Engineering Science and Technology, an International Journal 18 (2015) 150–162

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: http://www.elsevier.com/locate/jestch

Full length article

Vehicle speed prediction via a sliding-window time series analysis and an evolutionary least learning machine: A case study on San Francisco urban roads

Ladan Mozaffari [a], Ahmad Mozaffari [b], Nasser L. Azad [b, *]

[a] Department of Management Sciences and Operational Research, Islamic Azad University, Golestan, Iran
[b] Systems Design Engineering Department, University of Waterloo, Ontario, Canada

ARTICLE INFO

Article history:
Received 27 August 2014
Received in revised form
20 October 2014
Accepted 6 November 2014
Available online 19 December 2014

Keywords:
Vehicle powertrains
Speed prediction
Sliding window time series forecasting
Predictive control
Intelligent tools

ABSTRACT

The main goal of the current study is to take advantage of advanced numerical and intelligent tools to predict the speed of a vehicle using time series. It is clear that the uncertainty caused by temporal behavior of the driver as well as various external disturbances on the road will affect the vehicle speed, and thus, the vehicle power demands. The prediction of upcoming power demands can be employed by the vehicle powertrain control systems to improve significantly the fuel economy and emission performance. Therefore, it is important to systems design engineers and automotive industrialists to develop efficient numerical tools to overcome the risk of unpredictability associated with the vehicle speed profile on roads. In this study, the authors propose an intelligent tool called evolutionary least learning machine (E-LLM) to forecast the vehicle speed sequence. To have a practical evaluation regarding the efficacy of E-LLM, the authors use the driving data collected on the San Francisco urban roads by a private Honda Insight vehicle. The concept of sliding window time series (SWTS) analysis is used to prepare the database for the speed forecasting process. To evaluate the performance of the proposed technique, a number of well-known approaches, such as auto regressive (AR) method, back-propagation neural network (BPNN), evolutionary extreme learning machine (E-ELM), extreme learning machine (ELM), and radial basis function neural network (RBFNN), are considered. The performances of the rival methods are then compared in terms of the mean square error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), median absolute percentage error (MDAPE), and absolute fraction of variances (R^2) metrics. Through an exhaustive comparative study, the authors observed that E-LLM is a powerful tool for predicting the vehicle speed profiles. The outcomes of the current study can be of use for the engineers of automotive industry who have been seeking fast, accurate, and inexpensive tools capable of predicting vehicle speeds up to a given point ahead of time, known as prediction horizon (H_P), which can be used for designing efficient predictive powertrain controllers.

1. Introduction

Automotive companies are under extreme economic and societal pressures to improve the fuel economy and emission performance of their products. Therefore, they need to develop and apply significant technological advancements continuously to meet today's increasingly tight emission and fuel standards and regulations. Recently, the development of route-based or predictive powertrain control systems has received significant attention from the automotive industry to achieve these objectives. These predictive controllers use the prediction of vehicle's upcoming power demands, which is strongly a function of the future speed profile, to improve the powertrain performance. For instance, the vehicle speed prediction has been used to develop a predictive automatic gear shift controller to optimize the gear shifting to increase the fuel economy [1]. Another example is the use of predicted upcoming speed profiles to develop predictive power management controllers for hybrid electric vehicles (HEVs). An HEV powertrain system consists of a combustion engine and an electric motor to propel the vehicle. To improve the HEV's fuel economy, a power management controller is needed to optimally divide the vehicle power demand between its two propulsion systems. Researchers have indicated that additional fuel savings up to 4% can be obtained

* Corresponding author.
E-mail address: nlashgar@uwaterloo.ca (N.L. Azad).
Peer review under responsibility of Karabuk University.

http://dx.doi.org/10.1016/j.jestch.2014.11.002

for each HEV equipped with a route-based power management controller [2]. Broad implementation of predictive powertrain control systems for power management, gear shift scheduling, and so on would have a huge impact, saving millions of gallons of gasoline annually, and also, leading to significant emission reductions.

Accurate short-range speed profiles are critical for real-time implementations of these predictive powertrain control systems; therefore, future vehicle speed trajectory should be predicted in small time horizons. Data required to comprehensively describe future driving speeds may be collected using advanced communication systems and radar sensor technologies. In this regard, the emerging field of intelligent transportation system (ITS) has attracted an increasing attention of industrialists, and in particular, automotive engineers [3]. ITS is an advanced concept in the field of communication technologies which is based on the fact that it is a reasonable policy to develop driving condition prediction devices for improving the vehicle performance characteristics. So far, different technologies, e.g. vehicle telematics (VTs), have been created and used to gather beneficial information regarding the short-term and long-range traffic speed profiles, which can be employed by the predictive powertrain controllers to improve the fuel economy and emission performance. Unfortunately, most of the existing technologies which serve as the VT for ITS-based vehicles require an expensive infrastructure. These expensive infrastructures cannot be found in all regions, and thus, ITS technologies, for instance vehicle-to-infrastructure communication systems, cannot be used globally. Moreover, many cars are still built without the on-board sensors necessary to access ITS information, for example vehicle-to-vehicle communication systems. A near-term (and cost-effective) solution is to use historical driving data as well as currently available information from factory-made vehicle sensors to predict online the vehicle's driving cycle for near future.

The aforementioned flaws have instigated the researchers of automotive engineering and systems sciences to take advantage of computationally efficient numerical methods as well as intelligent tools to predict the vehicle speeds while driving over a given drive cycle. In this context, the concept of time series analysis is taken into account to analyze the history of vehicle's motion and develop a predictive tool capable of forecasting the vehicle or traffic speed along the road.

So far, intelligent time series forecasting has successfully been utilized for a wide range of applications. A comprehensive review on the progress and development of such techniques can be found in Ref. [27]. In particular, time series forecasting models can be implemented in a state-space representation form. Hyndman et al. [28] proposed an exponential smoothing method to develop a state-space framework for automatic forecasting tasks. Later in 2005, Hyndman et al. [29] conducted a comprehensive simulation and demonstrated that the time series forecasting with state-space representation can be easily used as a model for controllers. George et al. [30] comprehensively studied the applicability of time series for forecasting and control tasks. The results demonstrated that time series predictors are best suited to be used in controlling schemes for a wide spectrum of engineering applications. This point has been substantially indicated by many other researchers working in different fields.

Performing time series analysis and employing intelligent tools such as artificial neural networks (ANNs) has been proven to be efficient tools for accurate forecasting of vehicle speed, power demands, etc. which then can be used for designing more efficient and cost-effective predictive automotive controllers. In a research paper by Abdulhai et al. [4] a neuro-genetic predictive tool was proposed for forecasting the short-term traffic flow on roads. The proposed model was validated using the actual traffic flow data acquired from ATMS test bed in Orange Country, California. Three years later, Vlahogianni et al. [5] utilized genetic algorithm (GA) for the both optimization and evolving of ANN architectures for short-term traffic flow prediction. The simulations demonstrated that the proposed forecasting tool can yield satisfactory results for modeling both univariate and multivariate traffic databases. Jiang et al. [6] proposed the application of dynamic wavelet neural networks for forecasting the traffic flow, and vehicle speed trajectories. By considering a comprehensive database, the authors have been enabled to come up with a dynamic recurrent intelligent tool capable of forecasting the long-term and short-term traffic flow and vehicles speeds. They concluded that their model is able to be used by traffic engineers and highway agencies to create effective traffic management plans to alleviate freeway congestions. Zheng et al. [7] proposed an integrated Bayesian network with a neural network for accurate short-term traffic flow prediction. The numerical simulations proved the authenticity of the proposed tool for prediction of the traffic flow. Chan et al. [8] developed an ANN based on an exponential smoothing method to come up with an accurate intelligent tool for forecasting the traffic flow. The proposed study firstly applied the exponential smoothing method to handle the non-smooth and discontinuous features of a database collected from the traffic conditions on a section of a freeway in Western Australia. Thereafter, ANN was utilized to forecast the traffic flow on the considered road. The authors observed that the proposed method is accurate and can be used for real-time implementations. Later, Chan et al. [9] verified the authenticity of their technique by repeating the same simulations using a Levenberg–Marquardt ANN (LM-ANN). The simulations demonstrated that LM-ANN can even improve the prediction accuracy of standard back-propagation ANN in terms of the robustness and accuracy.

In all of the abovementioned research studies, the traffic flow has been predicted with respect to the positioning of the vehicle on the road. Thus, in spite of the usefulness of the proposed intelligent tools for traffic management, they cannot be employed to provide the required information for the powertrain control unit of the vehicle to improve the performance. In fact, to enhance the functionality of the vehicle controller, intelligent tools should be used to predict the future vehicle speeds with respect to the real-time speed profile of a moving vehicle [10]. A similar approach has been pursued by Fotouhi et al. [10] to design an efficient power management controller for a HEV. Moreover, there are other investigations which demonstrate that intelligent tools can be used for the prediction of vehicle speeds. Shu et al. [24] indicated that fuzzy algorithms can be used for the prediction of parallel hybrid electric vehicle speeds. Mahmoudabadi [25] took advantage of artificial neural networks for estimating the average speed of a vehicle in rural roads. Park et al. [26] used a neural network for real-time vehicle speed predictions and indicated the validity of their methodology. Considering the previous work, it sounds that using intelligent tools for designing vehicle speed predictors at the heart of a predictive powertrain controller is a reasonable option.

In this work, the authors have two main goals. From an automotive engineering viewpoint, they would like to realize whether the same concept, i.e. developing an intelligent tool for predicting the vehicle speed in a real-time fashion, can be implemented with an acceptable accuracy for a database collected by the movements of a Honda Insight personal vehicle on the San Francisco urban roads. If so, the predictive tool can then be used for designing a computationally efficient powertrain predictive controller. From a numerical methods' point of view, the authors aim at developing an advanced intelligent tool, evolutionary least learning machine (E-LLM) [11], which can be trained very fast, and also, can be effectively combined with sliding window time series analysis tools. By

implementing the numerical tool based on the state-space concept, through an exhaustive comparative study, the capability of E-LLM for the accurate and fast prediction of vehicle speeds over a predefined prediction horizon (H_P) is evaluated. If the model works properly, it can be inferred that it has a high potential to be used for real-time applications.

