Prediction of Short-term Traffic Variables using Intelligent Swarm-based Neural Networks

[*]**Kit Yan Chan (Member IEEE), Tharam Dillon (Fellow IEEE), J. Singh (Member IEEE), Elizabeth Chang (Senior Member IEEE)**

Digital Ecosystem and Business Intelligence Institute, Curtin University of Technology, WA 6102, Australia

Abstract: This paper presents an innovative algorithm integrated with particle swarm optimization and artificial neural networks to develop short-term traffic flow predictors, which are intended to provide traffic flow forecasting information for traffic management in order to reduce traffic congestion and improve mobility of transportation. The proposed algorithm aims to address the issues of development of short-term traffic flow predictors which have not been addressed fully in the current literature namely that: a) strongly non-linear characteristics are unavoidable in traffic flow data; b) memory space for implementation of short-term traffic flow predictors is limited; c) specification of model structures for short-term traffic flow predictors which do not involve trial and error methods based on human expertise; d) adaptation to newly-captured, traffic flow data is required. The proposed algorithm was applied to forecast traffic flow conditions on a section of freeway in Western Australia, whose traffic flow information is newly-captured. These results clearly demonstrate the effectiveness of using the proposed algorithm for real-time traffic flow forecasting.

Keywords: Adaptive neural network, particle swarm optimization, traffic flow forecasting, time-varying modeling, traffic management, evolutionary algorithm

1. Introduction

Forecasting accurate traffic flow conditions has long been identified as a proactive approach to regional traffic control [27], which can be broadly classified as short-term or long-term traffic flow forecasting [1]. Long-term forecasting provides monthly or yearly traffic flow conditions forecasting and is commonly used for long-term planning of transportation. Short-term forecasting, on the other hand, focuses on making predictions based on roadway sensor data, about the likely traffic flow changes in the short-term (typically within minutes), and provides the predictive functionality required for a proactive approach to traffic operations and control [33]. This paper focuses on short-term forecasting, where the interest is on producing forecasts after the system receives current traffic flow data from an on-road traffic facility. Exponential filtering [25] and Kalman filtering [21] approaches have been applied for forecasting short-term traffic flow based on past data. However, these filtering techniques can achieve only reasonable accuracies in short-time ahead traffic flow forecasting, when the traffic flows are relatively constant. More recently, linear statistical time series models, such as the autoregressive integrated moving average [28] and the seasonal one [16] have been used to improve the accuracies. However, their ability to capture the dynamics and nonlinearities that exist in the traffic flow is questionable, where unavoidable nonlinear characteristics of traffic flow are caused by the drivers' behaviors or reaction time regarding current traffic flow [3]. For example, different drivers have different reaction times when having to apply their brakes to stop the vehicle, when an obstacle is in front. Also, they have different behaviors when using their accelerators to control their car speeds, in order to match the current traffic flow conditions. Even though Alessandri et al. [2] proposed a macrosopoic approach to model nonlinear freeway traffic flow, uncertainties and measurement noise in terms of traffic flow have to be assumed to be small, which cannot be guaranteed in the real world.

[*] Kit Yan Chan is the corresponding author. His email address is Kit.Chan@curtin.edu.au

Another commonly used forecasting approach, neural networks (NNs), has been used to model highly non-linear traffic flow [5, 8, 9, 15] with more accurate forecasting results than those obtained by the classical forecasting approaches, where prior knowledge of the functional forms of traffic flow is not required by NNs. Also, NNs can learn from collected traffic flow data in the presence of uncertainty, which is not apparent because of the measurement noise or errors caused by traffic flow sensors. To further enhance the generalization capability of short-term traffic flow predictors, hybrid neural networks have been studied in recent research. For example, the Takagi-Sugeno fuzzy neural networks [12], which combine the mechanisms of fuzzy logic and feed-forward NNs, have been used to develop short-term traffic flow predictors [11, 24, 37]. Stathopoulos et al. [30] proposed to forecast short-term traffic flow by combining the forecasting outputs from neural networks and Kalman filters. Srinivasan et al. [29] developed a short-term traffic flow predictor which consists of two components: a fuzzy filter and a feed-forward neural network. Tan et al. [32] have developed a short-term traffic flow predictor, which combines the mechanisms of neural networks and the classical forecasting methods including moving average and autoregressive moving averages. Even if more accurate results can be obtained by these methods than by solely using NNs, their common limitations have been recognized: i) they were developed based on historical collected traffic flow data. 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; ii) the number of parameters used by the hybrid neural networks is great, and their memory footprints are larger than those required by the NNs with simpler configurations. More memory space and greater computational powers are therefore required to implement these hybrid NNs for adaption to newly-captured traffic flow data; iii) specification of NN structures such as NN configuration and number of hidden nodes is required to be pre-defined by a trial and error method which cannot guarantee that optimal ones will be obtained. Also, the pre-defined NN structures are fixed and cannot be adapted to newly-captured traffic flow data.

In this paper, an effective framework, namely adaptive particle swarm optimization, APSO, is proposed based on simple multi-layer neural networks, in order to address the issues related to the development of short-term traffic flow predictors namely: a) strongly non-linear characteristics which are unavoidable in traffic flow data; b) limited memory space for implementing short-term traffic flow predictors; c) specification of model structures for short-term traffic flow predictors which is required without involving trial and error method based on human expertise; d) short-term traffic flow predictors need to adapt to newly captured traffic flow data. To address issues (a) to (d), the APSO is proposed with the following features:

Simple multi-layer neural networks: The APSO is configured with the simple multi-layer neural networks, which demonstrably are able to forecast nonlinear traffic flow [2, 5, 8, 9, 15]. It tackles issue (a), that nonlinear characteristics exist in traffic flow data. Also, the number of parameters required by simple multi-layer neural networks is smaller than those required by the hybrid neural networks. Therefore, they are more suitable for adaptation to newly-captured traffic flow data, as less memory space and computational power is required that those required by hybrid neural networks for implementation, which tackles issue (b), that memory space for implementing short-term traffic flow predictors is limited.

Particle representation: In the APSO, each particle consists of two types of elements [31], control elements and parameter elements, in order to represent the NNs. Control elements represent the NN structures including the NN configurations and the number of hidden nodes. Parameter elements represent the NN parameters. Optimal structures and parameters can be adapted with newly-captured traffic flow data,

and also determination of optimal structures of NNs can be handled automatically by the APSO without involving trial and error methods. It addresses issue (b) that overcomes the limitations of the existing NN approaches [2, 5, 8, 9, 15] for traffic flow forecasting which is that the structure of the NN, such as the number of hidden nodes, is required to be pre-defined and cannot be adapted with respect to time.

Movement of particle swarm and multi-swarm: In APSO, the mechanism of particle swarm optimization, which is inspired by the social behaviours of animals like fish schooling and bird flocking [13], is employed to optimize short-term traffic flow predictors, as it can effectively solve the many difficulty optimization problems such as developing NNs [34, 35]. Also, Chan et al. [6] show that it can effectively adapt optimal structures and parameters of time-varying systems, where data is newly captured. Further to enhance the adaptive capability, the mechanism of sub-swarms [36] is employed. It groups particles with similar characteristics into a sub-swarm, which effectively create spatially organised internal representations of the features and abstractions detected in the particles. It is intended to locate the global optimum of the traffic flow forecasting predictor more effectively. Both these mechanisms are designed to address issue (d), that short-term traffic flow predictors need to adapt to newly-captured traffic flow data.

To evaluate the performance of the APSO, it is applied to develop short-term traffic flow predictors to forecast traffic flow conditions on a section of the freeway in Western Australia. Results show that the short-term traffic flow predictor developed by APSO is more accurate than those developed by the other tested algorithms including the Kalman filter [21], genetic algorithm [17] and particle swarm optimization approaches [4, 10, 18, 22]. The rest of the paper is organized as follows. Section 2 describes the mechanism of the short-term traffic flow predictor. Section 3 discusses the architecture of the APSO which is designed to improve the accuracy of the short-term traffic flow predictor. Section 4 reports the results obtained in a simulation case study. Finally, the conclusion is given in Section 5.

2. Short-term traffic flow predictor

Figure 1 shows the common configuration of a section of a freeway, consisting of an on-ramp and an off-ramp at both starting point A and end point B. While drivers at location A intend to go to destination C, forecasting information of future traffic flow condition at B is essential in order for the traffic control center to determine the best route for them. If the traffic flow conditions are forecast to be smooth at B, the traffic control center will allow them to go directly via the freeway to B, leave the freeway by the off-ramp at B, and then reach C. Alternatively, if the traffic flow conditions are forecast to be congested at B, the traffic control center will allow them to leave the freeway by the off-ramp in A, and then use the minor road to reach C.

