while a much smaller forecasting error was produced by the IPSO. Although, small impulses with relatively large error were produced by the IPSO at the 100th sampling time and the 160th sample time, after the contingencies occurred, the forecasting errors immediately decreased to low levels after a short time. Therefore, these simulation results demonstrate the effectiveness of the IPSO since it can adapt to changing on-road sensor configurations, which are damaged from time to time.

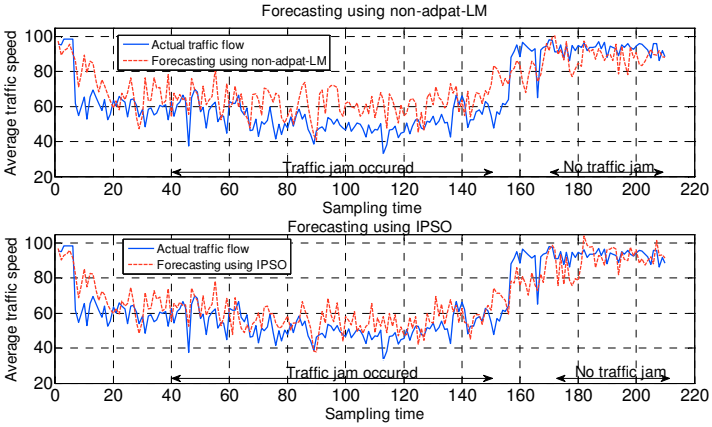



**Fig. 7** Forecasting results for Case 1

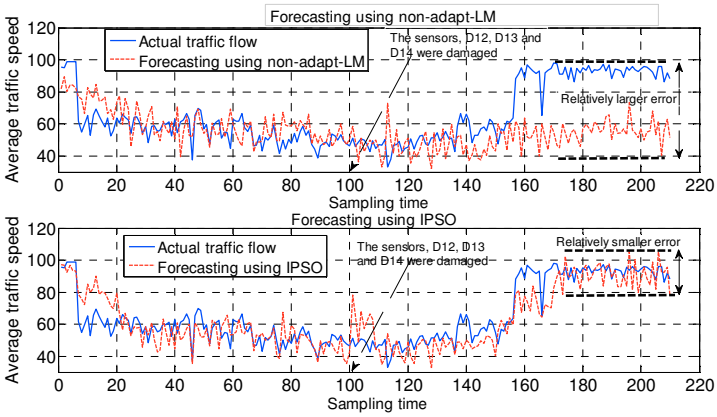



**Fig. 8** Forecasting results for Case 2

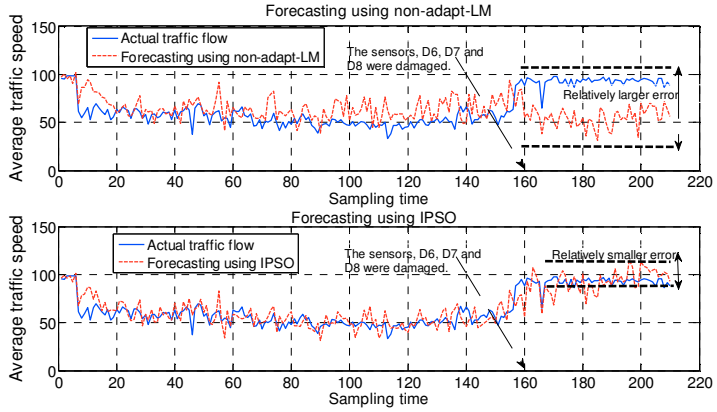



**Fig. 9** Forecasting results for Case 3

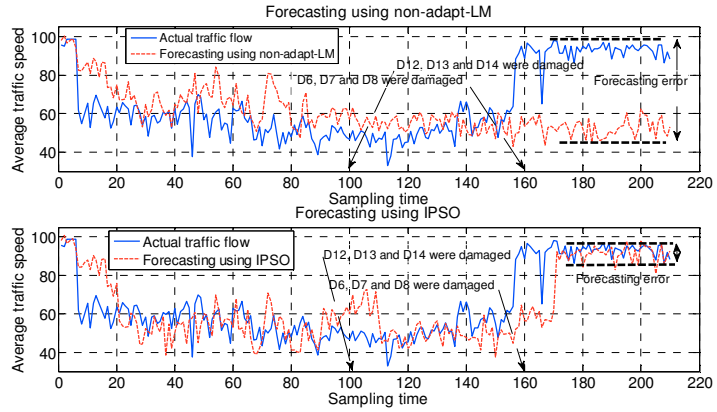



**Fig. 10** Forecasting results for Case 4

## VI. Conclusion

In this paper, an intelligent particle swarm optimization algorithm (IPSO) is proposed for the development of short-term traffic flow predictors, in order to tackle the time-varying assumptions underlying the currently used methods, where: i) the characteristics of current data captured by on-road sensors are assumed to be time-invariant with respect to those of the historical data which was used to developed short-term traffic flow predictors; and ii) the configuration of the on-road sensor systems is assumed to be time-invariant. By tackling these two time-varying assumptions, the IPSO is developed by integrating the mechanisms of particle swarm optimization, neural network and fuzzy inference systems, in order to develop short-term traffic flow predictors, which can adapt to the time-varying traffic flow data and the time-varying configurations of on-road sensor systems.

In the IPSO, the particle in the swarm is used to represent the neural network with a flexible structure, in order to forecast short-term traffic flow, without these two time-invariant assumptions. Then, the IPSO uses the mechanisms of particle movements, in order to locate with the time-varying optimum of the short-term traffic flow predictor. To further enhance the adaptive capability of the IPSO, the fuzzy inference system is used to inject activating components into the particles, in order to let the particles search for good solutions when the traffic flow accuracy obtained by the IPSO is low.