The rest of the paper is organized as follows. Section 2 is devoted to the description of the steps required for implementing the database in a state-space format, as well as the description of sliding window and H_P considered for this study. The structure of the proposed evolvable E-LLM is described in Section 3. The parameter settings and statistical metrics required for conducting the simulations are given in Section 4. The simulation results are presented in Section 5. Finally, the paper is concluded in Section 6.

2. Sliding window time series analysis of the collected data

2.1. Data collection

The driving data used in this study have been extracted from a database created as part of the ChargeCar project in the Robotic Institute at Carnegie Mellon University [12]. The database is very comprehensive, and includes many driving cycles information representing the traffic flow, and vehicle speed for different automobiles in several cities and states of the USA. In this study, the authors intend to adopt a very challenging driving data which represent the speed variations of a Honda Insight personal vehicle on the San Francisco urban roads [13]. The data have been gathered by contriving an advanced vehicle location (AVL) system, working based on the Garmin global positioning system (GPS), into the Honda Insight automobile. The Honda Insight has the frontal area of 29.28 ft^2, and the weight of 3300 lbs. The total passengers' weight was equal to 140 lbs. The total collected data comprise of 5 different segments and each one has been collected at a different condition: from home to work, from work to a restaurant for lunch, back from restaurant to work, from work to home, and a trip from home to a specific destination. This results in comprehensive information regarding the possible driving cycles of a typical car on the urban roads of San Francisco. The data have been recorded at every second, which forms a vector including the time (sec), speed (m/s), acceleration (m/s^2), power based model (kW), and distance (m). It is worth pointing out that the original database hosts all of the stop-start information, and also, it considers the idling durations. However, in this study, our main interest is to develop an intelligent tool for predicting the vehicle speed in a real-time fashion. Thus, we do not need to know when and where the driver stops driving, and for instance, goes for lunch. Therefore, the idling periods which provide no additional information have been removed from the database, and the driving data are divided into five separate segments. Consequently, 5 different trajectories representing the vehicles speeds on the San Francisco urban roads are created. Fig. 1 indicates the collected speed profiles for these sub-cycles.

Obviously, each of the considered scenarios offers a completely different speed pattern, and thus, by applying the proposed method to these patterns, the reliability and efficacy of the resulting predictor can be verified.

2.2. Sliding window time series analysis

The implementation of sliding window time series (SWTS) analysis enables us to later use the resulting predictor at the heart of the powertrain predictive controller. To conduct SWTS, the driving data should be presented in the state-space format. SWTS partitions the database into a number of finite-length segments and tries to relate z past data to the p ahead data. From a model predictive control (MPC) design perspective (which is used commonly for predictive controllers design), it can be interpreted that SWTS enables us to use the history of the vehicle's motions to forecast the

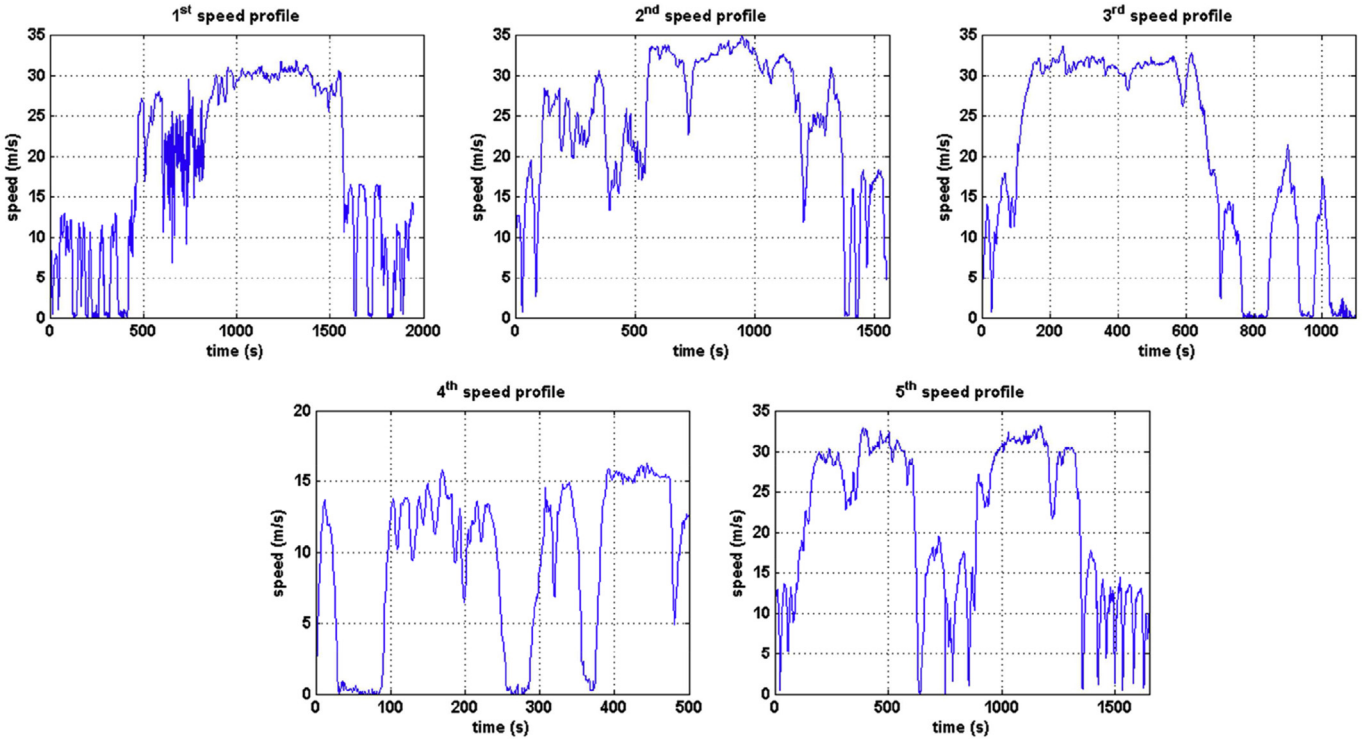

Fig. 1. The five considered urban speed profiles of Honda Insight.

future speeds up to a predefined prediction horizon (H_P). A schematic illustration of treating a time-dependent database in a sliding window format is depicted in Fig. 2.

Assume that a given database represents the time-dependent profile of vehicle speed ($V(t)$) from $t = 0, 1,\ldots, T$. Then, SWTS implies that at each set point, e.g. $t = t_n$, the vehicle speed for $t = t_{n+1}$, $t_{n+2},\ldots, t_{n+p}$ can be modeled using the vehicle speed history over a finite previous time sequence, i.e. $t = t_{n-1}, t_{n-2},\ldots, t_{n-z}$. It is worth pointing out that, in real-time implementations, the resulting algorithm can be updated after eacht' seconds, where $\{t' \in [t_{n+1}, t_{n+p}]/t' \in \mathbb{Z}\}$.

2.3. How the proposed scheme can be used by the predictive powertrain control unit

As it was stated before, the main goal of proposing such a scheme is to later devise it into the predictive powertrain control unit to improve the fuel economy and emission performance of the vehicle. In the previous sub-section, the authors described how SWTS enables us reform any time-dependent database into the standard state-space format for time series-based predictions. Let's assume that the predictive powertrain control actions should be repeated at every 10 s during the driving cycle, which means that $t' = 10$ sec. Now, two important parameters which should be verified are the input sliding window and the output sliding window. It is well-known that the performance of an MPC-based predictive controller is highly sensitive to the size of both input sliding window and output sliding window (H_P). In fact, by taking into account that each working point should be evaluated at every 1 s, it is very important to determine the optimum value of H_P. It is clear that by increasing the prediction horizon of a predictive controller, we can come up with a more reliable control strategy. However, by increasing the output sliding window, the performance of the intelligent machine may be undermined. Hence, in this study, the authors consider the prediction horizons within the range of [10,20] to extract the optimum/feasible value of H_P. After designing an efficient method capable of predicting the vehicle speeds for the next H_P states, so many H_P segments representing the characteristics of the driving data can be created and used to calculate the respective control laws for the predictive controller as an offline look-up table. In this context, at each updating point, the intelligent predictor, i.e. E-LLM, uses the past z (equal to 7 in this study) previously captured temporal speed values and predict the H_P ahead time states. It is worth pointing out that to select the value of z, a model order selection mechanism has been utilized. In this way, the authors have considered five different values of z, namely 5, 7, 9, 11, 15, and checked the accuracy, and also, the computational complexity of the derived prediction systems. Based on some numerical experiments, it has been observed that the prediction error of E-LLM when uses 7 or more previously captured temporal speeds is relatively the same. However, it is clear that as the considered intelligent system will be used as an online predictor, its computational complexity is of the highest importance. Therefore, the authors have chosen z of 7 for designing the intelligent system. It is worth pointing out that, after K-fold training, the authors observed that the prediction accuracy of E-LLM cannot be improved if the intelligent system uses $z > 7$. After the prediction, the similarity of the predicted speed profile with the segments used to create the control laws is investigated by a similarity search algorithm to find the proper control law from the look-up table for next $t' = 10$ seconds of the driving cycle. One of the salient assets of the resulting predictive controller is that it does not rely on expensive ITS infrastructures and on-board sensors to get the future vehicle speed profile. Rather, the proposed drive cycle prediction algorithm provides the required information. A schematic illustration of the predictive powertrain control scheme concept based on the driving cycle time-series prediction is depicted in Fig. 3.