To forecast future traffic flow conditions at B, a short-term traffic flow predictor was developed based on traffic flow data collected by five detector stations as illustrated in Figure 1, where D_{off}^{A} and D_{on}^{A} are located at the off-ramp and on-ramp of A respectively, as well as D_{off}^{B}, D_{on}^{B} and D_{thrf}^{B} are located at the off-ramp, on-ramp and through-road of B respectively, where traffic flow conditions, $y_{off}^{A}(t)$, $y_{on}^{A}(t)$, $y_{off}^{B}(t)$, $y_{off}^{B}(t)$ and $y_{thro}^{B}(t)$, at time *t* are captured by the five detection stations D_{off}^{A}, D_{on}^{A}, D_{off}^{B}, D_{on}^{B} and D_{thro}^{B} respectively with a sampling period T_s. Figure 1 shows only the short-term traffic flow predictor which aims to forecast traffic flow conditions at location B. In fact, this location can be set at any segment of the freeway, which is important for forecasting the short-term traffic flow conditions. For different segments of the freeway which are important, different short-term traffic flow predictors can be developed. Then, based on the

traffic flow conditions forecasts for different segments of the freeway, drivers are more likely to make the correct decision. For example, if two short-term traffic flow predictors are developed to forecast future traffic flow conditions at both locations A and B, then the future traffic flow conditions at both locations A and B are available. Drivers finding themselves before location A can determine more correctly whether they should use the freeway or use the minor road to reach location C.

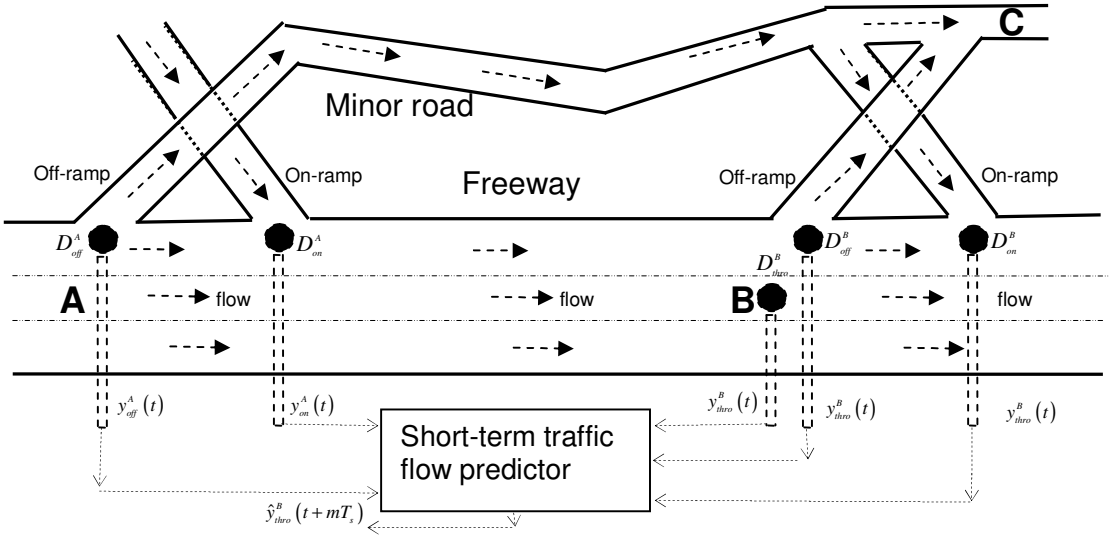

Figure 1 Schematic of short-term traffic flow predictor of a section of the freeway

The traffic flow conditions can be reflected by average speed of vehicles, flow of vehicles and density of vehicles. In this research, the average speed and density of vehicles are used as indicators of the traffic flow conditions. 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. These two measures are captured by the detector station, which consists of two inductive loop sensors separated with a small distance. To measure speeds of vehicles, the time difference of the vehicle passing between the two sensors is first captured. Then, the speed of the vehicle is calculated based on the time difference between the two captions and the distance between the two sensors. To measure density of vehicles, the detector station first captures the time taken between two consecutive vehicles passing through it. Then, the distance between two vehicles is calculated, based on the speed of the first vehicle passing through the detector station and the time difference between the two vehicles passing through the detector station. After the average distance between vehicles is calculated, the density of vehicles can be estimated by inversing the average distance between vehicles.

In this paper, the short-term traffic flow predictor is developed based on a time-varying neural network model namely *TVNN*, which is configured as a three-layer neural network consisting of only three layers: input layer, hidden layer and output layer. A three-layer neural network is used because it is able to forecast nonlinear traffic flow [2, 5, 8, 9, 15]. Also, it requires fewer parameters than those of hybrid neural networks, which incorporate neural networks approaches with other computational methods [11, 24, 29, 30, 32, 37]. Hence, it is more suitable for adapting to newly-captured traffic flow data. *TVNN* forecasts future traffic flow conditions at location B, $\hat{y}^{B}_{thro}\left(t+mT_s\right)$, with *m* sampling time ahead. The inputs of *TVNN* are the current traffic flow conditions, $y^{A}_{off}\left(t\right)$, $y^{A}_{on}\left(t\right)$, $y^{B}_{off}\left(t\right)$, $y^{B}_{off}\left(t\right)$ and $y^{B}_{thro}\left(t\right)$, as well as the past traffic flow conditions, $y^{A}_{off}\left(t-j\cdot T_s\right)$, $y^{A}_{on}\left(t-j\cdot T_s\right)$, $y^{B}_{off}\left(t-j\cdot T_s\right)$, $y^{B}_{on}\left(t-j\cdot T_s\right)$ and $y^{B}_{thro}\left(t-j\cdot T_s\right)$, collected at time ($t$-$j$·$T_s$) with j=1, 2, …, p, where the past traffic flow conditions within p sampling time was collected. *TVNN* can be formulated as:

$$\hat{y}^{B}_{thro}(t+mT_s)=TVNN\left(\bar{Q}(t),\bar{W}(t)\right)=\sum_{j=1}^{M}\left[\delta_j(t)\beta_j(t)\cdot\Psi\left(\gamma_{j,0}(t)+\sum_{k=1}^{5}\sum_{i=0}^{p}\gamma_{j,i,k}(t)\,y_k(t-iT_s)\right)\right]+\alpha_0(t) \quad (1)$$

where $y_1(t-i\cdot T_s)=y^{A}_{off}(t-i\cdot T_s)$, $y_2(t-i\cdot T_s)=y^{A}_{on}(t-i\cdot T_s)$, $y_3(t-i\cdot T_s)=y^{B}_{off}(t-i\cdot T_s)$, $y_4(t-i\cdot T_s)=y^{B}_{thro}(t-i\cdot T_s)$ and $y_5(t-i\cdot T_s)=y^{B}_{on}(t-i\cdot T_s)$, are the inputs of the *TVNN*; M is the number of hidden nodes; $\bar{Q}(t)=[y_1(t-i\cdot T_s), y_2(t-i\cdot T_s), y_3(t-i\cdot T_s), y_4(t-i\cdot T_s), y_5(t-i\cdot T_s)$ with $i=0,1,\ldots,p]$; $\alpha_0(t)$ 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 links from the input node 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 [2]; $\delta_j(t)$ is the activator with respect to the *j*-th hidden node, with $\delta_j(t)=0$ or $\delta_j(t)=1$. If $\delta_j(t)=1$, the *j*-th hidden node is activated. Then, the corresponding links between the input nodes to the *j*-th hidden node, as well as the corresponding links between the *j*-th hidden node to the output node are also activated. Otherwise, the *j*-th hidden node is inactivated, if $\delta_j(t)=0$. Then, the corresponding links connected with the *j*-th hidden node are also inactivated. This overcomes the limitations of the commonly used fixed-connected neural networks, where all the input nodes, hidden nodes and output nodes are restricted by a fixed configuration. Hence, the structures of the fixed-connected neural networks are fixed and cannot be adapted with respect to time. Also, some links between nodes are useless. If the fixed connected network is used, then some extra but unnecessary time is required to tune or optimize the weights between these useless links. It increases the time needed to adapt to the newly-captured traffic flow data. By introducing $\delta_j(t)$, the *TVNN* can adapt to newly-captured traffic flow data, by varying both the weights on the connections and the neural network structure. In equation (2), $\bar{W}(t)$ is denoted as the *TVNN* parameters, which are written as:

$$\bar{W}(t)=[\delta_j(t),\alpha_0(t),\beta_j(t),\gamma_{j,0}(t),\gamma_{j,i,k}(t) \text{ with } i=0,1,\ldots,p,\ j=1,2,\ldots,M \text{ and } k=1,2,\ldots,5] \quad (2)$$

$\bar{W}(t)$ is determined by two stages, off-line stage and on-line stage, as shown in Figure 2. In the off-line stage, $\bar{W}(t)$ is initialized based on a set of N_{hist} pieces of historical traffic flow data, $\Pi^{off-line}$, which were collected before the on-line time $T_{on\text{-}line}$ at time $t(i)$ with $i=1,2,\ldots,N_{hist}$, where $\Pi^{off-line}=\left\{\left[y^{B}_{thro}(t(i)),\bar{Y}(t(i)-mT_s)\right] \text{ with } i=1,2,\ldots,N_{hist}\right\}$, and $\bar{Y}(t(i)-m\cdot T_s)=[y^{A}_{off}(t(i)-(j+m)\cdot T_s), y^{A}_{on}(t(i)-(j+m)\cdot T_s), y^{B}_{off}(t(i)-(j+m)\cdot T_s), y^{B}_{on}(t(i)-(j+m)\cdot T_s), y^{B}_{thro}(t(i)-(j+m)\cdot T_s)$ with $j=0,1,\ldots,p]$.

