Applied Computational Intelligence and Soft Computing

Applied Neural Intelligence to Modeling, Control, and Management of Human Systems and Environments

Guest Editors: Toly Chen, P. Balasubramaniam, Quek Hiok Chai, and Yi-Chi Wang

Applied Neural Intelligence to Modeling, Control, and Management of Human Systems and Environments

Applied Neural Intelligence to Modeling, Control, and Management of Human Systems and Environments

Guest Editors: Toly Chen, P. Balasubramaniam, Quek Hiok Chai, and Yi-Chi Wang

Editorial Board

Contents

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 595041, 2 pages
doi:10.1155/2012/595041

Editorial

Applied Neural Intelligence to Modeling, Control, and Management of Human Systems and Environments

Toly Chen,[1] P. Balasubramaniam,[2] Quek Hiok Chai,[3] and Yi-Chi Wang[1]

[1] *Department of Industrial Engineering and Systems Management, Feng Chia University, Taichung 407, Taiwan*
[2] *Department of Mathematics, Gandhigram Rural University, Gandhigram 624302, India*
[3] *School of Computer Engineering, Nanyang Technological University, Singapore 639798*

Correspondence should be addressed to Toly Chen, tcchen@fcu.edu.tw

Received 22 August 2012; Accepted 22 August 2012

Planning, scheduling, and the control of resources and activities are key elements to survive and compete. Humans play a critical role in these activities. Actually, most systems could be viewed as an environment surrounded by human beings. Over the years, some novel solutions have been proposed to solve the problems of human systems and environments in modeling, control, and management. Advanced computing systems and artificial neural network approaches continue to be one of the most promising solutions.

The purpose of this *special issue* was to provide details on the development of advanced artificial neural network approaches and their applications to modeling, control, and the management of human systems and environments. The target audiences were researchers in information management, system engineering, environmental protection, as well as practicing managers and engineers. After a strict review, five articles from researchers around the world were finally accepted.

Predicting the stock market is an important facet of financial forecasting, attracting great interest from stock buyers and sellers, investors, policy makers, applied researchers, and many others who are involved in the capital market. Neural networks have been used extensively for stock market forecasting. S. Banik, M. Anwer, and M. K. Khan conducted a comparative study to predict the stock index values using soft computing models and a time series model. They used well-known models such as the genetic algorithm (GA) model and the adaptive network fuzzy integrated system (ANFIS) model as soft computing forecasting models, while considering the generalized autoregressive conditional heteroscedastic (GARCH) model as a time series model. The experimental results showed that the use of soft computing models is more successful than the time series model.

S. K. Boddhu and J. C. Gallagher described a novel frequency grouping based analysis technique, developed to qualitatively decompose the evolved controllers into explainable functional control blocks. They also provided a summary of their previous work related to evolving flight controllers for two categories of controllers and demonstrated the applicability of the newly developed decomposition analysis for both categories. Their proposed methodology has been successfully applied to autonomous and nonautonomous controllers, and it has been demonstrated that the methodology can indeed be used to decompose the evolved controllers into logically explainable control blocks for further control analysis.

K. H. Lim, K. P. Seng, and L.-M. Ang developed the Lyapunov theory-based radial basis function neural network (RBFNN) for traffic sign recognition. Their methodology, inserted multidimensional inputs into the RBF nodes, linked to multiple weights. An iterative weight adaptation scheme was then designed based on the Lyapunov-stability theory to obtain a set of optimum weights. After comparing the performances of the proposed classifier to some existing conventional techniques, the simulation results revealed that the proposed system achieved a better performance with a lower number of training iterations.

Drought forecasts can be an effective tool for mitigating some of the more adverse consequences of drought. A. M. Belayneh and J. F. Adamowski compared the effectiveness of three data driven models for forecasting drought conditions in the Awash River Basin of Ethiopia. The Standard

Precipitation Index (SPI) was forecasted using an artificial neural network (ANN), support vector regression (SVR), and the wavelet neural network (WN), and the performances were compared afterwards. The forecasting results indicated that the coupled wavelet neural network (WN) models were the best models for forecasting SPI values over multiple lead times in the Awash River Basin in Ethiopia.

T. Chen proposed a nonlinear programming and artificial neural network approach to optimize the performance of a job dispatching rule in a wafer fabrication factory. The proposed methodology fused two existing rules and constructed a nonlinear programming model for choosing the best values of the parameters in the two rules by dynamically maximizing the standard deviation of the slack, which has been shown in several studies to benefit the scheduling performance. In addition, a more effective approach was applied to estimate the remaining cycle time of a job, which was empirically shown to be conducive to improve the scheduling performance. Based on the experimental results, the optimization of the adjustable factors in the two rules is an appropriate tool for enhancing the scheduling performance of the dispatching rule.

Acknowledgment

We thank all the authors and reviewers for their valuable contributions to this special issue.

Toly Chen
P. Balasubramaniam
Quek Hiok Chai
Yi-Chi Wang

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 794061, 13 pages
doi:10.1155/2012/794061

Research Article

Standard Precipitation Index Drought Forecasting Using Neural Networks, Wavelet Neural Networks, and Support Vector Regression

A. Belayneh and J. Adamowski

Department of Bioresource Engineering, Faculty of Agricultural and Environmental Sciences, McGill University, QC, Canada H9X 3V9

Correspondence should be addressed to J. Adamowski, jan.adamowski@mcgill.ca

Received 24 February 2012; Accepted 18 July 2012

Academic Editor: Quek Hiok Chai

Drought forecasts can be an effective tool for mitigating some of the more adverse consequences of drought. Data-driven models are suitable forecasting tools due to their rapid development times, as well as minimal information requirements compared to the information required for physically based models. This study compares the effectiveness of three data-driven models for forecasting drought conditions in the Awash River Basin of Ethiopia. The Standard Precipitation Index (SPI) is forecast and compared using artificial neural networks (ANNs), support vector regression (SVR), and wavelet neural networks (WN). SPI 3 and SPI 12 were the SPI values that were forecasted. These SPI values were forecast over lead times of 1 and 6 months. The performance of all the models was compared using RMSE, MAE, and R^2. The forecast results indicate that the coupled wavelet neural network (WN) models were the best models for forecasting SPI values over multiple lead times in the Awash River Basin in Ethiopia.

1. Introduction

Droughts, a natural occurrence in almost all climatic zones, are a result of the reduction, for an extended period of time, of precipitation from normal amounts. Extended periods of drought can lead to several adverse consequences, which include a disruption of the water supply, low agricultural yields, and reduced flows for ecosystems. Consequently, the ability to forecast and predict the characteristics of droughts, specifically their initiation, frequency, and severity, is important. Effective drought forecasts are an effective tool for water resource management as well as an effective tool for the agricultural industry.

Currently, drought monitoring in Ethiopia is conducted by the National Meteorological Services Agency (NMSA). The NMSA regularly produces a 10-day bulletin that gives an analysis of rainfall based on the long-term average or normal. This bulletin is then circulated to a wide range of users, ranging from local development agents to decision makers at a national level. In addition to rainfall analysis, the normalized vegetation index (NDVI) is provided, which is a satellite-based index widely used to monitor vegetation and drought conditions. The NMSA produces a regular 10-day bulletin regarding NDVI variation that compares the current vegetation condition with normal or conditions of the previous year [1]. However, the NDVI is sensitive to changes in vegetative land cover and may not be effective in areas where vegetation is minimal. In addition, the NMSA of Ethiopia produces medium and seasonal forecasts of precipitation using the aforementioned NDVI.

Unlike other natural hazards, droughts have a slow evolution time [2]. The consequences of droughts take a significant amount of time to come into effect with respect to their inception, and when they are perceived by ecosystems and hydrological systems. Due to this feature, effective mitigation of the most adverse drought impacts is possible, more than in the case of other extreme hydrological events such as floods, earthquakes, or hurricanes, provided

a drought monitoring system, which is able to promptly warn of the onset of a drought and to follow its evolution in space and time, is in operation [3].

A common tool utilized to monitor current drought conditions is a drought index. Several drought indices can be used to forecast the possible evolution of an ongoing drought, in order to adopt appropriate mitigation measures and drought policies for water resources management [4]. This is because a drought index is expressed by a numeric number, which is believed to be far more functional than raw data during decision-making [2]. Several drought indices have been developed around the world in the past based on rainfall as the single variable, including the widely used Deciles [5], Standardized Precipitation Index (SPI) [6], and Effective Drought Index (EDI) [7]. There is also the well-known Palmer Drought Severity Index (PDSI) [8], which considers temperature along with rainfall. The SPI drought index was chosen to forecast drought in this study due to its simplicity, its ability to represent droughts on multiple time scales, and because it is a probabilistic drought index. In addition, the study by Ntale and Gan [9] determined that the SPI is the most appropriate index for monitoring the variability of droughts in East Africa because it is easily adapted to local climate, has modest data requirements, and can be computed at almost any time scale.

Forecasting any hydrologic phenomena can be done using either a physical, conceptual, or data-driven approach. The latter approach is widely used in hydrologic forecasting because data-driven models have low information requirements with respect to the number of variables required for inputs compared to physically based models. Data-driven models also have rapid development times. Unlike physical and conceptual models, data-driven models are not difficult to implement for the purposes of real-time forecasting. Artificial neural networks (ANNs) have been used in several studies as a drought-forecasting tool [10–16]. The most popular type of ANN used for the purposes of drought forecasting is the multilayer perceptron (MLP) that is usually optimized with a back propagation algorithm. However, ANNs are limited in their ability to deal with nonstationarities in the data, a weakness also shared by multiple linear regression (MLR) and autoregressive integrated moving average (ARIMA) models.

This limitation with nonstationary data has led to the recent formation of hybrid models, where data is preprocessed for nonstationary characteristics and then run through a forecasting method such as ANNs to cope with the nonlinearity. Wavelet analysis, an effective tool to deal with nonstationary data, has recently been applied in hydrological forecasting to examine the rainfall-runoff relationship in a Karstic watershed [17], to characterize daily streamflow [18, 19] and monthly reservoir inflow [20], to evaluate rainfall-runoff models [21], to forecast river flow [22–24], to forecast future precipitation values [25], and for the purposes of drought forecasting [26]. The study conducted by Kim and Valdes [26] is the only study that has explored the ability of a wavelet-neural network conjunction model (WN) to forecast a given drought index. However, no studies that assess the ability of WN models to forecast the SPI drought index in particular have been explored.

Support Vector Machines (SVMs) are a relatively new form of machine learning that was developed by Vapnik [27]. The term SVM is used to refer to both classification and regression methods as well as the terms Support Vector Classification (SVC) and Support Vector Regression (SVR), which refer to the problems of classification and regression, respectively [28]. There are several studies where SVRs were used in hydrological forecasting. Khan and Coulibaly [29] found that an SVR model was more effective at predicting 3–12 month lake water levels than ANN models. Rajasekaran et al. [30] used SVR successfully for storm surge predictions, and Kisi and Cimen [31, 32] used SVR to estimate daily evaporation and daily streamflow, respectively. Finally, SVR have been successfully used to predict hourly streamflow by Asefa et al. [33] and were shown to perform better than ANN and ARIMA models for monthly streamflow prediction by Wang et al. [34] and Maity et al. [35], respectively. Yuan and Tan [36] used SVRs as a screening tool to test for drought resistance of rice. However, to date SVRs have not been applied to forecast a given drought index.

This study compared the effectiveness of three data-driven models for forecasting drought conditions in the Awash River Basin of Ethiopia. The Standard Precipitation Index (SPI) was forecasted and compared using artificial neural networks (ANNs), support vector regression (SVR), and wavelet networks (WN). SPI 3 and SPI 12 were forecast over lead times of 1 and 6 months. The forecast lead times were chosen because a 1-month lead time is a typical short-term lead time and a 6-month lead time is representative of the bimodal rainfall pattern in the Awash River Basin. Forecast results of this study are useful for the agricultural water management sector and have the potential to be applied by water resources managers to effectively manage water resources in the region. In addition, accurate forecasts using these data-driven models can complement the forecasts already being used by the NMSA of Ethiopia.

2. Theoretical Development

In the following section, the computation of the SPI is briefly described. In addition to the description of the SPI, this section also describes the data-driven models that were used to forecast the SPI.

2.1. The Standard Precipitation Index (SPI). The Standard Precipitation Index (SPI) was developed by McKee et al. [6]. As mentioned in the previous section, one of the main advantages of the SPI is that it only requires precipitation data as an input, which makes it ideal for areas where data collection is not as extensive (such as in Ethiopia). The fact that the SPI is based solely on precipitation makes its evaluation relatively easy [37]. The SPI is a standardized index. Standardization of a drought index ensures independence from geographical position as the index in question is calculated with respect to the average precipitation in the same place [37].

Table 1: Drought classification based on SPI [6].

SPI values	Class
>2	Extremely wet
1.5–1.99	Very wet
1.0–1.49	Moderately wet
−0.99 to 0.99	Near normal
−1 to −1.49	Moderately dry
−1.5 to −1.99	Very dry
<−2	Extremely dry

The computation of the SPI drought index for any location is based on the long-term precipitation record (at least 30 years) cumulated over a selected time scale [38]. This long-term precipitation time series is then fitted to a gamma distribution, which is then transformed through an equal probability transformation into a normal distribution [38, 39]. Positive SPI values indicate wet conditions with greater than median precipitation, and negative SPI values indicate dry conditions with lower than median precipitation [38]. Table 1 below indicates SPI drought classes.

In most cases, the probability distribution that best models observational precipitation data is the Gamma distribution [37]. The density probability function for the Gamma distribution is given by the expression [37]:

$$g(x) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}, \quad \text{for } x > 0, \tag{1}$$

where $\alpha > 0$ is the shape parameter, $\beta > 0$ is the scale parameter, and $x > 0$ is the amount of precipitation. $\Gamma(\alpha)$ is the value taken by the standard mathematical function known as the Gamma function, which is defined by the integral [37]:

$$\Gamma(\alpha) = \int_0^{\infty} y^{\alpha-1} e^{-y} dy. \tag{2}$$

In general, the Gamma function is evaluated either numerically or using the values tabulated depending on the value taken by parameter α.

In order to model the data observed with a gamma distributed density function, it is necessary to estimate parameters α and β appropriately. Different methods have been suggested in the literature for the estimate of these two parameters. For example, the Thom [40] approximation is used for maximum probability in Edwards and McKee [41]:

$$\hat{\alpha} = \frac{1}{4A}\left(1 + \sqrt{1 + \frac{4A}{3}}\right), \qquad \hat{\beta} = \frac{\overline{x}}{\hat{\alpha}}, \tag{3}$$

where for n observations

$$A = \ln(\overline{x}) - \frac{\sum \ln(x)}{n}. \tag{4}$$

The estimate of the parameters can be further improved by using the interactive approach suggested in Wilks [42].

After estimating coefficients α and β the density of probability function $g(x)$ is integrated with respect to x and we obtain an expression for cumulative probability $G(x)$ that a certain amount of rain has been observed for a given month and for a specific time scale [37]:

$$G(x) = \int_0^x g(x)dx = \frac{1}{\hat{\beta}\Gamma(\hat{\alpha})} = \int_0^x x^{\hat{\alpha}-1} e^{-x/\beta} dx. \tag{5}$$

The Gamma function is not defined by $x = 0$, and since there may be no precipitation, the cumulative probability becomes [37]

$$H(x) = q + (1 - q)G(x), \tag{6}$$

where q is the probability of no precipitation. $H(x)$ is the cumulative probability of precipitation observed. The cumulative probability is then transformed into a normal standardized distribution with null average and unit variance from which we obtain the SPI index.

The above approach, however, is neither practical nor numerically simple to use if there are many grid points of many stations on which to calculate the SPI index. In this case, an alternative method is described in Edwards and McKee [41] using the technique of approximate conversion developed in Abramowitz and Stegun [43] that converts the cumulative probability into a standard variable Z. The SPI index is then defined as

$$Z = \text{SPI} = \begin{cases} -\left(t - \dfrac{c_0 + c_1 t + c_2 t_2}{1 + d_1 t + d_2 t_2 + d_3 t_3}\right), & \text{for } 0 < H(x) \le 0.5, \\ +\left(t - \dfrac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}\right), & \text{for } 0.5 < H(x) < 1, \end{cases} \tag{7}$$

where

$$t = \begin{cases} \sqrt{\ln\left[\dfrac{1}{(H(x))^2}\right]}, & \text{for } 0 < H(x) \le 0.5, \\ \sqrt{\ln\left[\dfrac{1}{(1 - H(x))^2}\right]}, & \text{for } 0.5 < H(x) < 1, \end{cases} \tag{8}$$

where x is precipitation, $H(x)$ is the cumulative probability of precipitation observed, and c_0, c_1, c_2, d_0, d_1, d_2 are constants with the following values:

$$c_0 = 2.515517, \quad c_1 = 0.802853, \quad c_2 = 0.010328, \quad d_0 = 1.432788, \quad d_1 = 0.189269, \quad d_2 = 0.001308. \tag{9}$$

2.2. Artificial Neural Networks (ANNs). Artificial neural networks (ANNs) are flexible computing frameworks that resemble the structure of a nerve system. ANNs have been used to model a broad range of hydrologic time series over the past two decades. The main advantage of using ANNs is that there is no need to define the physical processes between the inputs

and outputs [11]. This feature makes ANNs suitable for the purposes of drought forecasting, where all the variables that may cause a drought are not fully understood.

In this paper, the multilayer perceptron (MLP) feed-forward network was used to forecast the SPI time series. Figure 1 is an illustration of a typical feed-forward neural network. ANN models in this study were trained with the Levenberg Marquardt (LM) back propagation algorithm. MLPs have been used extensively in hydrologic forecasting studies [10, 12, 23, 26, 44, 45] due to their simplicity. In terms of their architecture, MLPs consist of an input layer, one or more hidden layers, and an output layer. The hidden layer contains the neuron-like processing elements that connect the input and output layers and is given by [26]

$$y'_k(t_s) = f_0\left[\sum_{j=1}^{m} w_{kj}.f_n\left(\sum_{i=1}^{n} w_{ji}x_i(t_s) + \left(w_{j0}\right)\right) + w_{k0}\right], \tag{10}$$

where n is the number of input variables; m is the number of hidden neurons; $x_i(t)$ = the ith input variable at time step t_s; w_{ji} = weight that connects the ith neuron in the input layer and the jth neuron in the hidden layer; w_{j0} = bias for the jth hidden neuron; f_n = activation function of the hidden neuron; w_{kj} = weight that connects the jth neuron in the hidden layer and kth neuron in the output layer; w_{k0} = bias for the kth output neuron; f_0 = activation function for the output neuron; $y'_k(t_s)$ is the forecasted kth output at time step t_s [26].

2.3. Support Vector Regression. Support vector machines (SVM) were developed by Vapnik [27] as a tool for classification and regression. SVMs embody the structural risk minimization principle, while neural networks embody the empirical risk minimization principle. In contrast to ANNs that seek to minimize training error, SVMs attempt to minimize the generalization error. SVMs have two components: support vector classification (SVC) and support vector regression (SVR). Since the main objective of this study is to forecast the SPI, the SVR was used.

Support vector regression (SVR) is used to describe regression with SVMs [27]. In regression estimation with SVR, the purpose is to estimate a functional dependency $f(\vec{x})$ between a set of sampled points $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_l\}$ taken from R^n and target values $Y = \{y_1, y_2, \ldots, y_l\}$ with $y_i \in R$ (the input and target vectors (x_i's and y_i's) refer to the monthly records of the SPI index). Assuming that these samples have been generated independently from an unknown probability distribution function $P(\vec{x}, y)$ and a class of functions [27]:

$$F = \left\{f \mid f(\vec{x}) = \left(\vec{W}, \vec{x}\right) + B : \vec{W} \in R^n, R^n \longrightarrow R\right\}, \tag{11}$$

where $\vec{W}$ and B are coefficients that have to be estimated from the input data. The main objective is to find a function $f(\vec{x}) \in F$ that minimizes a risk functional [46]:

$$R[f(\vec{x})] = \int l(y - f(\vec{x}), \vec{x})\, dP(\vec{x}, y), \tag{12}$$

where l is a loss function used to measure the deviation between the target, y, and estimate $f(\vec{x})$, values. As the probability distribution function $P(\vec{x}, y)$ is unknown, one cannot minimize the risk functional directly, but can only compute the empirical risk function as [46]

$$R_{\text{emp}}[f(\vec{x})] = \frac{1}{N}\sum_{i=1}^{N} l(y_1 - f(\vec{x}_i)), \tag{13}$$

where N is the number of samples. This traditional empirical risk minimization is not advisable without any means of structural control or regularization. To avoid this issue a regularized risk function with the smallest steepness among the functions that minimize the empirical risk function can be used as [46]

$$R_{\text{reg}}[f(\vec{x})] = R_{\text{emp}}[f(\vec{x})] + \gamma\left\|\vec{W}\right\|^2, \tag{14}$$

where γ is a constant ($\gamma \geq 0$). This additional term reduces the model space and thereby controls the complexity of the solution resulting in the following form of this expression [46, 47]:

$$R_{\text{reg}}[f(\vec{x})] = C_c \sum_{x_i \in X} l_\varepsilon(y_1 - f(\vec{x}_i)) + \frac{1}{2}\left\|\vec{W}\right\|^2, \tag{15}$$

where C_c is a positive constant that has to be selected beforehand. The constant C_c that influences a trade-off between or an approximation error and the regression (weight) vector $\|\vec{W}\|$ is a design parameter. The loss function in this expression, which is called an ε-insensitive loss function (l_ε), has the advantage that it will not need all the input data for describing the regression vector $\|\vec{W}\|$and can be written as [46]

$$l_\varepsilon(y_1 - f(\vec{x}_i)) = \begin{cases} 0, & \text{for} \mid y_1 - f(\vec{x}_i) \mid < \varepsilon \\ y_1 - f(\vec{x}_i), & \text{otherwise.} \end{cases} \tag{16}$$

This function behaves as a biased estimator when it is combined with the regularization term ($\gamma\|\vec{W}\|^2$). The loss is equal to 0 if the difference between the predicted and observed value is less than ε. The nonlinear regression function is described by the following expression [27, 46, 48]:

$$f(x) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(x, x_i) + B, \tag{17}$$

where $\alpha_i, \alpha_1^* \geq 0$ are the Lagrange multipliers, B is a bias term, and $K(x, x_i)$ is the Kernel function which is based upon Reproducing Kernel Hilbert Spaces [32]. The Kernel function enables operations to be performed in the input space as opposed to the potentially high-dimensional feature space. Several types of functions are treated by SVR such as polynomial functions, Gaussian radial basis functions, exponential radial basis functions, multilayer perception functions, and functions with splines and so forth [32].

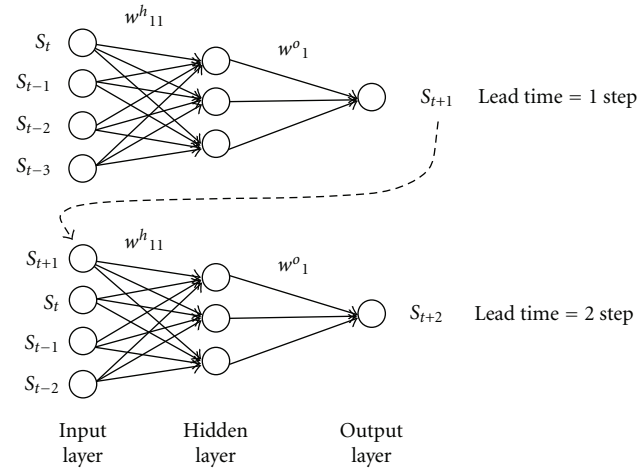

Figure 1: Typical Feed-forward Neural Network.

2.4. Wavelet Transforms. Wavelet transforms are mathematical functions that can be used for the analysis of time-series that contain nonstationarities. Wavelet transforms allow for the use of long time intervals for low frequency information and shorter intervals for high frequency information. They are capable of revealing aspects of data like trends, breakdown points, and discontinuities that other signal analysis techniques might miss [26]. Another advantage of wavelet analysis is the flexible choice of the mother wavelet according to the characteristics of the investigated time series [45].

An important step in the use of wavelet transforms is the choice of a mother wavelet (ψ). The continuous wavelet transform (CWT) is defined as the sum over all time of the signal multiplied by scale and shifted versions of the wavelet function ψ [26]:

$$W(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} x(t)\psi^* \left(\frac{t-\tau}{s} \right) dt, \tag{18}$$

where s is the scale parameter; τ is the translation and $*$ corresponds to the complex conjugate [26]. The CWT produces a continuum of all scales as the output. Each scale corresponds to the width of the wavelet; hence, a larger-scale means that more of a time series is used in the calculation of the coefficient than in smaller scales. The CWT is useful for processing different images and signals; however, it is not often used for forecasting because its computation is complex and time consuming. As an alternative, in forecasting applications, the discrete wavelet transform (DWT) is used, due to its simplicity and shorter computation time. DWT scales and positions are usually based on powers of two (dyadic scales and positions). This is achieved by modifying the wavelet representation to [49]

$$\psi_{j,m}(m) = \frac{1}{\sqrt{\left| s_0^j \right|}} \sum_k \psi \left(\frac{k - m\tau_0 s_0^j}{s_0^j} \right) x(k), \tag{19}$$

where j and m are integers that control the scale and translation, respectively, while $s_0 > 1$ is a fixed dilation step and τ_0 is a translation factor that depends on the aforementioned dilation step. The effect of discretizing the wavelet is that the time-space scale is now sampled at discrete levels. The DWT operates two sets of functions: high-pass and low-pass filters. The original time series is passed through high-pass and low-pass filters, and detailed coefficients and approximation series are obtained.

One of the inherent challenges of using the DWT for forecasting applications is that if we change values at the beginning of our time series, all of the wavelet coefficients will subsequently change. To overcome this problem, a redundant algorithm, known as the à trous algorithm can be used, given by [50]

$$C_{i+1}(k) = \sum_{l=-\infty}^{+\infty} h(l) c_i \left(k + 2^i l \right), \tag{20}$$

where h is the low pass filter and the finest scale is the original time series. To extract the details, $w_i(k)$, that were eliminated in (21), the smoothed version of the signal is subtracted from the coarser signal that preceded it, given by [51]

$$w_i(k) = c_{i-1}(k) - c_i(k), \tag{21}$$

where $c_i(k)$ is the approximation of the signal and $c_{i-1}(k)$ is the coarser signal. Each application of (20) and (21) creates a smoother approximation and extracts a higher level of detail. Finally, the nonsymmetric Haar wavelet can be used as the low pass filter to prevent any future information from being used during the decomposition [52].

3. The Awash River Basin

This study forecasted the SPI in the Awash River Basin of Ethiopia. The mean annual rainfall of the basin varies from about 1,600 mm in the highlands north east of Addis Ababa, to 160 mm in the northern point of the basin [53]. The total amount of rainfall also varies greatly from year to year, resulting in severe droughts in some years and flooding in

others. The total annual surface runoff in the Awash Basin amounts to some $4,900 \times 106\,m^3$ [54].

The Awash River Basin (Figure 2) was separated into three smaller basins for the purpose of this study on the basis of various factors such as location, altitude, climate, topography, and agricultural development. A study conducted by Edossa et al. [54] separated the Awash Basin in a similar fashion. The subbasins were called the Upper, Middle, and Lower Awash Basins, respectively. The reasoning behind the use of these three subbasins was to ensure the methods used in this study were effective in forecasting short-term drought in different conditions. The characteristics of each sub-basin are briefly described in the following sections.

3.1. Upper Awash Basin. The Upper Awash Basin has a temperate climate with annual mean temperatures ranging between 15–22°C and an annual precipitation of between 500–2000 mm [54]. Rainfall distribution in the Upper Awash Basin is unimodal. Seven rainfall gauges located in the Upper Awash River Basin were chosen for this study (Table 2). These stations were chosen because their precipitation records from 1970–2005 were either complete or relatively complete. Any station, which had over 10% of their records missing was not selected.

3.2. Middle Awash Basin. The Middle Awash Basin is in the semiarid climatic zone with a long hot summer and a short mild winter. Annual rainfall varies between 200–1500 mm [54]. The rainfall distribution is bimodal in this subbasin. Minor rains normally occur in March and April and major rains from July to August. Eight rainfall gauges located in the Middle Awash Basin were selected using the same criteria as in the Upper Awash Basin and are shown in Table 2.

3.3. Lower Awash Basin. The Lower Awash River Basin has a hot, semi-arid climate. The annual mean temperature of the region ranges between 22 and 32°C with average annual precipitation between 500 and 700 mm [54]. Five rainfall gauges were selected form the Lower Awash Basin using the same criteria used in the two other sub-basins and are shown in Table 2.

4. Methodology

The methodology section of this paper describes how the SPI was calculated and then forecast over two separate lead times using ANN, WN, and SVR models.

4.1. SPI Calculation. In order to calculate the SPI, a probability density function that adequately describes the precipitation data must be determined. The gamma distribution function was selected to fit the raw rainfall data from each station in this study. The SPI is a z-score and represents an event departure from the mean, expressed in standard deviation units. The SPI is a normalized index in time and space. SPI values can be categorized according to classes. In this study, the near normal class is established from the aggregation of two classes: $-1 < \text{SPI} < 0$ (mild drought) and $0 \leq \text{SPI} \leq 1$ (slightly wet). The departure from the mean is a probability indication of the severity of the wetness or drought that can be used for risk assessment. The time series of the SPI can be used for drought monitoring by setting application-specific thresholds of the SPI for defining drought beginning and ending times. Accumulated values of the SPI can be used to analyze drought severity. In this study, the SPI_SL_6 program developed by the National Drought Mitigation Centre, University of Nebraska-Lincoln, was used to compute time series of drought indices (SPI) for each station in the basin and for each month of the year at different time scales.

In each sub-basin, for each station, SPI 3 and SPI 12 were computed. These SPI values were subsequently forecast over lead times of 1 and 6 months. A 3-month SPI compares the precipitation for that period with the same 3-month period over the historical record. For example, a 3-month SPI at the end of September compares the precipitation total for the July–September period with all the past totals for that same period. A 3-month SPI indicates short and medium term trends in precipitation and is still considered to be more sensitive to conditions at this scale than the Palmer Index. A 3-month SPI can be very effective in showing seasonal trends in precipitation and is a good indicator of agricultural drought. SPI 12 reflects long-term precipitation patterns. SPI 12 is a comparison of the precipitation for 12 consecutive months with the same 12 consecutive months during all the previous years of available data and is a good indicator of long-term drought conditions. Because these time scales are the cumulative result of shorter periods that may be above or below normal, the longer SPIs tend toward zero unless a specific trend is taking place. Forecast lead times of 1 and 6 months were chosen because 1 month is the shortest possible monthly lead time and 6 months is representative of the bimodal rainfall pattern in parts of the Awash River Basin discussed in Section 3.2.