3. Evolutionary least learning machine

In the previous section, the authors described how the database should be treated to be prepared for the proposed time series-based analysis. In this section, the authors explain the steps required for implementation of the time series-based predictor. As it was mentioned, the contribution of the intelligent predictor is to relate a sequence of the previously captured speed signals with H_P

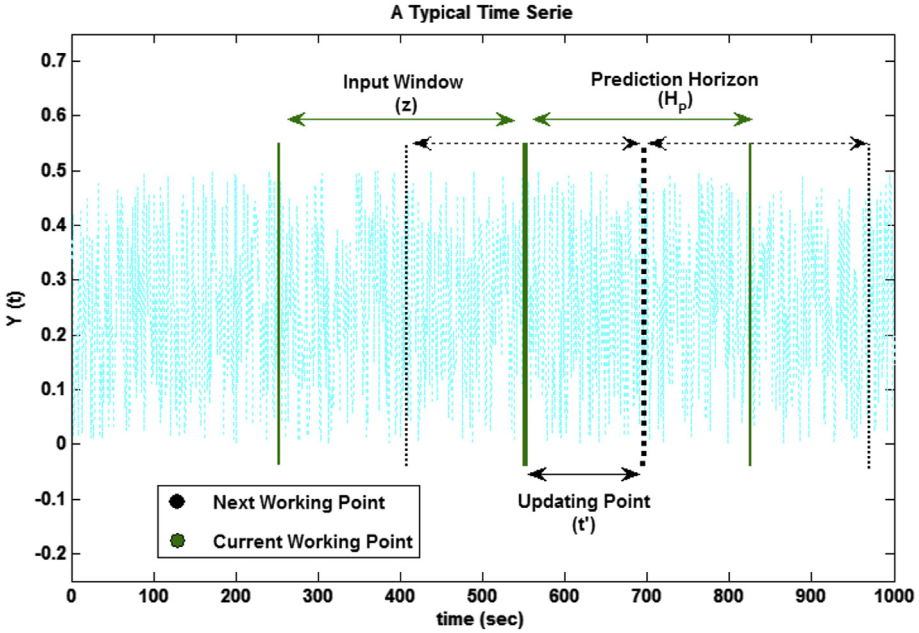

Fig. 2. The sliding window based time series analysis.

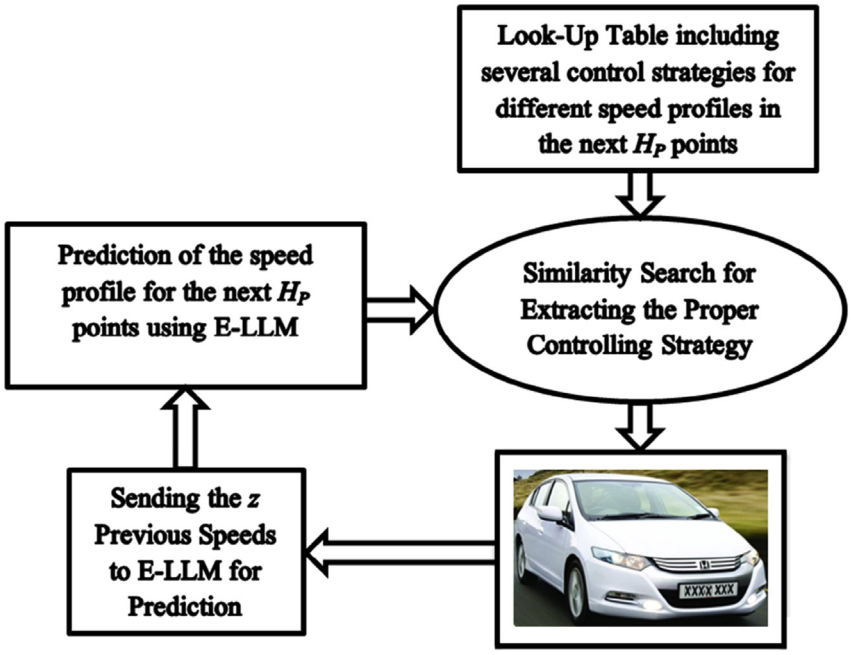

Fig. 3. A schematic illustration of the proposed time series prediction-based powertrain controller.

proceeding speed signals. Let us assume that the z previously captured speed signals form an input vector X of z arrays, and the H_P ahead signals form an output vector Y_d of H_P arrays, then, the intelligent predictor (Ψ) can be considered as a system comprising of several mathematical formulations aim at making a map between X and Y_d, as given below:

$$Y_p = \Psi(X) \tag{1}$$

where Y_P is the vector of H_P proceeding signals predicted by the intelligent predictor. The art of designing a time series-based predictor is to form the function Ψ such that the output of the predictor Y_P has the lowest possible deviation from the desired ahead sequences vector Y_d.

Given the abovementioned facts, in this section, the authors describe how they designed such an intelligent predictor. The proposed E-LLM system is a hybrid bi-level machine which uses an evolutionary algorithm called mutable smart bee algorithm (MSBA) as a meta-optimizer for evolving the architecture of least learning machine (LLM) so that the prediction accuracy is increased to the highest possible degree. In what follows this section, the authors describe the architecture of E-LLM. Firstly, the steps required for implementing LLM are explained. Thereafter, the contribution of MSBA as a meta-optimizer for evolving the architecture of LLM is scrutinized.

3.1. Least learning machine

LLM is an extended version of extreme learning machine (ELM) which enables us design multi-layer feed-forward neural network architectures [11]. In contrast to ELM which has only one hidden layer, LLM may have several hidden layers working altogether. Indeed, such a characteristic is very suited when we would like to design a time series-based predictor. In a study by Vafaipour et al. [14], it was demonstrated that multi-layered feed-forward (MLFF) architectures with more than one hidden layer can afford much more accurate results for time series predictions. This is because, in most of the cases, the relation between the previously captured signals and the proceeding signals is highly nonlinear and requires network architectures of higher order of nonlinearity (i.e. those having several hidden layers). In this investigation, it has been demonstrated that by integrating the concept of hidden feature space ridge regression (HFSR) and ELM, a powerful tool can be developed which is suited for multi-input multi-output (MIMO) systems, and thus, is useful for the time series analysis.

The mathematical steps required for the implementation of LLM is the same as those of ELM. However, as the architecture of LLM is different from ELM, a number of subtle remarks should be considered to properly implement LLM. Assume that the MIMO database has N data points in which the input vector has D_{inp} elements and output vector has D_{out} elements. This can be mathematically expressed as:

$$\text{Data} = \left\{ \left(x_j, y_j\right) \middle| x_j \in R^{D_{\text{inp}}}, y_j \in R^{D_{\text{out}}}, j = 1, 2, \ldots, N \right\} \tag{2}$$

Assume that LLM has $l = 1, 2, \ldots, L$ hidden layers, and l_{th} layer has HN_l hidden nodes. In this study, the activation function of each node is set to be *logsig*. In LLM, three categories of structural features should be considered: (1) those dealing with the input layer and the first hidden layer (I–H), (2) those dealing with the interaction of hidden layers (H–H), and (3) those dealing with the last hidden layer and the output layer (H-O). Let us indicate the synaptic weights connecting (1) input layer to the first hidden layer with (W_{IH}), (2) l_{th} hidden layer to l'_{th} hidden layer with ($W_{HH}^{l-l'}$), and (3) the last hidden layer to the output layer with (W_{HO}). Just like ELM, LLM works based on the algebraic multiplication of the synaptic weights with their respective regression matrices. As ELM has only one hidden layer, it has only one regression matrix, known as the hidden layer output matrix (H) [15]. However, based on the description provided before, LLM requires two different types of regression matrices: (1) one H_{IH} matrix, and (2) a number of $H_{HH}^{l-l'}$ matrices. H_{IH} is formed based on the multiplication of W_{IH} with the nodes of the first hidden layer and $H_{HH}^{l-l'}$ is formed by multiplication of $W_{HH}^{l-l'}$ with the hidden nodes of l'_{th} hidden layer. The value of H matrices depends on the characteristics of H matrices for all of the previous layers. Except the synaptic weights of the last layer, the weights of the previous layers can be randomly set. To calculate the synaptic

weights connecting the last hidden layer to the output layer, the pseudo-inverse least square method (LSM) is used. Before proceeding with this approach, we need to explain how the regression matrices can be determined.

Calculation of H_{IH}: This matrix is formed by concatenation of the firing signals of each hidden node with respect to all of the arrays of W_{IH}. Assume that the *logsig* activation function of each node is indicated with g and each node has an external bias (b). Then, based on the standard inference of any neural system, the matrix can be defined as:

$$H_{IH}=\begin{bmatrix} g\left(W_1^{IH}\cdot x_1+b_1\right) & \cdots & g\left(W_{HN_1}^{IH}\cdot x_1+b_{HN_1}\right) \\ \vdots & \ddots & \vdots \\ g\left(W_1^{IH}\cdot x_{D_{inp}}+b_1\right) & \cdots & g\left(W_{HN_1}^{IH}\cdot x_{D_{inp}}+b_{HN_1}\right) \end{bmatrix}_{D_{inp}\times HN_1} \tag{3}$$

Now, imagine that we want to proceed with the next layer, i.e. forming the matrix H which represents the regressors of the first hidden layers. Obviously, this matrix (H_{HH}^{1-2}) is formed by concatenation of the firing signals of each hidden node in the second hidden layer with respect to all of the arrays of H_{IH}. Again, by considering the *logsig* activation function for each node and a bias, H_{HH}^{1-2} can be formed as given below [16]:

$$H_{HH}^{1-2}=\begin{bmatrix} g\left(W_1^{HH^{1-2}}\cdot\left(\sum_{u=1}^{D_{inp}}H_{IH}(u,1)\right)+b_1\right) & \cdots & g\left(W_{HN_2}^{HH^{1-2}}\cdot x_1\left(\sum_{u=1}^{D_{inp}}H_{IH}(u,1)\right)+b_{HN_2}\right) \\ \vdots & \ddots & \vdots \\ g\left(W_1^{HH^{1-2}}\cdot\left(\sum_{u=1}^{D_{inp}}H_{IH}(u,HN_1)\right)+b_1\right) & \cdots & g\left(W_{HN_2}^{HH^{1-2}}\cdot\left(\sum_{u=1}^{D_{inp}}H_{IH}(u,HN_1)\right)+b_{HN_2}\right) \end{bmatrix}_{HN_1\times HN_2} \tag{4}$$