The initial *TVNN* parameters $\bar{W}(T_{on-line})$ can be initialized by minimizing either of the following two cost functions (3) or (4):

$$J^{MAE}_{hist}:\min e^{MAE}_{hist}\left(\bar{W}(T_{on-line}),\Pi^{off-line}\right)=\frac{1}{N_{hist}}\sum_{i=1}^{N_{hist}}\left|\frac{y^{B}_{thro}(t(i))-\hat{y}^{B}_{thro}(t(i))}{y^{B}_{thro}(t(i))}\right|, \quad (3)$$

and

$$J^{RMS}_{hist}:\min e^{RMS}_{hist}\left(\bar{W}(T_{on-line}),\Pi^{off-line}\right)=\frac{1}{N_{hist}}\sqrt{\sum_{i=1}^{N_{hist}}\left(\frac{y^{B}_{thro}(t(i))-\hat{y}^{B}_{thro}(t(i))}{y^{B}_{thro}(t(i))}\right)^2}. \quad (4)$$

where $y_{thro}^{B}\left(t(i)\right)$ is the traffic flow condition collected by the detector station D_{thro}^{B} at the past time $t(i)$, and $\hat{y}_{thro}^{B}\left(t(i)\right)=TVNN\left(\bar{Y}\left(t(i)-mT_s\right),\bar{W}\left(T_{on-line}\right)\right)$. J_{hist}^{RMS} in equation (3) and J_{hist}^{MAE} in equation (4) represent the root mean square error and the mean absolute error in terms of the differences between actual conditions and forecasts respectively. J_{hist}^{MAE} is an average of the absolute errors, where both small and large errors have the same weights. J_{hist}^{RMS} tends to exaggerate significant errors by weighting the large errors more heavily than smaller errors by squaring them.

In the on-line stage, $\bar{W}(t)$ is fine-tuned by adapting a set of *m* pieces of new traffic flow data, $\Pi^{on-line}(t)$, which are captured after the on-line time $T_{on\text{-}line}$, where $\Pi^{on-line}(t)=\left\{\left[y_{thro}^{B}\left(t-i\cdot T_s\right),\bar{Y}\left(t-(i+m)T_s\right)\right] \text{ with } i=0,1,...,m\right\}$, and $\bar{Y}\left(t-(i+m)\cdot T_s\right)=\left[y_{off}^{A}\left(t-(j+m+i)\cdot T_s\right),\right.$ $y_{on}^{A}\left(t-(j+m+i)\cdot T_s\right), y_{off}^{B}\left(t-(j+m+i)\cdot T_s\right), y_{on}^{B}\left(t-(j+m+i)\cdot T_s\right), y_{thro}^{B}\left(t-(j+m+i)\cdot T_s\right) \text{with } j=0,1,...,p\left.\right]$. $\bar{W}(t)$ is determined by minimizing the following cost functions (5) and (6) in terms of mean absolute error and root mean square error respectively:

$$J_{adapt}^{MAE}:\min e_{adapt}^{MAE}\left(\bar{W}(t),\Pi^{on-line}\right)=\frac{1}{m}\sum_{i=1}^{m}\left|\frac{y_{thro}^{B}\left(t-i\cdot T_s\right)-\hat{y}_{thro}^{B}\left(t-i\cdot T_s\right)}{y_{thro}^{B}\left(t-i\cdot T_s\right)}\right|, \tag{5}$$

and

$$J_{adapt}^{RMS}:\min e_{adapt}^{RMS}\left(\bar{W}(t),\Pi^{on-line}\right)=\frac{1}{m}\sqrt{\sum_{i=1}^{m}\left(\frac{y_{thro}^{B}\left(t-i\cdot T_s\right)-\hat{y}_{thro}^{B}\left(t-i\cdot T_s\right)}{y_{thro}^{B}\left(t-i\cdot T_s\right)}\right)^2} \tag{6}$$

where $y_{thro}^{B}\left(t-i\cdot T_s\right)$ is the traffic flow condition captured by the detection station D_{thro}^{B} at time $\left(t-i\cdot T_s\right)$, and $\hat{y}_{thro}^{B}\left(t-i\cdot T_s\right)=TVNN\left(\bar{Y}\left(t-(i+m)\cdot T_s\right),\bar{W}(t)\right)$.

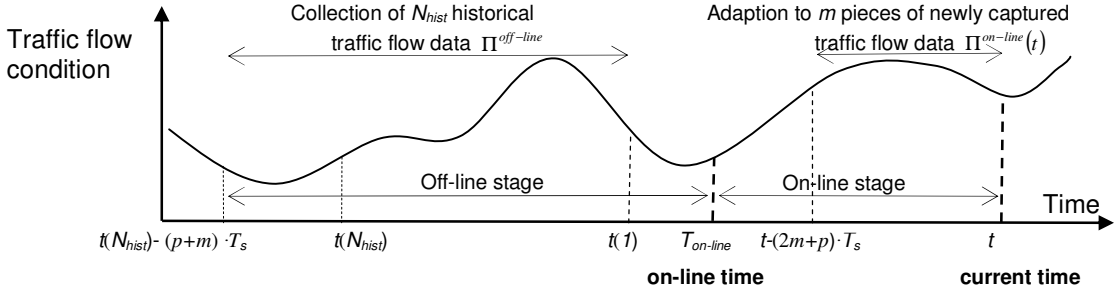

Figure 2 Collection of historical traffic flow data and adaption to newly captured traffic flow data

To solve the optimization problems (3) to (6), classical tools for nonlinear programming such as nonlinear branch-and-bound, sequential linearization, and Lagrangian relaxation methods can be used. However, their shortcoming is that they cannot cope with significantly non-smooth functions and multiple local minima of which the optimization problems (3) to (6) may involve. Therefore, we propose to use particle swarm optimization, to solve these optimization problems, as this method can obtain satisfactory solutions to optimization problems that are discontinuous, vastly multimodal, or involve noisy search spaces [23]. An algorithm based on particle swarm optimization, namely adaptive particle swarm optimization APSO, is developed in the following section to determine *TVNN* parameters, $\bar{W}(t)$.

3. Adaptive Particle Swarm Optimization

The operations of APSO can be divided into two stages. In the first stage, APSO determines the initial *TVNN* parameters, $\bar{W}(T_{on-line})$, based on the historical traffic data, $\Pi^{off-line}$, collected before time $T_{on-line}$. In the second stage, APSO updates the *TVNN* parameters, $\bar{W}(t)$, based on

the new traffic data, $\Pi^{on-line}(t)$, captured after time $T_{on-line}$. By doing this, the *TVNN* can adapt to newly-captured traffic flow data by updating $\bar{W}(t)$, in order to obtain better traffic flow accuracy. It overcomes the neural network approaches in the literature which have been developed based only on the historical traffic flow data [2, 5, 8, 9, 15], but are not capable of adapting to newly-captured traffic flow data. The operations of APSO are shown in Algorithm 1.

In the first stage, APSO starts by generating an initial swam based on the particle representation described in Section 3.1. Then, a set of sub-swarms is created by Algorithm 2, namely the Sub-swarm Creation Algorithm described in Section 3.2. After the sub-swarms have been created, Algorithm 3, namely the Particle Updating Algorithm described in Section 3.3, is used to update each particle in the sub-swarms, in order to locate the optima of the cost function. In Algorithm 3, either e_{hist}^{MAE} in equation (3), or e_{hist}^{MAE} in equation (4) can be used as the cost function, which aims to fit the historical traffic flow data, $\Pi^{off-line}$, collected before the on-line time $T_{on-line}$. When the termination condition is reached, the best of the best of all particles of all sub-swarms is returned as the initial *TVNN* parameter, $\bar{W}(T_{on-line})$.