4.2. Wavelet Decomposition. In the proposed WN model, the SPI data for each of the rainfall stations was decomposed into subseries of approximations and details (DWs). The process consists of a number of successive filtering steps. The original SPI time series is first decomposed into an approximation and accompanying detail signal. The decomposition process is then iterated, with successive approximation signals being decomposed in turn. As a result the original SPI time series is broken down into many lower resolution components.

When conducting wavelet analysis, the number of decomposition levels that is appropriate for the data must be chosen. A commonly used method to determine the number of decomposition levels is based on the signal length [55] and is given by $L = \text{int}[\log(N)]$, where L is the level of decomposition and N is the length of the signal. The training set in this study comprised between 1290 and 3017 samples (samples varied depending on the number of inputs for each rainfall station). Thus, the decomposition level was selected as $L = 3$.

As discussed in Section 2.4, the "a trous" wavelet algorithm with a low pass Haar filter was used to create four

Table 2: Descriptive statistics for the Awash River Basin.

Basin	Station	Mean annual precipitation (mm)	Max annual (1970–2005) precipitation (mm)	Standard deviation (mm)
Upper Awash Basin	Bantu Liben	91	647	111
	Tullo Bullo	94	575	114
	Ginchi	97	376	90
	Sebeta	111	1566	172
	Ejersalele	67	355	75
	Ziquala	100	583	110
	Debre Zeit	73	382	81
Middle Awash Basin	Koka	97	376	90
	Modjo	76	542	92
	Nazereth	73	470	85
	Wolenchiti	76	836	95
	Gelemsso	77	448	75
	Hirna	78	459	86
	Dire Dawa	51	267	54
	Meisso	61	361	61
Lower Awash Basin	Dubti	15	192	23
	Eliwuha	44	374	57
	Mersa	87	449	89
	Mille	26	268	40
	Bati	73	357	80

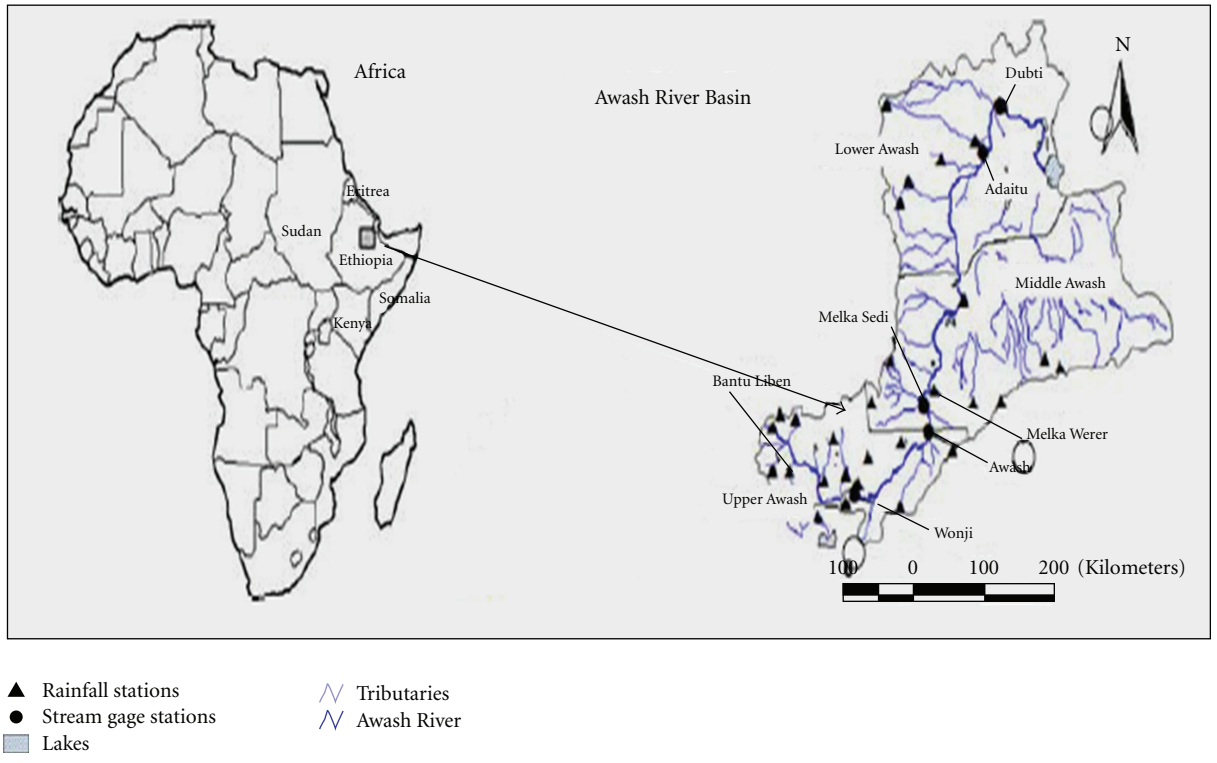

Figure 2: Awash River Basin (Source: [54]).

sets of wavelet subseries. These four sub-series included a low frequency component (the approximation) used to uncover the trend of each signal and a set of three high frequency components (the details) used to uncover the periodicity of the signal. All decomposed sub-series were added together to generate one time series and used as an input to the ANN models. Using the sum of all the sub-series as an input in this study provided more accurate results than using certain sub-series or sub-series that exhibited the highest correlations with the original time series.

4.3. ANN Models. All the ANN models were created with the MATLAB (R.2010a) ANN toolbox. The hyperbolic tangent sigmoid transfer function was the activation function for the hidden layer, while the activation function for the output layer was a linear function. All the ANN models in this study were trained using the LM back propagation algorithm. The LM back propagation algorithm was chosen because of its efficiency and reduced computational time in training models [45].

In this study, there were between 4–8 input neurons for each ANN model. The optimal number of input neurons for each station was selected using a trial and error procedure. The data-driven models were recursive models, where a model is forecast one lead time ahead, and the subsequent forecasts include the output from the previous forecast as an input. Hence, a forecast of 6 months lead time will have the outputs from forecasts of lead times of 1–5 months. Recursive models were used because it was determined that it would be simpler to use an ANN with one output neuron. Mishra and Desai [10] compared recursive ANN models and ANN models with more than one output neuron (direct ANN models) and found the results to be comparable for forecasting the SPI. The inputs and outputs were normalized between 0 and 1. A study by Wanas et al. [56] empirically determined that the best performance of a neural network occurs when the number of hidden nodes is equal to $\log(T)$, where T is the number of training samples. Another study conducted by Mishra and Desai [10] determined that the optimal number of hidden neurons is $2n + 1$, where n is the number of input layers. In this study the optimal number of hidden neurons was determined to be between $\log(T)$ and $(2n + 1)$. For example, if using the method proposed by Wanas et al. [56] gave a result of 4 hidden neurons and using the method proposed by Mishra and Desai [10] gave 6 hidden neurons, the optimal number of hidden neurons was between 4 and 6, thereafter the optimal number was determined using trial and error. These two methods helped establish an upper and lower bound for the number of hidden neurons.

For all the ANN models the cross validation technique [57] was used to partition the data sets; 80% of the data was used to train the models, while the remaining 20% of the data was used to test and validate the models, with 10% used for testing and 10% used for validation. The training set was used to compute the error gradient and to update the network weights and biases. The error from the validation set was used to monitor the training process. If the network overfits the data, the error in the validation set will begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned. The testing data set is an independent data set and is used to verify the performance of the model.

4.4. WN Models. The WN models were trained in the same way as the ANN models, with the exception that the inputs were made up from the wavelet decomposed subseries. In this study, the significant wavelets (approximation and detail series) were summed together once the insignificant coefficients were excluded, similar to what was done by Partal [58] and Kisi and Cimen [32]. In this study, the summed sub-series provided better results than using the individual wavelet coefficients as inputs.

For WN models, an input layer with 4–8 neurons, a single hidden layer composed of 4–6 neurons, and one output layer consisting of one neuron were developed. The number of neurons was determined in the same way as for the traditional ANN models. All the ANN models that had wavelet decomposed subseries as their inputs were also partitioned in a similar manner to the traditional ANN models.

4.5. SVR Models. All SVR models were developed using the OnlineSVR software created by Parrella [59]. OnlineSVR is a technique used to build support vector machines for regression. The OnlineSVR software partitions the data into only two sets: a training set and a testing set. The SVR models were partitioned in a similar manner to the ANN and WN models.

All SVR models used the nonlinear radial basis function (RBF) kernel. As a result, each SVR model consisted of three parameters that were selected: gamma (γ), cost (C), and epsilon (ε). The γ parameter is a constant that reduces the model space and controls the complexity of the solution, C is a positive constant that is a capacity control parameter, and ε is the loss function that describes the regression vector without all the input data [32]. These three parameters were selected based on a trial and error procedure. The combination of parameters that produced the lowest RMSE values for the training data sets was selected.

4.6. Performance Measures. The performance of the forecasts resulting from the data-driven models was evaluated by the following measures of goodness of fit:

$$\text{The coefficient of determination } (R^2) = \frac{\sum_{i=1}^{N}\left(\hat{y}_i - \overline{y}_i\right)}{\sum_{i=1}^{N}\left(y_i - \overline{y}_i\right)^2},$$
$$\overline{y}_i = \frac{1}{N}\sum_{i=1}^{N} y_i, \tag{22}$$

where $\overline{y}_i$ is the mean value taken over N, y_i is the observed value, $\hat{y}_i$ is the forecasted value, and N is the number of data

points. The coefficient of determination measures the degree of association among the observed and predicted values. The higher the value of R^2 (with 1 being the highest possible value), the better the performance of the model

$$\text{The Root Mean Squared Error (RMSE)} = \sqrt{\frac{\text{SSE}}{N}}, \quad (23)$$

where SSE is the sum of squared errors and N is the number of data points used. SSE is given by

$$\text{SSE} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad (24)$$

with the variables already having been defined. The RMSE evaluates the variance of errors independently of the sample size

$$\text{The Mean Absolute Error (MAE)} = \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{N}. \quad (25)$$

The MAE is used to measure how close forecasted values are to the observed values. It is the average of the absolute errors.

5. Results and Discussion

For each subbasin of the Awash River Basin, the station that showed the best performance results for each data driven model are presented below. In this study, SPI 3 and SPI 12 were forecast over lead times of 1 and 6 months to determine the effectiveness of the data-driven models over short- and long-term lead times.

As shown in Table 3(a), the best data-driven model in the Upper Awash Basin for forecasts of SPI 3 and 12 is the WN model. All the models exhibited better results for forecasts of a 1-month lead time (L1) compared to forecasts of 6-months lead time (L6). Forecasts of SPI 12, for all the data-driven models, had better performance results than forecasts of SPI 3 in terms of R^2, RMSE, and MAE, regardless of forecast lead time. The best 1-month lead time WN forecast of SPI 12 had results of 0.9534, 0.0600, and 0.0536 in terms of R^2, RMSE, and MAE, respectively. The second best results were from ANN models with results of 0.9451, 0.0610, and 0.0603 in terms of R^2, RMSE and MAE, respectively. Figures 3 and 4 show the ANN and WN 1-month forecast results for SPI 12 at the Ejersalele station.

The performance of both these models is quite similar, as indicated by Figures 3 and 4. Both models adequately represent the periods of abundant and acute precipitation as indicated by the peaks and valleys in the figures.

Similar to the results for the Upper Awash Basin, the best forecast results in the Middle Awash Basin were from WN models. The WN models had the best results for both SPI 3 and SPI 12, for forecast lead times of 1 and 6 months, respectively (Table 3(b)). The forecast results of all the data-driven models deteriorated when the forecast lead time was increased from 1 to 6 months.

Figure 5 illustrates the relationship between the observed SPI 12 and the predicted SPI 12 from the ANN model at

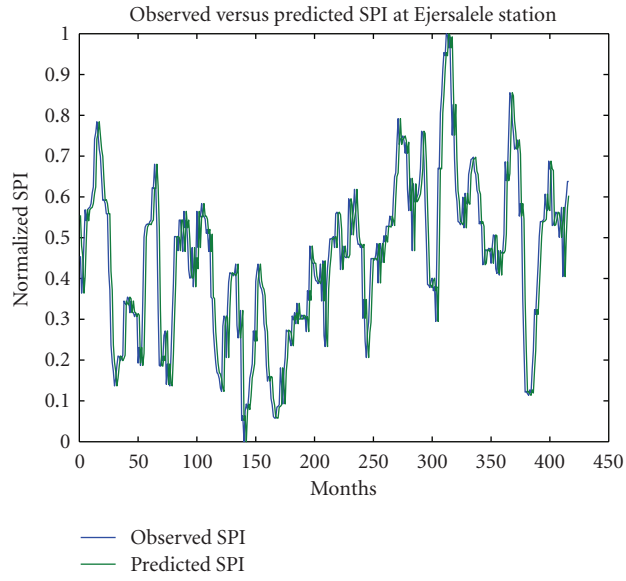

Figure 3: SPI 12 forecast results for the best ANN model at the Ejersalele station (1-month lead time).

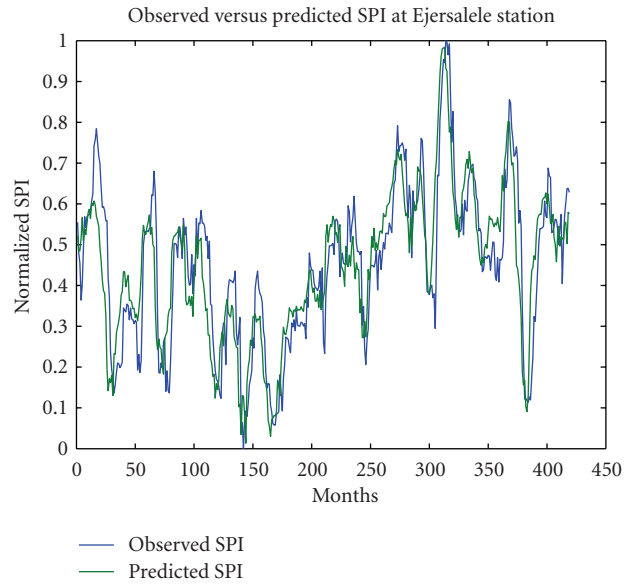

Figure 4: SPI 12 forecast results for the best WA-ANN model at the Ejersalele station (1-month lead time).

the Nazereth station. The ANN model underestimates the severity of the drought period at 112 months. In contrast, the WN model for SPI 12 at the Nazereth station displays improved results with respect to the drought period at 112 months (Figure 6).

In the Lower Awash Basin, the forecast results exhibited the same trend shown in the Upper and Middle sub-basins. The WN models had the best results for both SPI 3 and SPI 12, for forecast lead times of 1 and 6 months, respectively. Figures 7 and 8 illustrate the best SPI 12 forecasts at the Dubti station where both ANN and WN models predict

Table 3: (a) Performance results for the Ejersalele station, Upper Awash Basin, (b) Performance results of Nazereth Station, Middle Awash Basin, (c) Performance results of Dubti Station, Lower Awash Basin.

(a)

Model-Lead time	SPI 3			SPI 12		
	R^2	RMSE	MAE	R^2	RMSE	MAE
ANN-L1	0.7694	0.1574	0.1433	0.9451	0.0610	0.0603
ANN-L6	0.6232	0.1744	0.1567	0.8614	0.1011	0.0885
WN-L1	0.8829	0.0700	0.0352	0.9534	0.0600	0.0536
WN-L6	0.6433	0.1070	0.0356	0.8731	0.0790	0.0662
SVR-L1	0.7219	0.1046	0.0915	0.7611	0.1312	0.1129
SVR-L6	0.6647	0.1118	0.1042	0.6941	0.1341	0.1247

(b)

Model-Lead time	SPI 3			SPI 12		
	R^2	RMSE	MAE	R^2	RMSE	MAE
ANN-L1	0.7319	0.1170	0.1016	0.9158	0.1003	0.0911
ANN-L6	0.6546	0.1240	0.1142	0.7542	0.1104	0.0919
WN-L1	0.9483	0.0510	0.0441	0.9167	0.0753	0.0629
WN-L6	0.8641	0.0727	0.0512	0.8012	0.1072	0.0802
SVR-L1	0.7114	0.1216	0.1114	0.7713	0.1147	0.1130
SVR-L6	0.6540	0.1320	0.1217	0.7326	0.1244	0.1215

(c)

Model-Lead time	SPI 3			SPI 12		
	R^2	RMSE	MAE	R^2	RMSE	MAE
ANN-L1	0.7368	0.1175	0.1095	0.9188	0.0710	0.0648
ANN-L6	0.6806	0.1302	0.1147	0.7135	0.0938	0.0836
WN-L1	0.9018	0.0652	0.0581	0.9473	0.0648	0.0560
WN-L6	0.8119	0.0706	0.0642	0.8641	0.0846	0.0747
SVR-L1	0.6990	0.1146	0.1022	0.7041	0.1102	0.1009
SVR-L6	0.6331	0.1309	0.1242	0.6705	0.1107	0.1025

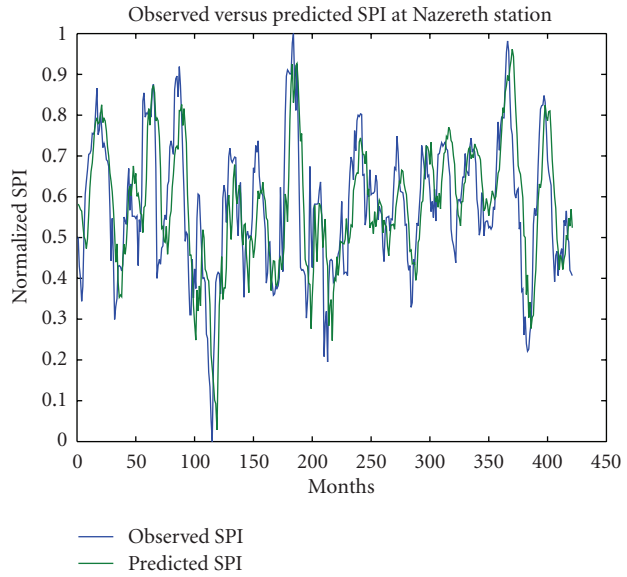

Figure 5: SPI 12 forecast results for the best ANN model at the Nazereth station (1 month lead time).

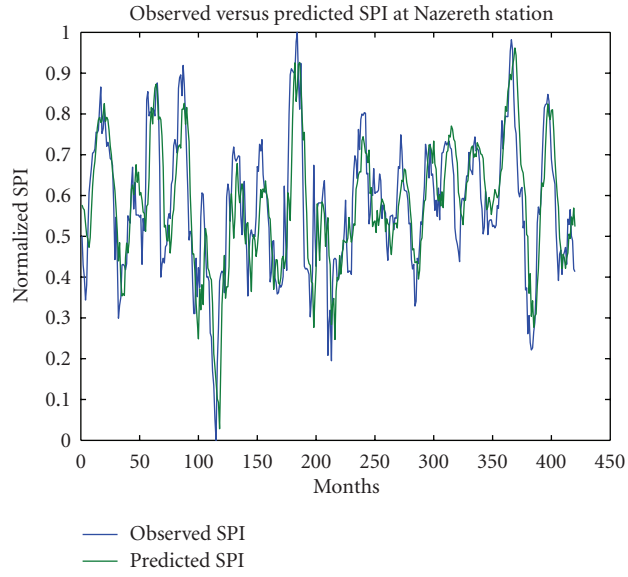

Figure 6: SPI 12 forecast results for the best WN model at the Nazereth station (1-month lead time).

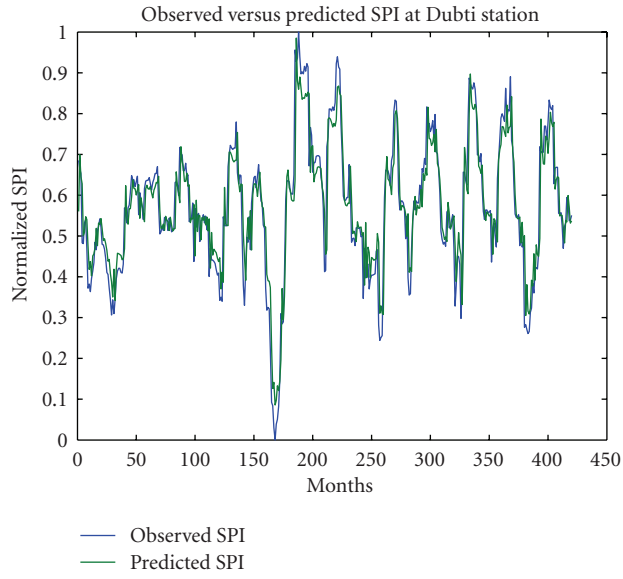

Figure 7: SPI 12 forecast results for the best ANN model at the Dubti station (1-month lead time).

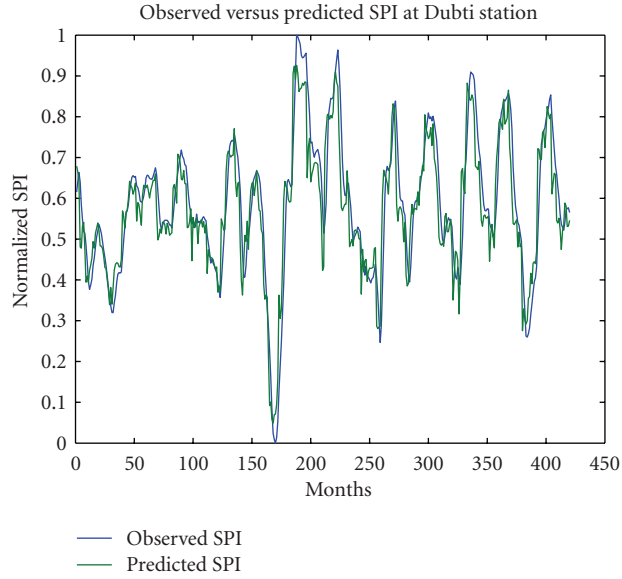

Figure 8: SPI 12 forecast results for the best WN model at the Dubti station (1-month lead time).

the periods of abundant and acute precipitation quite well. When the forecast lead time was increased, the performance of all the models deteriorated, especially with respect to R^2. Data-driven models in the Upper and Lower Awash basins exhibited their best results for forecasts of SPI 12, indicating that data-driven models are more effective in predicting long-term drought conditions in those two basins, while in the Middle Awash Basin most models also exhibited their best results for forecasts of SPI 12 except WN models, which exhibited their best results for forecasts of SPI 3. This trend could be due to the fact that long-term SPI, which is a cumulative of short-term time scales, tend toward zero unless a specific trend is taking place. The exception regarding the WN models in the Middle Awash Basin may be due to the fact that the precipitation record at this station is relatively stable, meaning there are not many changes from one month to the next and the SPI 3 is not sensitive to those changes.

Overall, all three data-driven models forecast SPI 3 and SPI 12 well for forecast lead times of 1 and 6 months. The results indicate that ANN models are more effective than SVR models at forecasting in this study. The use of wavelet analysis improved the forecast results of ANN models, specifically in predicting extreme events as shown in Figure 6. Indeed, using a measure for peak relative error as shown by

$$Z = 100 \left| \frac{q_s(\text{peak}) - q_o(\text{peak})}{q_o} \right|, \tag{26}$$

it was determined that the relative error of the ANN model, 95%, was reduced to 88% when a WN model was used.

The fact that wavelet analysis is an effective tool at revealing local discontinuities helps explain why it was more effective in predicting the extreme events in the Middle Awash Basin. Wavelet analysis may help de-noise the original SPI time series compared to a traditional ANN model. The forecast of this de-noised signal may further explain the fact that extreme events are forecast better using wavelet analysis.

An increase in forecast lead time results in a deterioration of performance in all the models. However, this deterioration does not result in poor models, indicating the stability of these data-driven models in predicting the SPI. The results in terms of RMSE and MAE do not deteriorate drastically with an increase of lead time. For example, for the Dubti station, the RMSE and MAE of SVR models deteriorate by 0.05 and 0.26%, respectively.

There is variability with regards to the best forecasts of both SPI 3 and SPI 12 amongst the three subbasins. For example, the best forecast of SPI 3 at a 1-month lead time occurred in the Middle Awash Basin (WN model), while the best forecast of SPI 12 at a 1-month lead time occurred in the Upper Awash Basin (WN model). While each subbasin has a different climatology, there does not seem to be a clear trend linking climatology with forecast accuracy. It seems that the reason behind the best models for each data-driven method being in various subbasins is linked with the characteristics of the individual station and not the characteristics of the subbasin as a whole.

In addition, the forecast results for SPI 12 are better than the forecast results for SPI 3 in almost all cases. For SPI 3 and other short-term SPI, each new month has a large impact on the period sum of precipitation [6]. As a result, the SPI 3 is sensitive to any change in precipitation from one month to another. In the case of SPI 12, each individual month has less impact on the total and the index is not as sensitive to changes in precipitation from one month to the next. The fact that SPI 3 is more sensitive to changes in precipitation results in less accurate forecast results than SPI 12. However, the effects of wavelet analysis are more significant for SPI 3 than for SPI 12, especially for forecast lead times of 6

months. As stated previously, the ANN forecasts of SPI 12 are not as sensitive to changes in precipitation and thus good results are obtained. The ability of wavelet analysis to improve these results exists as shown but is not as high as the improvement seen in SPI 3 forecasts because ANN forecasts of SPI 3 suffer due to the sensitivity of SPI 3 to slight changes in precipitation over the long-term record.

All three subbasins had a different climatology. The forecast results have all shown that WN models are the most effective at forecasting the SPI in all the sub-basins in terms of R^2, RMSE and MAE. Whether this is the case in all climatic zones needs to be explored in future studies.

6. Conclusion

This study tried to determine the most effective data-driven model for forecasts of the SPI drought index in the Awash River Basin of Ethiopia. WN models were shown to be the most effective model for forecasts of SPI 3 and 12 in all three subbasins. WN models showed greater correlation between observed and predicted SPI compared to simple ANNs and SVR models. WN models also consistently showed lower values of RMSE and MAE compared to the other data driven models explored in this study. All the data-driven models showed increased forecast results for SPI 12 compared to SPI 3. Forecast results deteriorated as the forecast lead time increased for all the models. Of the two machine learning techniques, ANNs are more effective in forecasting the SPI compared to SVR models. This trend occurs in all three subbasins and should be studied in other regions to determine if ANNs are more effective tools for drought forecasting compared to SVR models. It is thought that WN models provide more accurate results because preprocessing the original SPI time series with wavelet decompositions "denoises" the data. Future studies should attempt to explore WSVR models, ensemble WN and WSVR models, and explore SPI forecasts using these new methods in other regions with different characteristics. Future studies should also attempt to quantify time shift error as it is a part of forecasting problems with regression models.

Acknowledgments

An NSERC Discovery Grant and a FQRNT New Researcher Grant held by Jan Adamowski were used to fund this research.

References

[1] E. Mersha and V. K. Boken, "Agricultural drought in Ethiopia," in *Monitoring and Predicting Agricultural Drought: A Global Study*, V. K. Boken, A. P. Cracknell, and R. L. Heathcote, Eds., Oxford University Press, 2005.

[2] A. K. Mishra and V. P. Singh, "A review of drought concepts," *Journal of Hydrology*, vol. 391, no. 1-2, pp. 202–216, 2010.

[3] T. Ross and N. Lott, "A climatology of 1980–2003 extreme weather and climate events," National Climatic Data Center Technical Report No. 2003-01. NOAA/ NESDIS, National Climatic Data Center, Asheville, NC, USA.

[4] A. Cancelliere, G. di Mauro, B. Bonaccorso, and G. Rossi, "Stochastic forecasting of drought indices," in *Methods and Tools For Drought Analysis and Management*, G. Rossi, T. Vega, and B. Bonaccorso, Eds., Springer, 2007.

[5] W. J. Gibbs and J. V. Maher, *Rainfall Deciles as Drought Indicators*, vol. 48 of *Bulletin (Commonwealth Bureau of Meteorology, Australia)*, Bureau of Meteorology, Melbourne, Australia, 1967.

[6] T. B. McKee, N. J. Doesken, and J. Kleist, "The relationship of drought frequency and duration to time scales," in *Proceedings of the 8th Conference on Applied Climatology*, American Meteorological Society, Anaheim, Calif, USA, 1993.

[7] H. R. Byun and D. A. Wilhite, "Objective quantification of drought severity and duration," *Journal of Climate*, vol. 12, no. 9, pp. 2747–2756, 1999.

[8] W. Palmer, "Meteorological drought," Tech. Rep. 45, U.S. Weather Bureau, Washington, DC, USA, 1965.

[9] H. K. Ntale and T. Y. Gan, "Drought indices and their application to East Africa," *International Journal of Climatology*, vol. 23, no. 11, pp. 1335–1357, 2003.

[10] A. K. Mishra and V. R. Desai, "Drought forecasting using feed-forward recursive neural network," *Ecological Modelling*, vol. 198, no. 1-2, pp. 127–138, 2006.

[11] S. Morid, V. Smakhtin, and K. Bagherzadeh, "Drought forecasting using artificial neural networks and time series of drought indices," *International Journal of Climatology*, vol. 27, no. 15, pp. 2103–2111, 2007.

[12] U. G. Bacanli, M. Firat, and F. Dikbas, "Adaptive Neuro-Fuzzy inference system for drought forecasting," *Stochastic Environmental Research and Risk Assessment*, vol. 23, no. 8, pp. 1143–1154, 2009.

[13] A. P. Barros and G. J. Bowden, "Toward long-lead operational forecasts of drought: an experimental study in the Murray-Darling River Basin," *Journal of Hydrology*, vol. 357, no. 3-4, pp. 349–367, 2008.

[14] P. Cutore, G. Di Mauro, and A. Cancelliere, "Forecasting palmer index using neural networks and climatic indexes," *Journal of Hydrologic Engineering*, vol. 14, no. 6, pp. 588–595, 2009.

[15] M. Karamouz, K. Rasouli, and S. Nazif, "Development of a hybrid Index for drought prediction: case study," *Journal of Hydrologic Engineering*, vol. 14, no. 6, pp. 617–627, 2009.

[16] A. F. Marj and A. M. J. Meijerink, "Agricultural drought forecasting using satellite images, climate indices and artificial neural network," *International Journal of Remote Sensing*, vol. 32, no. 24, pp. 9707–9719, 2011.

[17] D. Labat, R. Ababou, and A. Mangin, "Wavelet analysis in karstic hydrology. 2nd part: rainfall-runoff cross-wavelet analysis," *Comptes Rendus de l'Academie de Sciences*, vol. 329, no. 12, pp. 881–887, 1999.

[18] P. Saco and P. Kumar, "Coherent modes in multiscale variability of streamflow over the United States," *Water Resources Research*, vol. 36, no. 4, pp. 1049–1067, 2000.