As it can be seen, the values of the arrays of the new formed H matrix depend on the value of the last H matrix. By proceeding with this layer by layer procedure, the matrix H formed in the last hidden layer ($H_{HH}^{L-1\_L}$) can be mathematically expressed as:

$$H_{HH}^{L-1\_L}=\begin{bmatrix} g\left(W_1^{HH^{L-1\_L}}\cdot\left(\sum_{u=1}^{HN_{L-2}}H_{HH}^{L-1\_L-2}(u,1)\right)+b_1\right) & \cdots & g\left(W_{HN_L}^{HH^{L-1\_L}}\cdot\left(\sum_{u=1}^{HN_{L-2}}H_{IH}(u,1)\right)+b_{HN_L}\right) \\ \vdots & \ddots & \vdots \\ g\left(W_1^{HH^{L-1\_L}}\cdot\left(\sum_{u=1}^{HN_{L-2}}H_{IH}(u,HN_{L-1})\right)+b_1\right) & \cdots & g\left(W_{HN_L}^{HH^{L-1\_L}}\cdot\left(\sum_{u=1}^{HN_{L-2}}H_{IH}(u,HN_{L-1})\right)+b_{HN_L}\right) \end{bmatrix}_{HN_{L-1}\times HN_L} \tag{5}$$

After the calculation of the last H matrix, LLM can easily map any feature to each other, as given below:

$$H_{HH}^{L-1\_L}\cdot W_j^{HO}=y_P^j \tag{6}$$

where $j=1,\ldots,D_{out}$. This means that, by repeating the pseudo-inverse LSM for D_{out} times, LLM can deal with any MIMO database. Hence, we expect that the final W^{HO} be a matrix with the following format:

$$W^{HO}=\begin{bmatrix} W_{1,1}^{HO} & \cdots & W_{1,D_{out}}^{HO} \\ \vdots & \ddots & \vdots \\ W_{HN_L,1}^{HO} & \cdots & W_{HN_L,D_{out}}^{HO} \end{bmatrix}_{HN_L,D_{out}} \tag{7}$$

The value of W^{HO} is calculated using the LSM method, as follows:

$$W^{HO}=\overset{\dagger}{H}_{HH}^{L-1\_L}\cdot Y_P \tag{8}$$

where $\overset{\dagger}{H}_{HH}^{L-1\_L}$ is calculated as:

$$\overset{\dagger}{H}_{HH}^{L-1\_L}=\left(\overset{T}{H}_{HH}^{L-1\_L}\cdot H\right)^{-1}\overset{T}{H} \tag{9}$$

where $\overset{T}{H}$ is the transpose of matrix H.

The architecture of LLM is depicted in Fig. 4.

3.2. Mutable smart bee algorithm

In this section, the authors describe how MSBA is used to evolve the architecture of LLM. MSBA is a relatively recent metaheuristic technique proposed by Mozaffari et al. [17]. The main motivation behind the development of MSBA was to improve the exploration/exploitation capabilities of artificial bee colony (ABC) algorithm, and also, to come up with a searching mechanism which is best suited for constraint optimization problems. The initial simulations proved that MSBA can outperform several state-of-the-art metaheuristics in constraint optimization problems. Later, MSBA has been applied to several engineering and numerical unconstraint optimization problems and the available reports confirm that it is a powerful optimization approach when we are dealing with unconstraint problems [18]. Recently, Mozaffari et al. [19] proposed an adaptive MSBA which adapts the mutation probability over the optimization procedure. Through an exhaustive numerical experiment, it has been concluded that the adaptive version of MSBA can even yield much more promising results as it provides a logical

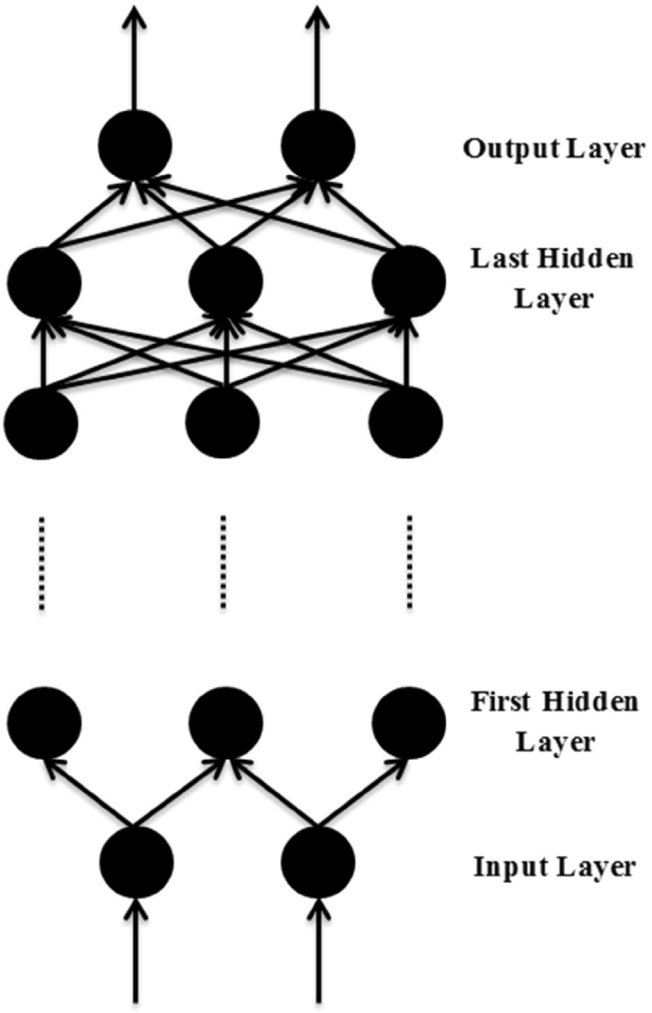

Fig. 4. A schematic illustration of the LLM architecture.

balance between the exploration and exploitation capabilities. A comprehensive review of the chronological advances of MSBA can be found in Mozaffari et al. [19], and for the sake of brevity, the authors refer the interested readers to this paper. From a numerical method's point of view, it has been demonstrated that the heuristic agents of MSBA, known as smart bees, provide the resulting metaheuristic with two advantageous features compared to the most of existing metaheuristics. Firstly, these agents are equipped with a finite-capacity (short-term) memory which enables them to compare their new positions with the previously detected regions and, based on a greedy selection, they can select the fittest solution. Secondly, the mutation operator devised in MSBA guarantees the diversification of the search, and thus, a successful search can be performed with even a low number of the heuristic agents. The mentioned numerical advantages have instigated us to take advantage of MSBA for evolving the architecture of LLM. The pseudo-code of MSBA is presented in Fig. 5.

To evolve the architecture of LLM, we should find out which controlling parameters have the potential of being optimized such that the accuracy of LLM is maximized. Based on the descriptions given previously, one can easily understand that there are two main controlling parameters which have a significant effect on the performance of LLM: (1) the number of hidden layers (L), and (2) the number of hidden neurons in each layer of LLM. Fortunately, the synaptic weights of LLM are verified randomly or analytically, and thus, it is not needed to consider them as the decision parameters of MSBA. The following steps should be taken so that MSBA can evolve the architecture of LLM:

Step 1: Randomly, initialize the population of s smart bees. Each of these smart bees has 9 decision variables with the following form:

$$S = [\, l \quad HN_1 \quad HN_2 \quad HN_3 \quad HN_4 \quad HN_5 \quad HN_6 \quad HN_7 \quad HN_8 \,] \tag{10}$$

where l is an integer value within the range of [1,8] and the other decision variables are integer values within the range of [2,10]. It should be noted that the first decision variable represents the number of hidden layers and the other 8 variables indicate the number of hidden nodes at each layer. Clearly, when the number of hidden layers is less than 8, for example 5, we only consider the 2nd to 5th decision variables and the value of the remaining decision variables will be equal to 0. The integer programming can be easily done in the Matlab software. To do so, we only need to initialize the value of each variable by using the command *rand*, and then, extract the corresponding integer value by using the command *round*.

Step 2: After the formation of the corresponding LLM, its accuracy is calculated using the equation below (which is the objective function of MSBA):

$$F = \min \quad \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D_{out}}\left(Y_P^{i,j} - Y_d^{i,j}\right)^2 \tag{11}$$

It is clear that, as a meta-optimizer, MSBA tries to optimize the characteristics of its heuristic agents so that the value of the objective function is minimized. Through the above steps, the structure of E-LLM is formed.