The effectiveness of the IPSO was evaluated by applying it to the development of short-term traffic flow predictors to forecast

traffic flow conditions on a section of freeway in Western Australia, whose traffic flow data was captured on-line by an on-road sensor system consisting of fourteen on-road sensors. The four cases, included one with traffic congestion, and three which involved on-road sensors damaged at different times and different locations, were considered. The forecasting results obtained by the short-term traffic flow predictor developed by IPSO were more accurate than those obtained by the other tested algorithms: the Levenberg-Marquardt approach, the genetic algorithm and particle swarm optimization approaches taken from the recent literature. These results clearly demonstrated the effectiveness of the proposed IPSO.

Future work will be carried out by further enhancing and evaluating the effectiveness of the proposed IPSO. Thus, it can be divided into two categories: a) Incorporation with the other heuristic algorithms may enhance the effectiveness of the proposed IPSO for short-term traffic flow forecasting. Investigation of the effectiveness of the incorporating ant colony algorithm, artificial immune systems or bee colony algorithm into IPSO will be conducted; and b) As manufacturing processes which involve control operations by sensor data are common [2, 6, 24], the effectiveness of the IPSO can be further evaluated by applying it to control manufacturing processes which are operated with sensor systems.

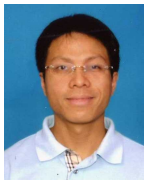
## ACKNOWLEDGEMENT

The authors wish to express their sincere thanks to Steve S.H. Ling and Jaipal Singh for many useful discussions and valuable suggestions. They would also like to acknowledge their very useful comments for this research.