[19] L. C. Smith, D. L. Turcotte, and B. L. Isacks, "Stream flow characterization and feature detection using a discrete wavelet transform," *Hydrological Processes*, vol. 12, no. 2, pp. 233–249, 1998.

[20] P. Coulibaly, F. Anctil, and B. Bobée, "Daily reservoir inflow forecasting using artificial neural networks with stopped training approach," *Journal of Hydrology*, vol. 230, no. 3-4, pp. 244–257, 2000.

[21] S. N. Lane, "Assessment of rainfall-runoff models based upon wavelet analysis," *Hydrological Processes*, vol. 21, no. 5, pp. 586–607, 2007.

[22] J. F. Adamowski, "Development of a short-term river flood forecasting method for snowmelt driven floods based on wavelet and cross-wavelet analysis," *Journal of Hydrology*, vol. 353, no. 3-4, pp. 247–266, 2008.

[23] J. Adamowski and K. Sun, "Development of a coupled wavelet transform and neural network method for flow forecasting of non-perennial rivers in semi-arid watersheds," *Journal of Hydrology*, vol. 390, no. 1-2, pp. 85–91, 2010.

[24] M. Özger, A. K. Mishra, and V. P. Singh, "Long lead time drought forecasting using a wavelet and fuzzy logic combination model: a case study in Texas," *Journal of Hydrometeorology*, vol. 13, no. 1, pp. 284–297, 2012.

[25] T. Partal and Ö. Kişi, "Wavelet and neuro-fuzzy conjunction model for precipitation forecasting," *Journal of Hydrology*, vol. 342, no. 1-2, pp. 199–212, 2007.

[26] T. W. Kim and J. B. Valdes, "Nonlinear model for drought forecasting based on a conjunction of wavelet transforms and neural networks," *Journal of Hydrologic Engineering*, vol. 8, no. 6, pp. 319–328, 2003.

[27] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[28] J. B. Gao, S. R. Gunn, C. J. Harris, and M. Brown, "A probabilistic framework for SVM regression and error bar estimation," *Machine Learning*, vol. 46, no. 1–3, pp. 71–89, 2002.

[29] M. S. Khan and P. Coulibaly, "Application of support vector machine in lake water level prediction," *Journal of Hydrologic Engineering*, vol. 11, no. 3, pp. 199–205, 2006.

[30] S. Rajasekaran, S. Gayathri, and T.-L. Lee, "Support vector regression methodology for storm surge predictions," *Journal of Ocean Engineering*, vol. 35, no. 16, pp. 1578–1587, 2008.

[31] O. Kisi and M. Cimen, "Evapotranspiration modelling using support vector machines," *Hydrological Sciences Journal*, vol. 54, no. 5, pp. 918–928, 2009.

[32] O. Kisi and M. Cimen, "A wavelet-support vector machine conjunction model for monthly streamflow forecasting," *Journal of Hydrology*, vol. 399, no. 1-2, pp. 132–140, 2011.

[33] T. Asefa, M. Kemblowski, M. McKee, and A. Khalil, "Multi-time scale stream flow predictions: the support vector machines approach," *Journal of Hydrology*, vol. 318, no. 1–4, pp. 7–16, 2006.

[34] W. C. Wang, K. W. Chau, C. T. Cheng, and L. Qiu, "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series," *Journal of Hydrology*, vol. 374, no. 3-4, pp. 294–306, 2009.

[35] R. Maity, P. P. Bhagwat, and A. Bhatnagar, "Potential of support vector regression for prediction of monthly streamflow using endogenous property," *Hydrological Processes*, vol. 24, no. 7, pp. 917–923, 2010.

[36] Z. M. Yuan and X. S. Tan, "Nonlinear screening indicators of drought resistance at seedling stage of rice based on support vector machine," *Acta Agronomica Sinica*, vol. 36, no. 7, pp. 1176–1182, 2010.

[37] C. Cacciamani, A. Morgillo, S. Marchesi, and V. Pavan, "Monitoring and forecasting drought on a regional scale: emilia-romagna region," *Water Science and Technology Library*, vol. 62, part 1, pp. 29–48, 2007.

[38] I. Bordi and A. Sutera, "Drought monitoring and forecasting at large-scale," in *Methods and Tools For Drought Analysis and Management*, G. Rossi, T. Vega, and B. Bonaccorso, Eds., pp. 3–27, Springer, New York, NY, USA, 2007.

[39] N. B. Guttman, "Accepting the standardized precipitation index: a calculation algorithm," *Journal of the American Water Resources Association*, vol. 35, no. 2, pp. 311–322, 1999.

[40] H. C. S. Thom, "A note on gamma distribution," *Monthly Weather Review*, vol. 86, pp. 117–122, 1958.

[41] D. C. Edwards and T. B. McKee, "Characteristics of 20th century drought in the United States at multiple scales," Atmospheric Science Paper 634, 1997.

[42] D. S. Wilks, *Statistical Methods in the Atmospheric Sciences an Introduction*, Academic Press, San Diego, Calif, USA, 1995.

[43] M. Abramowitz and A. Stegun, Eds., *Handbook of Mathematical Formulas, Graphs, and Mathematical Tables*, Dover Publications, New York, NY, USA, 1965.

[44] S. Morid, V. Smakhtin, and M. Moghaddasi, "Comparison of seven meteorological indices for drought monitoring in Iran," *International Journal of Climatology*, vol. 26, no. 7, pp. 971–985, 2006.

[45] J. Adamowski and H. F. Chan, "A wavelet neural network conjunction model for groundwater level forecasting," *Journal of Hydrology*, vol. 407, no. 1–4, pp. 28–40, 2011.

[46] M. Çimen, "Estimation of daily suspended sediments using support vector machines," *Hydrological Sciences Journal*, vol. 53, no. 3, pp. 656–666, 2008.

[47] A. J. Smola, *Regression Estimation with Support Vector Learning Machines [M.S. thesis]*, Technische Universitat Munchen, Munich, Germany, 1996.

[48] S. Gunn, "Support vector machines for classification and regression," ISIS Technical Report, Department of Electronics and Computer Science, University of Southampton, 1998.

[49] B. Cannas, A. Fanni, G. Sias, S. Tronci, and M. K. Zedda, "River flow forecasting using neural networks and wavelet analysis," in *Proceedings of the European Geosciences Union*, 2006.

[50] S. G. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, Calif, USA, 1998.

[51] F. Murtagh, J. L. Starck, and O. Renuad, "On neuro-wavelet modeling," *Decision Support Systems*, vol. 37, no. 4, pp. 475–484, 2004.

[52] O. Renaud, J. Starck, and F. Murtagh, *Wavelet-Based Forecasting of Short and Long Memory Time Series*, Department of Economics, University of Geneve, 2002.

[53] C. E. Desalegn, M. S. Babel, A. Das Gupta, B. A. Seleshi, and D. Merrey, "Farmers' perception of water management under drought conditions in the upper Awash Basin, Ethiopia," *International Journal of Water Resources Development*, vol. 22, no. 4, pp. 589–602, 2006.

[54] D. C. Edossa, M. S. Babel, and A. D. Gupta, "Drought analysis in the Awash River Basin, Ethiopia," *Water Resources Management*, vol. 24, no. 7, pp. 1441–1460, 2010.

[55] M. K. Tiwari and C. Chatterjee, "Development of an accurate and reliable hourly flood forecasting model using wavelet-bootstrap-ANN (WBANN) hybrid approach," *Journal of Hydrology*, vol. 394, no. 3-4, pp. 458–470, 2010.

[56] N. Wanas, G. Auda, M. S. Kamel, and F. Karray, "On the optimal number of hidden nodes in a neural network," in *Proceedings of the 11th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '98)*, pp. 918–921, May 1998.

[57] J. C. Principe, N. R. Euliano, and W. Curt Lefebvre, *Neural and Adaptive Systems*, John Wiley & Sons, 2000.

[58] T. Partal, "Modelling evapotranspiration using discrete wavelet transform and neural networks," *Hydrological Processes*, vol. 23, no. 25, pp. 3545–3555, 2009.

[59] F. Parrella, *Online support vector regression [M.S. thesis]*, University of Genoa, 2007.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 471973, 9 pages
doi:10.1155/2012/471973

Research Article

A Nonlinear Programming and Artificial Neural Network Approach for Optimizing the Performance of a Job Dispatching Rule in a Wafer Fabrication Factory

Toly Chen

Department of Industrial Engineering and Systems Management, Feng Chia University, No. 100 Wenhwa Road, Seatwen, Taichung 407, Taiwan

Correspondence should be addressed to Toly Chen, tcchen@fcu.edu.tw

Received 21 May 2012; Accepted 18 July 2012

Academic Editor: Yi-Chi Wang

A nonlinear programming and artificial neural network approach is presented in this study to optimize the performance of a job dispatching rule in a wafer fabrication factory. The proposed methodology fuses two existing rules and constructs a nonlinear programming model to choose the best values of parameters in the two rules by dynamically maximizing the standard deviation of the slack, which has been shown to benefit scheduling performance by several studies. In addition, a more effective approach is also applied to estimate the remaining cycle time of a job, which is empirically shown to be conducive to the scheduling performance. The efficacy of the proposed methodology was validated with a simulated case; evidence was found to support its effectiveness. We also suggested several directions in which it can be exploited in the future.

1. Introduction

This study attempts to optimize the performance of a job dispatching rule in a wafer fabrication factory. The production equation required by a wafer fabrication factory is very expensive and must be fully utilized. For this purpose, to ensure that the capacity does not substantially exceed the demand is a perquisite. Subsequently, how to plan the use of the existing capacity to shorten the cycle time and maximize the turnover rate is an important goal. In this regard, scheduling is undoubtedly a very useful tool.

However, some studies [1–4] noted that job dispatching is very difficult task in a semiconductor manufacturing factory. Theoretically, it is an NP-hard problem. In practice, many semiconductor manufacturing factories suffer from lengthy cycle times and are not able to improve on their delivery promises to their customers.

Semiconductor manufacturing can be divided into four stages: wafer fabrication, wafer probing, packaging, and final testing. The most important stage is wafer fabrication. It is also the most time-consuming one. In this study, we investigated the job dispatching for this stage. This field includes many different methods, including dispatching rules, heuristics, data-mining-based approaches [5, 6], agent technologies [5, 7–9], and simulation. Among them, dispatching rules (e.g., first-in first out (FIFO), earliest due date (EDD), least slack (LS), shortest processing time (SPT), shortest remaining processing time (SRPT), critical ratio (CR), the fluctuation smoothing rule for the mean cycle time (FSMCT), and the fluctuation smoothing rule for cycle time variation (FSVCT), FIFO+, SRPT+, and SRPT++) all have received a lot of attention over the last few years [5–7] and are the most prevalent methods used in practical applications. For details on the traditional dispatching rules, please refer to Lu et al. [10].

Some advances in this field are as follows. Altendorfer et al. [11] proposed the work in parallel queue (WIPQ) rule targeting maximizing throughput at a low level of work in process (WIP). Zhang et al. [12] proposed the dynamic bottleneck detection (DBD) approach by classifying workstations into several categories and then applied different dispatching rules to these categories. They used three dispatching rules including FIFO, the shortest processing time until the next bottleneck (SPNB), and CR. Based on the current conditions in the wafer fabrication factory, Hsieh et al. [6] chose one approach from FSMCT, FSVCT, largest

deviation first (LDF), one step ahead (OSA), or FIFO. Chen [13] modified FSMCT and proposed the nonlinear FSMCT (NFSMCT) rule, in which he smoothed the fluctuation in the estimated remaining cycle time and balanced it with that of the release time or the mean release rate. To diversify the slack, he applied the "division" operator instead. This was followed by Chen [14], in which he proposed the one-factor-tailored NFSMCT (1f-TNFSMCT) rule and the one-factor-tailored nonlinear FSVCT (1f-TNFSVCT) rule. Both rules contain an adjustable parameter to allow them to be customized for a target wafer fabrication factory. Chen [15] used more parameters and proposed 2f-TNFSMCT and 2f-TNFSVCT.

In a multiple-objective study, Chen and Wang [16] proposed a biobjective nonlinear fluctuation smoothing rule with an adjustable factor (1f-biNFS) to optimize both the average cycle time and the cycle time variation at the same time. More degrees of freedom seem to be helpful in the performance of customizable rules. For this reason, Chen et al. [17] extended 1f-biNFS to a biobjective fluctuation smoothing rule with four adjustable factors (4f-biNFS). For a summary of these rules please refer to Table 1. One drawback of these rules is that only static factors are used, and they must be determined in advance. To this end, most studies (e.g., [13–17]) performed extensive simulations. This is not only time-consuming but it also fails to consider enough possible combinations of these factors. Chen [18] established a mechanism that was able to adjust the values of the factor in 1f-biNFS dynamically (dynamic 1f-biNFS). However, even though satisfactory results were obtained in his experiment, there was no theoretical basis supporting the proposed mechanism. Chen [19] attempted to relate the scheduling performance to the factor values using a back propagation network (BPN). If that would have worked, then the factor values contributing to the optimal scheduling performance could have been found. However, the explanatory ability of the BPN was not good enough.

At the same time, Chen [18] stated that a nonlinear fluctuation smoothing rule uses the divisor operator instead of the subtraction operator, which diversifies the slack and makes the nonlinear fluctuation smoothing rule more responsive to changes in the parameters. Chen and Wang [16] proved that the effects of the parameters are balanced better in a nonlinear fluctuation smoothing rule than in a traditional one if the variation in the parameters is large. In addition, there will be fewer ties since the slack values are very different. Further, magnifying the difference in the slack seems to improve the scheduling performance, especially with respect to the average cycle time [20]. For these reasons, a slack-diversifying fuzzy-neural rule is used in chen et al. [20] for job dispatching in a wafer fabrication factory, in order to further improve the performance of job dispatching in a wafer fabrication factory. The slack-diversifying nonlinear fluctuation smoothing rule is modified from 1f-TNFSVCT by maximizing the difference in the slack measured with the standard deviation of the slack.

This study adopts several treatments to further improve Wang et al.'s approach.

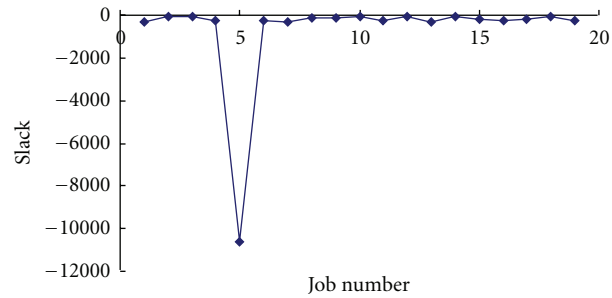

Figure 1: The extreme cases.

(1) In nonlinear fluctuation smoothing rules, it is common that some jobs have very large or small slack values, that is the extreme case (see Figure 1), which usually distorts the results of calculating the standard deviation of slacks. In this study, the extreme cases are excluded before calculation.

(2) Two objectives, the average cycle time and cycle time standard deviation, are considered at the same time by fusing the results from 2f-TNFSMCT and 2f-TNFSVCT.

(3) A nonlinear programming problem is solved to find the optimal values of parameters in 2f-TNFSMCT and 2f-TNFSVCT.

(4) On the other hand, the remaining cycle time of a job needs to be estimated in 2f-TNFSMCT and 2f-TNFSVCT. For this reason, we also propose a more effective fuzzy-neural approach to estimate the remaining cycle time of a job. The fuzzy-neural approach is a modification of the fuzzy c-means and back propagation network (FCM-BPN) approach [17] by incorporating in the concept of principal component analysis (PCA). According to Chen and Wang [3], with more accurate remaining cycle time estimation, the scheduling performance of a fluctuation smoothing rule can be significantly improved. In the original study, Chen and Wang used a gradient search algorithm for training the BPN, which is time-consuming and not very accurate. In this study, we use the Levenberg-Marquardt algorithm to achieve the same purpose, which is more efficient than that in Chen and Wang's study and can produce more accurate forecasts.

The differences between the proposed methodology and the previous methods are summarized in Table 1.

The remainder of this paper is arranged as follows. Section 2 provides the details of the proposed methodology. In Section 3, a simulated case is used to validate the effectiveness of the nonlinear programming and artificial neural network approach. The performances of some existing approaches in this field are also examined using the simulated data. Finally, we draw our conclusions in Section 4 and provide some worthwhile topics for future work.

Table 1: The differences between the proposed methodology and the previous methods.

Rule	Number of objectives	Objectives	Number of adjustable parameters	Optimized?	How to derive the rule?
NFSMCT	1	Average cycle time	1	No	(i) Generalizing FSMCT
1f-TNFSVCT	1	Cycle time standard variation	1	No	(i) Generalizing FSVCT (ii) Adding adjustable parameters
1f-TNFSMCT	1	Average cycle time	1	No	(i) Generalizing FSMCT (ii) Adding adjustable parameters
2f-TNFSVCT	1	Cycle time standard deviation	2	No	(i) Generalizing FSVCT (ii) Adding adjustable parameters
4f-biNFS	2	Average cycle time, cycle time standard deviation	2	Yes	(i) Fusing FSVCT and FSMCT (ii) Adding adjustable parameters
The proposed methodology	2	Average cycle time, cycle time standard deviation	2	Yes	(i) Fusing 2f-TFSMCT and 2f-TNFSVCT (ii) Nonlinear programming

2. Methodology

The variables and parameters that will be used in the proposed methodology are defined in the following.

(1) R_j: the release time of job j; $j = 1 \sim n$.

(2) BQ_j: the total queue length before bottlenecks at R_j.

(3) CR_{ju}: the critical ratio of job j at step u.

(4) CT_j: the cycle time of job j.

(5) CTE_j: the estimated cycle time of job j.

(6) D_j: the average delay of the three most recently completed jobs at R_j.

(7) DD_j: the due date of job j.

(8) FQ_j: the total queue length in the whole factory at R_j.

(9) Q_j: the queue length on the processing route of job j at R_j.

(10) RCTE_{ju}: the estimated remaining cycle time of job j from step u.

(11) RPT_{ju}: the remaining processing time of job j from step u.

(12) SCT_{ju}: the step cycle time of job j until step u.

(13) SK_{ju}: the slack of job j at step u.

(14) U_j: the average factory utilization before job j is released. If the utilization of the factory is reported on a daily basis, then U_j is the utilization of the day before job j is released.

(15) WIP_j: the factory work in progress (WIP) at R_j.

(16) λ: mean release rate.

(17) x_p: inputs to the three-layer BPN, $p = 1 \sim 6$.

(18) h_l: the output from hidden-layer node l, $l = 1 \sim L$.

(19) w_l^o: the connection weight between hidden-layer node l and the output node.

(20) w_{pl}^h: the connection weight between input node p and hidden-layer node l, $p = 1 \sim 6$; $l = 1 \sim L$.

(21) θ_l^h: the threshold on hidden-layer node l.

(22) θ^o: the threshold on the output node.

The proposed methodology includes the following seven steps.

Step 1. Replacing parameters using PCA.

Step 2. Use FCM to classify jobs. The required inputs for this step are the new variables determined by PCA. To determine the optimal number of categories, we use the S-test. The output of this step is the category of each job.

Step 3. Use the BPN approach to estimate the cycle time of each job. Jobs of different categories will be sent to different three-layer BPNs. The inputs to the three-layer BPN include the new variables of a job, while the output is the estimated cycle time of the job.

Step 4. Derive the remaining cycle time of each job from the estimated cycle time.

Step 5. Incorporate the estimated remaining cycle time into the new rule that is composed of two subrules–2f-TNFSMCT and 2f-TNFSVCT.

Step 6. Find out the optimal value of parameters in the new rule by solving a nonlinear programming problem.

The remaining cycle time of a job being produced in a wafer fabrication factory is the time still needed to complete the job. If the job is just released into the wafer fabrication factory, then the remaining cycle time of the job is its cycle time. The remaining cycle time is an important input for the scheduling rule. Past studies (e.g., [21–24]) have shown that the accuracy of the remaining cycle time forecasting can be improved by job classification. Soft computing methods (e.g., [3, 20, 25, 26]) have received much attention in this field.

2.1. PCA Analysis. First, PCA is used to replace the inputs to the FCM-BPN. The combination of PCA and FCM has

been shown to be a more effective classifier than FCM alone. Although there are more advanced applications of PCA, in this study PCA is used to enhance the efficiency of training the FCM-BPN. PCA consists of the four following steps:

(1) Raw data standardization: to eliminate the difference between the dimensions and the impact of large numerical difference in the original variables$\{U_j, Q_j, \mathrm{BQ}_j, \mathrm{FQ}_j, \mathrm{WIP}_j, D_j\}$, the original variables are standardized:

$$x_{ij}^* = \frac{x_{ij} - \overline{x}_j}{\sigma_j},$$
$$\overline{x}_j = \frac{\sum_{i=1}^n x_{ij}}{n}, \tag{1}$$
$$\sigma_j = \sqrt{\frac{\sum_{i=1}^n \left(x_{ij} - \overline{x}_j\right)^2}{n-1}},$$

where $\overline{x}_j$ and σ_j indicate the mean and standard deviation of variable j, respectively,

(2) Establishment of the correlation matrix R:

$$R = \frac{1}{n-1} X^{*T} X^*, \tag{2}$$

where X^* is the standardized data matrix. The eigenvalues and eigenvectors of R are calculated and represented as $\lambda_1 \sim \lambda_m$ and $u_1 \sim u_m$, respectively; $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$.

(3) Determination of the number of principal components: the variance contribution rate is calculated as:

$$\eta_i = \frac{\lambda_i}{\sum_{i=1}^m \lambda_i} \cdot 100\%, \tag{3}$$

and the accumulated variance contribution rate is

$$\eta_\Sigma(q) = \sum_{i=1}^q \eta_i. \tag{4}$$

Choose the smallest q value such that $\eta_\Sigma(q) \geq 85\%\sim 90\%$.

(4) Formation of the following matrixes:

$$U_{m\times q} = \left[u_1, u_2, ..., u_q\right],$$
$$Z_{n\times q} = X^*_{n\times m} U_{m\times q}. \tag{5}$$

After PCA, examples are then classified using FCM.

2.2. The FCM Approach. In the proposed methodology, jobs are classified into K categories using FCM. If a crisp clustering method is applied, then it is possible that some clusters will have very few examples. In contrast, an example belongs to multiple clusters to different degrees in FCM, which provides a solution to this problem.

FCM classifies jobs by minimizing the following objective function:

$$\mathrm{Min} \sum_{k=1}^K \sum_{j=1}^n \mu_{j(k)}^m e_{j(k)}^2, \tag{6}$$

where K is the required number of categories; n is the number of jobs; $\mu_{j(k)}$ indicates the membership that job j belongs to category k; $e_{j(k)}$ measures the distance from job j to the centroid of category k; $m \in [1, \infty)$ is a parameter to adjust the fuzziness and is usually set to 2. The procedure of FCM is as follows.

(1) Produce a preliminary clustering result.

(2) (Iterations) Calculate the centroid of each category as

$$\overline{x}_{(k)} = \left\{\overline{x}_{(k)p}\right\}; \quad p = 1 \sim q,$$
$$\overline{x}_{(k)p} = \frac{\sum_{j=1}^n \mu_{j(k)}^m x_{jp}}{\sum_{j=1}^n \mu_{j(k)}^m},$$
$$\mu_{j(k)} = \frac{1}{\sum_{q=1}^K \left(e_{j(k)}/e_{j(q)}\right)^{2/(m-1)}}, \tag{7}$$
$$e_{j(k)} = \sqrt{\sum_{\text{all } p} \left(x_{jp} - \overline{x}_{(k)p}\right)^2},$$

where $\overline{x}_{(k)}$ is the centroid of category k. $\mu_{j(k)}^{(t)}$ is the membership that job i belongs to category k after the tth iteration.

(3) Remeasure the distance from each job to the centroid of each category, and then recalculate the corresponding membership.

(4) Stop if the following condition is met. Otherwise, return to step (2):

$$\max_k \max_j \left| \mu_{j(k)}^{(t)} - \mu_{j(k)}^{(t-1)} \right| < d, \tag{8}$$

where d is a real number representing the threshold for the convergence of membership.

Finally, the separate distance test (S-test) proposed by Xie and Beni [24] can be applied to determine the optimal number of categories K:

$$\mathrm{Min}\ S \tag{9}$$

subject to

$$J_m = \sum_{k=1}^K \sum_{j=1}^n \mu_{j(k)}^m e_{j(k)}^2,$$
$$e_{\min}^2 = \min_{k1 \neq k2} \left(\sum_{\text{all } p} \left(\overline{x}_{(k1)p} - \overline{x}_{(k2)p}\right)^2 \right), \tag{10}$$
$$S = \frac{J_m}{n \times e_{\min}^2},$$
$$K \in Z^+.$$

The K value minimizing S determines the optimal number of categories.

2.3. The BPN Approach. After clustering, a portion of the jobs in each category is input as the "training examples" to the three-layer BPN to determine the parameter values. The configuration of the three-layer BPN is set up as follows. First, inputs are the six parameters associated with the jth example/job including the q new variables. These parameters have to be normalized before feeding into the three-layer BPN. Subsequently, there is only a single hidden layer with neurons that are twice that in the input layer. Finally, the output from the three-layer BPN is the (normalized) estimated cycle time (CTE_j) of the example. The activation function used in each layer is Log Sigmoid function:

$$f(x) = \frac{1}{(1 + e^{-x})}. \tag{11}$$

The procedure for determining the parameter values is now described. Two phases are involved at the training stage. At first, in the forward phase, inputs are multiplied with weights, summated, and transferred to the hidden layer. Then activated signals are outputted from the hidden layer as

$$h_l = \frac{1}{1 + e^{-n_l^h}}, \tag{12}$$

where

$$\begin{aligned} n_l^h &= I_l^h - \theta_l^h \\ I_l^h &= \sum_{p=1}^{q} w_{pl}^h \cdot x_{jp}. \end{aligned} \tag{13}$$

h_l's are also transferred to the output layer with the same procedure. Finally, the output of the BPN is generated as

$$o_j = \frac{1}{1 + e^{-n^o}}, \tag{14}$$

where

$$\begin{aligned} n^o &= I^o - \theta^o, \\ I^o &= \sum_{l=1}^{L} w_l^o \cdot h_l. \end{aligned} \tag{15}$$

Subsequently, in the backward phase, some algorithms are applicable for training a BPN, such as the gradient descent algorithms, the conjugate gradient algorithms, and the Levenberg-Marquardt algorithm. In this study, the Levenberg-Marquardt algorithm is applied. The Levenberg-Marquardt algorithm was designed for training with second-order speed without having to compute the Hessian matrix. It uses approximation and updates the network parameters in a Newton-like way, as described below.

The network parameters are placed in vector $\boldsymbol{\beta} = [w_{11}^h, \ldots, w_{qL}^h, \theta_1^h, \ldots, \theta_L^h, w_1^o, \ldots, w_L^o, \theta^o]$. The network output o_j can be represented with $f(\mathbf{x}_j, \boldsymbol{\beta})$. The objective function of the BPN is to minimize the root mean-squared error (RMSE) or equivalently the sum of squared error (SSE):

$$\mathrm{SSE}(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(N\left(\mathrm{CT}_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)\right)^2. \tag{16}$$

The Levenberg-Marquardt algorithm is an iterative procedure. In the beginning, the user should specify the initial values of the network parameters $\boldsymbol{\beta}$. Let $\boldsymbol{\beta}^{\mathbf{T}} = (1, 1, \ldots, 1)$ is a common practice. In each step, the parameter vector $\boldsymbol{\beta}$ is replaced by a new estimate $\boldsymbol{\beta} + \boldsymbol{\delta}$, where $\boldsymbol{\delta} = [\Delta w_{11}^h, \ldots, \Delta w_{qL}^h, \Delta\theta_1^h, \ldots, \Delta\theta_L^h, \Delta w_1^o, \ldots, \Delta w_L^o, \Delta\theta^o]$. The network output becomes $f(\mathbf{x}_j, \boldsymbol{\beta}+\boldsymbol{\delta})$ that is approximated by its linearization as

$$f\left(\mathbf{x}_j, \boldsymbol{\beta} + \boldsymbol{\delta}\right) \approx f\left(\mathbf{x}_j, \boldsymbol{\beta}\right) + \mathbf{J}_j\boldsymbol{\delta}, \tag{17}$$

where

$$\mathbf{J}_j = \frac{\partial f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} \tag{18}$$

is the gradient vector of f with respect to $\boldsymbol{\beta}$. Substituting (17) into (16),

$$\mathrm{SSE}(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{j=1}^{n} \left(N\left(CT_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right) - \mathbf{J}_j\boldsymbol{\delta}\right)^2. \tag{19}$$

When the network reaches the optimal solution, the gradient of SSE with respect to $\boldsymbol{\delta}$ will be zero. Taking the derivative of $\mathrm{SSE}(\boldsymbol{\beta} + \boldsymbol{\delta})$ with respect to $\boldsymbol{\delta}$ and setting the result to zero gives

$$\left(\mathbf{J}^{\mathbf{T}}\mathbf{J}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathbf{T}}\left(N\left(CT_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)\right), \tag{20}$$

where $\mathbf{J}$ is the Jacobian matrix containing the first derivative of network error with respect to the weights and biases. Equation (20) includes a set of linear equations that can be solved for $\boldsymbol{\delta}$.

Finally, the BPN can be applied to estimate the cycle time of a job, and then the remaining cycle time of the job can be derived as

$$\mathrm{RCTE}_{ju} = \mathrm{CTE}_j - \mathrm{SCT}_{ju}, \tag{21}$$

2.4. The New Rule. In traditional fluctuation smoothing (FS) rules there are two different formulation methods, depending on the scheduling purpose [22]. ne method is aimed at minimizing the average cycle time with FSMCT:

$$\mathrm{SKM}_{ju} = \frac{j}{\lambda} - \mathrm{RCTE}_{ju}. \tag{22}$$

The other method is aimed at minimizing the variance of cycle time with FSVCT:

$$\mathrm{SKV}_{ju} = R_j - \mathrm{RCTE}_{ju}. \tag{23}$$

Jobs with the smallest slack values (SKM_{ju} or SKV_{ju}) will be given higher priority. These two rules and their variants have

been proven to be very effective in shortening the cycle time in wafer fabrication factories [10, 14–17].