4. Parameter settings, rival techniques and performance evaluation metrics

To conduct the numerical simulations, a set of parameters should be set, and also, a number of performance evaluation metrics should be considered. It is also important to select a set of rival methods to endorse the applicability of the proposed time series forecasting technique. As mentioned previously, the considered predictor is comprised of two different levels, a meta-optimizer and a predictor. To evaluate the performance of MSBA, a set of the rival metaheuristics, i.e. particle swarm optimization (PSO) [20], genetic algorithm (GA) [21], and artificial bee colony (ABC) [22], are used. All of the metaheuristics perform the optimization with a population of 20 heuristic agents, for 100 iterations. Besides, all of the rival techniques conduct the optimization with a unique initialization to make sure that the obtained results are not biased. To capture the effects of uncertainty, the optimization is conducted for 20 independent runs with random initial seeding, based on the Monte-Carlo simulation. For PSO, a linear decreasing adaptive inertia weight (with initial value w_0 of 1.42) is taken into account. The values of social and cognitive coefficients are set to be 2. For the both ABC and MSBA, the trial number of 10 and the modification rate (MR) of 0.8 are considered. For MSBA, 5 bees violating the admissible trial number are sent to the mutation phase. The mutation operator is an arithmetic graphical search (AGS) operator. It is worth noting that the mutation probability of MSBA is a linear decreasing function with the initial

Pseudo-code of MSBA

begin
Objective function f(X), $X=(x_1,\ldots,x_d)^T$
Generate initial population of n employed bees X_i (i = 1,2,…,n)
Start " Mutable Smart Bee Algorithm"
while ($t < t_{max}$) or (stop criterion)
  for i = 1: Number of employed bees (n)
    Select a decision variable (d) of i^{th} bee randomly
    Select d^{th} decision variable of another bee
    Select the random number ϕ within the range of unity [0,1]
    Update the d^{th} decision variable of i^{th} bee as following:
    $$x_{i,d} = x_{i,d} + \phi\left(x_{i,d} - x_{j,d}\right)$$
  end
  Calculate the probability ($prob$) of all n agents based on their fitness
  for i = 1: Number of onlooker bees (n)
    if $rand < prob$ (i)
      Select a decision variable (d) of i^{th} bee randomly
      Select d^{th} decision variable of another bee
      Select the random number ϕ within the range of unity [0,1]
      Update the d^{th} decision variable of i^{th} bee as following:
      $$x_{i,d} = x_{i,d} + \phi\left(x_{i,d} - x_{j,d}\right)$$
    end
  end
  Discriminate m solutions which exceed the trial number
  for i = 1: m
    Select a decision variable (d) of i^{th} bee randomly
    if $rand \geq P_m$
      Select the random number ϕ within the range of unity [0,1]
      $$x_i = \begin{cases} x_i + \phi\left(x_i - lb\right)\left(1 - \frac{t}{T}\right)^{1.2} \\ x_i + \phi\left(ub - x_i\right)\left(1 - \frac{t}{T}\right)^{1.2} \end{cases}$$
    elseif $rand < P_m$
      $x_i = x_i$
    end
  end
end while
Post process results and visualization
end of MSBA

Fig. 5. Pseudo-code of MSBA.

value of 0.05. For GA, the tournament selection mechanism, Radcliff crossover and polynomial mutation are taken into account. The crossover and mutation probabilities are set to be 0.8 and 0.04, respectively. All of the considered metaheuristics are integrated with LLM and their performances are evaluated to make sure that the proposed meta-optimizer is powerful. After that, the performance E-LLM is compared against the auto regressive (AR) method [10], back-propagation neural network (BPNN) [14], evolutionary extreme learning machine (E-ELM) [23], extreme learning machine (ELM) [15], adaptive neuro-fuzzy inference system (ANFIS) [24], and radial basis function neural network (RBFNN) [14], to indicate that the considered predictor works properly. By performing a model order selection recommended by Huang et al. [15], the authors realized that ELM should have 12 hidden nodes in its hidden layer. To use BPNN, a sensitivity analysis, as recommended by Vafaeipour et al. [14], was performed and it was observed that BPNN should have three hidden layers with 2–5–6 hidden nodes. E-ELM is the same as the one proposed in Zhu et al. [23]. It is worth noting that to evade numerical instabilities and biased results, as well as to discern the

maximum computational potentials of the considered predictors, the database is normalized within the range of unity, using the following formulation:

$$X^i_{norm} = \frac{X^i - X^i_{\min}}{X^i_{\max} - X^i_{\min}} \tag{12}$$

where $i = 1,\ldots, D_{inp}$. The outputs are also normalized in the same fashion.

Furthermore, it is important to consider a set of the performance evaluation techniques to compare the accuracy of the considered predictors. In this study, the mean square error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), median absolute percentage error (MDAPE), and absolute fraction of variances (R^2) metrics are utilized to accurately compare the capabilities of the considered predictors. The mathematical formulations of the considered predictors are given below:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D_{out}}\left(Y_P^{i,j} - Y_d^{i,j}\right)^2 \tag{13}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D_{out}}\left(Y_P^{i,j} - Y_d^{i,j}\right)^2} \tag{14}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D_{out}}\frac{\left|Y_P^{i,j} - Y_d^{i,j}\right|}{Y_d^{i,j}} \times 100 \tag{15}$$

$$MDAPE = \sum_{j=1}^{D_{out}} median\left(\left|\frac{Y_P^j - Y_d^j}{Y_d^j}\right| \times 100\right) \tag{16}$$

$$R^2 = \frac{1}{D_{out}}\sum_{j=1}^{D_{out}}\left(\frac{\sum_{i=1}^{N}\left(Y_d^{i,j} - avg\left(Y_d^{i,j}\right)\right)\left(Y_P^{i,j} - avg\left(Y_P^{i,j}\right)\right)}{\sqrt{\sum_{i=1}^{N}\left[\left(Y_d^{i,j} - avg\left(Y_d^{i,j}\right)\right)^2\right]\left[\left(Y_P^{i,j} - avg\left(Y_P^{i,j}\right)\right)^2\right]}}\right) \tag{17}$$

To make sure that the identifiers are trained based on their prediction power (namely, the potential of forecasting upcoming speeds), and also, they are able to capture the effects of biased training, all of the identification scenarios are conducted based on a 10-fold cross-validation. This means that the datasets are divided into 10 subgroups and the training is performed at 10 different stages. In this context, 9 subgroups are used for the training, and then, the prediction performance is evaluated by the 1 remaining unseen dataset. Such a procedure is transacted so that we make sure all of the 10 subgroups have been considered as the unseen data. Finally, the average values of each of the 10 folds are again averaged and the final value is reported. As mentioned previously, 10-fold training/testing is repeated for 20 independent runs to provide statistical feedback regarding the performance of the rival identifiers. All of the simulations are performed in the Matlab environment with Microsoft Windows 7 operating system on a PC with a Pentium IV, Intel Dual core 2.2 GHz and 2 GBs RAM.

5. Simulation results

5.1. Evaluating the performance of meta-optimizers

At the first step of the experiments, the authors would like to understand whether the MSBA meta-optimizer selected for evolving the architecture of LLM can afford acceptable outcomes. Fig. 6 depicts the mean real-time convergence behavior of MSBA, PSO, GA and ABC for evolving the architecture of LLM. To verify the prominent performance of MSBA, the authors depict all of the

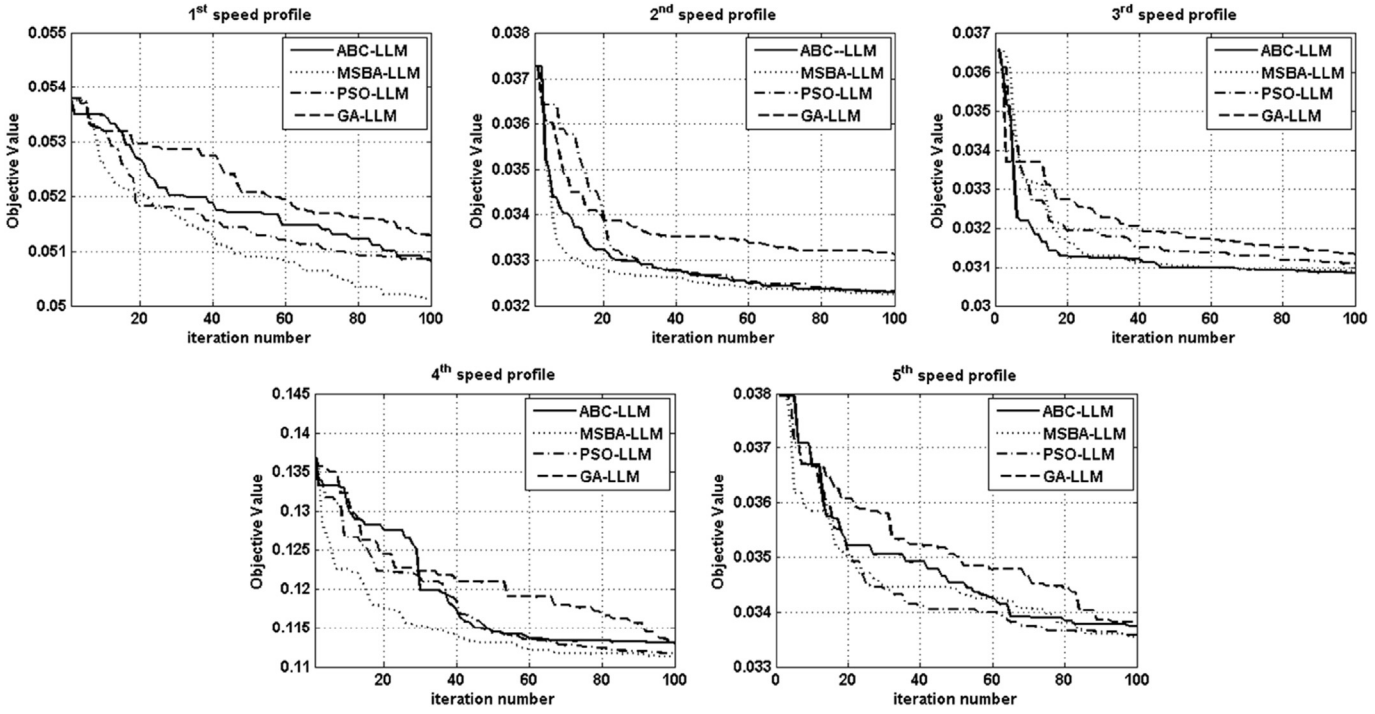

Fig. 6. The convergence behavior of the considered meta-optimizer.

Table 1
The statistical results yielded by meta-optimizers for 20 independent runs.