## REFERENCES

1. A. Alessandri, R. Bolla, M. Gaggero, M. Repetto, "Modeling and identification of nonlinear dynamics for freeway traffic by using information from a mobile cellular network", *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 952-959, 2009.
2. R. Antonello, R. Oboe, L. Prandi, and F. Biganzoli, "Automatic mode matching in MEMS vibrating gyroscopes using extremum-seeking control", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3880-3891, 2009.
3. J. Arif, N.R. Chaudhuri, S. Ray and B. Chaudhuri, "Online Levenberg-Marquardt algorithm for neural network based estimate and control of power systems", *Proceedings of International Joint Conference on Neural Networks*, pp. 199-206, 2009.
4. A. Aw and M. Rascle, "Resurrection of 'second order' models of traffic flow", *Siam Journal on Applied Mathematics*, vol. 60, no. 3, pp. 916-938, 2000.
5. F. Bergh and A.P. Engelbrecht, "A study of particle swarm optimization particle trajectories", *Information Sciences*, vol. 176, no. 8, pp.937-971, 2006.
6. F. Bonaccorso, L. Cantelli and G. Muscato, "An arc welding robot control for a shaped metal deposition plant: modular software interface and sensors", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3126-3132, 2011.
7. G. E.-P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experiments: Design, Innovation, and Discovery*, 2nd ed. New York: Wiley, 2005.
8. K.Y. Chan, T.S. Dillon, C.K. Kwong, "Polynomial modelling for time-varying systems based on a particle swarm optimization algorithm", *Information Sciences*, 181(9) 1623-40, 2011.
9. C. Chen, B. Zhang and G. Vachtsevanos, "Machine condition prediction based on adaptive neuro-fuzzy and high-order particle filtering", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4353-4364, 2011.
10. G.A. Davis, and N.L. Nihan, "Nonparametric regression and short-term freeway traffic forecasting", *Journal of Transportation Engineering*, vol. 177, no. 2, pp. 178–188, 1991.
11. H. Dia, "An object-oriented neural network approach to short-term traffic forecasting", *European Journal of Operational Research*, vol. 131, pp. 253-261, 2001.
12. M. Dougherty, "A review of neural networks applied to transport", *Transportation Research Part C: Emerging Technologies*, vol. 3, no. 4, pp. 247-260, 1995.
13. R.C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization", in *Evolutionary Programming VII*. New York: Springer-Verlag, LNCS, vol. 1447, 1998, pp. 611-616.
14. R.C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization", in *Proceedings of the IEEE Congress on Evolutionary Computation*, San Diego, USA, 2000, pp. 84–88.
15. Y. Gao and M.J. Er, "NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches", *Fuzzy Sets and Systems*, vol. 150, pp. 331-350, 2005.
16. M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural network design*, Boston ; London : PWS Pub, 1996.
17. W.C. Hong, P.F. Pai, S.L. Yang and C.Y. Lai, "Continuous ant colony optimization in a SVR urban traffic forecasting model", *Lecture Note in Computer Sciences*, vol. 4507, pp. 765-773, 2007.
18. W.C. Hong, Y. Dong, F. Zheng and C.Y. Lai, "Forecasting urban traffic flow by SVR with continuous ACO", *Applied Mathematical Modelling*, vol. 35, pp. 1282-1291, 2011.
19. W.C. Hong, "Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm", *Neurocomputing*, vol. 74, pp. 2069-2107, 2011.
20. W.C. Hong, "Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting", *Neural Computing and Applications*, vol. 21, pp. 583-593, 2012.
21. G.N. Karystinos and D.A. Pados, "On overfitting, generalization, and randomly expanded training sets", *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1050-1057, 2000.
22. J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proc. 30th IEEE Conf. Decision and Control*, vol. 4, pp. 1942-1948, 1995.
23. J. Kennedy and R.C. Eberhart, *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.
24. J. Kjellsson, A.E. Vallestad, R. Steigmann, and D. Dzung, "Integration of a Wireless I/O Interface for PROFIBUS and PROFINET for Factory Automation", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4279-4287, 2009.
25. C. Ledoux, "An urban traffic flow model integrating neural network", *Transportation Research*, vol. 5C, pp. 287-300, 1997.
26. F.H.F. Leung, H.K. Lam, S.H. Ling and P.K.S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm", *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 79-88, 2003.
27. F. Liang, "Bayesian neural networks for nonlinear time series forecasting", *Statistics and Computing*, vol. 15, pp. 13-29, 2005.
28. F.J. Lin, L.T. Teng, J.W. Lin, and S.Y. Chen, "Recurrent functional link based fuzzy-neural-network-controlled induction-generator system using improved particle swarm optimization", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1557-1577,2009.
29. Y.X. Liao, J.H. She and M. Wu, "Integrated hybrid-PSO and fuzzy-NN decoupling control for temperature of reheating furnace", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 7, pp. 2704-22714, 2009.
30. Q. Li, W.Chen, Y. Wang, S. Liu and J. Jia, "Parameter identification for PEM fuel cell mechanism model based on effective informed adaptive particle swarm optimization", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2410-2419, 2011.
31. F.J. Lin, L.T. Teng, J.W. Lin and S.Y. Chen, "Recurrent functional-link-based fuzzy-neural-network controlled induction generator system using improved particle swarm optimization", *IEEE Transactions Industrial Electronics*, vol. 56, no. 5, pp. 1557-1577, 2009.
32. C.H. Lu and C.C. Tsai, "Adaptive predictive control with recurrent neural network for industrial processes: an application to temperature control of a variable-frequency oil cooling machine", *IEEE Transactions on Industrial Electronics*, vol. 55, no. 3, pp. 1366-1375, 2008.
33. C.H. Lu, "Design and application of stable predictive controller using recurrent wavelet neural networks", *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3733-3742, 2009.
34. C.H. Lu, "Wavelet fuzzy neural networks for identification and predictive control of dynamic systems", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 3046-3058, 2011.
35. P.A. Mastorocostas and J.B. Theocharis, "An orthogonal least-squares method for recurrent fuzzy-neural modeling", *Fuzzy Sets and Systems*, vol. 140, no. 2, pp. 285-300, 2003.
36. G. Mirchandani and W. Cao, "On hidden nodes for neural nets", *IEEE Transactions on Circuits and Systems*, vol. 36, no. 5, pp. 661-664, 1989.