Chen [15] normalized the parameters and used the division operator instead and derived the 2f-TNFSVCT rule:

$$\mathrm{SKM}_{ju} = \left(\frac{\beta}{\alpha\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(R_j - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \min\left(R_j\right)\right)\right), \tag{24}$$

and the 2f-TNFSMCT rule:

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right), \tag{25}$$

where

$$\alpha = \max\left(R_j\right) - \min\left(R_j\right), \quad \beta = \max\left(\mathrm{RCTE}_{ju}\right) - \min\left(\mathrm{RCTE}_{ju}\right), \tag{26}$$

$0 \le \xi, \zeta \le 1$. There are many possible models to form the combination of ξ and ζ. For example,

$$\text{(Linear model) } \xi = \zeta, \quad \text{(Nonlinear model) } \xi = \zeta^k, \quad k \ge 0, \quad \text{(Logarithmic model) } \xi = \frac{\ln(1+\zeta)}{\ln 2}. \tag{27}$$

The new rule is composed of two rules. The first rule is derived by diversifying the slack in the 2f-TNFSVCT rule, aimed at minimizing the variation of cycle time [22]. To diversify the slack, the standard deviation of the slack is to be maximized as follows:

$$\sigma_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}_{ju}}\right)^2}{N-1}}. \tag{28}$$

However, in nonlinear fluctuation smoothing rules, it is common that two of the jobs will have very large or small slack values, that is, the extreme cases, which distort the sequencing results. For this reason, such jobs are put in a set EC that will be excluded from calculating the standard deviation:

$$\overline{\mathrm{SKM}'_{ju}} = \frac{\sum_{j=1\ j\notin \mathrm{EC}}^{N} \mathrm{SKM}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}'_{ju}}\right)^2}{N-3}}. \tag{29}$$

The second rule is derived by diversifying the slack in the 2f-TNFSMCT rule, aimed at minimizing the mean cycle time:

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right). \tag{30}$$

To diversify the slackness, the standard deviation of the slack is to be maximized:

$$\overline{\mathrm{SKV}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKV}_{ju}}{N-2}, \tag{31}$$

$$\sigma'_{\mathrm{SKV}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKV}_{ju} - \overline{\mathrm{SKV}'_{ju}}\right)^2}{N-3}}. \tag{32}$$

To generate a biobjective rule, the two rules need to be combined into a single one, for which the following nonlinear programming model is to be optimized:

$$\text{Max } Z = \omega_1 \sigma'_{\mathrm{SKM}_{ju}} + (1-\omega_1)\sigma'_{\mathrm{SKV}_{ju}} \tag{33}$$

s.t.

$$\overline{\mathrm{SKM}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKM}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}'_{ju}}\right)^2}{N-3}},$$

$$\overline{\mathrm{SKV}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKV}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKV}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(SKV_{ju} - \overline{SKV'_{ju}}\right)^2}{N-3}},$$

$$\mathrm{SKM}_{ju} = \left(\frac{\beta}{\alpha\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(R_j - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \min\left(R_j\right)\right)\right),$$

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right),$$

$$0 \le \xi, \qquad \zeta \le 1 \tag{34}$$

which is an NP problem.

3. A Simulation Study

To evaluate the effectiveness of the proposed methodology, simulated data were used to avoid disturbing the regular

operations of the wafer fabrication factory. Simulation is a widely used technology to assess the effectiveness of a scheduling policy, especially when the proposed policy and the current practice are very different. This investigation is not possible to implement in the actual production environment. The real-time scheduling systems will input information very rapidly into the production management information systems (PROMIS). To this end, a real wafer fabrication factory located in Taichung Scientific Park of Taiwan with a monthly capacity of about 25,000 wafers was simulated. The simulation program has been validated and verified by comparing the actual cycle times with the simulated values and by analyzing the trace report, respectively. The wafer fabrication factory is producing more than 10 types of memory products and has more than 500 workstations for performing single-wafer or batch operations using 58 nm~110 nm technologies. Jobs released into the fabrication factory are assigned three types of priorities, that is, "normal," "hot," and "super hot." Jobs with the highest priorities will be processed first. Such a large scale accompanied with reentrant process flows make job dispatching in the wafer fabrication factory a very tough task. Currently, the longest average cycle time exceeds three months with a variation of more than 300 hours. The wafer fabrication factory is therefore seeking better dispatching rules to replace first-in first-out (FIFO) and EDD, in order to shorten the average cycle times and ensure the on-time delivery to its customers. One hundred replications of the simulation are successively run. The time required for each simulation replication is about 30 minute using a PC with Intel Dual CPU E2200 2.2 GHz and 1.99G RAM. A horizon of twenty-four months is simulated.

To assess the effectiveness of the proposed methodology and to make comparison with some existing approaches–FIFO, EDD, SRPT, CR, FSVCT, FSMCT, Justice [27], NFS [16], 2f-TNFSMCT, and 2f-TNFSVCT all of these methods were applied to schedule the simulated wafer fabrication factory to collect the data of 1000 jobs, and then we separated the collected data by their product types and priorities. That is about the amount of work that can be achieved with 100% of the monthly capacity. In some cases, there was too little data, so they were not discussed.

To determine the due date of a job, the PCA-FCM-BPN approach was applied to estimate the cycle time, for which the Levenberg-Marquardt algorithm rather than the gradient descent algorithm was applied to speed up the network convergence. Then, we added a constant allowance of three days to the estimated cycle time, that is, $\kappa = 72$, to determine the internal due date.

Jobs with the highest priorities are usually processed first. In FIFO, jobs were sequenced on each machine first by their priorities, then by their arrival times at the machine. In EDD, jobs were sequenced first by their priorities, then by their due dates. In CR, jobs were sequenced first by their priorities, then by their critical ratios. In the proposed methodology, the nonlinear model with $k = 2$ is used. In Justice, jobs were sequenced on each machine first by their priorities, then according to the job speed matrix (Table 2).

Table 2: The job speed matrix.

		Machine's bottleneck status		
		Hungry	Proper	Crowded
Work progress status	Behind	Rapid	Rapid	Normal
	Just in time	Rapid	Normal	Suspended
	Advanced	Normal	Normal	Suspended

Subsequently, the average cycle time and cycle time standard deviation of all cases were calculated to assess the scheduling performance. With respect to the average cycle time, the FIFO policy was used as the basis for comparison, while FSVCT was compared in evaluating cycle time standard deviation. The results are summarized in Tables 3 and 4.

According to the experimental results, the following points can be made:

(1) For the average cycle time, the proposed methodology outperformed the baseline approach, the FIFO policy. The average advantage was about 16%.

(2) In addition, the proposed methodology surpassed the FSVCT policy in reducing cycle time standard deviation. The most obvious advantage was 59%.

(3) As expected, SRPT performed well in reducing the average cycle times, especially for product types with short cycle times (e.g., product A), but might give an exceedingly bad performance with respect to cycle time standard deviation. If the cycle time is long, the remaining cycle time will be much longer than the remaining processing time, which leads to the ineffectiveness of SRPT. SRPT is similar to FSMCT. Both try to make all jobs equally early or late.

(4) The performance of EDD was also satisfactory for product types with short cycle time. If the cycle time is long, it is more likely to deviate from the prescribed internal due date, which leads to the ineffectiveness of EDD. That becomes more serious if the percentage of the product type is high in the product mix (e.g., product type A). CR has similar problems.

(5) The proposed rule was also compared with the traditional one without slack diversification. Taking product type A with normal priority as an example, the comparison results are shown in Figure 2. Obviously, the proposed rule dominated most of the traditional rules without slack diversification. According to these results, slack diversification did indeed improve the performances of the fluctuation smoothing policies.

4. Conclusions and Directions for Future Research

For capital-intensive industries like wafer fabrication, efficient use of expensive equipment is very important. To this end, job dispatching is a challenging but important task. However, for such a complex production system, to optimize the scheduling performance is a tough task.

Table 3: The performances of various approaches in the average cycle time.

Avg. cycle time (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	1254	400	317	1278	426
EDD	1094	345	305	1433	438
SRPT	948	350	308	1737	457
CR	1148	355	300	1497	440
FSMCT	1313	347	293	1851	470
FSVCT	1014	382	315	1672	475
NFS	1456	407	321	1452	421
Justice	1126	378	322	1576	489
2f-TNFSMCT	1369	379	306	1361	399
2f-TNFSVCT	1465	416	318	1551	500
The proposed methodology	1076	289	269	1132	388

Table 4: The performances of various approaches in cycle time standard deviation.

Cycle time standard deviation (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	55	24	25	87	51
EDD	129	25	22	50	63
SRPT	248	31	22	106	53
CR	69	29	18	58	53
FSMCT	419	33	16	129	104
FSVCT	280	37	27	201	77
NFS	87	49	19	44	47
Justice	120	26	20	69	32
2f-TNFSMCT	75	37	17	47	19
2f-TNFSVCT	38	38	29	33	24
The proposed methodology	86	26	15	54	21

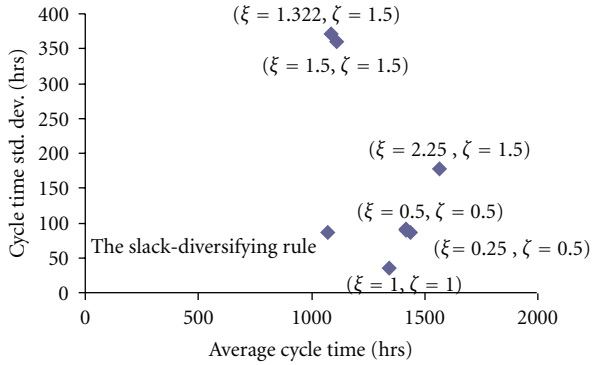

Figure 2: Comparing the slack-diversifying rule with traditional rules without slack diversification.

As an innovative attempt, this study presents a nonlinear programming and artificial neural network approach to optimize the performance of a slack-diversifying dispatching rule in a wafer fabrication factory, to optimize the average cycle time, and to optimize cycle time standard deviation.

The proposed methodology merges two existing rules—2f-TNFSMCT and 2f-TNFSVCT, and constructs a nonlinear programming model to choose the best values of parameters in the two rules. A more effective approach is also applied to estimate the remaining cycle time of a job, which is empirically shown to be conducive to the scheduling performance. To further enhance the accuracy of the remaining cycle time estimation, other dynamic parameters must be considered. In addition, some advanced methods for the cycle time estimation, such as data mining methods [28], can be applied as well.

After a simulation study, we observed the following phenomena.

(1) Through improving the accuracy of estimating the remaining cycle time, the performance of a scheduling rule can indeed be strengthened.

(2) Optimizing the adjustable factors in the two rules appears as an appropriate tool to enhance the scheduling performance of the rule.

(3) Slack diversification is indeed conducive to the performance of a fluctuation smoothing rule.

However, to further assess the effectiveness and efficiency of the proposed methodology, the only way is to apply it to an actual wafer fabrication factory. In addition, other rules can be optimized in the same way in future studies.

Acknowledgment

This work was supported by the National Science Council of Taiwan.

References

[1] C. N. Wang and C. H. Wang, "A simulated model for cycle time reduction by acquiring optimal lot size in semiconductor manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 34, no. 9-10, pp. 1008–1015, 2007.

[2] T. Chen and Y. C. Lin, "A fuzzy-neural fluctuation smoothing rule for scheduling jobs with various priorities in a miconductor manufacturing factory," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 17, no. 3, pp. 397–417, 2009.

[3] T. Chen and Y. C. Wang, "A nonlinear scheduling rule incorporating fuzzy-neural remaining cycle time estimator for scheduling a semiconductor manufacturing factory-a simulation study," *International Journal of Advanced Manufacturing Technology*, vol. 45, no. 1-2, pp. 110–121, 2009.

[4] T. Chen, "Optimized fuzzy-neuro system for scheduling wafer fabrication," *Journal of Scientific and Industrial Research*, vol. 68, no. 8, pp. 680–685, 2009.

[5] D. A. Koonce and S. C. Tsai, "Using data mining to find patterns in genetic algorithm solutions to a job shop schedule," *Computers and Industrial Engineering*, vol. 38, no. 3, pp. 361–374, 2000.

[6] B. W. Hsieh, C. H. Chen, and S. C. Chang, "Scheduling semiconductor wafer fabrication by using ordinal optimization-based simulation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 599–608, 2001.

[7] H. J. Yoon and W. Shen, "A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints," *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, no. 1, pp. 83–91, 2008.

[8] Y. Harrath, B. Chebel-Morello, and N. Zerhouni, "A genetic algorithm and data mining based meta-heuristic for job shop scheduling problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 280–285, October 2002.

[9] K. Sourirajan and R. Uzsoy, "Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication," *Journal of Scheduling*, vol. 10, no. 1, pp. 41–65, 2007.

[10] S. C. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Transactions on Semiconductor Manufacturing*, vol. 7, no. 3, pp. 374–388, 1994.

[11] K. Altendorfer, B. Kabelka, and W. Stöcher, "A new dispatching rule for optimizing machine utilization at a semiconductor test field," in *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC '07)*, pp. 188–193, June 2007.

[12] H. Zhang, Z. Jiang, and C. Guo, "Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology," *International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 110–121, 2009.

[13] T. Chen, "Fuzzy-neural-network-based fluctuation smoothing rule for reducing the cycle times of jobs with various priorities in a wafer fabrication plant: a simulation study," *Proceedings of the Institution of Mechanical Engineers Part B, Journal of Engineering Manufacture*, vol. 223, no. 8, pp. 1033–1043, 2009.

[14] T. Chen, "A tailored non-linear fluctuation smoothing rule for semiconductor manufacturing factory scheduling," *Proceedings of the Institution of Mechanical Engineers Part I, Journal of Systems and Control Engineering*, vol. 223, no. 2, pp. 149–160, 2009.

[15] T. Chen, "Intelligent scheduling approaches for a wafer fabrication factory," *Journal of Intelligent Manufacturing*, vol. 23, no. 3, pp. 897–911, 2012.

[16] T. Chen and Y. C. Wang, "A bi-criteria nonlinear fluctuation smoothing rule incorporating the SOM-FBPN remaining cycle time estimator for scheduling a wafer fab—a simulation study," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 5–8, pp. 709–721, 2010.

[17] T. Chen, Y.-C. Wang, and Y.-C. Lin, "A bi-criteria four-factor fluctuation smoothing rule for scheduling jobs in a wafer fabrication factory," *International Journal of Innovative Computing, Information and Control*, vol. 6, pp. 4289–4303, 2010.

[18] T. Chen, "Dynamic fuzzy-neural fluctuation smoothing rule for jobs scheduling in a wafer fabrication factory," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 223, no. 8, pp. 1081–1094, 2009.

[19] T. Chen, "An optimized tailored nonlinear fluctuation smoothing rule for scheduling a semiconductor manufacturing factory," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 317–325, 2010.

[20] T. Chen, Y. C. Wang, and H. C. Wu, "A fuzzy-neural approach for remaining cycle time estimation in a semiconductor manufacturing factory—a simulation study," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 8, pp. 2125–2139, 2009.

[21] Y. C. Wang, T. Chen, and C. W. Lin, "A slack-diversifying nonlinear fluctuation smoothing rule for job dispatching in a wafer fabrication factory," *Robotics & Computer Integrated Manufacturing*. In press.

[22] T. Chen and T. Wang, "Enhancing scheduling performance for a wafer fabrication factory: the bi-objective slack-diversifying nonlinear fluctuation-smoothing rule," *Computational Intelligence and Neuroscience*. In press.

[23] T. Chen and M. Huang, "A fuzzy-neural slack-diversifying NFS rule for job dispatching in a wafer fabrication factory," *ICIC Express Letters*, vol. 6, no. 9, pp. 2243–2248, 2012.

[24] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[25] T. Chen and Y. C. Lin, "A fuzzy back propagation network ensemble with example classification for lot output time prediction in a wafer fab," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 658–666, 2009.

[26] T. Chen, Y. C. Wang, and H. R. Tsai, "Lot cycle time prediction in a ramping-up semiconductor manufacturing factory with a SOM-FBPN-ensemble approach with multiple buckets and partial normalization," *International Journal of Advanced Manufacturing Technology*, vol. 42, no. 11-12, pp. 1206–1216, 2009.

[27] T. Nakata, K. Matsui, Y. Miyake, and K. Nishioka, "Dynamic bottleneck control in wide variety production factory," *IEEE Transactions on Semiconductor Manufacturing*, vol. 12, no. 3, pp. 273–280, 1999.

[28] T. Chen, "Job cycle time estimation in a wafer fabrication factory with a bi-directional classifying fuzzy-neural approach," *International Journal of Advanced Manufacturing Technology*, vol. 56, pp. 1007–1018, 2011.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 410832, 7 pages
doi:10.1155/2012/410832

Research Article

Modeling Chaotic Behavior of Chittagong Stock Indices

Shipra Banik, Mohammed Anwer, and A. F. M. Khodadad Khan

School of Engineering and Computer Science, Independent University, Bangladesh, Dhaka 1212, Bangladesh

Correspondence should be addressed to Shipra Banik, shiprabanik@yahoo.com.au

Received 18 December 2011; Revised 16 May 2012; Accepted 4 June 2012

Academic Editor: Yi-Chi Wang

Stock market prediction is an important area of financial forecasting, which attracts great interest to stock buyers and sellers, stock investors, policy makers, applied researchers, and many others who are involved in the capital market. In this paper, a comparative study has been conducted to predict stock index values using soft computing models and time series model. Paying attention to the applied econometric noises because our considered series are time series, we predict Chittagong stock indices for the period from January 1, 2005 to May 5, 2011. We have used well-known models such as, the genetic algorithm (GA) model and the adaptive network fuzzy integrated system (ANFIS) model as soft computing forecasting models. Very widely used forecasting models in applied time series econometrics, namely, the generalized autoregressive conditional heteroscedastic (GARCH) model is considered as time series model. Our findings have revealed that the use of soft computing models is more successful than the considered time series model.

1. Introduction

The stock index values play an important role in controlling dynamics of the capital market. As a result, the appropriate prediction of stock index values is a crucial factor for domestic/foreign stock investors, buyers and/or sellers, fund managers, policy makers, applied researchers (who want to improve the model specifications of this index), and many others. Many researchers, for example, [1–4] and others have found that the empirical distribution of stock is significantly nonnormal and nonlinear. Stock market data are also observed in practice chaotic and volatile by nature (e.g., see [5–8]). That is why stock values are hard to predict. Traditionally, the fundamental Box-Jenkins analysis has been the mainstream methodology that is used to predict stock values in applied literature. Due to continual studies of stock market experts, the use of soft computing models (such as artificial neural networks, fuzzy set, evolutionary algorithms, and rough set theory.) have been widely established to forecast stock market. Evidence [9, 10] suggests that the Box-Jenkins approach often fails to predict time series when the behavior of series is chaotic and nonlinear. Thus, soft computing systems have emerged to increase the accuracy of chaotic time series predictions. The reason is that these systems have the potential to provide a viable solution through the versatile approach to self-organization. Thus, in forecasting literatures [11–14], it has been found that soft computing systems yield better results compared to the statistical time series approaches when the series is chaotic. This paper compares forecasts of stock prices from soft computing forecasting models and the model introduced by [15]. Our motivation for this comparison lies in the recent increasing interest in the use of soft computing models for forecasting purposes of economic and finance variables. Thus, soft computing models are used to learn the nonlinear and chaotic patterns in the stock system. Several studies [7, 11, and many others] have compared soft computing models and the traditional Box-Jenkins model. However, there are only a few comparative analyses (according to our knowledge) between soft computing models and standard time series statistical models [13] in case of Bangladeshi stock indices. In this paper, we examine the performance of the daily Chittagong stock market indices using soft computing models and time series model. See [13] for prediction of the daily Dhaka stock market index values. Thus, we hope that findings of the study will be interesting to fund managers, many businesses investors, policy makers, academics, and others who are involved in this volatile

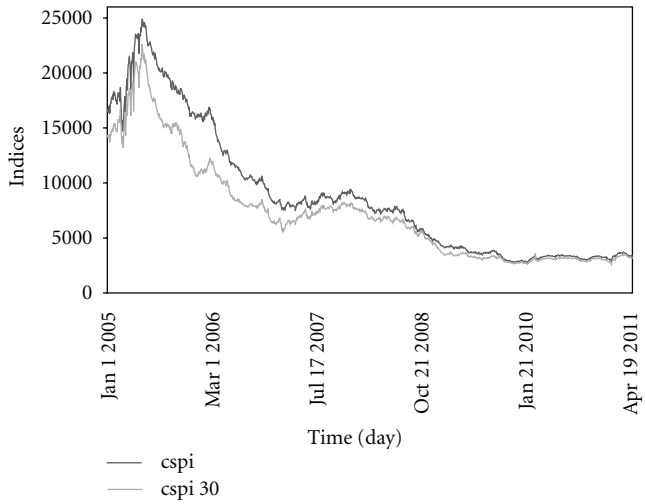

Figure 1: Time plots of the indices.

market. The structure of the paper is as follows: data and forecasting model is given in the next section. Statistical properties and various econometric noises are discussed in Section 3. A brief description of the considered forecasting models is described in Section 4. Performances of the different evaluation criterion are explained in Section 5. Finally, concluding remarks with some proposed future research are given in the final section.

2. Data and Forecasting Model

2.1. Data. The indices, the Chittagong Stock Exchange has been maintaining since October 10 1995, are the stock prices for all companies (cse-all) and for 30 selected companies (cse-30). Thus, we considered the daily cse-all and cse-30 [data source: http://www.cse.com.bd/] prices for the available periods (January 1, 2005 to May 5, 2011). For a description of the indices, see the above website.

It is true people trend to invest in stock market because it has high returns over time. Stock markets are generally affected by economic, social, political, and even psychological factors. These factors interact with each other in a very complicated manner. That is why stock data are observed, chaotic and volatile by nature. It is well known that chart is the best way to visualize trend and chaotic behavior if present in any price series. Thus, to understand the behaviors of considered indices, cse-all and cse-30 are plotted against time in Figure 1. It is very clear that there is a decreasing trend with respect to time. There are some reasons why these sorts of trend exist. See http://www.cse.com.bd/ for details. It is also observable from this plot the behaviors of these selected price series are not linear. This means series can appear volatile with moves that look chaotic. Some sort of nonlinearity can also be present in the selected series.

2.2. Forecasting Models. Since our series are time series, so we have selected the most commonly used time series model,

Table 1: Data description and proposed model.

Indices	Size	Proposed AR(p) model
cse-all	1540×1	AR(2)
cse-30	1540×1	AR(4)

namely, the autoregressive (AR) model of order p. The model is defined for each of considered series as follows:

$$\text{cse-all}_t = \alpha + \beta t + \rho_i \text{cse-all}_{t-i} + e_t, \quad t = 1, 2, 3, \ldots, n, \tag{1}$$

$$\text{cse-30}_t = \alpha + \beta t + \rho_i \text{cse-30}_{t-i} + e_t, \quad t = 1, 2, 3, \ldots, n, \tag{2}$$

where α is an intercept, β is the deterministic trend, t is the time variable, ρ_i are the lag orders of the AR(p) model, and $e_t \sim N(0, \sigma^2)$. The appropriate lags of the series are selected by the Bayesian Information Criterion (BIC). Other information criterions, for example, Akaike Information Criterion (AIC), Schwarz Information Criterion (SIC), and others can also be used to select the lag order of the AR components of our selected models. See Table 1 for data size and proposed AR(p) model.

3. Statistical Properties of Data

3.1. Numerical Summary. To understand the characteristics of the selected indices, summary statistics are tabulated in Table 2. It is clear from the above table, most of times, that cse-all and cse-30 indices are observed 8691 and 7259.9, respectively. Standard deviation measures confirm to us that the considered prices are not equal to 8691 and 7259.9. The expected range of cse-all and cse-30 prices can be estimated as 8691 ± 5883.3 and 7259.9 ± 4668.6, respectively. Skewness measures indicate to us that stock market indices display right skewed distributions. It means that most of prices are below the average prices. Kurtosis measures also indicate to us that price indices are not normal.

3.2. Time Series Properties. It is now a well-established stylized fact that most time series are nonstationary and contain a unit root (e.g., see [14]). The conventional approach of time series is based on the implicit assumption that the underlying data series is stationary. This assumption was rarely questioned until the early 1970s and numerical analysis proceeded if all-time series were stationary. Numerous studies (e.g., [14–16] and many others) have suggested that most time series are nonstationary and therefore, the assumption of stationarity is unrealistic. Thus, prior to model specifications and the estimations, the stationary property of the data series is routinely tested. Otherwise, the study can yield unrealistic results. That is why to select appropriate forecasting model for our study, we have tested first stationarity property of the considered series.

3.2.1. Stationarity Tests. There are many stationarity tests available in time series literature. For details, see [17–19] and

Table 2: Numerical properties of the selected indices.

Indices	Mean	Median	Standard deviation	Skewness	Kurtosis
cse-all	$8.6916e+003$	$7.6318e+003$	$5.8833e+003$	0.5404	2.9063
cse-30	$7.2599e+003$	$6.5434e+003$	$4.6686e+003$	0.4604	3.6754

Table 3: Stationarity test results.

Indices	ADF(p) statistic	P value (critical value) for the ADF test	PP(L) statistic	P value (critical value) for the PP test
cse-all	$-0.9888(2)$	$0.9434(-3.4144)$	$-1.13(12)$	$0.9227(-3.4144)$
cse-30	$-1.2330(4)$	$0.9022(-3.4144)$	$-1.29(10)$	$0.8877(-3.4144)$

"p" and "L" indicate lag order to remove serial correlation.
Decision rule: If P value < level of significance (α), then accept null hypothesis.

Table 4: Linearity test results.

Indices	cse-all	cse-30
Engle test statistic	53.52	54.35
P value	0.0	0.0

others. To test the nonstationarity behavior of our considered models (1)-(2), we have used most commonly applied unit root tests, namely, the Augmented Dickey Fuller (ADF) test proposed by Said and Dickey [20] and the test proposed by Phillips and Perron (PP) [21]. For test procedures, see [17]. MATLAB commands that *Adftest* and *pptest* are used to compute the ADF and the PP statistics and results are reported in Table 3. Note that under the null hypothesis of the ADF and PP tests the series assumed nonstationary and that under the alternative hypothesis the series is stationary. Results show us all series are nonstationary (because P value $> \alpha, \alpha = 0.05$). Thus, null hypothesis of the tests is accepted. Then we have taken the first difference of the series to remove non-stationarity and applied then again the ADF test and the PP test. These test results show us in first differences that considered series are stationary. These results are not reported for spaces but are available on request. The effect of these tests will be shown when our forecasting model is used, the model is used in first differences.

3.3. Linearity Test. There are many statistical techniques available in literature to test whether the series is linear or nonlinear. To select appropriate forecasting method, we have tested also linearity of the considered models (1) and (2). These tests are based on the ordinary least square method residuals. The statistical test proposed by Engle [5] is used to test the presence of nonlinear dependence. For details of the test procedure, see [22]. Linearity test results are tabulated in Table 4. Results show that at the 5% level of significance (α), nonlinearity is present (check the P-value) in our considered series. Just a note here under the null hypothesis, the series are considered linear and under the alternative hypothesis, the series are considered nonlinear. Table 4 results show us P-value is less than α which indicates rejection of the null hypothesis. So Table 4 results confirm to us that our considered series are nonlinear.

Table 5: GA options.

Step	Algorithm option
Creation	Uniform, normal
Fitness scaling	Rank, proportional, top (truncation), linear scaling, shift
Selection	Roulette, stochastic uniform selection, tournament, uniform
Crossover	Binary-valued (single-point, two-point, n-point, uniform) real-valued (intermediate, line)
Mutation	Gaussian, uniform
Reinsertion	Pure, uniform, elitist, fitness-based

4. Models Used for Prediction

Considered statistical tests results show that our selected series are nonstationary, nonlinear, and chaotic (Figure 1). To remove nonstationarity, we used the series in first differences. We have selected nonlinear forecasting models to forecast Chittagong stock indices, which has also the ability to capture chaotic behavior. We have chosen the following models to forecast considered indices. A brief description of the considered forecasting models is given below.

4.1. Soft Computing Model. We have chosen two very popular and widely used models, namely, the genetic algorithm (GA) model and the neuro-fuzzy model.

4.1.1. GA Model. Holland [23] introduces this technique. It is a technique based on the "Darwin's Principle of Natural Selection" and is used to solve optimization problems. The basic idea is to select the best, discard the rest. To handle the complex multidimensional behaviors of a system, this approach has been used efficiently in forecasting literature (e.g., [23–26] and others). See Figure 2, for the flowchart that illustrates the basic steps in a GA. See Table 5, for the standard GA options. A brief explanation of each step is as follows.

Step 1. Create an initial population consisting of random chromosomes. To understand the GA process, for example,

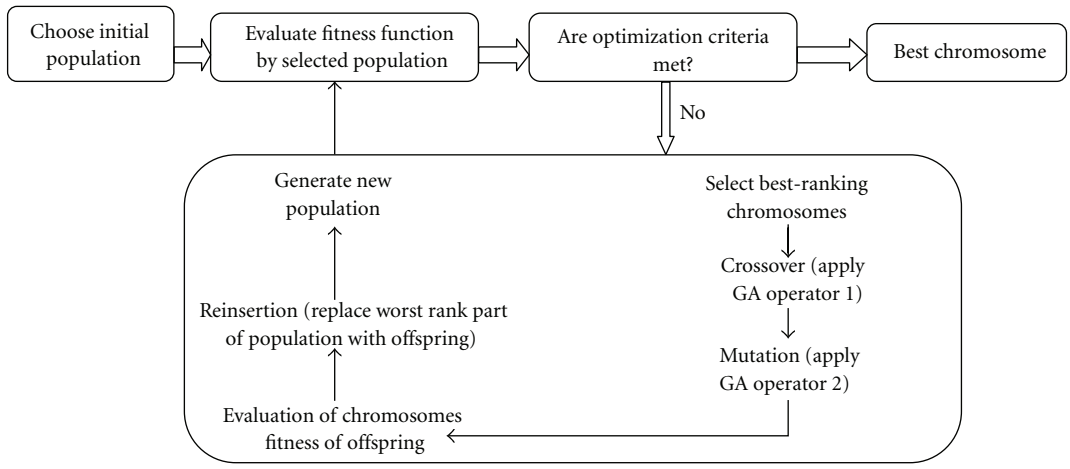

Figure 2: Flowchart of a GA model.

Table 6

RMSE:	26.19	32.09	53.75	20.18	18.67	66.64
Fit (RMSE):	1.2	0.8	0.4	1.6	2.0	0
pdf of RMSE:	0.2	0.13	0.06	0.26	0.33	0
cdf of RMSE:	0.2	0.33	0.39	0.65	0.98	1.0

for an AR(2) model, consider the following random population of 6 chromosomes with 4 parameters each: (0.13, 0.01, 0.84, 0.68), (0.20, 0.74, 0.52, 0.37), (0.19, 0.44, 0.20, 0.83), (0.60, 0.93, 0.67, 0.50), (0.27, 0.46, 0.83, 0.70), and (0.19, 0.41, 0.01, 0.42). Population size is chosen usually from 100–500. Larger population may produce more robust solution.

Step 2. Fitness scaling is used to provide a measure of how selected chromosomes perform in the problem domain. The AR(2) model fitness is evaluated through a criterion like RMSE (CC, MAE can be also used). For the AR(2) model, we get RMSE: 26.19, 32.09, 53.75, 20.18, 18.67, and 66.64. Using the linear-ranking process, for example, (details, see Pohlheim [27]), Fit(RMSE): 1.2, 0.8, 0.4, 1.6, 2.0, 0.