Algorithm 1: *APSO*
Begin
if $t < T_{on-line}$ */ first stage of APSO*
Step 1: Create an initial swarm, $\bar{S}(T_{on-line})$, with N_s particles which is described in Section 3.1
Step 2: Divide the initial swarm, $\bar{S}(T_{on-line})$, into sub-swarms, $\bar{s}_i(T_{on-line})$, with *i*=1,2,…, $N_{\bar{S}(T_{on-line})}$ using **Algorithm 2**, $multiswarm(\bar{S}(T_{on-line}))$, described in Section 3.2
Step 3: Update the particles of each sub-swarm based on **Algorithm 3**, $update(\bar{s}_i(T_{on-line}))$, with *i*=1,2,…, *k*, described in Section 3.3
**/ e_{hist}^{MAE} in (3) or e_{hist}^{MAE} in (4) is used as the cost function in Algorithm 3*
Step 4: If the termination condition is reached, **goto** Step 5. Otherwise, goto Step 2.
Step 5: **Return** $\bar{W}(T_{on-line})$ */* $\bar{W}(T_{on-line})$ is the best TVNN parameters obtained by the best particle*
Endif */ end of the first stage of APSO*
If $t \geq T_{on-line}$ */ second stage of APSO*
Step 6: $t = T_{on-line} + T_s$;
Step 7: Update the particles of each sub-swarm by **Algorithm 3**, $update(\bar{s}_i(t))$
**/ either e_{adapt}^{MAE} in (5) or e_{adapt}^{RMS} in (6) is used as the cost function of Algorithm 3*
Step 8: **Return** $\bar{W}(t)$ */* $\bar{W}(t)$ is the best TVNN parameters obtained by the best particle*
Step 9: Determine the change of the characteristics of the traffic flow data $\Pi^{on-line}(t)$ captured at time *t* and those $\Pi^{on-line}(t-T_s)$ captured at time $(t-T_s)$ based on **Algorithm 4**, $change(\Pi^{on-line}(t), \Pi^{on-line}(t-T_s))$ described in Section 3.4.
If $(change(\Pi^{on-line}(t), \Pi^{on-line}(t-T_s)) = True)$ **then**
Step 10.1: Re-generate N_s particles randomly to form a new swarm $\bar{S}(t)$ with N_s particles
Step 10.2: Re-group the new swarm $\bar{S}(t)$ into sub-swarms $\bar{s}_i(t)$ with *i*=1,2,…, $N_{\bar{S}(t)}$ based on **Algorithm 2,** $multiswarm(\bar{S}(t))$.
Endif
Step 11: If *TVNN* is still on-line, **goto** Step 6. Otherwise **Break** */ end of the second stage of APSO*
Endif
Ends

In the second stage, each particle of each sub-swarm is updated based on Algorithm 3, in order to exploit the time-varying peak covered by the sub-swarm, which adapts the new characteristic of the newly-captured traffic flow data, $\Pi^{on-line}(t)$. Either e_{adapt}^{MAE} in equation (5), or e_{adapt}^{RMS} in equation (6) can be used as the cost function in Algorithm 3. The best particle among all particles in all sub-swarms is returned as the *TVNN* parameter, $\bar{W}(t)$, at time *t*. If a significant change is detected by Algorithm 4, namely the Change Detection Algorithm described in Section 3.4, the following actions are taken by the APSO to adapt to the newly captured traffic flow data, $\Pi^{on-line}(t)$, whose characteristic

of $\Pi^{on-line}(t)$ is significantly different from those of the previously captured traffic flow data. First, all particles are regenerated randomly, and then a set of sub-swarms is created by Algorithm 2 which will relocate all the *TVNN*s represented by the particles to adapt significantly to the newly-captured data. If no significant change is detected, the *TVNN*s represented by the particles are slightly fine-tuned to adapt to the newly-captured data by Algorithm 3. The process continues until the on-line condition is terminated.

3.1 Particle Representation

The swarm with N_s particles in hierarchical form [31] is used in the APSO for representing both the structure and the weights of the *TVNN*. The i_1-th particle, $\bar{p}_{i_1}(t)$, is used to represent the *TVNN* parameters, $\bar{W}(t)$, defined in equation (2), where $\bar{p}_{i_1}(t)$ consists of two types of elements, the control elements, $\bar{\sigma}_{i_1}(t)$, and parameter elements, $\bar{\rho}_{i_1}(t)$. $\bar{p}_{i_1}(t)$ is denoted by,

$$\bar{p}_{i_1}(t)=\left[p_{i_1,1}(t), p_{i_1,2}(t),..., p_{i_1,n_p}(t)\right]=\left[\bar{\sigma}_{i_1}(t),\bar{\rho}_{i_1}(t)\right],$$ where $p_{i_1,i_2}(t)\in[-1..1]$ with i_1=1,2,…, N_s, i_2=1,2,…, n_p, and n_p is the total number of elements. After transforming $\bar{\sigma}_{i_1}(t)$ to binary bits, $\bar{\sigma}_{i_1}(t)$ can be used by the particle swarm optimization for solving optimization problems with discrete variables [14]. $\bar{\sigma}_{i_1}(t)$ consists of the first M elements of $\bar{p}_{i_1}(t)$, and each element of $\bar{\sigma}_{i_1}(t)$ is used to represent an activator with respect to a hidden node in the *TVNN*, where $\bar{\sigma}_{i_1}(t)=\left[p_{i_1,1}(t), p_{i_1,2}(t),..., p_{i_1,M}(t)\right]$. The activator of the j-th hidden node, $\delta_j(t)$ defined in equation (2), is given by:

$$\begin{cases}\delta_j(t)=0, \text{ if } p_{i_1,j}(t)<0\\ \delta_j(t)=1, \text{ if } p_{i_1,j}(t)>0\end{cases} \text{ with } j=1,2,\ldots M.$$

The parameter elements, $\bar{\rho}_{i_1}(t)$, consist of the $(1+M)$-th to the (n_p)-th elements of $\bar{p}_{i_1}(t)$. $\bar{\rho}_{i_1}(t)$ is represented as the weights of links and biases of the *TVNN* defined in equation (2): $\alpha_0(t)$, $\beta_j(t)$, $\gamma_{j,0}(t)$ and $\gamma_{j,i,k}(t)$, with i=0,1, …, p, j=1, 2, …, M and k=1, 2, …, 5. $\bar{\rho}_{i_1}(t)$ are denoted as follow:

$$\bar{\rho}_{i_1}(t)=\left[p_{i_1,1+M}(t), p_{i_1,2+M}(t),..., p_{i_1,n_p}(t)\right]=\left[\alpha_0(t),\beta_j(t),\gamma_{j,0}(t),\gamma_{j,i,k}(t) \text{ with } i=0,1,...,p,\ j=1,2,...,M \text{ and } k=1,2,...,5\right],$$

where (n_p-M) is the number of weights, with $n_p=(1+M+5\cdot(1+p)\cdot M)$. With i=1,2,…,p and k=1,2,…,5, $\beta_j(t)$, $\gamma_{j,0}(t)$ and $\gamma_{j,i,k}(t)$ are the corresponding weights, which are connected with the j-th hidden nodes. Their values are forced to be zeros, if $\delta_j(t)=0$. Otherwise, their values are the same as the values of the corresponding elements of $\bar{\rho}_{i_1}(t)$, if $\delta_j(t)=1$. Based on this particle representation, both structure and the weights of the *TVNN* can be kept varying with respect to time, in order to adapt newly-captured traffic flow data. This particle representation overcomes the limitation of fixed-connected neural networks for traffic flow forecasting [2, 5, 8, 9, 15], because the neural network structures are fixed and cannot be adapted to newly-captured traffic flow data.

3.2 **Multi-swarm Creation Algorithm**

As traffic flow patterns vary with respect to time, the optimal structure and weights of the *TVNN* also vary with respect to time. To assist the APSO to locate this optimum, the multi-swarm creation algorithm, $multiswarm(\bar{S}(t))$, based on [36], is proposed to be used in the APSO, where the multi-swarm creation algorithm has been demonstrated to assist particle swarm optimization approaches for solving time-varying problems effectively. The multi-swarm creation algorithm creates sub-swarms, with the particles in each sub-swarm having similar

characteristics. It is intended to position sub-swarms on different peaks of the time-varying landscape with respect to the structure and weights of the *TVNN*. The aim is to locate effectively the global time-varying optimum. It first creates N_s sub-swarms in which each particle is treated as a sub-swarm $\overline{s}_i(t)$ with i=1,2, …, N_s. Then, the two sub-swarms, $\overline{s}_i(t)$ and $\overline{s}_j(t)$, with the smallest distance between them are identified by the following equation:

$$d(i,j) = \min_{\overline{p}_i(t)\in\overline{s}_i(t),\overline{p}_j(t)\in\overline{s}_j(t)} \sqrt{\sum_{k=1}^{M} c\left(p_{i,k}(t) - p_{j,k}(t)\right)^2 + \sum_{k=M+1}^{n_p} \left(p_{i,k}(t) - p_{j,k}(t)\right)^2}, \tag{7}$$

where $i, j = 1, 2, \ldots, N_s$, but $i \neq j$. Yang and Li's [36] algorithm assumes that all elements in the particle have the same priority, which may not be appropriate to use here, as two elements, control elements and parameter elements, are used in APSO. Therefore, a priority factor, c>1, is introduced to let the control elements have a higher priority than that of the parameter elements, where $p_{i,k}(t)$ with k=1, 2, … , h_{max}, are the control elements and $p_{j,k}(t)$ with k=1, 2, … , h_{max}, are the parameter elements. By doing this, the control elements, which can determine the structure of the *TVNN*, can dominate the distance measure. If the total number of particles in $\overline{s}_i(t)$ and $\overline{s}_j(t)$ is not greater than a pre-defined maximum sub-swarm size, a new merging sub-swarm $\overline{s}_i(t)$ is created by combining the particles in both $\overline{s}_i(t)$ and $\overline{s}_j(t)$. This iteration continues until all sub-swarms contain more than one particle.