37. I. Okutani and Y.J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory", *Transportation Research Part B: Methodology,* vol. 18, no. 1, pp. 1–11, 1984.
38. K.E. Parsopoulos and M.N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization", *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211-224, 2004.
39. L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria", *Neural Networks*, vol. 11, no. 4, pp. 761-777, 1998.
40. C. Quek, M. Pasquier and B.B.S. Lim, "POP-TRAFFIC: A Novel Fuzzy Neural Approach to Road Traffic Analysis and Prediction", *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 2, pp. 133-146, 2006.
41. P. Ross, "Exponential filtering of traffic data", *Transportation Research Record*, vol. 869, pp. 43–49, 1982.
42. B.L. Smith, and M.J. Demetsky, "Traffic flow forecasting: comparison of modeling approaches", *Journal of Transportation Engineering*, vol. 123, no. 4, pp. 261–266, 1997.
43. B.L. Smith, B.M. Williams and R.K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting", *Transportation Research Part C*, vol. 19, pp. 303-321, 2002.
44. D. Srinivasan, C.W. Chan and P.G. Balaji, "Computational intelligence based congestion prediction for a dynamic urban street network", *Neurocomputing*, vol. 72, pp. 2710-2716, 2009.
45. A. Stathopoulos, L. Dimitriou and T. Tskeris, "Fuzzy modeling approach for combined forecasting of urban traffic flow", *Computer Aided Civil and Infrastructure Engineering*, vol. 23, no. 7, pp. 521-535, 2008.
46. M.C. Tan, S.C. Wong, J.M. Xu, Z.R. Guan and P. Zhang, "An aggregation approach to short term traffic flow prediction", *IEEE Transactions on Intelligent Transportation systems*, vol. 10, no. 1, pp. 60-69, 2009.
47. K.S. Tang, K.F. Man, Z.F. Liu and S. Kwong, "Minimal fuzzy memberships and rules using hierarchical genetic algorithms", *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 162-169, 1998.
48. Y. Wang, M. Papageorgiou and A. Messmer, "A real-time freeway network traffic surveillance tool", *IEEE Transactions on Control Systems Technology*, vol. 14, no. 1, 18-32, 2006.
49. E.I. Vlahogianni, M.G. Karlaftis and J.C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach", *Transportation Research Part C*, vol. 13, no. 3, pp. 211-234, 2005.
50. B.M. Williams, P.K. Durvasula, D.E. Brown, "Urban Freeway Traffic Flow Prediction: Application of Seasonal Autoregressive Integrated Moving Average and Exponential Smoothing Models", *Transportation Research Record*, no. 1644, pp. 132-141, 1998.
51. H. Yin, S.C. Wong, J. Xu and C.K. Wong, "Urban traffic flow prediction using a fuzzy-neural approach", *Transportation Research Part C,* vol. 10, pp. 85-98, 2002.
52. Z.H. Zhan, J. Zhang and H.S.H. Chung, "Adaptive particle swarm optimization", *IEEE Transactions on Systems, Man, and Cybernetic – Part B: Cybernetics*, vol. 39, no. 6, pp. 1362-1381, 2009.
53. R.J. Wai, J.D. Lee and K.L. Chuang, "Real-time PID control strategy for Maglev transportation system via particle swarm optimization", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 629-646,, 2011.