Step 3. Based on Step 2 results, choose parents for the next generation. To understand it, consider the distribution found in Tables 5 and 6.

It is observed that chromosome 5 is the fittest chromosome, because it occupies the largest interval, whereas chromosome 3 is the second least fit chromosome as the smallest interval. Chromosome 6 is the least fit interval that has a fitness value of 0 and gets no chance for reproduction. Using for example, the roulette wheel method (purpose is to eliminate the worst chromosomes and to regenerate better substitutes), selected 4 parents are: (0.20, 0.74, 0.52, 0.37), (0.13, 0.01, 0.84, 0.68), (0.27, 0.46, 0.83, 0.70), and (0.13, 0.01, 0.84, 0.68).

Next step is to produce offspring from selected parents by combining entries of a pair of parents (known as crossover) and also by making random changes to a single parent (known as mutation).

Step 4 (GA operator-1). Basic operator for producing new (improved) chromosomes is known as crossover (a version of artificial mating). It produces offspring that have some parts of both parents genetic material. Offspring are produced using the intermediate crossover method, because this is a method proposed to recombine for parents with real-valued chromosomes (see details, Pohlheim [27]). Thus, crossover offspring are (0.16, 0.16, 0.85, 0.57), (0.13, 0.22, 0.76, 0.43), (0.13, 0.15, 0.83, 0.69), and (0.26, 0.45, 0.84, 0.68).

Step 5 (GA operator-2). Offspring are mutated after producing crossover offspring and this GA operator increases the chance that the algorithm will generate better fittest RMSE than the Step 4. GA creates 3 types of offspring: elite offspring (number of best RMSE values in the current generation that are guaranteed to survive to the next generation), crossover offspring, and mutation offspring. To understand it, consider an example: suppose that the population size is 20 and the elite count is 2. If crossover fraction is 0.8, then the distribution of offspring is 2 elites, 14 (18*0.8) are crossover offspring, and the remaining 4 are mutation offspring. Just to know, a crossover fraction of 1 means all offspring other than elite are crossover offspring, while a crossover fraction of 0 means that all offspring are mutation offspring. How offspring are produced under the mutation process, see Pohlheim [27]. The mutation offspring are found (0.16, 0.17, 0.85, 0.56), (0.13, 0.22, 0.76, 0.43), (0.13, 0.14, 0.83, 0.69), (0.26, 0.45, 0.84, 0.68), respectively.

Step 6. Once offspring have been produced using Steps 4–5, offspring fitness (i.e., RMSE values) must be determined (procedure similar to Step 2). We get improved RMSE: 23.37, 28.13, 24.11,18.62.

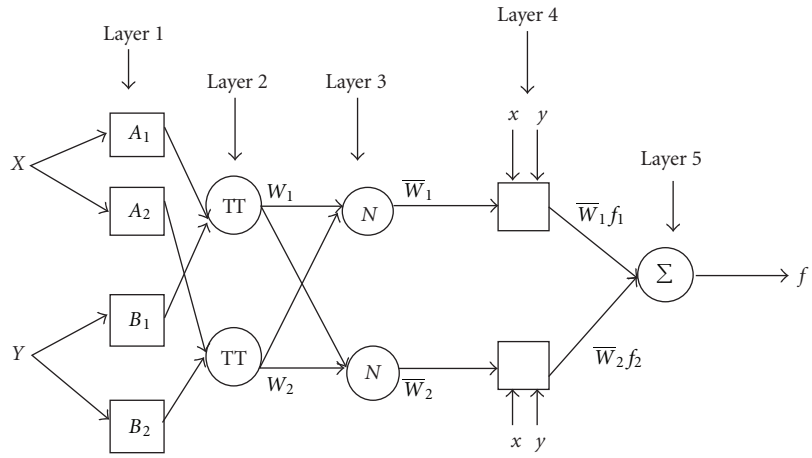

Figure 3: AN ANFIS system.

If less offspring are produced than the original population size, then to maintain size, offspring have to be reinserted into the old population. By this step, it is determined which chromosomes will be replaced by offspring. Using for example, the fitness-based reinsertion method, the following RMSE are found: 20.18, 18.67, 18.62, 23.37, 24.11, and 28.13.

If termination criteria are not defined, GA returns to Step 3 and continues up to Step 6. It is satisfied when either maximum number of generations is achieved or when all chromosomes in the population are identical (i.e., converge). The creator sets this number before running GA, which ensures that GA does not continue indefinitely.

4.1.2. Adaptive Network Fuzzy Integrated System (ANFIS) Model. The second widely used soft computing model we have selected is the ANFIS model. In computing the literature, Jang [8] proposed this model. This model is a combination of two intelligence systems: neural network (NN) system and fuzzy inference system (FIS). This is also known as NN-fuzzy integrated system, where the NN learning algorithm is used to determine parameters of FIS. NNs are nonlinear statistical data modeling tools and can capture and model any input-output relationships. FIS is the process of formulating the mapping from a given input to an output using the fuzzy logic. This mapping provides a basis from which decisions can be made or patterns can be discerned. The process of FIS involves membership functions (mfs), fuzzy logic operators, and if-then rules. The structure of ANFIS (see Figure 3 for its architecture) has 5 layers: (a) 1 input layer, (b) 3 hidden layers that represents mfs and fuzzy rules, and (c) 1 output layer. ANFIS uses the Sugeno-fuzzy inference model to be the learning algorithm. As an example, the fuzzy if-then rules for the first-order Sugeno-fuzzy model can be expressed as follows.

Rule 1. If x (input 1) is A_1 and y (input 2) is B_1, then f_1 (output) $= p_1x + q_1y + r_1$.

Rule 2. If x (input 1) is A_2 and y (input 2) is B_2, then f_2 (output) $= p_2x + q_2y + r_2$.

The learning algorithm of ANFIS is a hybrid algorithm, which combines the gradient descent (GD) method and the least square estimation (LSE) for an effective search of parameters. ANFIS uses a two-pass of learning algorithm to reduce error: forward pass and backward pass. The hidden layer is computed by the GD method of the feedback structure and the final output is estimated by the LSE method (for details, see [8]).

4.2. Time Series Model: GARCH Model. Since our considered series are time series, to compare the performances of the soft computing models, a very popular time series model, namely, generalized autoregressive conditional heteroscedastic (GARCH) model is selected from time series econometrics literature. A brief description of this model is discussed.

In 1986, Bollerslev invented the GARCH model. To understand it clearly, consider an AR(1) model:

$$\text{stockprice}_t = \text{Constant} + \text{stockprice}_{t-1} + e_t, \quad t = 1, 2, \ldots, n, \tag{3}$$

where stockprice_t is the observed cse-all and cse-30 prices. Here, the current volatility depends not only on the past errors, but also on the past volatilities. Suppose $e_t = \varepsilon_t\sigma_t$, where $\sigma_t = \beta_0 + \sum_{i=1}^{q}\beta_i e_{t-i}^2 + \sum_{j=1}^{p}\gamma_j\sigma_{t-j}$ with $\beta_0 > 0, \beta_i \geq 0$ $(i = 1, 2, \ldots, q), \gamma_j \geq 0$ $(j = 1, 2, \ldots, p)$, and $\varepsilon_t \sim N(0, 1)$, known as an GARCH process of orders p and q. This model is widely used to know the volatility nature that exists generally in the time series data. For details, see [6].

5. Discussion of Results

Forecasting with 100% accuracy may be impossible, but we can do our best to reduce forecasting errors. Thus, to find the error level between observed and predicted stock series that means the forecasting performances of the considered models are evaluated against the following widely used statistical measures, namely, root mean square error (RMSE), correlation coefficient (CC), and coefficient of determination (R^2).

Table 7: Computational settings for selected forecasting models.

GA	ANFIS	GARCH
(1) Population size—3000, generated at random from the uniform probability distribution.	(1) Input—Lags of cse-all and cse-30	cse-all and cse-30: Fitted GARCH (2,1) and GARCH (1,1) Selection Process: BIC.
(2) Fitness function—RMSE of model (2.2.1-2.2.2); Fitness scaling—Linear ranking process.	(2) 2 mfs (type Gaussian) for each of input variables. Thus, 8 if-then fuzzy rules were learned.	
(3) Selection-Roulette wheel.		
(4) Crossover-fraction: 0.95; Intermediate method.		
(5) Mutation-fraction: 0.05; Uniform method.		
(6) Reinsertion: The fitness-based reinsertion method.		
(7) Termination Criteria—Maximum number of generations, assumed 200.		

Table 8: Performances for training data.

Forecasting models	cse-all			cse-30		
	RMSE	CC	R^2	RMSE	CC	R^2
GA	6.79	0.94	0.88	6.95	0.91	0.82
ANFIS	6.89	0.93	0.86	6.83	0.92	0.84
GARCH	7.34	0.91	0.82	7.56	0.88	0.77

Note: We considered January 1, 2005 to January 1, 2009 as a training period.

Table 9: Performances for testing data.

Forecasting models	cse-all			cse-30		
	RMSE	CC	R^2	RMSE	CC	R^2
GA	5.37	0.96	0.92	6.48	0.94	0.88
ANFIS	6.93	0.94	0.88	5.59	0.97	0.94
GARCH	8.58	0.89	0.79	7.89	0.90	0.81

Note: We considered February 1, 2005 to May 5, 2011 as a testing period.

Note that a smallest value of RMSE indicates higher accuracy in forecasting, and higher R^2 value indicates better prediction. All computational works were carried out using the programming code of MATLAB (version 7.0). We have selected January 1, 2005 to January 1, 2009 as the training periods and rest of periods as the testing periods. See Table 7 for computational parameters for all selected forecasting models. Tables 8 and 9 summarize the performances of different considered forecasting models where the training and testing data are achieved for prediction of stock values using the considered error measures RMSE, CC, and R^2. In terms of all measures, our training results show that for the cse-all price series, the GA forecasting model performed better (noted smallest RMSE values, highest CC, and R^2 values) than other forecasting models, followed by the ANFIS forecasting model and the GARCH forecasting model. For the cse-30 series, we found that the ANFIS forecasting model performed better than the other forecasting models. After the models are built using the training data, considered series forecasted over the testing data and performances are reported in Table 9. The testing results when compared to our considered forecasting models show again that the daily cse-all price series forecasting ability of the GA forecasting model is higher than the other forecasting models. We noted for the cse-30 price index, the ANFIS forecasting model performed (lowest value of RMSE, highest values of CC and R^2) better than the other forecasting models.

6. Conclusion and Future Works

It is well known that soft computing models pay particular attention to nonlinearties which in turn help to improve complex data predictions. In this paper, we forecasted Chittagong stock price index for all companies and stock price index for 30 selected companies for the period from January 1, 2005 to May 5, 2011. Recent time series literature suggests that most stock price series are nonstationary, contains a unit root. For this reason to make appropriate predictions, using unit root tests, first we tested nonstationarity properties because our considered series are time series. Our test results suggested that the series are nonstationary. To remove this noise from the series, we used the series in first differences. Then we tested linearity of the series using the statistical linear test. Test result showed us that the two considered series are nonlinear. Thus, we selected two very well-known soft computing models, namely, the GA forecasting model and the ANFIS forecasting model. To compare the performances of these two models, we also selected most popular nonlinear time series forecasting model. According to our findings, we would like to conclude that applied workers should select the GA forecasting model to forecast future daily stock price index for all selected companies. In case of daily stock price index for 30 selected companies, the ANFIS forecasting model is more successful than the other considered forecasting models. We believe our findings will be helpful for researchers who are planning to make appropriate decisions with this complex variable. Our next

step is to improve and compare the predictions with other recently proposed model, for example, rough set theory and other. This is left for future research.

Acknowledgments

The authors are grateful to the participants of the 17th International Mathematics conference held on 22–24 December 2011 and organized by the Bangladesh Mathematical Society and the Department of Mathematics, Jahangirnagar University, Dhaka, Bangladesh. They greatly acknowledge comments and very useful suggestions from anonymous referees, which improved greatly the presentation of the paper. They are also very thankful to the Editor Yi-Chi Wang and editorial staff Badiaa Sayed for their very valuable cooperation. They greatly acknowledge it.

References

[1] B. Mandelbrot, "The variation of certain speculative prices," *Journal of Business*, vol. 36, pp. 394–419, 1963.

[2] E. F. Fama, "The behavior of stock market prices," *Journal of Business*, vol. 38, pp. 34–105, 1965.

[3] D. A. Hsu, R. B. Miler, and D. W. Wichern, "On the stable paretian behavior of stock market prices," *Journal of the American Statistical Association*, vol. 69, pp. 108–113, 1974.

[4] D. Kim and S. J. Kon, "Alternative models for the conditional heteroskedasticity of stock returns," *Journal of Business*, vol. 67, pp. 563–598, 1994.

[5] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of UK Inflation," *Econometrica*, vol. 50, pp. 987–1008, 1982.

[6] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[7] S. Banik, M. Anwer, K. Khan, R. A. Rouf, and F. H. Chanchary, "Neural network and genetic algorithm approaches for forecasting bangladeshi monsoon rainfall," in *Proceedings of the 11th International Conference on Computer and Information Technology (ICCIT '08)*, December 2008.

[8] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[9] D. F. Cook and M. L. Wolfe, "A back-propagation neural network to predict average air temperatures," *AI Applications in Natural Resource Management*, vol. 5, no. 1, pp. 40–46, 1991.

[10] A. Abraham, N. S. Philip, and P. Saratchandran, "Modeling chaotic behavior of stock indices using intelligent paradigms," *International Journal of Neural, Parallel and Scientific Computations*, vol. 11, no. 1-2, pp. 143–160, 2003.

[11] J. Kamruzzaman and R. A. Sarker, "Comparing ANN based models with ARIMA for prediction of forex rates," *Bulletin of the American Schools of Oriental Research*, vol. 22, no. 2, pp. 2–11, 2003.

[12] G. E. P. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*, Cambridge University Press, Cambridge, UK, 1970.

[13] S. Banik, F. H. Chanchary, R. A. Rouf, and K. Khan, "Modeling chaotic behavior of Dhaka Stock Market Index values using the neuro-fuzzy model," in *Proceedings of the 10th International Conference on Computer and Information Technology (ICCIT '07)*, pp. 80–85, December 2007.

[14] C. R. Nelson and C. R. Plosser, "Trends and random walks in macroeconmic time series. Some evidence and implications," *Journal of Monetary Economics*, vol. 10, no. 2, pp. 139–162, 1982.

[15] W. F. Mitchell, "Testing for unit roots and persistence in OECD unemployment rates," *Applied Economics*, vol. 25, no. 12, pp. 1489–1501, 1993.

[16] R. S. McDougall, "The seasonal unit root structure in New Zealand macroeconomic variables," *Applied Economics*, vol. 27, pp. 817–827, 1995.

[17] W. H. Greene, *Econometric Analysis*, Prentice Hall, Upper Saddle River, NJ, USA, 7th edition, 2008.

[18] S. Banik, *Testing for Stationarity, Seasonality and Long Memory in Economic and Financial Time Series [Ph.D. thesis]*, School of Business, La Trobe University, Bundoora, Australia, 1999, Unpublished.

[19] S. Banik and P. Silvapulle, "Testing for seasonal stability in unemployment series: international evidence," *Empirica*, vol. 26, no. 2, pp. 123–139, 1999.

[20] S. E. Said and D. A. Dickey, "Testing for unit roots in autoregressive-moving average models of unknown order," *Biometrika*, vol. 71, no. 3, pp. 599–607, 1984.

[21] P. C. B. Phillips and P. Perron, "Testing for a unit root in time series regression," *Biometrika*, vol. 75, no. 2, pp. 335–346, 1988.

[22] R. L. Thomas, *Modern Econometrics: An Introduction*, Addision-Wesley, New York, NY, USA, 1997.

[23] J. N. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[24] Y. H. Lee, S. K. Park, and D. E. Chang, "Parameter estimation using the genetic algorithm and its impact on quantitative precipitation forecast," *Annales Geophysicae*, vol. 24, no. 12, pp. 3185–3189, 2006.

[25] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.

[26] Z. Wei, W. U. Zhi-ming, and Y. Gen-Ke, "Genetic programming-based chaotic time series modeling," *Journal of Zhejiang University*, vol. 5, no. 11, pp. 1432–1439, 2004.

[27] H. Pohlheim, *Documentation for Genetic and Evolutionary Algorithm Toolbox for Use with MATLAB*, 2005.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 705483, 21 pages
doi:10.1155/2012/705483

Research Article

Qualitative Functional Decomposition Analysis of Evolved Neuromorphic Flight Controllers

Sanjay K. Boddhu and John C. Gallagher

Department of Computer Science and Engineering, Wright State University, Dayton, OH 45435, USA

Correspondence should be addressed to Sanjay K. Boddhu, sboddhu@cs.wright.edu

Received 21 February 2012; Accepted 21 May 2012

Academic Editor: P. Balasubramaniam

In the previous work, it was demonstrated that one can effectively employ CTRNN-EH (a neuromorphic variant of EH method) methodology to evolve neuromorphic flight controllers for a flapping wing robot. This paper describes a novel frequency grouping-based analysis technique, developed to qualitatively decompose the evolved controllers into explainable functional control blocks. A summary of the previous work related to evolving flight controllers for two categories of the controller types, called autonomous and nonautonomous controllers, is provided, and the applicability of the newly developed decomposition analysis for both controller categories is demonstrated. Further, the paper concludes with appropriate discussion of ongoing work and implications for possible future work related to employing the CTRNN-EH methodology and the decomposition analysis techniques presented in this paper.

1. Introduction

Most, if not all, existing bird-sized and insect-sized flapping-wing vehicles possess only a small number of actively controlled degrees of freedom. In these vehicles, the bulk of the wing motions are generated via a combination of actively driven linkages (motors and armatures, piezoelectric beams, etc.) and passively driven elements (wing flex or rotation via dynamic pressure loading, etc.) [1, 2]. The number of controlled degrees of freedom is often minimized to simplify control and to limit the number of bulky actuators carried on board. In theory, both bird-sized [1] and insect-sized [2] robots can sustain stable flight with controllers generating actuation signals for only few degrees of freedom. But it would require taking advantage of every possible degree of freedom available in the robot to achieve sophisticated maneuvers that are possible in their biological counterparts. Thus, there exists a possibility that applying a learning or adaptable controller techniques [1, 3–6] to the control of the insect-sized flapping wing vehicles, hereafter referred to as Microlevel Flapping Wing Robots (MFWRs), will likely to produce more biomimetic control and maneuver patterns that evade traditional controller design. One can imagine two basic approaches to the "adaptable controller" problem. First, one might attempt to hybridize an adaptive system to a traditional controller in the hope that the combined system could learn the specific needs of an individual vehicle by augmenting a base controller. Second, one might attempt to construct an adaptable controller that could learn acceptable control laws tabula rasa either all-at-once or via a staged approach. Even if tabula rasa methods could be made to work, one would incur a responsibility to explain the operation of controllers that, though functional, might operate in ways not conformant with existing explanatory paradigms. In previous work [4, 7–9], the authors were able to demonstrate controllers could be "learned from scratch" by verifying the idea within a framework of neuromorphic evolvable hardware. Further, this previous work also demonstrated the feasibility of neuromorphic adaptive hardware implementations that provide computational advantage over existing adaptive control techniques using similar neural substrates [10, 11]. The work mentioned in this paper focuses more intensely on that problem of explaining what those controllers do and how they do it. It will discuss subsequent work that was undertaken to analyze those evolved flight controllers with newly developed frequency-based and

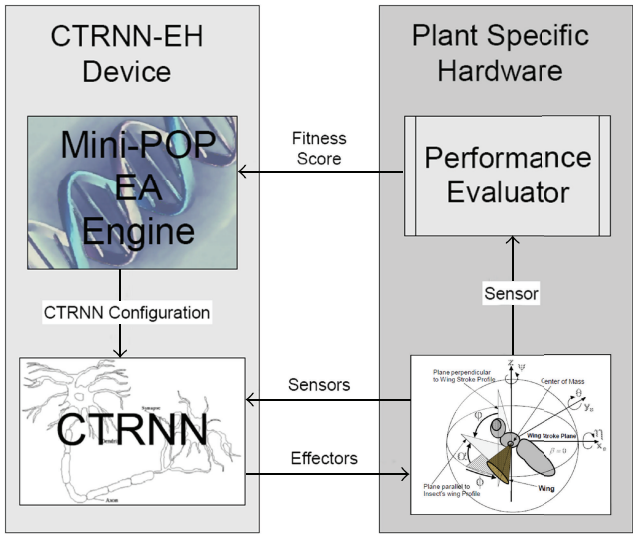

Figure 1: Schematic of CTRNN-EH Framework.

preexisting modularization decomposition methods. In this vein, the background knowledge necessary to understand the terminology, methods, and approaches employed as part of the above-mentioned CTRNN-EH framework are briefly explained in Section 2. Following is Section 3 describing the specific methods and models employed to successfully evolve CTRNN-EH flight controllers for a specific MFWR model. Section 4 describes the details of the proposed analysis methods and their applicability to the evolved flight controllers, followed by Section 5 with concluding remarks on the current work and ongoing future work.

2. Background and Previous Work

2.1. CTRNN-EH Framework. The CTRNN-EH framework introduced in the previous works [5, 12, 13] is summarized schematically in Figure 1. The CTRNN-EH framework is a neuromorphic variant of the standard Evolvable Hardware paradigm using Continuous Time Recurrent Neural Networks (CTRNNs) as the reconfigurable hardware substrate. CTRNNs are networks of Hopfield continuous model neurons [13] with unconstrained connection weight matrices. CTRNN neural activity and outputs are governed by an nth degree differential equation of the form:

$$\tau \frac{dy_i}{dt} = -y_i + \sum_{j\text{-}1}^{N} w_{ji}\sigma\left(y_j + \theta_j\right) + s_i I_i(t), \qquad (1)$$

where y_i is the state of neuron i, τ_i is time constant of neuron i, w_{ji} is the strength of the connection from the jth to ith the neuron, θ is a bias term, $\sigma(x) = 1/(1 + e^{-x})$ is the standard logistic activation function, and $I_i(t)$ represents a weighted sensory input with strength s_i. The set of parameters defining a neuron I (incoming weights, time constant, and bias) are called an individual *neuron configuration*, and a collection of these individual neuron configurations for a given network is called a *network configuration* or *CTRNN configuration*. These CTRNNs have been proven to be universal approximators of any smooth dynamics [14]. The practical benefit of this is that in principle, any possible control law can be approximated arbitrarily well given enough CTRNN neurons. In practice, even small networks of CTRNN neurons are capable of producing complex dynamical behavior. Based on the Evolvable Hardware (EH), principle of evolving optimized and desired configurations in a reconfigurable substrate using evolutionary algorithm techniques. CTRNNs are evolved to produce, the right control signal, using the evolutionary algorithms. The training of the CTRNNs is finding the appropriate parameter settings, that is, configuring the neuron settings in the network. The evolutionary algorithms search the given possible settings of the neuron and find the optimal settings for the network as a whole, to produce the required control signals. The CTRNNs functioning with optimal settings produced by the EA is called as the evolved controller for the given control problem being dealt. Based on previous work, the Minipop algorithm [15] is chosen as the evolutionary algorithm for the CTRNN-EH framework. The Minipop algorithm is a light weight evolutionary algorithm driven by mutation and hypermutation [15], more details of the same can be found in [13, 15].

2.2. Previous Work. The above-mentioned CTRNN-EH framework has been successfully employed to control legged locomotion in both real and simulated hexapod walkers [3, 16] by author's colleagues. These efforts concentrated on solving a learning locomotion control problem for the hexapod robots with twelve degrees of freedom, from scratch, without any preknowledge of the robot's physical characteristics (like weight and any leg damages). Conceptually, each leg of the hexapod (with two degrees of freedom in its actuators) would require optimal oscillatory patterns in its two actuators, with appropriate phase relations to aid in generating forces to move the hexapod forward or backward directions. Moreover, any controller that claims to provide optimal locomotion controller for the hexapod has to take into consideration the required optimal oscillatory dynamics in each leg as well as the needed collaborative dynamics among all the six legs to generate optimal and energy-efficient motion in the hexapod [3]. This complex mix of local and distributed locomotion pattern generation problem was successfully addressed by the CTRNN-EH architecture mentioned in [6]. Moreover, the evolved oscillatory CTRNN-EH locomotion controllers in those experiments embedded a large amount of practical functionality in very small numbers of neurons. These evolved controllers were capable of optimally controlling variously weighted bodies with or without damaged legs. The controllers could do this without reevolution and could adapt their dynamics on the fly by entraining to external sensory input [6, 13]. Further the work conducted to understand these evolved CTRNN-EH controllers (for legged locomotion) has resulted in a set of dynamical module analysis concepts [5, 6, 16] that can be employed to predict and explain the behavior of these

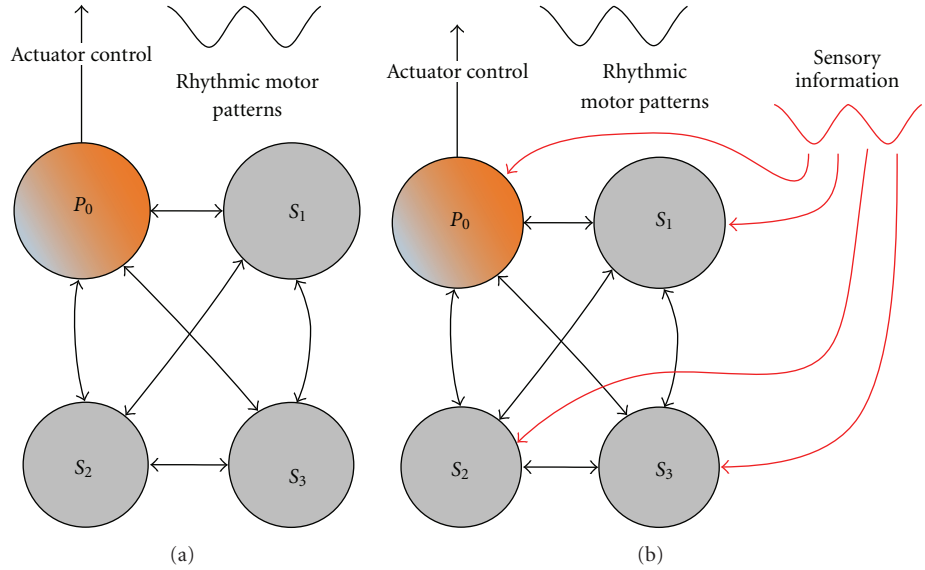

FIGURE 2: Illustrative representation of the assembly of neurons forming the autonomous neuromorphic (shown in (a)) and nonautonomous Neuromorphic (shown in (b)) controllers. "*P*" notation is used for primary neuron directly controlling the actuator of the robot under control and "*S*" is the secondary neuron, which aids in generating appropriate external dynamics required for the primary neuron. The numbering of the neurons in the network is arbitrarily chosen as appropriate.

evolved controllers, and controllers with similar nature, in terms of their functional sustainability and failure.

However, conceptually the flapping-wing flight problem shares the requirement of generating optimal oscillatory dynamics for desired flight behavior; with the hexapod walker problem, the former has inherent instability in its body dynamics, introduced by virtue of the medium of its flight (i.e., in three, dimensional space with constantly varying center of mass). This possible inherent instable body dynamics present in flapping-wing vehicles might make the CTRNN-EH based learning more challenging to be effective than when applied to the hexapod walkers, to generate optimal actuator dynamics. Further, the dynamical module analysis [6, 16] that was successful to understand the evolved locomotion controllers might not be applicable for the possible CTRNN-EH flight controllers. Nonetheless, the capabilities of CTRNN-EH controllers to produce smooth dynamics and provide provisions to adapt and modulate those produced dynamics observed in the previous work [5] sufficiently justifies the needed efforts presented in the authors published paper [4, 7–9] to evolve flapping flight controllers. Although providing the indetail description of varied possible modes of the CTRNN-EH controllers is beyond the scope of this paper, two basic modes are defined below, which are more pertinent to understand the experiments presented here in the paper.

Autonomous Controllers. These are neural network configurations that produce oscillatory signals without any external sensory inputs [3, 5]. As shown in the concept illustrative Figure 2(a), these configurations can generate autonomous and periodic dynamics in the neural network without any external triggers/sensor inputs; thus these would be referred to as autonomous neuromorphic controllers in the later sections.

Nonautonomous Controllers. These are neural network configurations that can produce appropriate oscillatory patterns only when coupled to some other oscillatory system. They are more completely discussed in [8]. As shown in the concept illustrative Figure 2(b), these configurations can generate varied dynamics in the neural network only in sync with specific external triggers/sensor inputs provided to them, thus these would be referred as nonautonomous neuromorphic controllers in the later sections.

3. Evolved Flapping Wing Flight Controllers

This section describes authors's successful efforts aimed at evolving autonomous and non-autonomous CTRNN-EH controllers for a number of flapping-wing vehicle flight modes [4, 7, 9]. The section will begin with a brief description of the microlevel flapping winged robot (MFWR) model employed, followed by description of MFWR-specific CTRNN-EH control architecture and the evolution strategy applied at evolving different flight controllers.

3.1. Microlevel Flapping Winged Robot (MFWR). Micromechanical flying insect model was developed by MFI team at UC Berkeley [2] to facilitate the investigative study to build a real flapping wing robot with a 100 mg mass and a 25 mm wings span. Based on the available MFI literature on that robot's wing aerodynamics and body dynamics, we reconstructed a model with linear actuator dynamics called Microlevel Flapping Winged Robot (MFWR) [4]. Our model was verified in simulation and found to be a match for the

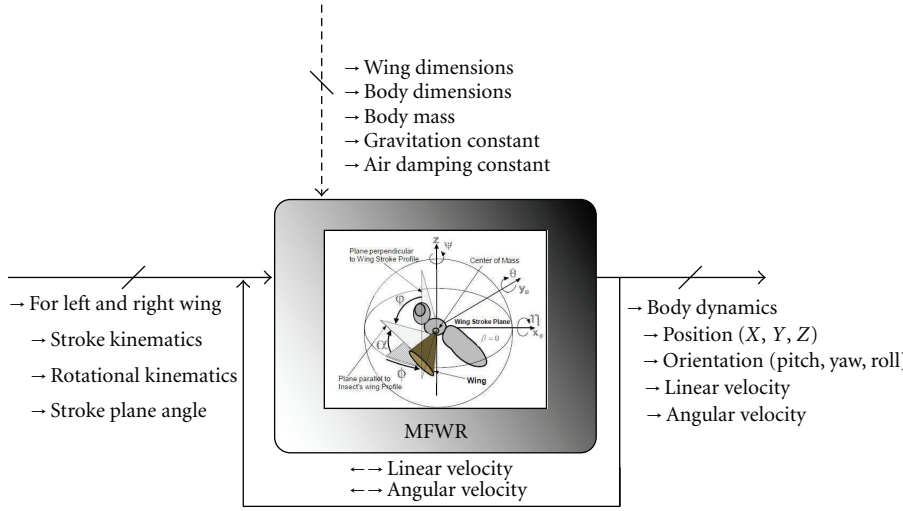

Figure 3: Schematic of the simulated Microlevel Flapping Winged Robot (MFWR).