Algorithm 2: $multiswarm(\overline{S}(t))$

Begin

Step 1: Create N_s sub-swarm $\overline{s}_i(t)$ by $\overline{s}_i(t) = \overline{p}_i(t)$ with i = 1, 2, …, N_s

Step 2: Identify a pair of sub-swarms $\overline{s}_i(t)$ and $\overline{s}_j(t)$ with $i, j = 1, 2, \ldots, N_s$, but $i \neq j$, which is separated by a shortest distance based on equation (7).

Step 3: **If** the total number of particles in $\overline{s}_i(t)$ and $\overline{s}_j(t)$ is not greater than a pre-defined maximum number of particles in a sub-swarm, **then**

Step 3.1: Merging $\overline{s}_i(t)$ and $\overline{s}_j(t)$ together into a single sub-swarm $\overline{s}_i(t) := \overline{s}_i(t) + \overline{s}_j(t)$

Step 3.2: $N_s = N_s - 1$

Step 3.2: Recalculate the distance between all the sub-swarms based on equation (7).

Step 4: **If** all $\overline{s}_i(t)$ with $i = 1, 2, \ldots, N_s$, contain more than one particle

then return ($\overline{s}_i(t)$ with $i = 1, 2, \ldots, N_s$) and **terminate**

otherwise goto Step 2

End

3.3 Particle Updating Algorithm

When a sub-swarm $\overline{s}_i(t)$ is created using the above multi-swarm creation algorithm, each particle $\overline{p}_{i_1}(t) = \left[p_{i_1,1}(t), p_{i_1,2}(t), \ldots, p_{i_1,n_p}(t)\right]$ with $i_1 = 1,2,\ldots,N_{\overline{s}_i(t)}$ in $\overline{s}_i(t)$ is updated based on Algorithm 3, $update(\overline{s}_i(t))$, where $N_{\overline{s}_i(t)}$ is the number of particles in the sub-swarm $\overline{s}_i(t)$. In order for a sub-swarm to locate a local peak for the *TVNN* quickly, each particle in a sub-swarm learns from both the global best positions found by itself and the sub-swarms. Each element of $\overline{p}_{i_1}(t)$, and the velocity vector of the particles, $\overline{v}_{i_1}(t) = \left[v_{i_1,1}(t), v_{i_1,2}(t), \ldots, v_{i_1,n_p}(t)\right]$, are updated using the inertia weight approach [26]:

$$v_{i_1,i_2}(t)' = \omega \cdot v_{i_1,i_2}(t) + \eta_1 \cdot rand_1 \cdot \left(p_{pbest_{i_1},i_2}(t) - p_{i_1,i_2}(t)\right) + \eta_2 \cdot rand_2 \cdot \left(p_{gbest,i_2}(t) - p_{i_1,i_2}(t)\right) \tag{8}$$

$$p_{i_1,i_2}(t)' = p_{i_1,i_2}(t) + v_{i_1,i_2}(t)' \tag{9}$$

where $p_{i_1,i_2}(t)$' and $v_{i_1,i_2}(t)$' represent the updated element and the updated velocity of the i_2-th element of the the i_1-th particle respectively; $\overline{p}_{pbest_{i_1}}(t)=\left[p_{pbest_{i_1},1}, p_{pbest_{i_1},2},..., p_{pbest_{i_1},n_p}\right]$ is the best elements found so far by the particle, $\overline{p}_{i_1}(t)$; $\overline{p}_{gbest}(t)=\left[p_{gbest,1}, p_{gbest,2},..., p_{gbest,n_p}\right]$ is global best particle found so far in, $\overline{s}_i(t)$; η_1 and η_2 are the acceleration constants; $rand_1$ and $rand_2$ are random numbers generated in the interval [0.0, 1.0]; ω is an inertia weight. After updating each particle, $\overline{p}_{i_1}(t)$, the fitness of each particle is evaluated based on the cost function. When the APSO is in the off-line stage, the particles intend to initialize $W(T_{off})$ based on historical traffic flow data. Then, either e_{hist}^{MAE} in equation (3) or e_{hist}^{MAE} in equation (4) is used as the cost function. When the APSO is in the on-line stage, the particles intend to generate new $W(t)$ in order to adapt to newly captured data. Then, either e_{adapt}^{MAE} in equation (5) or e_{adapt}^{RMS} in equation (6) is used as the cost function. After evaluating the fitness of each particle based on the specified cost function, the best particle, $\overline{p}_{pbset_1}(t)$, and the global best particle, $\overline{p}_{gbest_i}(t)$, can be updated.

Algorithm 3: $update\left(\overline{s}_i(t)\right)$

Begin

Step 1: Update the velocity of each particle, $\overline{v}_{i_1}(t)$, and update each particle, $\overline{p}_{i_1}(t)$, based on (8) and (9) respectively.

Step 2: Evaluate the fitness of each particle $\overline{p}_{i_1}(t) \in \overline{s}_i(t)$ with $i_1 = 1,2,..., N_{\overline{s}_i(t)}$ based on the cost function:

If $(t < T_{on-line})$

Then either e_{hist}^{MAE} in (3) or e_{hist}^{MAE} in (4) is used as the cost function

**/ aims to initialize $\overline{W}(T_{on-line})$ based on historical traffic flow data*

**/ e_{hist}^{MAE} for mean absolute error; e_{hist}^{MAE} for root mean square error*

Elseif $(t \geq T_{on-line})$

Then either e_{adapt}^{MAE} in (5) or e_{adapt}^{RMS} in (6) is used as the cost function

**/ aims to generate new $\overline{W}(t)$ to adapt to newly captured traffic flow data*

**/ e_{adapt}^{MAE} for mean absolute error; e_{adapt}^{RMS} for root mean square error*

Endif

Step 3: Update the best particle $\overline{p}_{pbest_{i1}}(t)$ found so far by the particle $\overline{p}_{i1}(t)$:

If $\overline{p}_{i1}(t)$ is better than $\overline{p}_{pbest_{i1}}(t)$

Then $\overline{p}_{pbest_{i1}}(t) = \overline{p}_{i1}(t)$

Endif

Step 4: Update the global best particle $\overline{p}_{gbest_i}(t)$ found so far by $\overline{s}_i(t)$:

If $\overline{p}_{i1}(t)$ is better than $\overline{p}_{gbest_i}(t)$

Then $\overline{p}_{gbest_i}(t) = \overline{p}_{i1}(t)$

Endif

End

3.4 Change Detection Algorithm

Change Detection Algorithm $\left(change\left(\Pi^{on-line}(t), \Pi^{off-line}(t-T_s)\right) = True\right)$, uses the global best particle, $\overline{p}_{gbest}(t-T_s)$, over all sub-swarms as the monitoring particle to detect whether there is significant change of the characteristics of the newly captured traffic flow data, $\Pi^{on-line}(t)$ and $\Pi^{on-line}(t-T_s)$, at time, t, and at time, $(t-T_s)$, respectively. The fitness of the global best particle is re-evaluated based on the cost function either e_{adapt}^{RMS} in equation (5) or e_{adapt}^{MAE} in equation (6), before it is updated. If the fitness of the global best particle changes significantly, this indicates that a significant environmental change occurs. For example, if the *TVNN* parameters $W(t)$ represented by the best particle can achieve 80% accuracy for the new traffic flow data captured at time $(t-T_s)$, but it can only achieve 20% accuracy for those captured at time

t, then it is assumed that a significant change to the traffic flow conditions has been detected. If the accuracy obtained by the best particle changes only slightly, then it is assumed that no significant change has been detected.