**Kit Yan Chan** (M'11) received his MPhil degree in Electronic Engineering from City University of Hong Kong, Hong Kong in 2003 and his PhD degree in Computing from London South Bank University, United Kingdom in 2006. After his PhD study, he worked as a Postdoctoral Research Fellow in the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, until 2009. Currently, he is a Senior Research Fellow in the Department of Electrical and Computer Engineering, Curtin University, Australia. He has published 40 scientific papers in international journals and 2 research monographs. His research interests include computational intelligence and its applications to new product design, manufacturing process design, speech recognition and traffic flow forecasting.

**Tharam Dillon** (M'74–SM'87–F'98–LF'10) is a Fellow of the IEEE, ACS and IE(Aust). He is the professor at Department of Computer Science and Computer Engineering, La Trobe University, Australia. He is head of the IFIP International Task Force WG2.12/24 on Semantic Web and Web Semantics, the chairman of the IFIP WG12.9 on computational intelligence, the IEEE/IES Technical Committee on Industrial Informatics, and the IFIP Technical Committee 12 on Artificial Intelligence. He has published more than 750 papers published in international conferences and journals and is the author of 5 books and has another 5 edited books. His current research interests include Web semantics, ontologies, Internet computing, e-commerce, hy-brid neurosymbolic systems, neural nets, software engineering, database systems, and data mining. He is the Editor-in-Chief of the *International Journal of Computer Systems Science and Engineering* as well as the *Engineering Intelligent Systems*.

**Elizabeth Chang** (M'02–SM'07) received a Bachelor's Degree in Computer Science in 1985 from Beijing University. She received her Master's Degree and PhD in Computer Science in 1991 and 1996 respectively from La Trobe University. Currently, she is the professor at the School of Information Systems, Curtin University, Australia. She has co-authored 3 books and has published over 350 scientific papers as book chapters, and journals and conferences She is currently holding 6 ARC grants and a Tier 1 Centre of Excellence grant and obtained cash from ARC, industry partners as well as Research Centre of Excellence funds of over $5 million for 2002-2011. Her research interest includes: ontology and multi-agent systems, data mining for business intelligence, trust, security and risk in e-Business, XML, Web Services, P2P for collaborative environments, web engineering, IT for business and commerce, IT for health informatics, and IT for education.