Berkeley model in that we could reproduce their published flight envelopes and behaviors. The *Wing Aerodynamics* (WA) module of model is based on the mathematical model, developed from empirical study conducted on the Robofly [17]. The WA module generates the aerodynamic forces and torques for given wing kinematics. The *Body Dynamics Module* takes the aerodynamics forces and torques generated by the wing kinematics and integrates them along with the dynamical model of the MFI body, thus computing the body's position and the attitude as a function of time. The readers are directed to [18] for detail descriptions of these modules.

In brief, the implemented MFWR model takes wing (left and right) actuation parametric inputs like stroke and rotation trajectories and produces the position and attitude information of the MFWR in the world coordinate system as shown in Figure 3. Additionally, the model also takes the body linear velocity and angular velocity from the previous simulation step making it an internal feedback system. The MFWR model has been simulated with realistic and envisioned physical and environmental parameters like robot's mass of 100 mg, envisioned wing length of 25 mm, acceleration due to gravity value of 9.8 m/sec^2, air dampening coefficient of 62.3×10^{-6} N-sec/m, stroke angle range of -60 to $+60$, rotational angle range of -90 to $+90$, and with some constant parameters derived and mentioned in [2, 17]. Further, the differential equations characterizing the internal dynamics of the robot model have been computed using Runge-Kutta (RK4) numerical method.

3.2. Control Architecture and Evolutionary Algorithm Specifications. After some preliminary experimentation conducted with the MFWR model, the custom CTRNN-EH control architecture shown in Figure 4 was chosen to be less-redundant architecture and with enough flexibility to embed into its dynamics the optimal control laws required for different flight modes [4]. The actuation dynamics modeled for each wing can presently control the stroke and rotational parameters of the wing, and the stroke plane of the wing is fixed at a constant angle. This distributed CTRNN architecture shown in Figure 4 has a central core network block that can produce signals, which are delivered to each wing trajectory actuation after being processed by a delay network block. Figure 4 shows the interfacing of the controller architecture with the MFI wing parameters. The delay networks are placed to produce asymmetric/symmetric actuations of left wing with respect to right wing or vice versa to produce net nonzero/zero torques and forces in Y-direction [2, 4]. For the set of experiments and results, being described in this paper, a fullyconnected eight neuron network is chosen for the central core network block and the evolutionary algorithm is employed to evolve the central core network and/or the delay duration in outer delay networks [4]. Further, the central core can accept sensory input, from the MFWR's status or external command, which is feed to each neuron in the core. The next subsection provides the details of the evolutionary algorithm employed in the present experiments.

As mentioned earlier, each individual CTRNN neuron is specified by one bias, one time constant, and eight weighted connections from all neurons in the network (seven connections to other neurons, one self-connection, and one sensor). Thus, the central core network, with eight-neurons, is fully specified by the numeric value settings of eighty-eight parameters, with eleven parameters for each neuron in a fully connected eight-neuron network. The aforementioned minipop EA is implemented with a population size of 4 and a mutation rate of 0.005. The genome length chosen was equal to total number of bits employed to encode a given CTRNN configuration. Each neuron parameter in the configuration is encoded in eight bits, which aggregates to a genome length of 704 bits to represent the central core network. The delay input interval duration for gate networks is encoded in an eight-bit string. Employing the aforementioned architecture and algorithm specifications

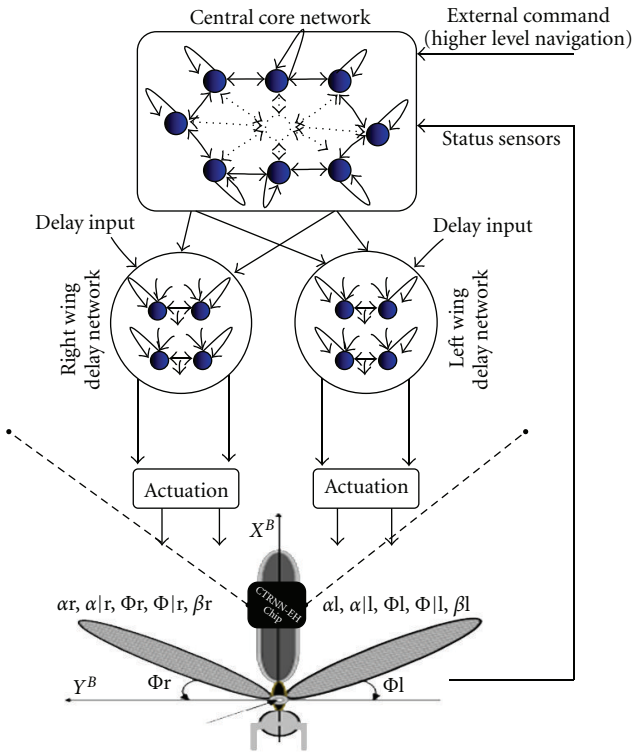

Figure 4: Interfacing CTRNN Architecture to MFWR Model to evolve and analyze flight controllers. In the figure, α (with range −90 to 90) indicates the instantaneous rotational angle of the wing, Φ (with range −60 to 60) indicates the instantaneous stroke angle of the wing, and β (with range −45 to 45) indicates the instantaneous stroke plane angle of the wing. The parameters with "|" indicate the rate of change for the parameter it is referring to and these parameters with "r" and "l" indicate the right (r) and left (l) wing parameters, respectively.

the next sections provide the details of the evolutionary runs and evaluation criterion applied to evolving autonomous and nonautonomous controllers.

3.3. Evolved Autonomous Flight Controllers. Three kinds of autonomous flight controllers, namely, cruising, altitude gain, and steering, were successfully evolved using the aforementioned architecture and algorithm, but with type-specific fitness evaluation criterion. For example, an acceptable behavior of MFWR under an evolved cruise mode controller is to produce motion in a forward direction that is greater than the motion in altitude or sideward directions. Moreover, it should also maintain zero angular velocity along the three vehicle frame axes (zero pitch, roll, and yaw). The later criteria of the expected controller can be met by employing preevolved CTRNN-EH gate networks with symmetric delays. But the first and primary criteria of the controller should be evolved in central core, since this is the only module capable of generating any dynamics to drive the wings. Thus, an evaluation function to capture this established cruise criteria should observe the motion of the MFWR under the control of the potential controller

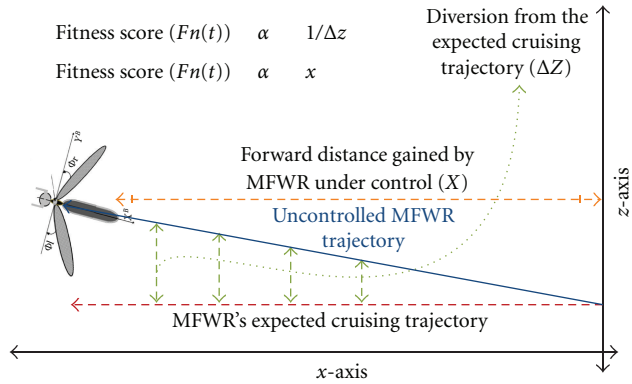

Figure 5: The figure shows the relation between the expected cruise behavior of MFWR and the fitness evaluation function employed to evolve the CTRNN-EH controllers to achieve the same behavior under control. An acceptable cruise controller has to propel the MFWR in forward direction and minimize the variation in the altitude. Thus, the fitness score employed to evolve the cruise controllers should reward any forward motion (in x-direction) and penalize any variations in altitude (in z-direction).

and reward the controller on generation of the forward motion and penalize it on generation of altitude variations. A pictorial representation of the expected autonomous cruise behavior and the established relation to its potential evaluation function is shown in Figure 5. Thus, the below evaluation function is designed with a minimizing fitness strategy [4], to capture the expected cruise behavior in MFWR:

$$\frac{\sum_{i=0}^{i=N}\left(|P_{zi}| - \left|P_{yi}\right| - P_{xi}\right)}{N}, \tag{2}$$

where P_{zi}, P_{xi}, and P_{yi} are the instantaneous positional data of MFWR moment in Z(altitude), X(forward), and Y(sideward) directions under the control of the wing kinematics generated by the controller and N is the total number of time steps present in each evaluation period.

It can be observed that the above evaluation function captures the expected forward motion by placing constraints on the controller to maximize P_{xi} term, because it is a negating summation variable in the above minimizing fitness strategy function. Further, the altitude sustainability constraint is enforced by $|P_{zi}|$ term, which captures the averaging absolute measure of the variation in the altitude across the evaluation time, and evolved cruising controller's fitness should minimize this factor so as to favor the overall fitness value contributed by the P_{xi} term. Thus, at least theoretically, the established fitness evaluation function for evolving cruising mode controllers rewards the forward motion of the MFWR and penalizes the variations in its altitude, when placed under the control. Figure 6 shows the behavior of the MFWR under an evolved cruise controller, which has successfully evolved to produce appropriate stroke and rotation kinematics in the wings of the MFWR. Based on the above-mentioned fitness evaluation strategy, that is, to capture the flight mode behavior in terms of the positional information of the MFWR over a specified period

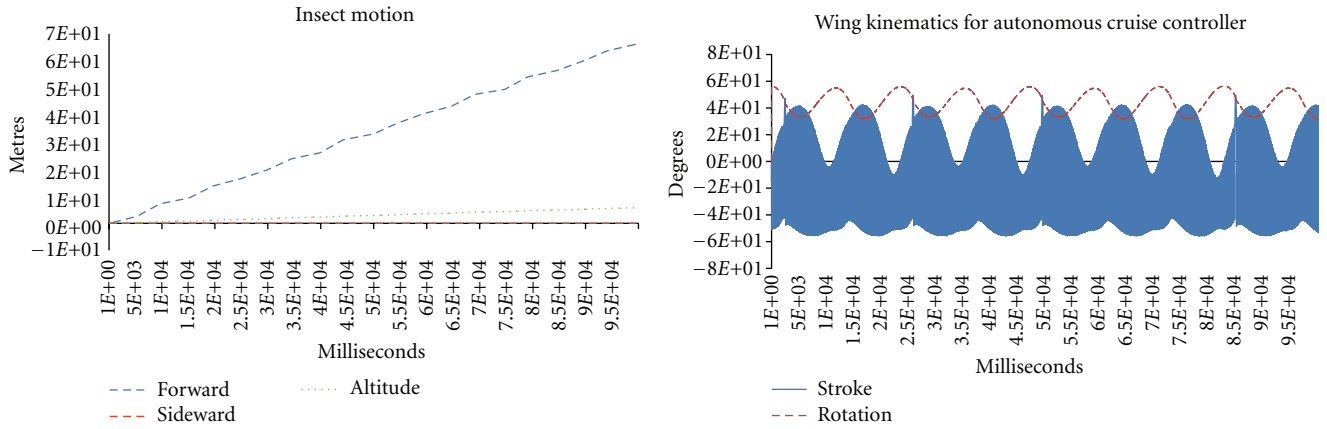

Figure 6: The motion of the MFWR in three dimensions (shown in (a)) controlled by a Cruise Mode Controller actuated wing kinematics (shown in (b)). Here the stroke kinematics has relatively higher beats rate than that of the rotation for the above controller shown in the figure, thus the oscillation cycles are cluttered, making it hard to visualize them with respect to the rotation kinematics.

of control, other two autonomous controllers, Altitude-gain and Steer, were also evolved successfully [8, 9]. Further details of the autonomous flight mode controller experiments can be found in [7–9].

3.4. Evolved Nonautonomous Flight Controllers. Two kinds of non-autonomous flight controllers were successfully evolved, namely, adaptive cruise mode controllers and polymorphic controllers [7, 8]. The adaptive cruise mode controllers were similar to their autonomous counterpart except that they were forced to sense the altitude of the MFWR and adapt accordingly during the evolution. The polymorphic controllers on the other hand were forced to change the behavior of the core central CTRNN module between autonomous altitude-gain controllers and autonomous cruise mode controllers, based on the external command. Both of these controllers employed the same fitness evaluation criterion, which was mentioned in the context of the autonomous controllers, on a varied range of the sensory inputs. An evolved nonautonomous cruise mode controller's effect on the wing kinematics of the MFWR is shown in Figure 7(a), along with the corresponding positional data of the MFWR and the sensory input. Also, one of the evolved CTRNN-EH polymorphic flight controller's effects on the wing kinematics is shown in Figure 7(b), along with the corresponding positional data of the MFWR robot and the external sensory input for invoking the desired modes.

4. Analysis of the Evolved Flight Controllers

The evolved CTRNN-EH flight controllers would be better accepted for practical deployment, at least for engineers, if their functionality can be explained using known general principles of engineering. As with all evolvable hardware-based methods, there exists a possibility that the acceptance of the evolved flight controllers, merely in terms of fitness score value (which is based on closely approximating the acceptable overall body trajectory behavior), could have been exploited the possible underlying noise in the MFWR model to gain optimal controller status. Thus, the first possible analysis to accept the evolved flight controller is to diligently observe and validate the insect's temporal behavior when coupled with the evolved controller's dynamics and determine if they satisfy the known principal physical characteristics of the MFWR model flight behavior. Further, it would be of interest to explain the evolved controllers by possible decomposition of the CTRNN-EH layer in terms of logical control blocks. The next subsections deal with analyzing the evolved controllers with two deduced approaches mentioned below.

4.1. Acceptability Analysis. During the course of this work, it was deduced that the acceptability of the physical behavior of the MFWR flight, produced by the evolved flight controllers could be readily understood by qualitatively contrasting them, with the information discerned from the empirical study conducted on the MFI insect model [2, 18]. During the mentioned empirical study, it was demonstrated that an appropriately parameterized wing's rotational trajectory (the parameters being frequency, amplitude, and phase) can produce thrust in the wing motion plane that can counter air damping and drag on the insect's body, which in turn leads to proportional motion of the robot in forward direction. Additionally, it has been deduced that an appropriate and steady stroke trajectory envelope in the wing kinematics, at positive rotational position and rate of the wing (i.e., upstroke of the wing), can produce positive lift in MFI (which would counter the gravitational forces and leads to rise in the altitude), and the same stroke envelope at negative rotational position and rate would generate antilift (which leads to drop in the altitude). Thus, any designed or evolved controllers for MFWR model should at least qualitatively satisfy this empirically deduced criterion, established for the MFI insect model flight behavior.

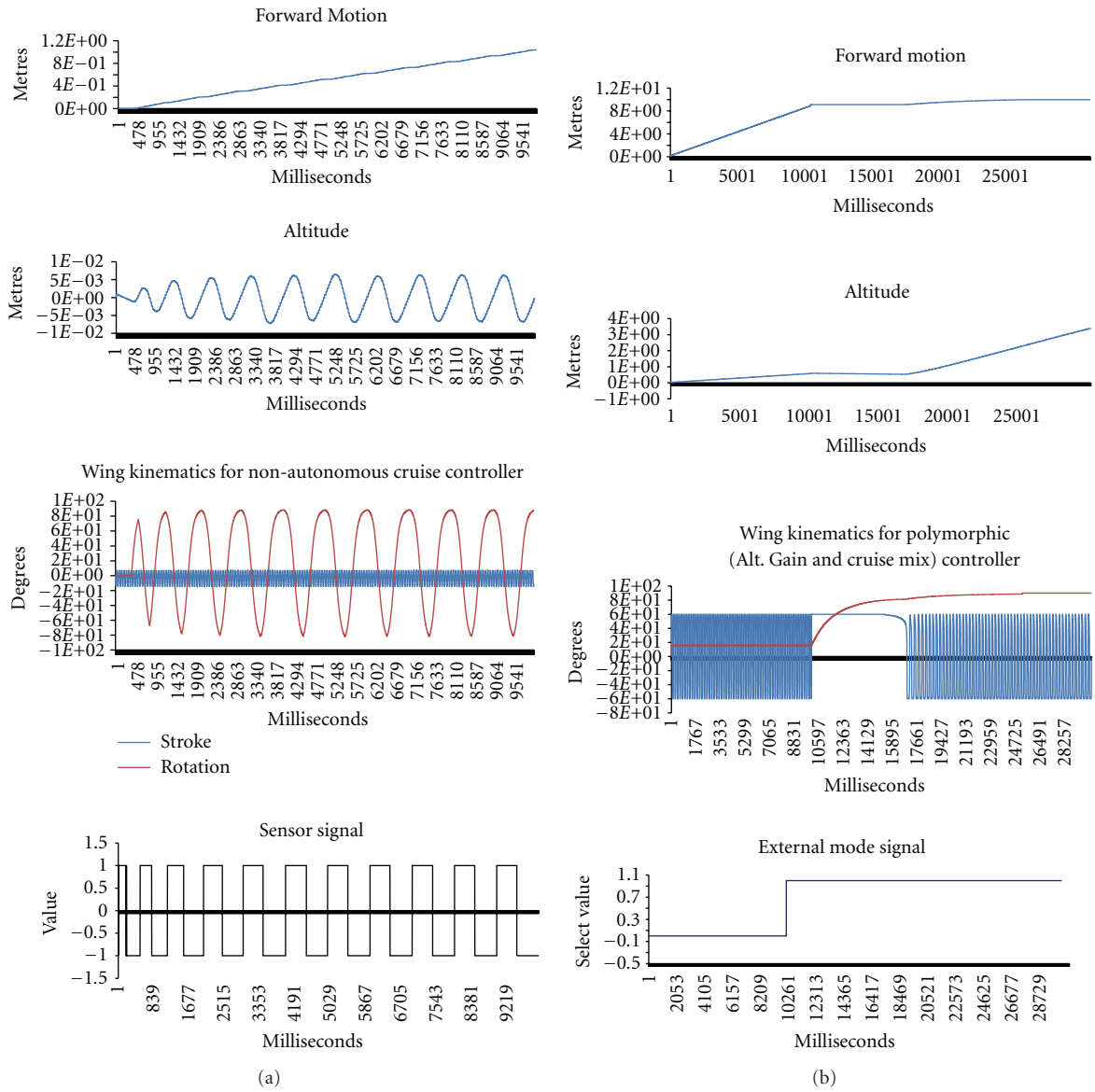

Figure 7: (a) shows the Wing Kinematics generated by the Central Core network of a Non-Autonomous Cruising Controller and the corresponding insect motion and sensory update (the stroke kinematics relatively has higher beats rate than that in rotation for the above controller). (b) shows the Wing Kinematics generated by the Central Core network of a Polymorphic flight Controller and the corresponding insect motion, when acted upon by external sensory inputs. One can see the initial cruising behavior is been switched to Alt. Gain behavior (with brief switching delay in wing kinematics).

It was demonstrated that the autonomous and nonautonomous controllers, evolved merely based on simple fitness evaluation functions, produced an acceptable physical behavior in the MFWR model in terms of overall body trajectory [8, 9]. Furthermore, when diligently observed, the wing trajectory (rotational and stroke) produced by the evolved flight controllers (for a given fitness criteria) seems to abide with the empirically established wing trajectory criterion (from MFI insect model flight behavior study). A detailed description of acceptability analysis is not in the scope of this paper and readers are directed to [8, 9] for more details on this analysis performed on individual flight mode controllers.

4.2. Qualitative Functional Decomposition Analysis. It would be of interest to interpret the evolved controllers by possible decomposition into easily explainable logical control blocks, and there exists a previous work [5, 16, 19] that relies on identifying the internal dynamics of the CTRNN-EH controller into Central Pattern Generators (CPGs) and Reflexive Pattern Generators (RPGs). To summarize succinctly, one can perceive the CPG patterned CTRNN controller as the collection of neuron modules that have been evolved appropriately at individual neuron level, to produce autonomous oscillatory dynamics, without any external oscillations or bias. The possibility of a two-neuron CTRNN-EH controller producing autonomous oscillatory dynamics has been

demonstrated in [16, 19], and further the later work in the same realm [5, 6] provided a logical CPG template, in which neurons inhibit each other with a time delay, which further leads to continuously destabilizing each other to generate oscillatory dynamics. Further, an RPG patterned CTRNN-EH controllers can be perceived as the collection of neuron modules that have been evolved appropriately to produce oscillatory dynamics in presence of the external oscillations or bias.

The evolved autonomous altitude gain, cruising, and steering and controllers are suspected to fall under the CPG template and could be decomposed into a collection of explainable oscillatory and nonoscillatory neuron groups that produced desired control of the evolved flight behaviors.

On other hand, the nonautonomous cruising mode controllers and polymorphic mode controller (as a whole) are likely to fall under the RPG template and could be decomposed into a collection of sensor-dependent or -independent oscillatory neuron groups. Thus, it would be necessary to find and separate the possible independent and dependent oscillatory control modules in an evolved controller that could aid in characterizing a given controller using known CPG or RPG templates. Further this decomposition process could provide a qualitative view and human understandable structure of the lower-level coordination among these separated modules, which primarily govern the behavior of a given evolved controller. In this vein, a three-step frequency-based analysis procedure is proposed to qualitatively decompose the evolved controllers.

Dynamics-Deprived Neuron Elimination. To simplify the process of decomposing, the evolved controllers into a group of functional units, a step-by-step neuron elimination technique, shown in Figure 8, has been employed to possibly reduce the size of the existing 8-neuron CTRNN-EH controllers. As shown in Figure 8, one can assign a role to individual neurons in a given CTRNN-EH controller architecture, based on their functional value. The neurons that are connected directly to the effectors module of the MFWR can be designated as primary neurons, and others can be designated as secondary neurons. It is obvious that primary neurons cannot be dynamics-deprived neurons, but some of the secondary neurons that saturate to minimum or maximum of neuron output level during flight controller period qualify to be dynamics-deprived neurons. These detected dynamics-deprived neurons can be folded into the existing neurons by modifying the biases appropriately (i.e., "Bias-Forwarding"). Once the reduced architecture's dynamics qualitatively match the dynamics of the original complete network, the reduce network can be employed for further decomposition process.

Frequency-Based Grouping. Based on the previously mentioned general principle of acceptable controller dynamics, it was deduced that the steady oscillatory dynamics in the wing (stroke or rotation) dictate the flight behavior. Thus, based on this controller acceptability knowledge, it would be appropriate to group the neurons in the reduced network, based on their individual time constants, into no more than two groups (one each for rotation and stroke). As shown in the Figure 9, the clustering criteria are based on the idea that the neurons with relatively lower time constants (i.e., higher frequency) are separated from the neurons with relatively higher time constants (i.e., lower frequency). As shown in Figure 9, the grouping of the neurons will simplify the decomposition process in the way that one can logically relate the individual wing kinematics (stroke or rotation) to the individual neuron groups (clustered) based on the qualitative difference in the frequency and phase of the wing kinematics.

Lesion Study . Once the frequency clustered neuron control modules are obtained for a given controller, it is necessary to understand the interactions of the individual neurons within those control modules and with the other existing control modules to qualitatively deduce the underlying governing principle of the controller functionality. Thus, in this lesion study, a general method of diligently observing the variations in the dynamics of an individual or group of neurons, while some of its connections are amputated from rest of the network, has been adopted. Though the number of lesion operations cannot be quantified and will vary depending on the complexity of the evolved controller, but as shown in Figure 10, the initial intuitive regions of the amputations across all the evolved controllers would be between the frequency clustered neuron groups to verify their interdependency followed by a series of further lesions, like intragroup lesion study, wherever deemed appropriate for the controller in the context. Employing the above-mentioned qualitative decomposition process; the best five of every evolved controller in each category have been analyzed as discussed below.

4.3. Analysis of Autonomous Controllers

4.3.1. Autonomous Cruise Mode Controllers. This section provides the detailed qualitative decomposition process for one of the best evolved autonomous cruise mode controllers, using the above-mentioned three general steps. For qualitative comparisons and to better understand the controller decomposition an unaltered original eight-neuron architecture of the controller to be analyzed is shown in Figure 11. The architecture has two primary neurons 0 and 1, which are directly connected to the stroke and rotation effectors of the MFWR and the neurons from 2 to 7 are the secondary neurons of the controller, whose role in governing the controller behavior would be determined as part of this decomposition process. The original outputs of each neuron in the controller's architecture, which are responsible for producing the desired cruising behavior in MFWR, are shown in Figure 12.

Moreover, the flight trajectory of the MFWR under the control of the original controller in the context is shown in Figure 13, which would be useful for qualitative comparisons that would be performed later when the original architecture of the controller has been simplified for analysis. As shown in Figure 12, it can be observed that the secondary neurons

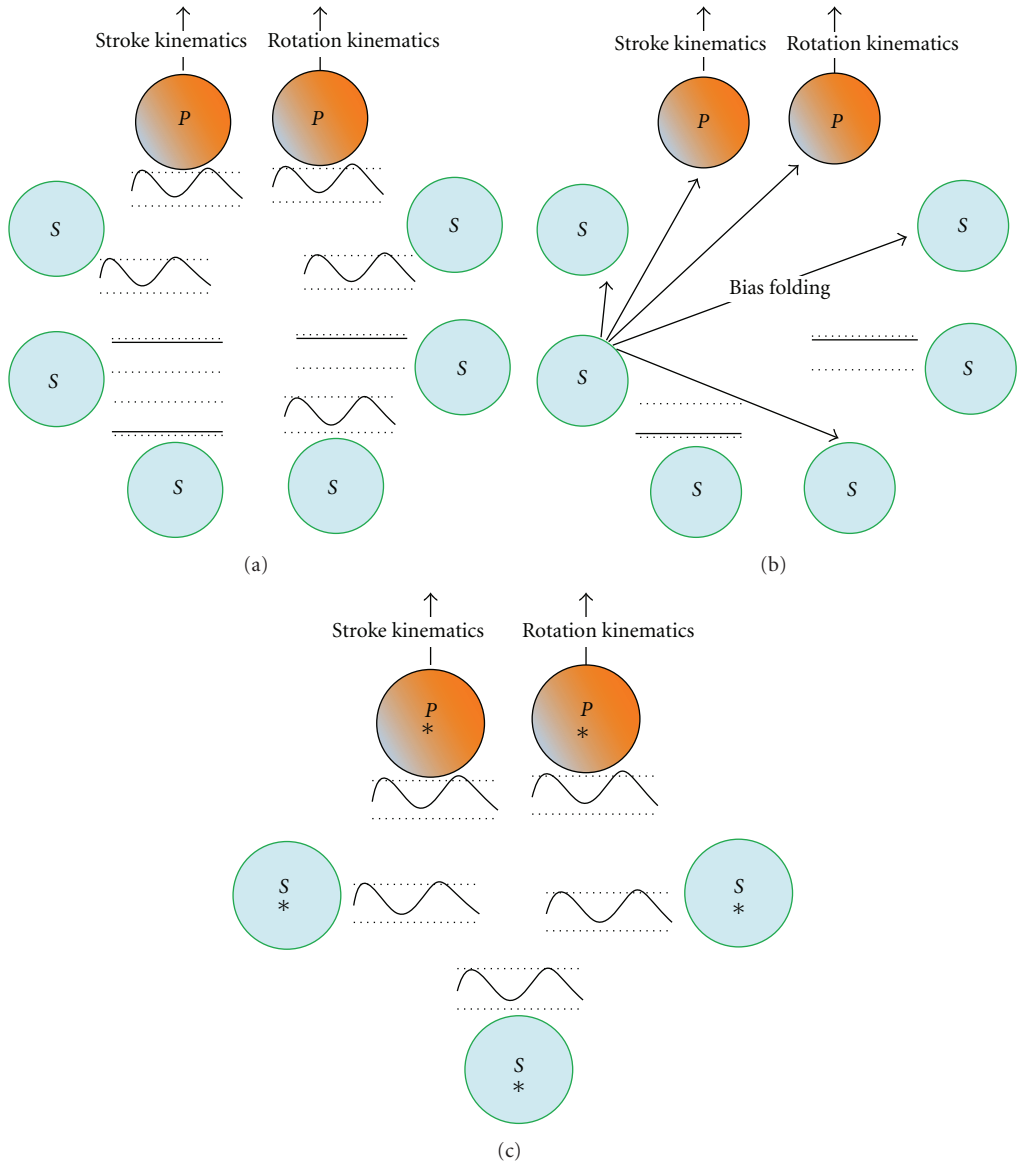

Figure 8: A pictorial representation of the "dynamics-deprived neuron elimination" process. The primary neurons are labeled as "*P*" and subsidiary neurons are labeled as "*S*". A three step process is adopted here, starting with eliminating the neurons with saturated dynamics in them as shown in (a), followed by folding the saturated output of the eliminated neurons as bias into the survival neurons as shown in (b). The final step shown in (c) is to verify the qualitative match of the dynamics produced by the bias modified (labeled with "*") individual neurons to their counterparts in the original architecture.

3, 5, and 6 seem to be saturated at constant output value during the flight control. Though it can be deduced, at least, from observations that these three neurons may not have contributed to the overall output dynamics produced by the controller, a detailed step-by-step process mentioned in the "Dynamics-deprived Neuron Elimination" procedure is necessary to rule out the possibility that these neurons might have played a critical role during the initialization of the controller by providing transient dynamics before saturating in the steady state. The obvious neurons that are contributing to the controller dynamics are 0, 1, 2, 4, and 7, but there exist distinct differences in the output envelope and frequency characteristics of 0 and 1 neurons from 2, 4, and 7, which could be used for "frequency-based grouping" process later on the successful reduction of the architecture size, as shown in Figure 17. Since, the candidate dynamics-deprived neurons are determined by the neuron output state observations; the biases of the neurons 0, 1, 2, 4, and 7 are modified appropriately, as pictorially represented in Figure 8(b), by treating the individual input weight of the survival neuron from each eliminated neuron as an additional bias value to its output state. The resultant reduced neuron architecture of five neurons in it is pictorial represented in Figure 14. To further validate

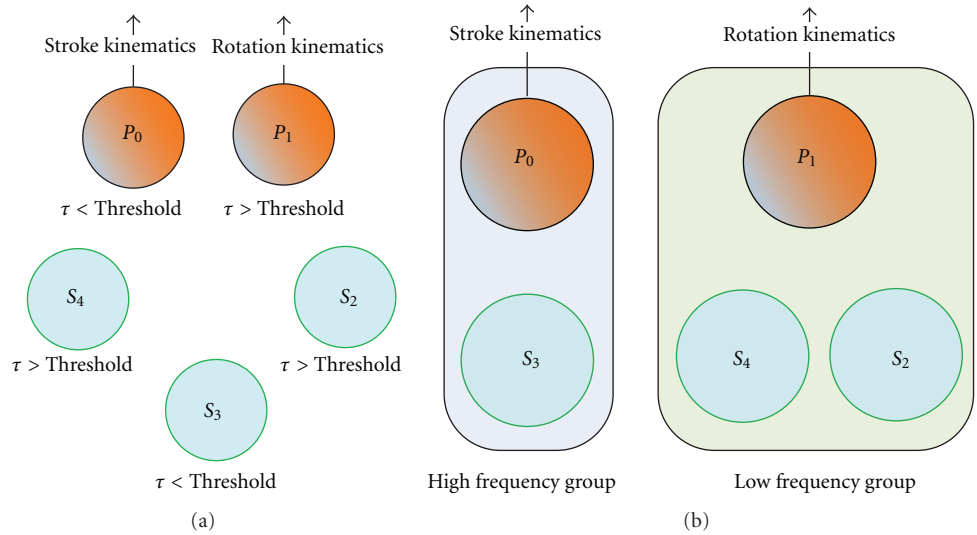

Figure 9: A pictorial representation of the "frequency-based grouping" process. The first step of the process, as shown in (a), is to determine a relative threshold time constant (Tau) for the reduced network, reduced by "dynamics-deprived neuron elimination" process, followed by grouping the neurons in the architecture based on the frequency of the output produced by individual neuron (i.e., the neurons with time constant less than the relative threshold are clustered into high frequency group, and neurons with time constants more than the relative threshold are clustered into low frequency group) as shown in (b).