Methods	Mean	Std.	Min	Max
1st speed profile				
ABC-LLM	0.0508343	3.22e0-4	0.0502261	0.0509921
MSBA-LLM	0.0501197	4.36e0-5	0.0501031	0.0501516
PSO-LLM	0.0508546	6.32e0-4	0.0502162	0.0509317
GA-LLM	0.0512877	1.56e0-3	0.0497417	0.0521522
2nd speed profile				
ABC-LLM	0.0322939	1.51e0-4	0.0320831	0.0323152
MSBA-LLM	0.0322461	2.57e0-5	0.0322197	0.0322636
PSO-LLM	0.0323027	3.66e0-5	0.0322841	0.0323552
GA-LLM	0.0331316	4.77e0-3	0.0324621	0.0363381
3rd speed profile				
ABC-LLM	0.0308893	4.42e0-5	0.0308263	0.0309153
MSBA-LLM	0.0308362	9.57e0-7	0.0308125	0.0308731
PSO-LLM	0.0310872	4.41e0-7	0.0310015	0.0311251
GA-LLM	0.0313186	1.89e0-4	0.0311031	0.0314436
4th speed profile				
ABC-LLM	0.1130143	6.34e0-3	0.1116721	0.1155351
MSBA-LLM	0.1112692	1.17e0-3	0.1108994	0.1136735
PSO-LLM	0.1117513	4.49e0-3	0.1106893	0.1123215
GA-LLM	0.1130989	6.63e0-2	0.1121553	0.1155435
5th speed profile				
ABC-LLM	0.0337523	8.53e0-4	0.0335521	0.0339492
MSBA-LLM	0.0335239	1.33e0-5	0.0334931	0.0335392
PSO-LLM	0.0335624	2.02e0-5	0.0335443	0.0335846
GA-LLM	0.0338298	3.11e0-4	0.0336732	0.0341536

optimization scenarios, namely for the entire speed profiles. It can be seen that the performance of MSBA is clearly superior to the other techniques for the first speed profile. For the other optimization scenarios, the performances of PSO and ABC are also comparable to those of MSBA. However, the main point is, for all of those optimization scenarios, MSBA is among the two best optimizers (and in fact, it yields the best results for all of those problems). However, PSO and ABC cannot retain their acceptable quality for all of the considered problems. Also, PSO shows a weak performance for the third profile and ABC shows an inferior performance in the fourth and fifth optimization scenarios. The other important observation refers to the fast convergence behavior of MSBA. MSBA is not only able of outperforming the other optimizers, but also can show an acceptable exploration/exploitation behavior, and consequently, has a fast convergence speed. It is only in the third optimization scenario that MSBA stands in the second place with regard to the convergence speed. Table 1 lists the statistical results obtained by the execution of the meta-optimizers for 20 independent runs. To clearly visualize the performance of the optimizers, the statistical results are reported in terms of the standard deviation (std.), mean obtained value, min obtained value, and max obtained value. As it can be seen from Table 1, the mean performance of MSBA is better than the other algorithms. Furthermore, the obtained *std*. values indicate that the both MSBA and PSO have an acceptable robustness rate as they converge to the relatively unique solutions over the independent optimization runs.

The results of the meta-optimization experiment reveal that MSBA can be used as a powerful method for evolving the architecture of LLM. Hence, the considered E-LLM uses MSBA as its meta-optimizer. The results of optimization indicate that LLM with 2 hidden nodes with 3–5 architecture can afford the best prediction results. To make sure that the considered meta-optimization policy

Table 2
Analysis of the effect of layers and nodes on the performance of LLM.

LLM features	MSE	RMSE	MAPE	MDAPE	R^2
1st speed profile					
4–2–3	0.0515	0.2269	10.5533	57.3862	0.9426
2–2–5	0.0524	0.2290	10.7352	57.6316	0.9422
3–4–4	0.0502	0.2241	10.4790	56.7873	0.9453
3–5	**0.0501**	**0.2238**	**10.2574**	**54.9487**	**0.9468**
2–5–2–4	0.0508	0.2253	10.5282	55.4233	0.9440
2nd speed profile					
4–2–3	0.0334	0.1828	5.8215	26.8463	0.9437
2–2–5	0.0339	0.1840	5.8779	27.0362	0.9435
3–4–4	0.0328	0.1812	5.7931	26.6044	0.9456
3–5	**0.0322**	**0.1798**	5.6949	**26.3141**	**0.9472**
2–5–2–4	0.0325	0.1802	**5.6944**	25.9646	0.9465
3rd speed profile					
4–2–3	0.0315	0.1774	**8.6273**	38.9877	0.9792
2–2–5	0.0323	0.1798	8.7926	41.6610	0.9788
3–4–4	0.0312	0.1765	8.7085	**37.6601**	0.9793
3–5	**0.0308**	**0.1758**	8.6830	38.4777	**0.9797**
2–5–2–4	0.0316	0.1778	8.7794	39.4638	0.9792
4th speed profile					
4–2–3	0.1132	0.3340	15.3006	173.9580	0.7774
2–2–5	0.1124	0.3365	14.9605	170.2491	0.7807
3–4–4	0.1115	0.3353	14.9710	170.2070	0.7852
3–5	**0.1113**	**0.3258**	**14.6646**	**163.0288**	**0.7938**
2–5–2–4	0.1119	0.3345	14.4705	165.2509	0.7884
5th speed profile					
4–2–3	0.0344	0.1855	6.6994	31.9600	0.9577
2–2–5	0.0354	0.1880	6.7836	32.7133	0.9558
3–4–4	0.0340	0.1845	6.6812	31.4183	0.9581
3–5	**0.0336**	**0.1833**	**6.5609**	**31.2470**	**0.9593**
2–5–2–4	0.0340	0.1844	6.6918	32.7278	0.9586

Bold numbers show the most qualified results.

Table 3
Comparison of the performance of the rival predictors.

Predictors	MSE	RMSE	MAPE	MDAPE	R^2
1st speed profile					
E-LLM	**0.0501**	**0.2238**	**10.2574**	**54.9487**	**0.9468**
E-ELM	0.0509	0.2245	10.5287	55.4298	0.9462
ELM	0.0530	0.2302	10.7010	57.2932	0.9405
BPNN	0.0528	0.2297	10.7691	57.2123	0.9409
RBFNN	0.0527	0.2295	10.7248	57.2009	0.9410
AR	0.0571	0.2390	11.2402	62.6303	0.9340
ANFIS	0.0523	0.2287	10.5862	57.2825	0.9407
2nd speed profile					
E-LLM	**0.0322**	**0.1798**	5.6949	26.3141	**0.9472**
E-ELM	0.0326	0.1805	**5.6794**	**26.1174**	0.9464
ELM	0.0339	0.1840	5.8664	27.4541	0.9430
BPNN	0.0339	0.1841	5.9238	26.3134	0.9427
RBFNN	0.0335	0.1829	5.8228	26.7757	0.9438
AR	0.0344	0.1855	6.0146	26.8101	0.9412
ANFIS	0.0326	0.1804	5.7046	26.3932	0.9462
3rd speed profile					
E-LLM	**0.0308**	**0.1758**	8.6830	**38.4777**	**0.9797**
E-ELM	0.0314	0.1771	8.7279	38.8436	0.9791
ELM	0.0326	0.1806	8.7629	43.2178	0.9788
BPNN	0.0323	0.1797	8.6912	41.4519	0.9789
RBFNN	0.0324	0.1801	**8.6323**	40.5796	0.9787
AR	0.0347	0.1862	8.9772	47.0630	0.9771
ANFIS	0.0312	0.1764	8.7039	38.6950	0.9793
4th speed profile					
E-LLM	**0.1113**	0.3236	**14.6646**	**163.0288**	**0.7938**
E-ELM	0.1114	**0.3338**	15.2600	173.2470	0.7774
ELM	0.1131	0.3363	15.1720	169.9092	0.7759
BPNN	0.1333	0.3651	16.5251	178.2751	0.7196
RBFNN	0.1283	0.3582	15.4540	170.2315	0.7413
AR	0.1419	0.3767	17.0010	184.2615	0.6893
ANFIS	0.1116	0.3340	15.0129	167.7441	0.7839
5th speed profile					
E-LLM	0.0336	0.1833	**6.5609**	**31.2470**	**0.9593**
E-ELM	**0.0335**	**0.1831**	6.6998	32.1198	0.9592
ELM	0.0358	0.1892	6.7926	32.7657	0.9523
BPNN	0.0364	0.1908	7.0172	33.7094	0.9536
RBFNN	0.0352	0.1876	6.8026	32.6206	0.9563
AR	0.0353	0.1878	6.9017	32.0872	0.9561
ANFIS	0.0337	0.1836	6.5660	31.8809	0.9591

Bold numbers show the most qualified results.