Algorithm 4: $change\left(\Pi^{on-line}(t),\Pi^{off-line}(t-T_s)\right)$

Begin

Step 1: Identify the global best particle, $\bar{p}_{gbest}(t-T_s)$, from all sub-swarms

Step 2: **If** $\dfrac{\left|e_{adapt}\left(\bar{W}_{gbest}(t-T_s),\Pi^{on-line}(t)\right)-e_{adapt}\left(\bar{W}_{gbest}(t-T_s),\Pi^{on-line}(t-T_s)\right)\right|}{e_{adapt}\left(\bar{W}_{gbest}(t-T_s),\Pi^{on-line}(t-T_s)\right)} > (\textit{pre-defined significant difference})$

**/ where the best TVNN parameters,* $\bar{W}_{gbest}(t-T_s)$ *is represented by* $\bar{p}_{gbest}(t-T_s)$

**/ either* $e_{adapt}=e_{adapt}^{RMS}$ *or* $e_{adapt}=e_{adapt}^{MAE}$

Then

Return (True)

Else

Return (False)

Endif

End

4 Simulation Results

The performance of the APSO is evaluated by developing the TVNNs based on traffic flow data, which was collected by the two detector stations, D_{on}^{Reid} and D_{off}^{Reid}, at the intersection of Reid Highway and Mitchell Freeway, as well as the three other detector stations, D_{on}^{Hutton}, D_{off}^{Hutton} and D_{on}^{Hutton}, at the intersection of Hutton Road and Mitchell Freeway, Western Australia. D_{on}^{Reid} and D_{off}^{Reid} are installed at the on-ramp and off-ramp of Reid Highway respectively. D_{on}^{Hutton}, D_{off}^{Hutton} and D_{thr}^{Hutton}, are installed at the on-ramp, off-ramp and through-road of Hutton Road respectively. Traffic flow data were collected from Week 6, Week 7, Week 8, Week 9, Week 11 and Week 12 in 2009, for the five business days, Monday to Friday, over the 2-hour peak traffic period (7.30 – 9.30 am). Sixty seconds (or one minute) were used as the sampling time. When drivers want to travel from Reid Hwy to the Perth City via the Mitchell Freeway, they have to cross the Hutton Road intersection. Therefore, it is necessary to forecast traffic flow conditions at the intersection of Hutton Road. As the distance between the intersections of Reid Highway and Hutton Road is about 7 kilometers, and the speed limit of the Mitchell Freeway between these two locations is 100 kilometer per hour, it usually takes about 0.07 hour (or 4.2 minutes) to travel from Reid Highway to Hutton Road via the Mitchell Freeway, when the traffic flow conditions between these two locations is smooth. When a short-term traffic flow predictor is available to forecast traffic flow conditions five minutes ahead at the location of Hutton Road, people from Reid Highway can decide whether to use the Mitchell Freeway to go to Hutton Road or to reach Hutton Road in another way. Therefore, the short-term traffic flow predictor is developed here to forecast the future traffic flow conditions, $\hat{y}_{thro}^{Hutton}$, at the location of the detection station, D_{off}^{Hutton} five sampling times (or five minutes) ahead.

Also, we observed that people have different behaviors on different business days. On Monday, people drive more frequently through the section between Reid Highway and Hutton Road, during the peak hours. From Tuesday to Thursday, fewer people drive through this section during the peak hours than on Monday. On Friday, much fewer people drive through this section. These behaviors of people affect the traffic flow conditions of the freeway. Therefore, five networks, namely $TVNN_{Mon}$, $TVNN_{Tue}$, $TVNN_{Wed}$, $TVNN_{Thu}$ and $TVNN_{Fri}$, were developed by the APSO for forecasting traffic flow for each business day, Monday, Tuesday, Wednesday, Thursday and Friday, respectively. They were trained based on the historical traffic flow data collected from their own corresponding business day, and then they were retrained based on the newly-captured traffic flow data from their own corresponding business day also. Apart from these five networks, we further

evaluated whether the APSO can be used in practice to train a network for all the five business days. A single network, namely $TVNN_{All}$, was also developed for all business days of the week. It was trained based on the traffic flow data collected from all business days, and it was retrained based on the newly-captured traffic flow data from all business days.

To evaluate the performance of all these networks, 6-fold cross validation was used based on these six weeks traffic flow data. The traffic flow data for each week was treated as a subsample. We selected the data from five weeks and treated these data as historical traffic flow data (i.e. training data), which were used for training the networks by determining the initial TVNN parameters. The data for the remaining week were treated as newly-captured data (i.e. test data), which were used for testing the networks by evaluating their generalization capability. The cross validation process was repeated 6 times, with each sub-sample being used exactly once for testing the networks. The 6 validation results were then averaged to produce a single estimation. For the $TVNN_{Mon}$, the training data collected from Monday (from7.30am to 9.30am) were used to determine the initial parameters of the $TVNN_{Mon}$, and the test data collected from Monday (from7.30am to 9.30am) were used to evaluate the generalization capability of the $TVNN_{Mon}$. Hence, there were 480 pieces of training data were used to determine the initial parameters of $TVNN_{Mon}$, and 120 pieces of test data for evaluating its generalization capability. The same was done for the four networks for the other four business days: $TVNN_{Tue}$, $TVNN_{Wed}$, $TVNN_{Thu}$, and $TVNN_{Fri}$. For $TVNN_{all}$, the training data were collected for all the five business days (from 7.30am to 9.30am) for all five weeks, and these training data were used to determine the initial parameters of $TVNN_{all}$. The test data were collected for all the five business days (from7.30am to 9.30am) of the remaining one week, and these test data were used to evaluate the generalization capability of $TVNN_{all}$. Hence, 2400 pieces of data were used for determining the initial parameters of $TVNN_{all}$, and 600 pieces of data for evaluating its generalization capability.

Both mean absolute error and root mean square error (defined in equations (10) and (11) respectively) were used to evaluate the generalization capabilities of the networks.

$$MAE = \frac{1}{N_T} \frac{\sum_{i=1}^{N_T} \left| y_{thro}^{Hutton}\left(i \cdot T_s\right) - \hat{y}_{thro}^{Hutton}\left(i \cdot T_s\right) \right|}{y_{thro}^{Hutton}\left(i \cdot T_s\right)} \cdot 100\% , \tag{10}$$

and

$$RMSE = 100\% \cdot \frac{1}{N_T} \sqrt{\sum_{i=1}^{N_T} \left(\frac{y_{thro}^{Hutton}\left(i \cdot T_s\right) - \hat{y}_{thro}^{Hutton}\left(i \cdot T_s\right)}{y_{thro}^{Hutton}\left(i \cdot T_s\right)} \right)^2} \tag{11}$$

$y_{thro}^{Hutton}\left(i \cdot T_s\right)$ is the *i*-th test data captured by the detection station, D_{thr}^{Hutton}, at time $\left(i \cdot T_s\right)$, where the mean speed of vehicles and the density of vehicles were considered; $\hat{y}_{thro}^{Hutton}\left(i \cdot T_s\right)$ is the estimate of traffic flow conditions at the detection station, D_{thr}^{Hutton}, which is forecast by the TVNN, 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 $m=5$. N_T is the number of pieces of newly-captured traffic flow data.

The following parameters are used in the APSO: both the acceleration constants ϕ_1 and ϕ_2 were set at 2.05, and the maximum velocity $v_{\max}$ was 0.2, which can be found in reference [13]; the number of particles in the swarm was 70, and the maximum number of particles in each swarm is 3, as recommended by [36]; for $TVNN_{Mon}$, $TVNN_{Tue}$, $TVNN_{Wed}$, $TVNN_{Thu}$, and $TVNN_{Fri}$, the maximum number of hidden nodes, M, is recommended by [20], where $M = 9 \approx \log_2\left(480\right)$ and 480 pieces of historical traffic flow data were used for determining the

initial parameters for these networks. For $TVNN_{All}$, the maximum number of hidden nodes , $M = 11 \approx \log_2(2400)$, is used, as there were 2400 pieces of data for training. For the termination condition of the training stages, the networks keep training by the APSO, until the training errors are below 10%. The performance of APSO is compared with a number of PSO algorithms taken from the recent literature. The PSO algorithms used for comparison include Pure-PSO [10], Lung-PSO [18], Par-PSO [22] and Black-PSO [4]. Apart from PSO algorithms, a commonly used stochastic algorithm such as a genetic algorithm, namely GA was used for comparison, where the operations of the GA can be found in [17]. The swarm sizes used on these PSO algorithms and the population size used on the GA algorithms was the same as the swarm size used on APSO. Also, the termination conditions used on all these stochastic algorithms are the same as that used on the APSO. Apart from these population-based stochastic algorithms, the Kalman filter [21] has also been used to compare with the performance of the networks developed by the APSO. Six Kalman filters were developed. Five of the Kalman filters were developed for a particular business day (either Monday, Tuesday, Wednesday, Thursday or Friday). One of the six Kalman filters was developed for all business days of the week.