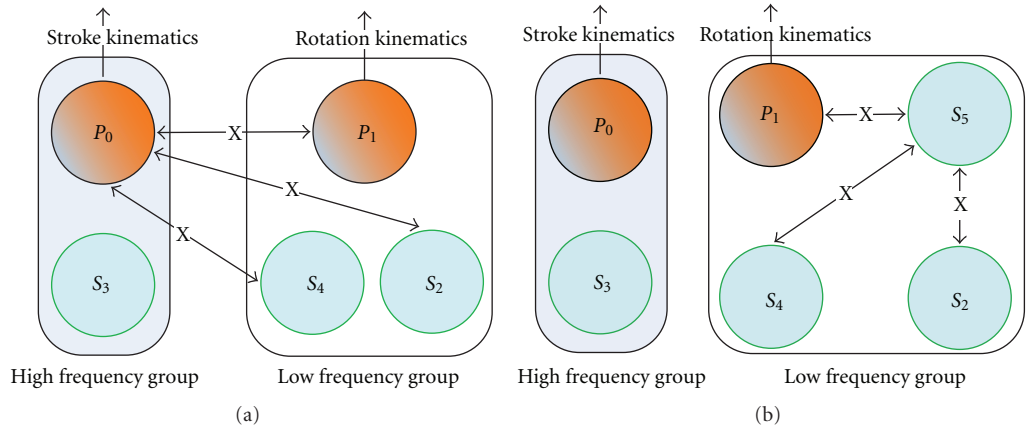

Figure 10: A pictorial representation of the "lesion study" process. The lesion study is based on the idea that it is possible to determine the underlying governing functional principle of the network with rigorously observing the behavior changes in the network for appropriate combinations of the amputations. Based on the complexity of the controller, the lesion study can be performed between neurons in distinct frequency groups, which is performing intergroup amputations, shown in (a), or between the neurons in the same frequency group, that is, intragroup amputations shown in (b).

that the dynamics-deprived neuron elimination process is applicable for this controller, two qualitative comparisons are necessary, primarily the architecturally reduced controller should at least qualitatively control the MFWR trajectory behavior that was intended by the original controller, and, moreover, the survival neuron output state envelopes during the flight control should match their output state envelopes from the original architecture.

The later condition eliminates the possibility that the reduced architecture could have changed dramatically and lost its internal dynamics, although it could have satisfied the primary condition to produce the desired cruise behavior in MFWR. Thus, the reduced five neuron controller is evaluated against the MFWR, and the individual neuron output state envelope of the five-neurons is captured and shown in Figure 15, and accordingly the trajectory of MFWR under the control of the reduced controller is shown in Figure 16. It can be observed that there exists an acceptable qualitative match between the produced neuron outputs in the reduced five-neuron controller to its counterparts original eight neuron controller, including the in sync variations of the frequency and amplitude in the primary stroke neuron and the neuron 4 (of original architecture) to its new neuron position 3.

Moreover, the most convincing evidence that the reduced controller qualitatively controls the MFWR trajectory to

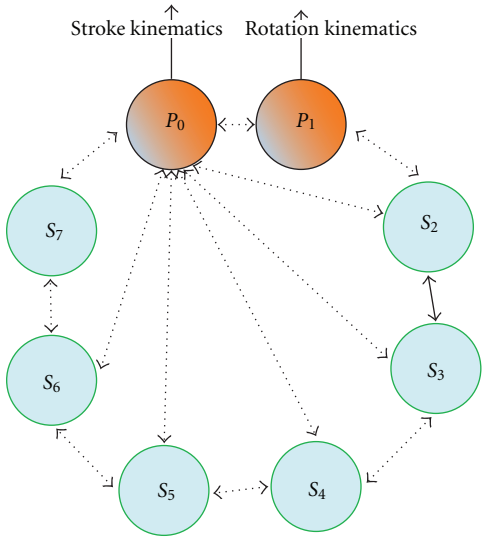

Figure 11: A pictorial representation of the fully connected eight neuron architecture of the autonomous cruise mode controller chosen for qualitative decomposition analysis. As mentioned earlier, the stroke and rotation neurons are marked "*P*" as primary and "*S*" as secondary for the other neurons and numbered accordingly from 0 to 7.

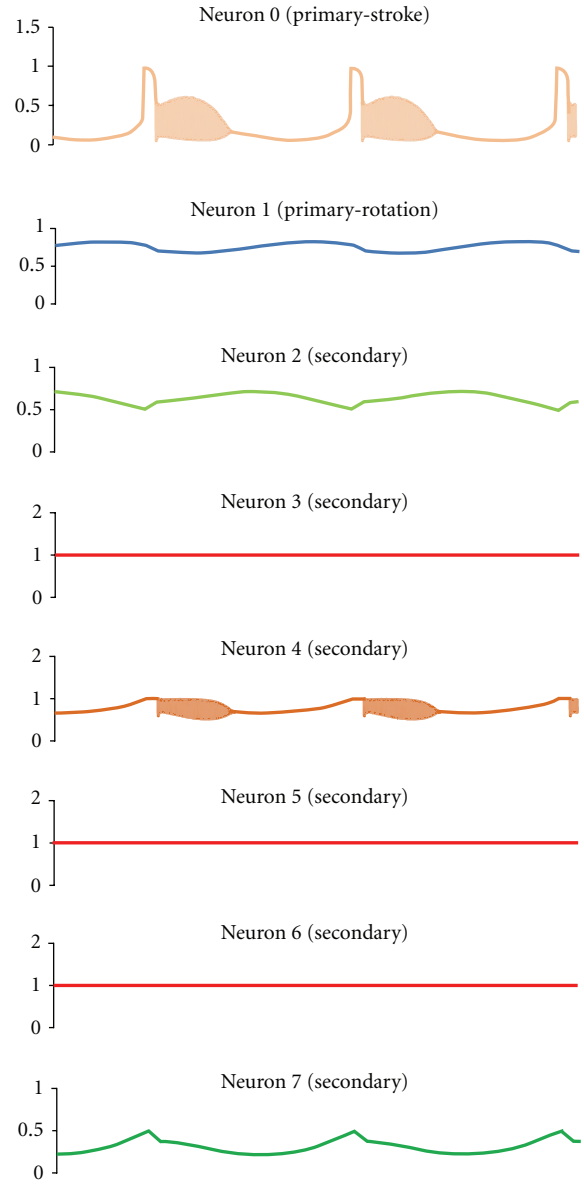

Figure 12: The above figure shows the neuron output state dynamics of each neuron in the fully connected original eight neuron controllers produced during the flight control of the MFWR to provide optimal cruise behavior.

produce desired cruise behavior justifies that the dynamics-deprived neuron elimination process is applicable for this controller. Thus, moving forward with the reduced five-neuron architecture, applying frequency-based grouping would be uncomplicated, since it can be observed from the five neuron output envelopes that the primary stroke neuron and third secondary neuron seem to share a peculiar in sync frequency and amplitude variations, intuitively belonging to high frequency group. Moreover, the evolved time constant for both of these neurons is same and is 0.010000 units and on other hand, the neurons 1 and 4 along with the rotation primary neuron can be allocated to low frequency group with corresponding time constants 10.546157, 9.558393, and 20.176863, respectively. Thus, if a relative time constant threshold of 9 units is chosen, then there exist two distinct frequency-based groups as shown in Figure 18. After the grouping of the primary stroke neuron and third secondary neuron in a comparable frequency group, further interpretation on their interconnection weight revealed that there strongly inhibit each other, and further there exists a strong possibility that these two neurons can form a two-neuron (high frequency) oscillator with any other input dynamics from the low frequency group (consists of rotation primary neuron, fourth secondary neuron, and second secondary neuron). Thus, an intergroup lesion study, as shown in Figure 10(a), to amputate the neuron connections between the high frequency group neurons and the low frequency group neurons is performed. When this amputated network is evaluated, the above intuitive possibility of two-neuron oscillator formation in the high frequency group was validated along with a revelation of two-neuron oscillator formation in a low frequency group, as shown in Figure 18. It can observed that the primary stroke neuron and the third secondary neurons oscillate at same frequency consistently, and their output amplitude is more than the lower frequency group consisting of primary rotation neuron and second secondary neuron along with a saturated fourth secondary neuron during the amputated evaluation. It is evident that the fourth secondary neuron's dynamics are not completely isolated from the high frequency stroke oscillator group (primary stroke neuron and third secondary neuron) since this fourth neuron has shown perfect oscillatory behavior when the reduced network was fully connected through this neuron. Moreover, when this amputated network controller is coupled to the MFWR, it has been observed that the controller was able to control the MFWR trajectory in an

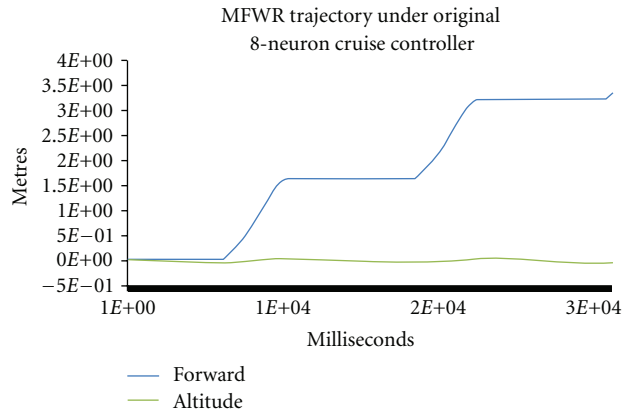

Figure 13: The MFWR trajectory produced by the original fully connected eight-neuron controller. It can be observed that the evolved controller was successful in producing forward motion in the MFWR without any overall gain in the altitude.

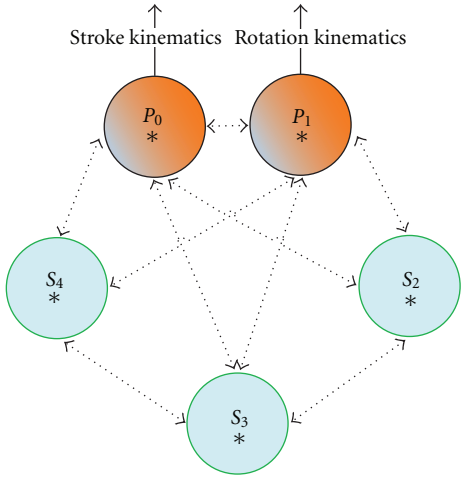

Figure 14: A pictorial representation of the reduced five-neuron architecture of the cruise controller referred in Figure 12. It should be noticed that the primary neurons and secondary neurons retain their position in the network, but neurons 4 and 7 from the original network are positioned in 3rd and 4th locations respectively. The "∗" indicates that the neurons in this "dynamics-deprived neuron elimination" process-based architecture differ from the original neuron in the way that, their bias has been accounted for the eliminated neuron's saturated output effect on them. So, at least in steady state this reduced network should perform functionally equivalent to the original architecture.

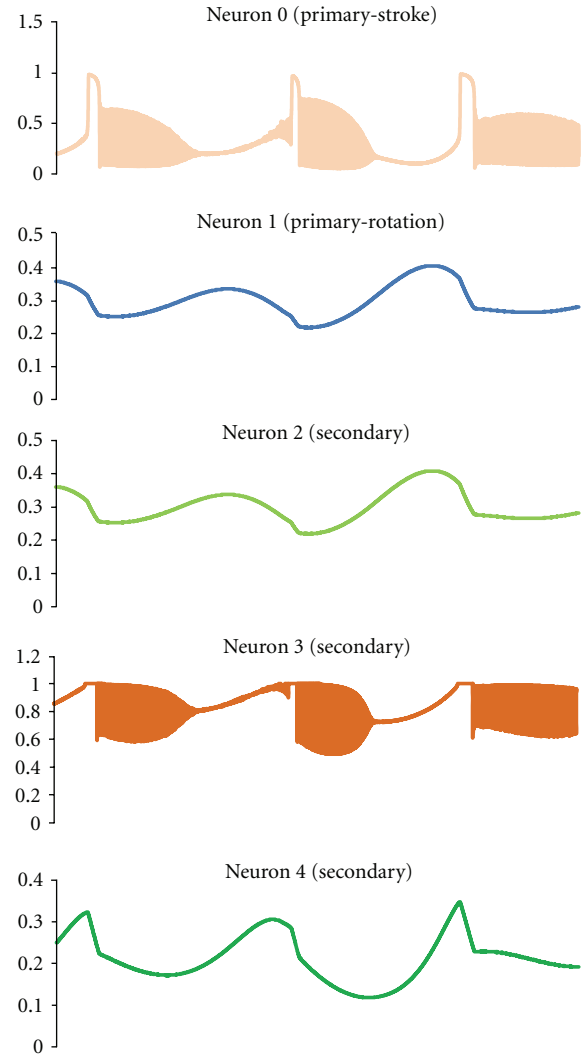

Figure 15: The above figure shows the neuron output state dynamics of each neuron in the reduced five-neuron architecture of the cruise controller architecture shown in Figure 12. It can be noticed that qualitatively the output dynamics of each neuron do not differ significantly from their original behavior shown in Figure 13.

acceptable cruise mode behavior template seen before in Figures 16 and 15, but gradually it drifted from the acceptable behavior and resulted in significant rise in MFWR's altitude, with a rise rate proportional to the MFWR forward motion rate, as shown in Figure 19. It would of interest to do a diligent comparison of the dynamics of the fully connected reduced five-neuron controller to that of the amputated controller shown in Figures 18 and 15 from behavior change perspective in each neuron output states during the flight evaluations. Though one might argue that quantitatively the dynamics of neurons 0 to 3 (includes primary stroke and rotation neurons and two secondary neurons) are different in both the scenarios, for qualitative analysis purposes these neurons do project similar oscillatory dynamics, but the drastic difference of the dynamics is observed in the fourth secondary neuron, which seized to oscillate when amputated from the high frequency neuron group indicating a strong connection to the controller's performance degradation noticed in the Figure 24. When analyzing the neuron outputs of the fully connected controller, the dynamics of stroke neuron group (high frequency group) were altered periodically by a slow moving signal, with a period equivalent to the rotation neuron group (low frequency group). Moreover, it was already demonstrated that there exists the fourth secondary neuron in this low frequency group (rotation)

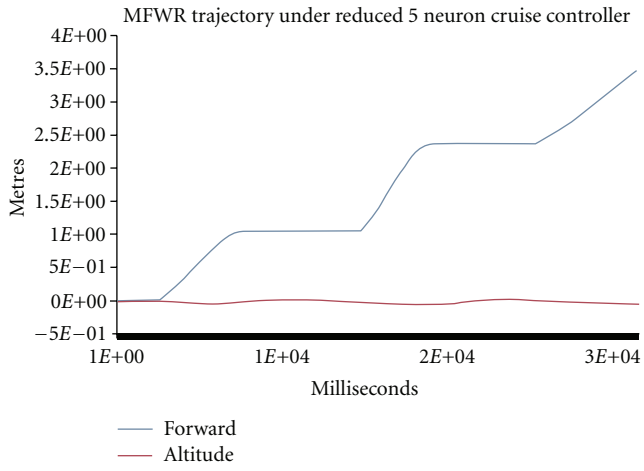

Figure 16: The MFWR trajectory produced by the reduced five neuron controller. It can be observed that the architectural reduced controller was successful in producing qualitatively same cruise behavior possible by the fully connected eight neuron network. The MFWR trajectory produced by the eight-neuron network is shown in Figure 14.

that is susceptible and depends on the dynamics of the high frequency group (stroke), and though both groups were capable of producing independent oscillatory dynamics to control the MFWR trajectory, the control lasted for a short period of time in absence of the possible dynamics modification by the fourth secondary neuron. Further, an optimal control behavior was only possible with inclusion of this fourth secondary neuron, which now can be treated as a monitoring neuron, that was evolved appropriately to take the responsibility of performing complex dynamics computation across both the oscillator groups and provide the high frequency group with periodic signals to alter its amplitude and frequency to satisfy the cruise behavior in the MFWR. Additionally, it was mentioned earlier that the fourth neuron has an intermediate time constant value of 9.558, which makes it have enough temporal summation ability, the ability which could have possiblely made it an observer (sink in the dynamics of the other neurons) of the other neurons and further have sufficient internal dynamics (sufficient firing rate to generate spikes) to modify their dynamics at slower but in a strong way thus modifying their frequency and amplitude periodically.

Based on the above analysis, it can be deduced that the evolved autonomous cruise mode controllers can be qualitatively explained as a composition of two steady and independent frequency oscillators, one governing the stroke kinematics of the wing with higher beat rate and another it is rotation with lower beat rate, in presence of a monitoring neuron which periodically tunes the amplitude and frequency of the stroke oscillator, which periodicity synchronized with the rotation oscillator. A pictorial representation of the above deduced compositional template is shown Figure 20. All of the 5 best evolved autonomous cruise mode controllers were reducible from an eight neuron to a five-neuron architecture using the "dynamics-deprived neuron elimination" process. Further, four of them had distinct frequency features in their architecture that could be exploited by the "Frequency-based Grouping" process and were successfully reduced to two kinematics control modules. Only three of the best five controllers complied with the decomposed template discussed above, with steady independent oscillator blocks and a monitoring neuron, and others performed the same functionality with closely dependent oscillator blocks that were not complaint with frequency-based clustering criteria. Nonetheless, the rigorous intragroup lesion study on them exhibited the presence of monitor neuron, which aided in controlling the amplitude of the rotation dynamics for acceptable cruise behavior.

4.3.2. Autonomous Altitude Gain Mode and Steer Mode Controllers. The above decomposition analysis mentioned in the context of the cruise mode controllers is performed on the entire best five autonomous altitude gain mode and steer mode controllers. The individual controller architectures were reducible from an 8-neuron to 4-neuron architecture using "Dynamics-deprived Neuron Elimination" process in both categories. Only some of the best altitude gain controllers were complaint with clustering criteria and thus two functional templates were derived using the lesion study performed on the individual neurons in the reduced network. As shown in Figure 21(a), this derived functional template employed single oscillatory control group encompassing both the primary neurons in it, performing close-looped oscillations required for wing kinematics, with aid of two subsidiary neurons in the network resembling a typical CPG-like control module described earlier in the section. This decomposition template is applicable for the steer controller's entire central core and only for two of the altitude gain controllers. The other template, shown in Figure 21(b), had only the stroke primary neuron in an oscillatory control group with a saturated rotation kinematics in separate control module, which is applicable for only altitude gain controllers.

4.4. Analysis of Nonautonomous Controllers

4.4.1. Nonautonomous Cruise Mode Controllers. This section provides the detailed qualitative decomposition process for one of the best evolved non-autonomous cruise mode controllers. The applicability of the established three-step decomposition using Dynamics-Deprived Neuron Elimination, Frequency-based grouping, and Lesion Study methods will be presented and possible oscillatory level decomposition will be deduced. For qualitative comparisons and to better understand the controller decomposition, an unaltered original eight-neuron architecture of the controller to be analyzed is shown in Figure 22. The architecture has two primary neurons 0 and 1, which are directly connected to the stroke and rotation effectors of the MFWR and the neurons from 2 to 7 are the secondary neurons of the controller, whose role in governing the controller behavior would be determined as part of this decomposition process. Apart from the interconnections among the neurons, every

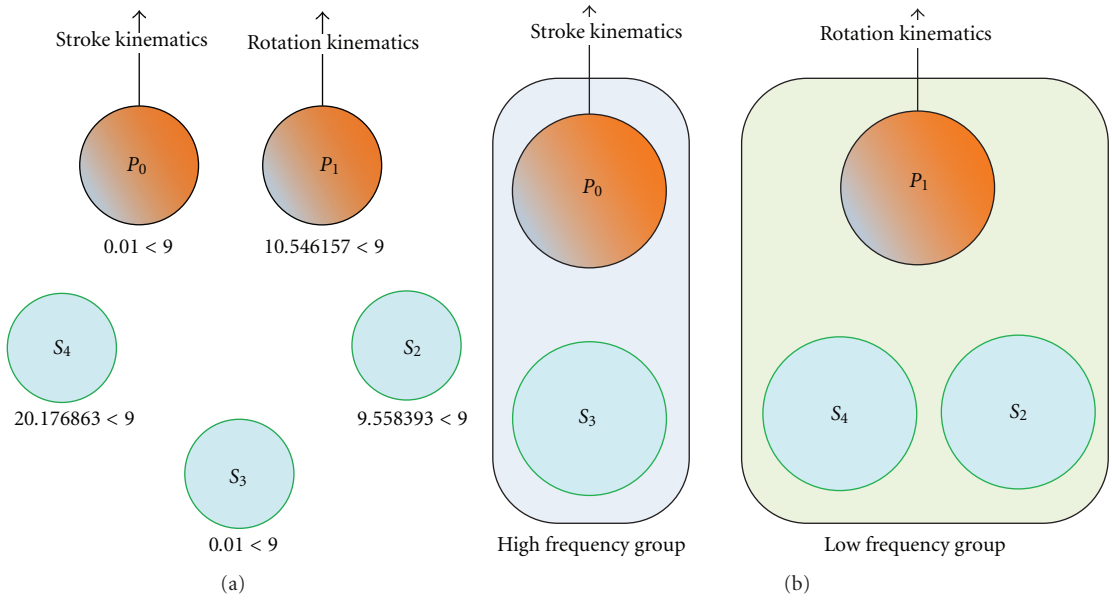

Figure 17: A pictorial representation of the "frequency-based grouping" process for the cruise mode controller shown in Figure 12. The first step of the process, as shown in (a), is to compare each neuron's time constant with determined relative threshold of 9.0 units, followed by grouping the neurons in the architecture based on the frequency of the output produced by individual neuron (i.e., the neurons with time constant less than the relative threshold are clustered into high frequency group and neurons with time constants more than the relative threshold are clustered into low frequency group) as shown in (b).

neuron is connected to an external altitude sensor, which is modeled to provide a relative altitude status of the MFWR from its initial altitude during the evaluation. The original outputs of each neuron in the controller's architecture, which are responsible for producing the desired cruising behavior in MFWR, along with the altitude sensor output, are shown in Figure 23. Moreover, the flight trajectory of the MFWR under the control of the original controller in the context is shown in Figure 24, which would be useful for qualitative comparisons that would be performed later when the original architecture of the controller has been simplified for analysis. As shown in the Figure 23, it can be observed that the secondary neurons 3 and 6 seem to be saturated at constant output value during the flight control. Though it can be deduced, at least, from observations that these three neurons, may not have contributed to the overall output dynamics produced by the controller, a detailed step-by-step process mentioned in the "dynamics-deprived neuron elimination" procedure is necessary to rule out the possibility that these neurons might have played a critical role during the initialization of the controller by providing transient dynamics before saturating in the steady state. The obvious neurons that are contributing to the controller dynamics are 0, 1, 2, 4, 5, and 7, but there exist distinct differences in the output envelope and frequency characteristics of 0, 2, 5, and 7 neurons from 1 and 4, which could be used for "frequency-based grouping" process later on the successful reduction of the architecture size. Moreover, as mentioned during the initial physical validation step of the evolution process, there exists an entrainment of the primary rotation neuron output state in amplitude and frequency with that of the sensor status output characteristics. Since the candidate dynamics-deprived neurons are determined by the neuron output state observations, the biases of the neurons 0, 1, 2, 4, 5, and 7 are modified appropriately, by treating the individual input weight of the survival neuron from each eliminated neuron as an additional bias value to its output state. The resultant reduced neuron architecture of six neurons in it is pictorially represented in Figure 25. To further validate that the dynamics-deprived neuron elimination process is applicable for this controller, two qualitative comparisons are necessary; primarily the architecturally reduced controller should at least qualitatively control the MFWR trajectory behavior that was intended by the original controller, and, moreover, the survival neuron output state envelopes during the flight control should match their output state envelopes from the original architecture. As mentioned earlier, the later condition eliminates the possibility that the reduced architecture could have changed dramatically and lost its internal dynamics, although it could have satisfied the primary condition to produce the desired cruise behavior in MFWR.

Further, the interesting entrainment behavior between the sensor output and the rotation neuron output (and if possible the third (old designated position-fourth) secondary neuron output) should be maintained, at least qualitatively. Thus, the reduced six-neuron controller is evaluated against the MFWR, and the individual neuron output state envelope of the six neurons and the sensor status are captured and shown in Figure 26, and accordingly the trajectory of MFWR under the control of the reduced controller is shown in Figure 27. It can be observed that there exists an acceptable qualitative match between the produced neuron outputs in

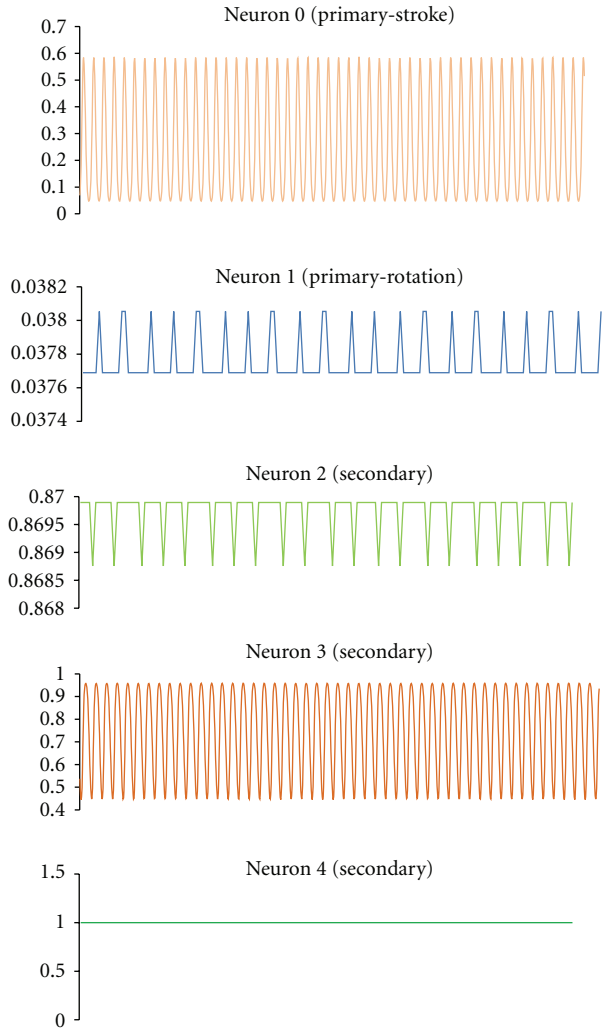

Figure 18: The output state dynamics of each neuron in the interfrequency group amputated network, amputated as part of the lesion study on the reduced five-neuron network. It can be observed that the primary stroke neuron and the third secondary neuron produced perfect in sync oscillations forming a two-neuron independent oscillator. On the other hand, the primary rotation neuron with second neuron formed a feeble two neuron oscillator. It should be noticed that the fourth neuron dynamics are saturated, in the amputated network, compared to its original oscillatory behavior seen in Figure 16 of the fully connected reduced controller.

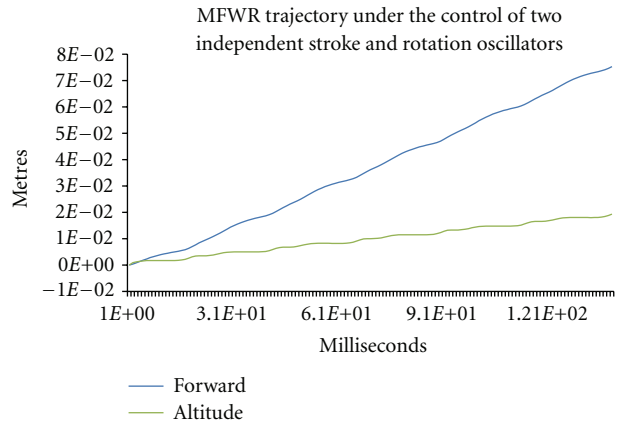

Figure 19: The trajectory of MFWR produced under the control of the amputated cruise controller. It can be noticed that the controller, with two independent oscillators for stroke, and rotation produces an acceptable cruise behavior during initial phases of the flight, but immediately loose its ability to control and reduce the altitude variations, in absence of the monitor neuron.

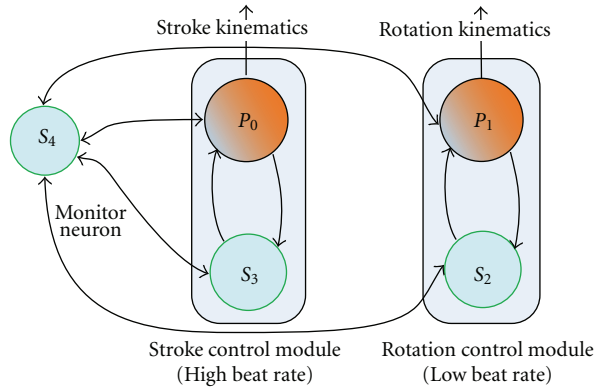

Figure 20: The Qualitative functional decomposition template derived for the autonomous cruise mode. Most of the autonomous cruise mode controllers can be decomposed into the above-shown template with a high frequency stroke control oscillator module and a low frequency rotation control oscillator along with an intermediate neuron called monitor neuron, which is responsible to coordinate and fine-tune the amplitude and frequency of the stroke oscillator with a period derived from the rotation oscillator. This functional template explains the general evolved behavior of the amplitude and frequency modulation of the stroke kinematics with rotation period for optimal cruise control of MFWR.

the reduced six-neuron controller to its counterparts in the original eight-neuron controller, including the in entrainment behavior between the primary stroke neuron and the sensor status.