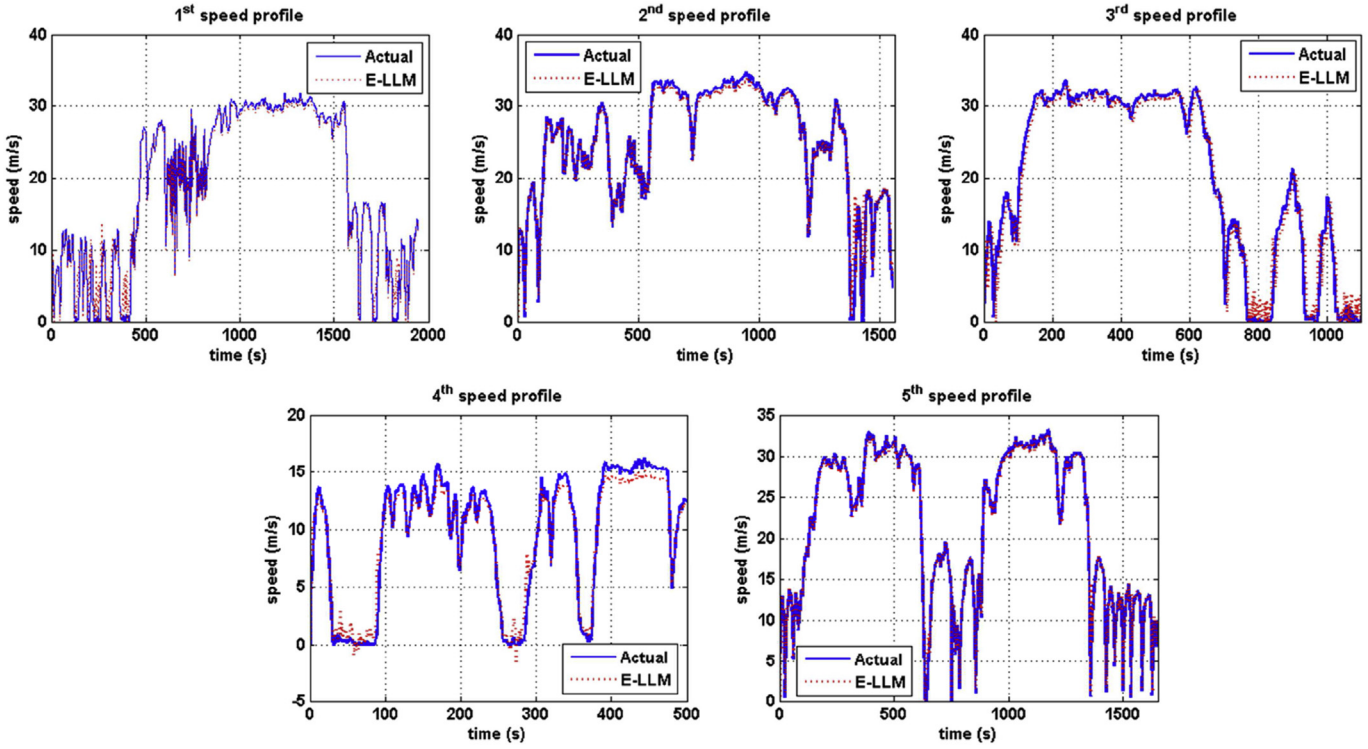

Fig. 7. The prediction capability of E-LLM.

Table 4
The prediction potentials of E-LLM for different prediction horizons.

E-LLM	MSE	RMSE	MAPE	MDAPE	R^2
1st speed profile					
$H_P = 10$	0.0501	0.2238	10.2574	54.9487	0.9468
$H_P = 12$	0.0556	0.2357	10.8811	75.1463	0.9356
$H_P = 14$	**0.0614**	**0.2477**	**11.3178**	**95.6694**	**0.9233**
$H_P = 16$	0.0648	0.2546	11.6167	116.7796	0.9157
$H_P = 18$	0.0685	0.2617	11.8876	139.0551	0.9077
$H_P = 20$	0.0717	0.2678	12.1616	162.9350	0.9002
2nd speed profile					
$H_P = 10$	0.0322	0.1798	5.6949	26.3141	0.9472
$H_P = 12$	0.0374	0.1935	6.2357	35.9852	0.9311
$H_P = 14$	**0.0425**	**0.2060**	**6.8459**	**47.6387**	**0.9127**
$H_P = 16$	0.0462	0.2149	7.1972	57.9305	0.8966
$H_P = 18$	0.0507	0.2252	7.6965	70.6010	0.8771
$H_P = 20$	0.0543	0.2331	8.0271	85.7020	0.8609
3rd speed profile					
$H_P = 10$	0.0308	0.1758	8.6830	38.4777	0.9797
$H_P = 12$	0.0361	0.1899	8.9149	52.6572	0.9731
$H_P = 14$	0.0411	0.2027	9.3285	72.4886	0.9667
$H_P = 16$	**0.0451**	**0.2125**	**9.5384**	**92.0856**	**0.9603**
$H_P = 18$	0.0494	0.2223	9.8980	112.4178	0.9527
$H_P = 20$	0.0532	0.2307	10.0496	132.7884	0.9461
4th speed profile					
$H_P = 10$	0.1113	0.3258	14.6646	163.0288	0.7938
$H_P = 12$	**0.1222**	**0.3495**	**15.2666**	**219.6018**	**0.7388**
$H_P = 14$	0.1381	0.3716	16.0476	275.1391	0.6866
$H_P = 16$	0.1495	0.3867	16.1277	335.2952	0.6500
$H_P = 18$	0.1614	0.4018	16.6770	404.1512	0.6067
$H_P = 20$	0.1716	0.4143	17.1430	450.5651	0.5577
5th speed profile					
$H_P = 10$	0.0336	0.1833	6.5609	31.2470	0.9593
$H_P = 12$	0.0379	0.1947	7.0849	42.8629	0.9484
$H_P = 14$	0.0421	0.2052	7.7319	54.2151	0.9365
$H_P = 16$	**0.0456**	**0.2135**	**8.1085**	**66.5465**	**0.9249**
$H_P = 18$	0.0501	0.2238	8.5307	84.9255	0.9124
$H_P = 20$	0.0538	0.2319	9.0122	99.7662	0.9003

Bold numbers show the most qualified results.

is reliable, the authors performed a sensitivity analysis and compared the prediction power of LLM with different architectures with that suggested by MSBA. Table 2 indicates the obtained results. For further assurance on the performance of the considered LLM architectures, the authors compare the prediction power of the rival LLMs in terms of all of the considered performance evaluation metrics. The obtained results indicate that the architecture proposed by MSBA is superior to the other LLMs. All to all, there are only two cases in which the meta-optimized architecture cannot show a superior performance; otherwise, none of the rival architectures can beat the LLM with 3–5 structural organization.

5.2. Comparing the rival predictors

After demonstrating the efficacy of the proposed meta-optimization scheme and ensuring the veracity of E-LLM, now it is the time to find out whether it can outperform the other well-known predictors. To avoid any biased conclusions, all of the rival predictors conduct the time series forecasting based on the concept of SWTS. Table 3 indicates the obtained results. As it can be seen, for the first scenario, E-LLM outperforms the other rival techniques. For this case, the performance of E-ELM is also acceptable compared to the other techniques. For the second and fifth speed profiles, it can be seen that E-LLM and E-LM outperform the other techniques. However, it can be seen that the performance of E-LLM is better than E-ELM. To be more precise, E-ELM only outperforms E-LLM in terms of MAPE and MDAPE for the second scenario, as well as MSE and RMSE for the fifth scenario. For the remaining metrics, the performance of E-LLM is superior. An interesting issue is observed in the third prediction scenario. It seems that RBFNN outperforms E-LLM in terms of MAPE. Also, the kernel-based feature transition of RBFFN can be a promising strategy for the prediction of the future speeds of the vehicle. As a general remark, it can be expressed that ELM, BPNN and RBFNN have a relatively equal

prediction power and the classical AR cannot show comparative results in comparison with its intelligent counterparts. ANFIS which works based on the integration of both fuzzification concept and neural computation also has an acceptable performance. As it can be seen, in the most of the testing scenarios, its performance is superior to BPNN, RBFNN, ELM and AR. However, it cannot outperform the E-LLM and E-ELM predictors.

The outcomes of the comparative study reveal that E-LLM can serve as a promising predictor for forecasting the future speeds of Honda Insight vehicle. In the next experiment, the authors scrutinize the prediction capabilities of E-LLM and evaluate its potentials to be used in the vehicle's predictive controller.

5.3. Studying the potentials of E-LLM for applications in the vehicle's predictive powertrain control unit

The time series prediction of E-LLM for all of the speed profiles is depicted in Fig. 7. As it can be seen, the prediction accuracy of E-LLM with respect to the actual vehicle speed time series is quite acceptable. The only non-negligible flaw of the predictor refers to the rare overshoots which take place when a significant change in the future speed values happens (especially, for the first speed profile). Otherwise, it can be seen that E-LLM accurately tracks the actual speed profile. As it was mentioned, the main role of the considered machine is to be embedded within the structure of a predictive powertrain controller. The controller employs the entire estimated speeds over a prediction horizon to calculate the control action sequence. Therefore, such prediction errors at over-shoot points won't cause a significant impact on the controller performance. Given the functionality of the devised intelligent system to enhance the vehicle fuel economy, and most importantly, considering its trivial cost compared to ITS-related infrastructures, and also, the way that the predictive controller is using the estimated speeds, its performance is acceptable.

In all of the previous numerical experiments, all of the simulations were conducted by H_P of 10. After ascertaining the acceptable performance of E-LLM, at this stage of the numerical experiments, the authors would like to apply it for predicting greater numbers of H_P to find out its maximum potentials. As it was mentioned, the admissible and logical range of prediction horizon is confined within the range of [10,20]. For the sake of comprehensive sensitivity analysis, here, the authors evaluate the prediction performance of E-LLM for H_P values of 10, 12, 14, 16, 18, and 20. This is very critical to realize the optimum value of prediction horizon as this parameter has a significant effect on the performance of MPC-based predictive controllers. To be more consistent, it should be tried to find the greatest possible value of H_P in which the prediction accuracy of E-LLM is not undermined. Table 4 lists the results of sensitivity analysis. As expected, by increasing the length of prediction horizon, the prediction accuracy of E-LLM is decreased. To select the maximum admissible length of the prediction horizon, the authors confine themselves to the criterion that the value of absolute fractions obtained by the predictors should not be significantly less than that obtained for the prediction horizon length of 10. The selected horizon lengths are indicated in a bold format. It seems that the prediction horizon length of 14 is an acceptable/logical choice. By considering the other performance evaluation metrics, one can discern that the differences of the prediction accuracy of E-LLM for the prediction lengths of 10 and 14 are not very different. However, from an MPC design point of view, having a predictor which enables us to detect the vehicle speeds for the next 14 s is very advantageous compared to the one which predicts the next 10 s.