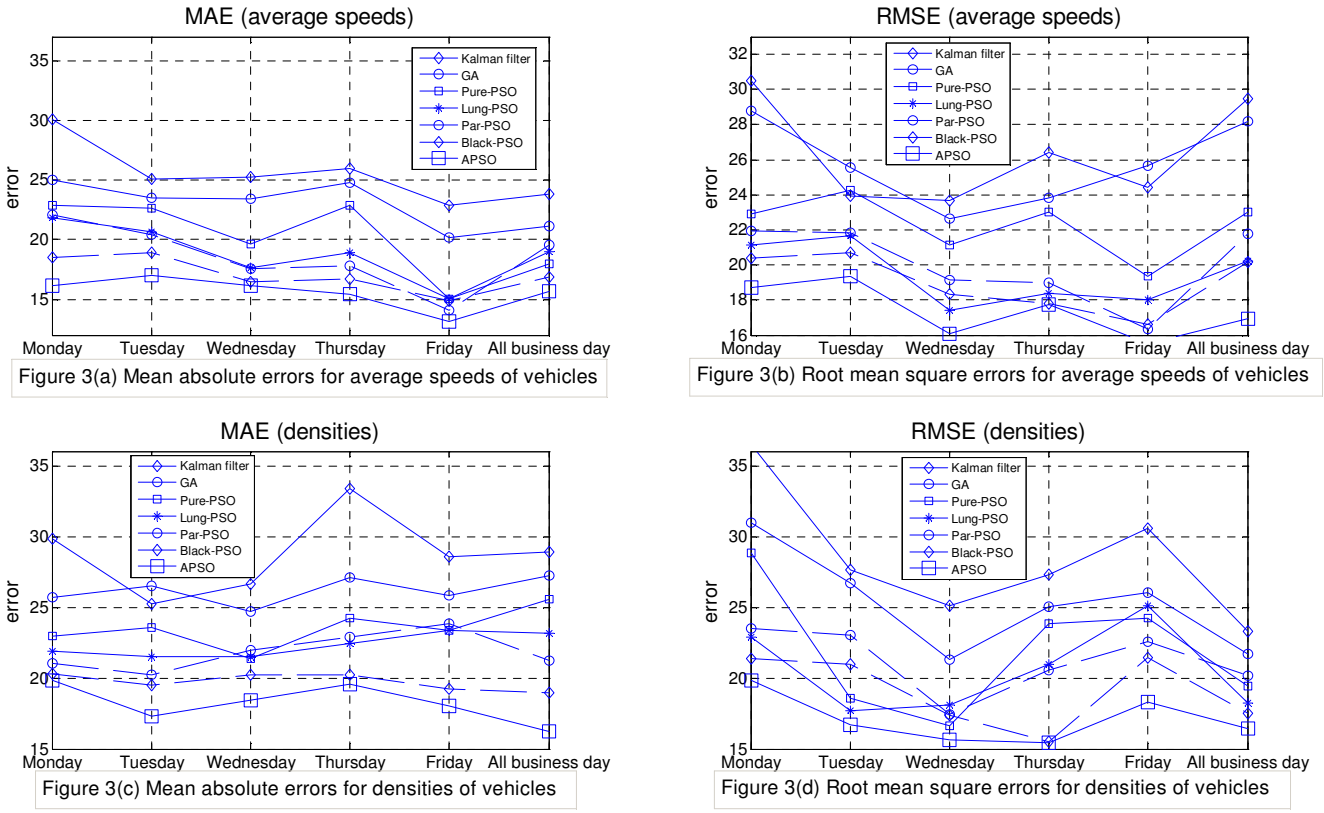

Figure 3(a) Mean absolute errors for average speeds of vehicles

Figure 3(b) Root mean square errors for average speeds of vehicles

Figure 3(c) Mean absolute errors for densities of vehicles

Figure 3(d) Root mean square errors for densities of vehicles

Figure 3 Cross validation for the algorithms

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. Regarding the average speeds of vehicles, the mean absolute errors and the root mean square errors obtained by all the methods are shown in Figure 3(a) and (b) respectively. Regarding the densities of vehicles, the errors obtained are also shown in Figure 3(c) and (d) respectively. The results obtained by the Kalman filters and the networks developed by all the population-based stochastic algorithms are shown in the figures. Results for each individual business day, Monday to Friday, as well as those for all business days are shown. The mean absolute errors and root mean square errors in terms of both average speeds and both densities obtained by the networks developed by the

five PSO algorithms are generally better than those of the GA, which are better than those obtained by the Kalman filters. Of the five PSO algorithms (APSO, Pure-PSO, Black-PSO, Par-PSO and Lung-PSO), the proposed APSO is generally the best among the five. Therefore, the proposed APSO can generally achieve the best generalization capabilities compared with the other methods.

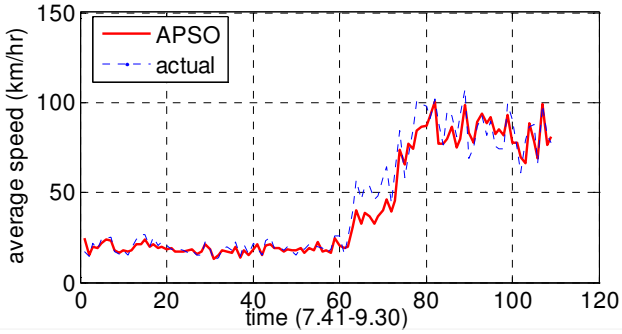

Figure 4(a) Average speed obtained by APSO - week 11 (Monday)

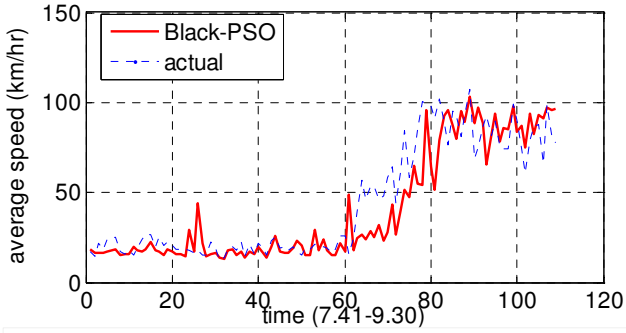

Figure 4(b) Average speed obtained by Black-PSO - week 11 (Monday)

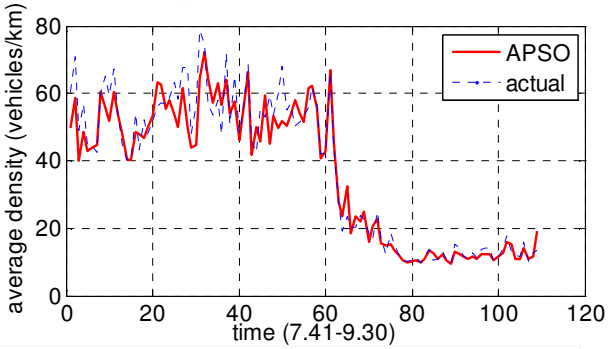

Figure 4(c) Density obtained by APSO - week 11 (Monday)

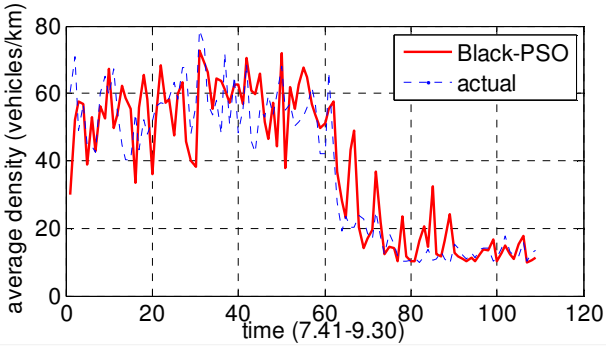

Figure 4(d) Density obtained by Black-PSO - week 11 (Monday)

Figure 4 Simulation results for Monday (week 11)

The simulation results for Monday (week 11) in terms of average speeds of vehicles and average densities of vehicles are depicted in Figure 4. Figure 4(a) shows that from the 1st sample to the 60th sample (about 7.41-8.41 am), the average speeds of vehicles were about 20km/hours, which is far below the speed limit. Hence, the traffic flow was congested and there was a traffic jam during the time. From the 80th sample to the 110th sample (about 9.00-9.30 am), the average speeds of vehicles are about 70 to 100 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. Therefore, the data being investigated contains traffic jam conditions. Figures 4(a) and 4(b) show the results of average speeds obtained by the proposed APSO and the second best algorithm, Black-PSO respectively. By comparing the results obtained by both APSO and Black-PSO, we can see that the better average speed estimates can be identified by the APSO than those by the Black-PSO. In general, the results forecast by APSO are more accurate than those forecast by Black-PSO. Also, Figure 4(c) and 4(d) shows that better average densities of vehicles can be estimated by the APSO comparing with those by the Black-PSO. Therefore, APSO can obtain more accurate results for traffic flow forecasting conditions in terms of average speeds and average density of vehicles.