As a final test, the authors extract the predicted values and actual speeds of the vehicle for a random segment. Fig. 8 compares the predicted speed profiles and the measured one for the next 14 s. As it can be seen, the obtained results are in a good agreement with

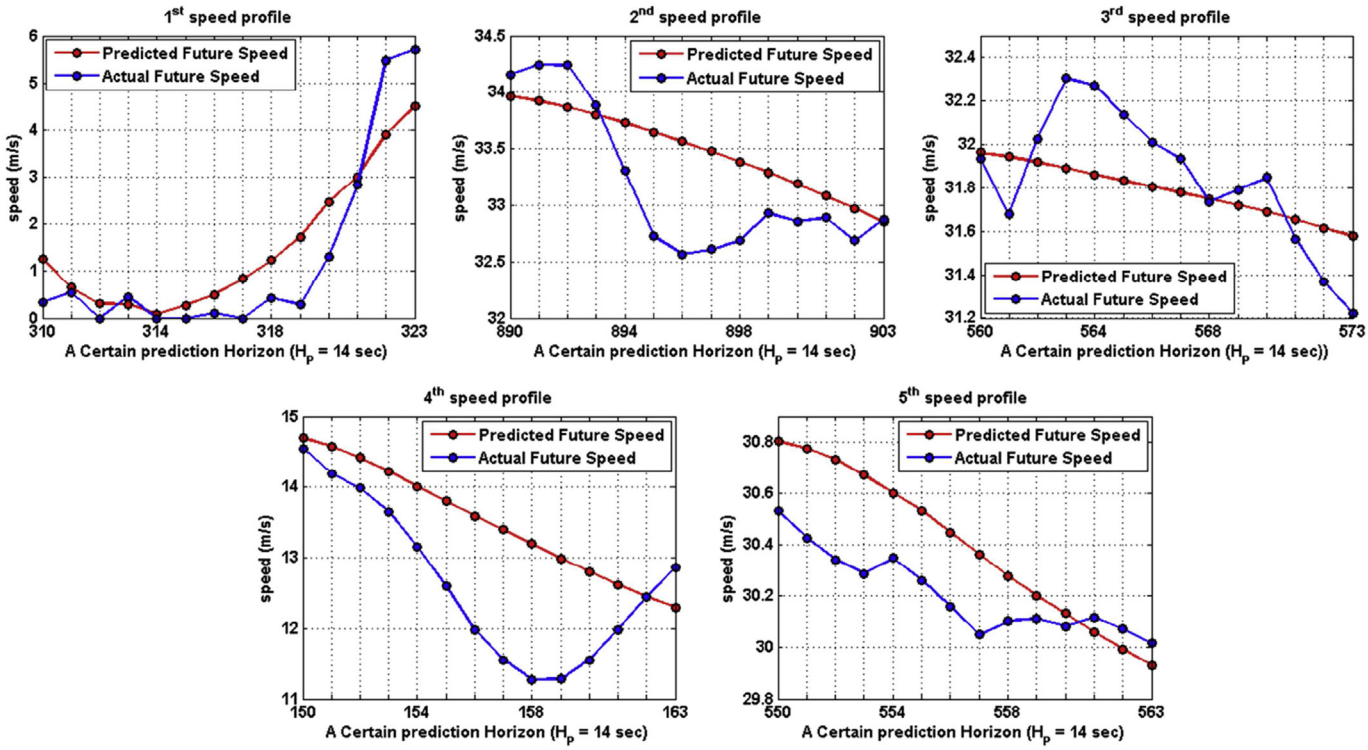

Fig. 8. Future speed profile predictions for a random 14 s segment.

each other and demonstrate that E-LLM can be used in the predictive powertrain control unit of the Honda Insight.

6. Conclusions

In this study, a hybrid intelligent predictive approach based on the mutable smart bee algorithm (MSBA) and the least learning machine (LLM) was presented to predict the vehicle speeds on the urban roads of San Francisco, California. The main objective of the proposed predictor was to correctly aid the predictive powertrain controller to ensure proper vehicle fuel economy and emission performance during the drive cycle. To this aim, the authors tried to find out whether the proposed E-LLM predictor is appropriate to be used as a time series predictor. Also, to ensure the proposed method is capable of being used as part of the predictive controller, the authors took advantage of the sliding window time series (SWTS)-based system implementation and derived a state-space representation. Based on the simulations, it was observed that SWTS enables us to derive an efficient time series predictor which can use the historical information of the vehicle speed and forecast the future vehicle speed profiles up to a predefined prediction horizon. Moreover, the comparative numerical experiments demonstrated that E-LLM can serve as a very powerful time series predictor, and also, can outperform several state-of-the-art methods with respect to the different performance evaluation metrics. The results of the current study draw the attention of automotive control engineers working within the realm of intelligent transportation systems (ITS) towards the fact that intelligent methods, for instance the proposed E-LLM, have good potentials to be used instead of the expensive telematics-based technologies used in smart vehicles.

Finally, it is worth to mention that the predicted speed profile has the most significant impact on the future vehicle power demands which should be known in advance by the powertrain controller to achieve a better performance in terms of the fuel economy. However, there are other pieces of route information, for instance the road grade, weather conditions and so on, which affect the future power demands. Certainly, having access to these additional variables, through measurements or predictions, will further improve the performance of the powertrain controller, but this was outside the scope of the current study. In future, the authors will demonstrate and discuss the structure and functioning of a predictive powertrain controller integrated with the devised speed prediction algorithm.

References

[1] M. Müller, M. Reif, M. Pandit, W. Staiger, B. Martin, Vehicle speed prediction for driver assistance systems, SAE Tech. Pap. (2004), 2004-01-0170.
[2] J.D. Gonder, Route-based control of hybrid electric vehicles, SAE World Congress, U. S. A. (2008).
[3] M. Vajedi, M. Chehresaz, N.L. Azad, Intelligent power management of plug-in hybrid electric vehicles, part II: real-time route based power management, Int. J. Electr. Hybrid Veh. 6 (1) (2014) 68–86.
[4] B. Abdulhai, H. Porwal, W. Recker, Short-term traffic flow prediction using neuro-genetic algorithms, Intell. Transp. Syst. J. 7 (1) (2002) 3–41.
[5] E.I. Vlahogianni, M.G. Karlaftis, J.C. Golias, Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach, Transp. Res. Part C Emerg. Technol. 13 (3) (2005) 211–234.
[6] X. Jiang, H. Adelim, Dynamic wavelet neural network model for traffic flow forecasting, J. Transp. Eng. 131 (10) (2005) 771–779.
[7] W. Zheng, D.H. Lee, Q. Shi, Short-term freeway traffic flow prediction: Bayesian combined neural approach, J. Transp. Eng. 132 (2) (2006) 114–121.
[8] K.Y. Chan, T.S. Dillion, J. Singh, E. Chang, Traffic Flow Forecasting Neural Networks Based on Exponential Smoothing Method, in: 6th IEEE Conference on Industrial Electronics and Applications, 2011, pp. 376–381. Beijing, China.
[9] K.Y. Chan, T.S. Dillion, S. Tharam, J. Singh, E. Chang, Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm, IEEE Trans. Intell. Transp. Syst. 13 (2) (2012) 644–654.
[10] A. Fotouhi, M. Montazeri-Gh, M. Jannatipour, Vehicle's velocity time series prediction using neural network, Int. J. Automot. Eng. 1 (1) (2011) 21–28.
[11] S. Wang, F.L. Chung, J. Wu, J. Wang, Least learning machine and its experimental studies on regression capability, Appl. Soft Comput. 21 (2014) 677–684.
[12] H.B. Brown, I. Nourbakhsh, C. Bartley, J. Cross, P. Dille, Charge car Community Conversions: Practical, Electric Commuter Vehicles Now!, Technical Report, Carnegie Mellon University, 2012.
[13] ChargeCar Project Website, (2012), 'http://www.chargecar.org/'
[14] M. Vafaeipour, O. Rahbari, M.A. Rosen, F. Fazelpour, P. Ansarirad, Application of sliding window technique for prediction of wind velocity time series, Int. J. Energy Environ. Eng. 5 (2014). Article ID: 105.
[15] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[16] A. Mozaffari, N.L. Azad, Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine coldstart hydrocarbon emission identification, Neurocomputing 131 (2014) 143–156.
[17] A. Mozaffari, M. Gorji-Bandpy, T.B. Gorji, Optimal design of constraint engineering systems: application of mutable smart bee algorithm, Int. J. Bio-Inspired Comput. 4 (3) (2012) 167–180.
[18] Z. Cui, Y. Zhang, Swarm intelligence in bio-informatics: methods and implementations for discovering patterns of multiple sequences, J. Nanosci. Nanotechnol. 14 (2) (2014) 1746–1757.
[19] A. Mozaffari, A. Ramiar, A. Fathi, Optimal design of classic Atkinson engine with dynamic specific heat using adaptive neuro-fuzzy inference system and mutable smart bee algorithm, Swarm Evol. Comput. 12 (2013) 74–91.
[20] J. Kennedy, R. Eberhart, Particel swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[21] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, Wiley, Chichester, London, 2001.
[22] D. Karaboga, Artificial bee colony, Scholarpedia 5 (3) (2010). Article ID: 6915.
[23] Q.Y. Zhu, A.K. Qin, P.N. Suganthan, G.B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (10) (2005) 1759–1763.
[24] H. Shu, L. Deng, P. He, Y. Liang, Speed prediction of parallel hybrid electric vehicles based on fuzzy theory, in: International Conference on Power and Energy Systems, Lecture Notes in Information Technology, vol. 13, 2012. Lecture Notes in Information Technology.
[25] A. Mahmoudabadi, Using artificial neural network to estimate average speed of vehicles in rural roads, Int. Conf. Intell. Network Comput. (2010) 25–30.
[26] J. Park, D. Li, Y.L. Murphey, J. Kristinsson, R. McGee, M. Kuang, T. Phillips, Real time vehicle speed prediction using a neural network traffic model, in: The 2011 International Joint Conference on Neural Networks, San Jose, 2011, pp. 2991–2996.
[27] J.G.D. Gooijer, R.J. Hyndman, 25 years of time series forecasting, Int. J. Forecast. 22 (2006) 443–473.
[28] R.J. Hyndman, A.B. Koehler, R.D. Snuder, S. Grose, A state space framework for automatic forecasting using exponential smoothing methods, Int. J. Forecast. 18 (2002) 439–454.
[29] J. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, Prediction intervals for exponential smoothing state space models, J. Forecast. 24 (2005) 17–37.
[30] E.P. George, G.M. Jenkis, G.C. Reinsel, Time Series Analysis: Forecasting and Control, John Wiley and Sons, 2008.