Also, the computational time in terms of both the off-line stage and on-line stage used on Monday, week 11 were shown in Figure 5 and Figure 6 respectively. For the off-line stage, Figure 5 shows the computational time taken by each population-based stochastic method, GA, Pure-PSO, Lung-PSO, Par-PSO, Black-PSO, and APSO, to reach the termination condition where the training error with 10% was achieved, as well as the time taken on the deterministic method, Kalman filter, to converge. It shows that the Kalman filter needed only about 0.5

seconds to converge, while the population-based stochastic methods needed more than 220 seconds to reach the termination conditions. For the on-line stage, Figure 6 shows that the computational time taken for each iteration when using the population-based stochastic methods, is about 2 to 3 seconds, while the computational time taken to adapt the Kalman filter with respect to the newly-captured data is about 0.02 seconds. Therefore, the computational time taken by Kalman filter is much less than that taken when using the stochastic methods. Even if the computational time taken by Kalman filter is much less than those taken by the stochastic methods, the accuracies in term of traffic flow forecasting conditions achieved by Kalman filter are poorer than those obtained by the 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 each iteration for all population-based stochastic methods, including the proposed APSO is about 2 to 3 seconds, which is much less than the sampling time of 60 seconds. Therefore, APSO is implementable, in order to adapt traffic flow data which is captured with the sample time of 60 seconds. It also shows that the computational time taken to adapt the Kalman filter is much less than that used on APSO. Therefore, the Kalman filter can be implemented with a smaller sample time, but the accuracies achieved by APSO are better than those obtained by the Kalman filter. The sample time of APSO can be set smaller, if the computational time taken for the APSO for one iteration is smaller. To achieve this, the following two approaches can be used: a) the APSO can be implemented on a more powerful microprocessor, in order to reduce the computational time; b) the APSO is currently implemented by Matlab, which is a high level programming language. If the APSO is implemented by a lower level programming language, the computational time required by the APSO is smaller, and thus, shorter sampling times can be used.

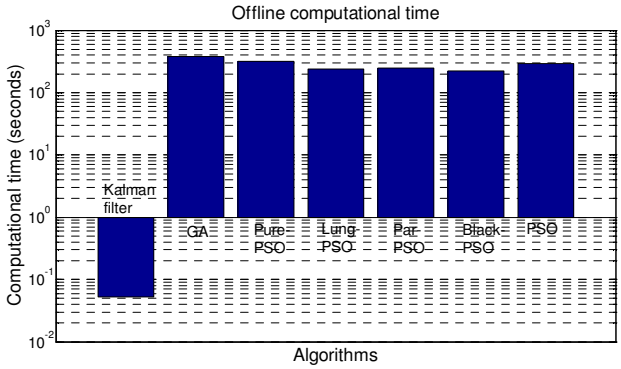

Figure 5 Offline computational time

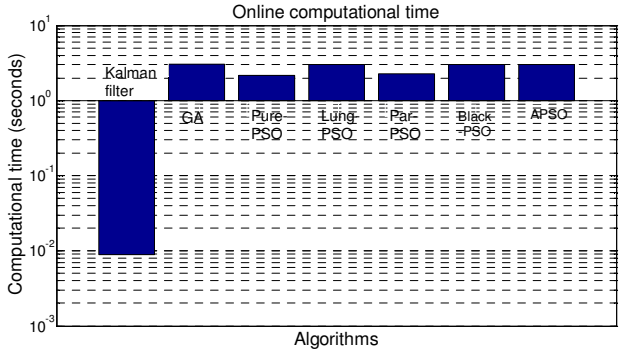

Figure 6 Online computational time

5 Conclusion

In this paper, a new algorithm, namely adaptive particle swarm optimization (APSO) based on the mechanisms of particle swarm optimization and artificial neural networks, has been proposed to develop short-term traffic flow predictors. It aims to address all the issues related to the development of short-term traffic flow predictors, which have not been fully addressed by existing approaches, including: a) strongly non-linear characteristics of short-term traffic flow data must be captured; b) large memory space for implementing short-term traffic flow predictors is usually not available; c) specification of model structures for short-term traffic flow predictors is required without involving trial and error method based on human expertise; d) it is essential to adapt short-term flow predictors to newly-captured traffic flow data.

To address (a), APSO uses simple multi-layer neural networks to develop short-term flow predictors, which can address the strongly non-linear characteristics of short-term traffic flow data. Also, these simple multi-layer neural networks need much less memory space than that required by the existing hybrid neural networks to develop short-term traffic flow predictors, and thus issue (b) can be addressed. To address issue (c), particles of the APSO represent both structures and parameters of short-term traffic flow predictors, such that both optimal structures and parameters can be determined automatically without involving trial and error methods based on human expertise. To address issue (d), APSO applies the mechanism of multi-swarms, in order to trace effectively the optima of short-term traffic flow predictors which vary with respect to newly-captured traffic flow data. The effectiveness of the APSO 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 is newly captured. The forecasting results obtained by the adaptive short-term traffic flow predictor developed by APSO are more accurate than those obtained by the other tested algorithms, Kalman filter, genetic algorithm and particle swarm optimization approaches taken from the recent literature. These results clearly demonstrate the effectiveness of the proposed APSO.

Acknowledgment The authors would like to thank Angelo Alessandri and the three anonymous referees for their helpful comments.

References

1. B. Abdulhai, H. Porwal and W. Recker, Short-term traffic flow prediction using neuro-genetic algorithms. ITS Journal 7, 3–41, 2002.
2. 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.
3. 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.
4. T.M. Blackwell and J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Transactions on Evolutionary Computations 10(4) 459-472, 2006.
5. S.C. Chang, R.S. Kim, S.J. Kim and B.H. Ahn, Traffic flow forecasting using a 3-stage model, Proceedings of the IEEE Intelligent Vehicles Symposium, pp. 451-456, 2000.
6. K.Y. Chan, T.S. Dillon, C.K. Kwong, Polynomial modelling for time-varying systems based on a particle swarm optimization algorithm, Information Sci., 181(9) 1623-40, 2011.
7. 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.
8. H. Dia, An object-oriented neural network approach to short-term traffic forecasting, European Journal of Operational Research, vol. 131, pp. 253-261, 2001.
9. M. Dougherty, A review of neural networks applied to transport, Transportation Research Part C: Emerging Technologies, vol. 3, no. 4, pp. 247-260, 1995.
10. R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in Proceedings of the 6th International Symposium on Micro Machine and Human Science, IEEE Service Center, Nagoya, pp.39-43, 1995.
11. 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.
12. C.F. Juang, Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes, Fuzzy Sets and Systems 142, pp. 199–219, 2004.
13. J. Kennedy and R.C. Eberhart, *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.
14. J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, IEEE Proceedings of International Conference on Systems, Man and Cybernetics, vol. 5, pp. 4104–4108, 1997.
15. Ledoux C., An urban traffic flow model integrating neural network, Transportation Research, vol. 5C, pp. 287-300, 1997.
16. S. Lee, and D.B. Fambro, Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting, Transportation Research Record, pp. 179–188, 1999.
17. S.H. Ling, F.H.K. Leung, H.K. Lam, Y.S. Lee and P.K.S. Tam, A novel genetic-algorithm-based neural network for short-term load forecasting, IEEE Transactions on Industrial Electronics, vol. 50, no. 4, pp. 793-799, 2003.
18. R.I. Lung and D. Dumitrescu, A collaborative model for tracking optima in dynamic environments, Proceedings of IEEE Con. Evolutionary Computation, pp. 564-567, 2007.
19. T. Luspay, B. Kulcsar, J.W. Wingerden, M. Verhaegen and J. Bokor, Linear parameter varying identification of freeway traffic models, IEEE Transactions on Control Systems Technology, vol. 19, no. 1, pp. 31-45, 2011.
20. 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.
21. I. Okutani and Y.J. Stephanedes, Dynamic prediction of traffic volume through Kalman filtering theory, Transportation Research, Part B: Methodology 18, no. 1, pp. 1–11, 1984.

22. D. Parrott and X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolutionary Computations, vol. 10, no. 4, pp. 440-458, 2006.
23. 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.
24. 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.
25. P. Ross, Exponential filtering of traffic data, Transportation Research Record C, vol. 869, pp. 43–49, 1982.
26. Y. Shi and R. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE Conference on Evolutionary Computation, pp. 69-73, 1998.
27. 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.
28. 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.
29. D. Srinivasan, C.W. Chan, P.G. Balaji, Computational intelligence based congestion prediction for a dynamic urban street network, Neurocomputing 72, pp. 2710-2716, 2009.
30. 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, pp. 521-535, 2008.
31. 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.
32. 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.
33. Y. Wang, M. Papageorgiou and A. Messmer, A real-time freeway network traffic surveillance tool, IEEE Trans. Control Systems Technology, vol. 14, no. 1, 18-32, 2006.
34. H.L. Wei. S.A. Billings, Y. Zhao and L. Guo, Lattice dynamical wavelet neural networks implemented using particle swarm optimization for spatio-temporal system identification, IEEE Transactions on Neural Networks, vol. 20 ,no. 1, pp. 181-185, 2009.
35. R. Xu, D.C. Wunsch II and R.L. Frank, Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization, IEEE Transactions on Computational Biology and Bioinformatics, vol. 4, no. 4, pp. 681-692, 2007.
36. S. Yang and C. Li, A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, IEEE Transactions on Evolutionary Computation, vol. 14, no. 6, pp. 959-974, 2010.
37. 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.
38. G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: the state of art, International Journal of Forecasting, vol. 14, pp. 35–62, 1998.