Thus, moving forward with the reduced six-neuron architecture, applying frequency-based grouping would be complicated, since it can observed from the six-neuron output envelopes that the primary stroke neuron, along with second, fourth, and fifth secondary neurons, seems to share the same frequency bandwidth, intuitively belonging to high frequency group.

Moreover, the evolved time constants for these neurons are in the range of 0.010000 to 0.05000 units. But, on the other hand, the rotation primary neuron and the third secondary neuron can be allocated to low frequency group with corresponding time range of 10.034 to 16.532 units. Moreover, since the sensor module output can be treated as a pseudoneuron (with dynamics equivalent to the MFWR model and with interneuron connections to the primary neurons only), there exist two options to decompose the architecture further. The first approach is to group the sensor pseudoneuron into the low frequency group and perform the intergroup lesion study, which will provide the insight into the high frequency group oscillator's (if at all the group

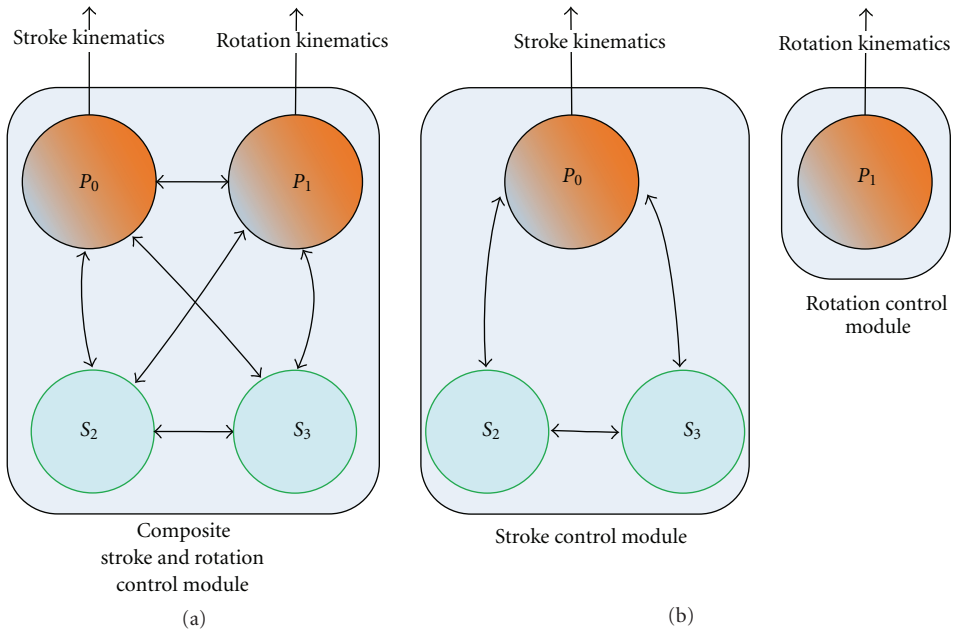

Figure 21: The qualitative functional decomposition derived for the evolved autonomous altitude gain controllers and steer controllers. Most of the steer controller's wing kinematics can be decomposed with a typical CPG-like functional template shown in (a) as a closely coupled stroke and rotation oscillators with steady beat rate and steady amplitude. Most of the altitude gain controllers can be decomposed with the functional template shown in (b), with a dedicated stroke oscillator along with a saturated rotation control module.

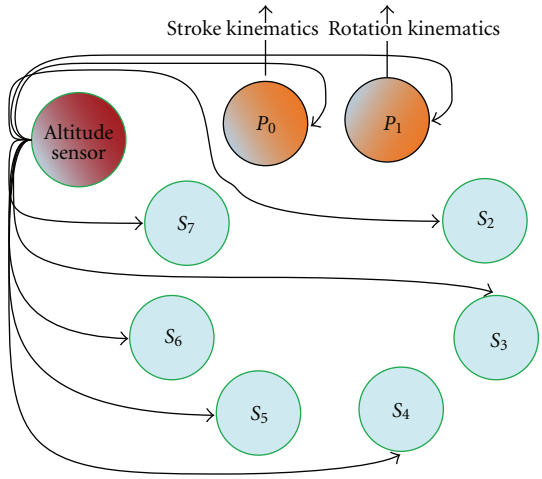

Figure 22: A pictorial representation of the fully connected eight-neuron architecture with a single altitude sensor, of the nonautonomous cruise mode controller chosen for qualitative decomposition analysis. Following the general neuron representation, the stroke and rotation neurons are marked with "P" as primary and "S" as secondary for the other neurons and numbered accordingly from 0 to 7.

exhibits independent oscillatory nature) dependency on the sensor state and further the same dependency can be derived by performing intragroup lesion study on the low frequency group by amputating the sensor pseudoneuron. The second approach is to group only the real neurons by completely ignoring the sensor signal (i.e., amputating the sensor signal) into a high and low frequency groups and study their behavior independently, checking for independent oscillatory behavior, in the absence of the external sensor signal, followed by introducing the sensor signal to detect any significant behavior changes for deducing any possible independent control modules. Though both approaches would yield the same conclusions, the second approach is chosen since the sensor dynamics of the MFWR can be treated separately from the actual neuron dynamics, in two easy steps of complete sensor-independent neuron dynamics decomposition (frequency grouping and intragroup lesion study) followed by the sensor status injection into the possible neuron-level decomposed modules. Thus, moving forward, the six-neuron architecture is disconnected from the external sensor and a frequency-based grouping, with groups mentioned earlier is performed as shown in the pictorial representation Figure 28. Further, an intergroup lesion study is performed, as described earlier (shown in Figure 10(a)), and the outputs of the each neuron in the two groups are presented in Figure 29. It can be noticed that these two frequency groups have indeed self-sufficient dynamics in them to be independent oscillators, with two frequency groups, the high-frequency group and the low-frequency group as demosntrated in above analysis. Moreover, the outputs of each neuron in the high-frequency group match their original output envelope from the reduced six-neuron architecture suggesting that this group's internal dynamics is immune to the external sensor dynamics. But the same cannot be deduced for the low-frequency group, which has high time constants and have already shown its affinity to entrain with the sensor signal. Moreover, this intragroup amputated

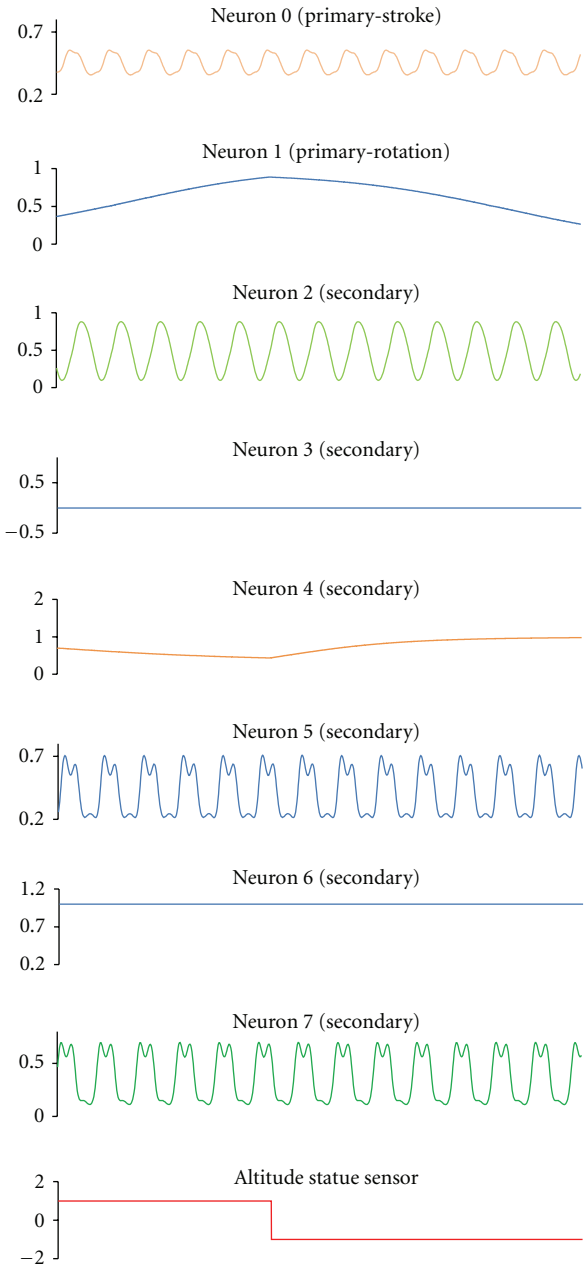

Figure 23: The above figure shows the neuron output state dynamics of each neuron in the fully connected original eight-neuron controller and the external altitude sensor, produced during the flight control of the MFWR to provide optimal cruise behavior. It can be observed that the output states of neurons 1 and 4 entrain with altitude sensor in phase and out of phase, respectively.

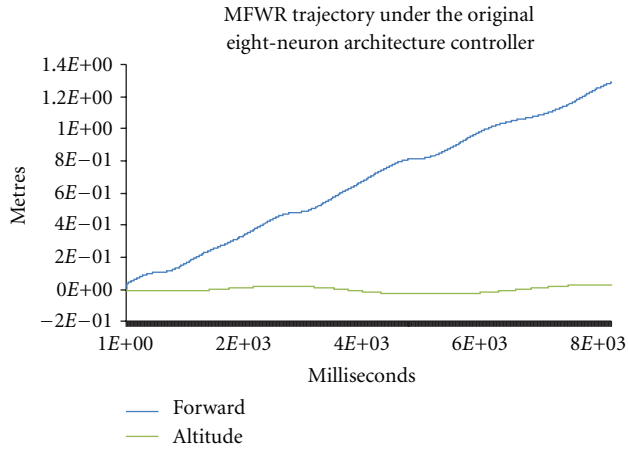

Figure 24: The MFWR trajectory produced by the original fully connected eight-neuron nonautonomous controller. It can be observed that the evolved controller was successful in producing forward motion in the MFWR without any overall gain in the altitude.

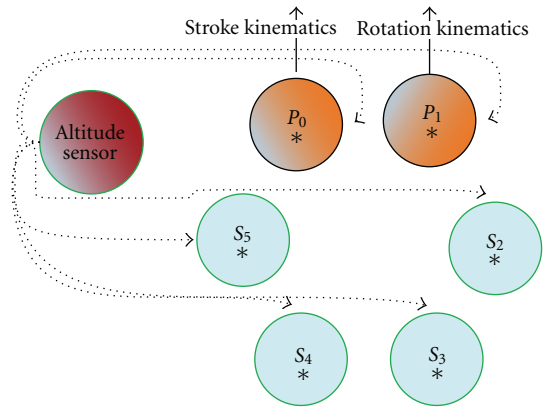

Figure 25: A pictorial representation of the reduced six-neuron architecture of the cruise controller referred in Figure 22. It should be noticed that the primary neurons and second secondary neuron retain their position in the network but the neurons 4, 5, and 7 from the original network are positioned in 3rd, 4th, and 5th locations respectively. The "*" indicates that the neurons in this "dynamics-deprived neuron elimination" process-based architecture differ from the original neuron in the way that, their bias has been accounted for the eliminated neuron's saturated output effect on them. So, at least in steady state, this reduced network should perform functionally equivalent to the original architecture.

stroke and rotation neuron group independent oscillators have been partially successful in controlling the MFWR's expected cruise behavior as shown in Figure 30, in which it can be observed that the altitude of the MFWR is lost with the progression of the time, though the rate of the altitude drop is very much less than the rate of forward motion gain. The next step in this process to deduce any possible modular control structure is to inject the sensor signal dynamics into the established two frequency groups and check for the entrainment behavior and controller's expected cruise mode acceptability on MFWR. While performing the agreed final step in the decomposition process, it was deduced that injecting the sensor signal dynamics only into low-frequency group is sufficient to produce qualitatively acceptable cruise mode behavior in MFWR. Thus, a general qualitative functional decomposition template shown in Figure 31 is derived explaining the evolved non-autonomous cruise mode controllers, as a combination of two independent oscillators, of which the high-frequency oscillator controlled the stroke

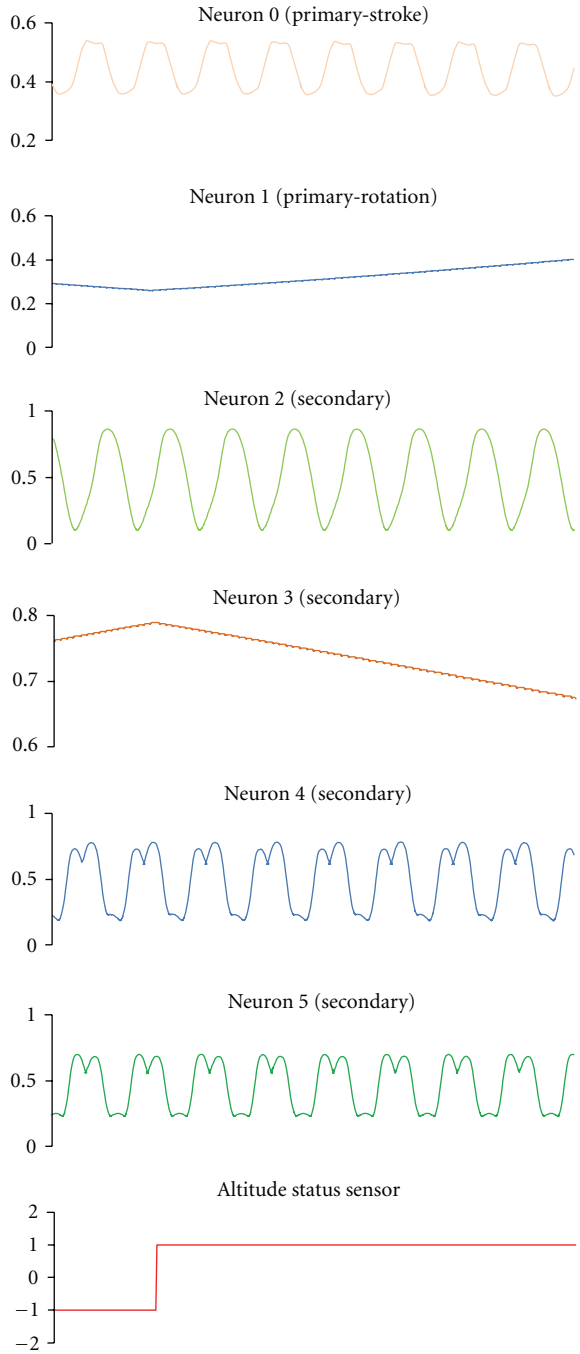

Figure 26: The above figure shows the neuron output state dynamics of each neuron in the reduced six neuron architecture of the cruise controller architecture shown in Figure 22. It can be noticed that the qualitative output dynamics of each neuron do not differ significantly from their original behavior shown in Figure 23. Moreover, It can be observed that the output states of neuron 1 and 3 entrain with altitude sensor in phase and out of phase, respectively.

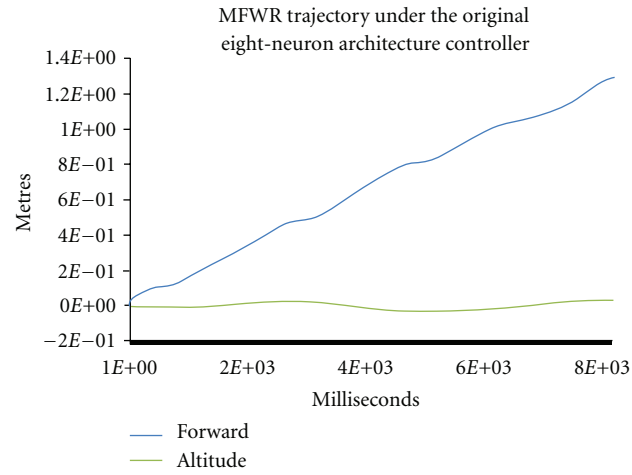

Figure 27: The MFWR trajectory produced by the reduced six-neuron controller. It can be observed that the architectural reduced controller was successful in producing qualitatively same cruise behavior possiblely by the fully connected eight-neuron network. The MFWR trajectory produced by the eight-neuron network is shown in Figure 24.

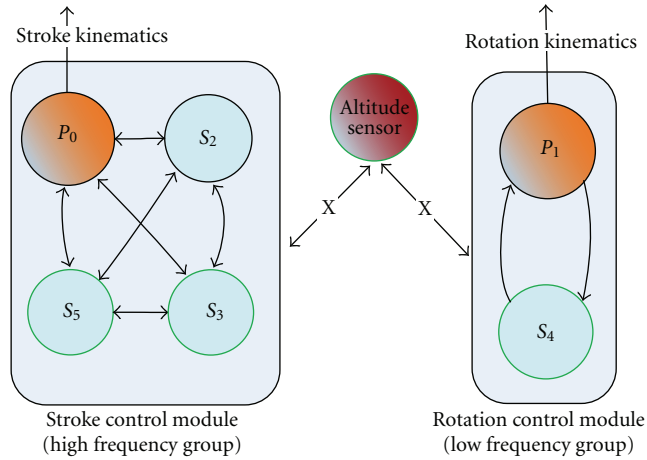

Figure 28: A pictorial representation of the "frequency-based grouping" process combined with intergroup lesion study in absence of the external sensor input for the nonautonomous cruise mode controller shown in Figure 22.

kinematics of the wing with steady amplitude and frequency and the low-frequency oscillator, which was evolved to monitor the altitude variations in the MFWR, through the available external sensor module, altered the rotation dynamics continuously to limit the variations in the altitude of the MFWR and simultaneously provided the forward motion in it, by generating required lift and antilift with the behavior verified by the general principles of the empirical study (mentioned in the acceptability analysis). Three of the best five evolved controllers followed the deduced template, and the other two followed more closed template that is only different from the predominant one in that the stroke frequency group has a dependency on the external sensor status.

4.4.2. Nonautonomous Polymorphic Controllers. Since the polymorphic controllers embed in their architecture both the autonomous altitude gain and cruise mode controllers, which can be invoked as a separate controllers in isolation with a static external signal not a continuous dynamic signal, the qualitative functional decomposition templates

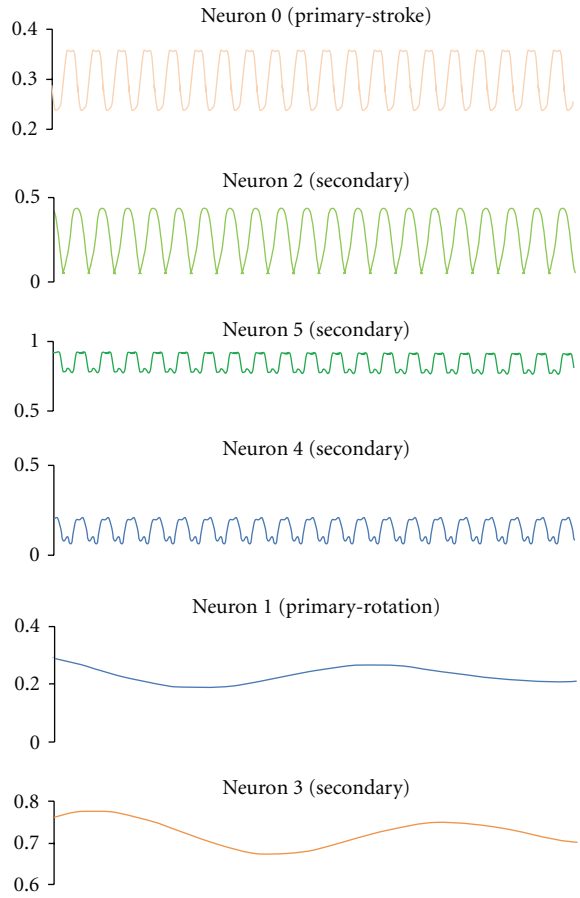

Figure 29: The above figure shows the neuron output state dynamics of each neuron in the low frequency group (1 and 3) and high frequency group (0, 2, 4, and 5) after intergroup amputation is performed and evaluated in absence of the external altitude sensor signal (represented in Figure 28). It can be noticed that each qualitative group is self-sufficient to generate internal dynamics to sustain steady oscillations independent of the external sensor signal. But, nonetheless low frequency group neurons seem to be susceptible to external sensor signal due to their high time constants.

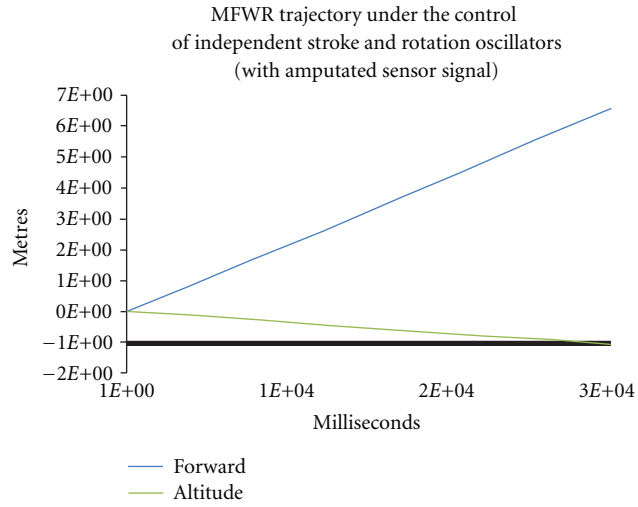

Figure 30: The MFWR trajectory produced by the controller during the lesion study performed with the techniques shown in Figure 28. It can be observed that the amputated two independent stroke and rotation oscillators, in absence of the external sensor, were partially successful in controlling the MFWR to have acceptable cruise behavior in it, suggesting the requirement of the external sensor ingestion to achieve the acceptable cruise control.

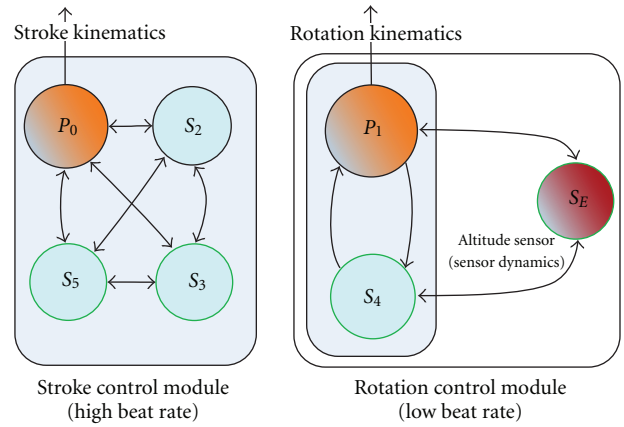

Figure 31: The Qualitative functional decomposition template derived for the non-autonomous cruise mode controllers. Most of the nonautonomous cruise mode controllers can be decomposed into the above-shown template, as a combination of two independent oscillator, of which the high-frequency oscillator, controlled the stroke kinematics of the wing with steady amplitude and frequency and the low-frequency oscillator which was evolved to monitor the altitude variations in the MFWR, through the available external sensor module, altered the rotation dynamics continuously to limit the variations in the altitude of the MFWR and simultaneously provided the forward motion in it, by generating required lift and anti-lift with the behavior verified by the general principles of the empirical study.

presented for the autonomous cruise and altitude gain controllers in the previous section would be applicable for decomposing the polymorphic controllers into two isolated general templates pictorial represented in Figure 21. To further validate the above presented templates, an evolved polymorphic controller's neuron outputs have been evaluated in isolation for cruise and altitude gain command (external sensor value of "0" and "1", resp.) and presented in Figure 32(a) and Figure 32(b), respectively. It can be observed from Figure 32(a) that there exist neurons 2, 3, 6, and 7 which meet dynamics-deprived criteria, and further when their saturated outputs are bias folded into neurons 0, 1, 4, and 5, the dynamics of the reduced four neuron network and its effect on MFWR behavior, namely, cruising behavior, matched the original eight-neuron network's generated behavior. Moreover, as anticipated, the reduced four-neuron network with same output dynamics frequency for all the neurons falls under the composite stroke and rotation control module template presented in Figure 21(a).

Moving forward, it can be observed from Figure 32(b), that there exist neurons 1, 2, 3, 4, and 7, which meet dynamics deprived criteria, and further when their saturated outputs are bias folded into neurons 0, 5, and 6, the dynamics

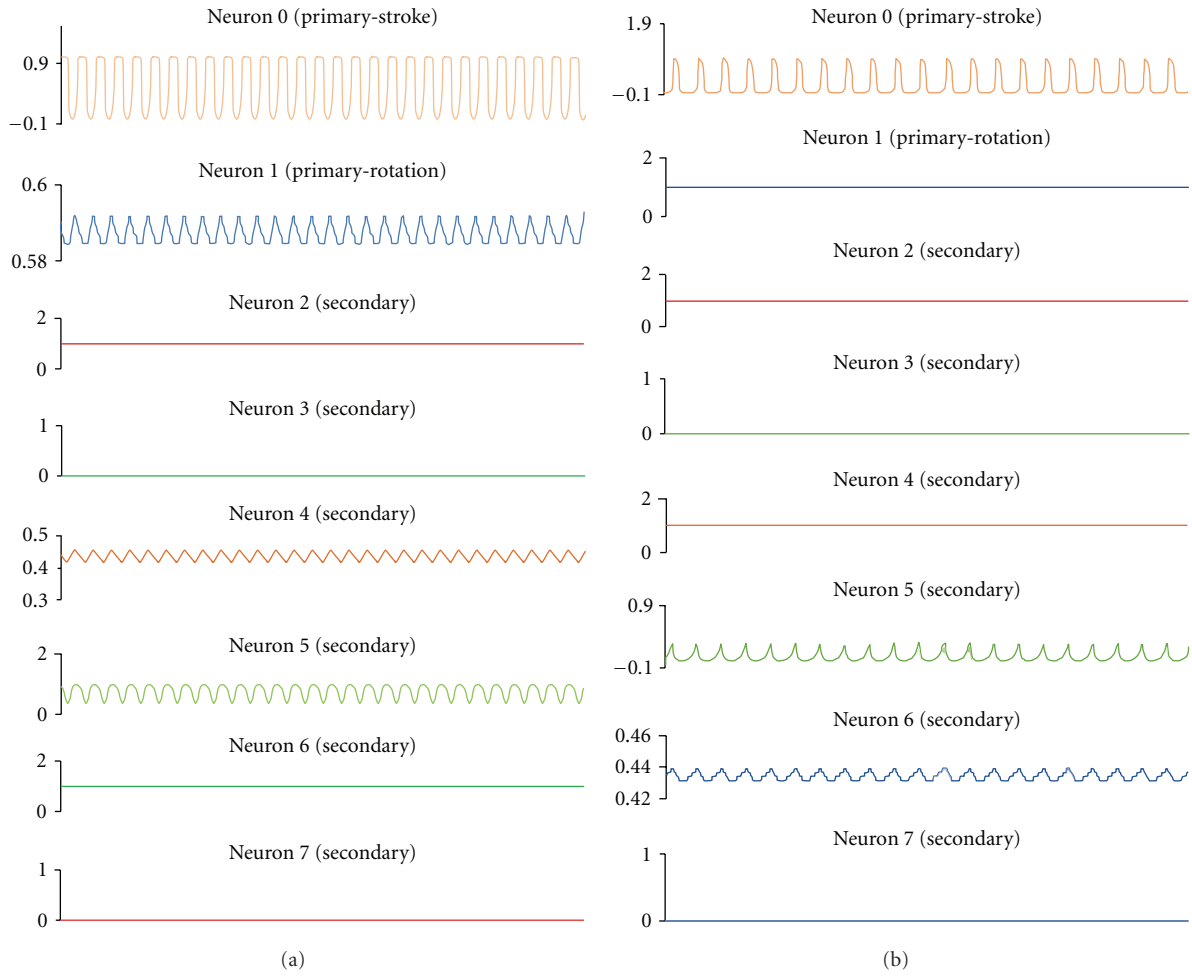

Figure 32: (a) shows the neuron output state dynamics of each neuron in the polymorphic controller when presented with a cruise command, whose the external sensor signal is "0." It can be seen that the neurons 0, 1, 4, and 5 form a composite module with same frequency and would not comply with established criteria for frequency-based grouping. But, the cruise mode controller has been verified to form a single frequency composite stroke and rotation control module as shown in Figure 21(a) and generating appropriate cruise behavior in the MFWR. (b) shows the neuron output state dynamics of each neuron in the polymorphic controller when presented with an altitude gain command whose external sensor signal is "1." It can be seen that the neurons 0, 5, and 6 forms same frequency group for stroke control and a constant rotation produced by saturated neuron 1. The altitude gain mode controller form a two separate independent stroke and rotation control blocks template as shown in Figure 21(b) and generates appropriate altitude gain behavior in the MFWR.

of the reduced four-neuron network and its effect on MFWR behavior, namely, altitude gain behavior, matched the original eight-neuron network's generated behavior. Moreover, as anticipated, the reduced four-neuron network with two independent stroke and rotation control module template presented in Figure 21(b). Thus, when presented with an appropriate external command (sensor) value, the static command would be folded into the exiting neurons in the polymorphic controller architecture, as an appropriately evolved external bias that is responsible to shift the dynamics of the rotation and stroke neurons between autonomous altitude gain and cruise mode controllers, generating appropriate wing kinematics in the MFWR as shown in Figure 7(b). Further, it can be observed by comparing the output neuron dynamics of the cruise mode (shown in Figure 32(a)) and the altitude gain mode (shown in Figure 32(b)) that the external sensor's dynamics modification process is evidently observed when the dynamically active neuron 4 in cruise mode saturates in altitude gain mode, and vice versa for dynamics of output of the neuron 6.

5. Conclusion

In this paper, we have summarized author's prior efforts using the Neuromorphic Evolvable Hardware (CTRNN-EH) framework to successfully evolve locomotion and different flight mode controllers, with detailed emphasis on the flight mode controllers. Further, a new frequency-based analysis procedure has been introduced to analyze the different evolved flight mode controllers, besides providing a brief qualitative analysis suggesting the acceptability of the evolved controllers for the given flight mode in the context. Moreover, the proposed frequency-based analysis methodology has been successfully applied to the evolved

autonomous and nonautonomous controllers, and it has been demonstrated that the methodology can be indeed used to decompose the evolved controllers into logically explainable control blocks for further control analysis. Finally, it can be perceived from the presented results and discussion that the proposed Neuromorphic Evolvable Hardware (CTRNN-EH) and frequency-based analysis methodologies can be employed to control problems that are similar to the flapping flight domain, using tabularasa approach. Though, it is not always an appropriate recommendation to employ a tabularasa approach to the control problems at hand; it can serve as an only approach where a suitably impressive closed-form traditional controller does not exist. Moreover, the above-proposed CTRNN-EH methodologies have also been successfully employed to design and evolve hybrid controllers, with evolvable module in the base traditional controller being evolved to supplement the control characteristics of the traditional controllers with rich dynamics of CTRNNs [20]. These CTRNN-EH-based hybrid controllers have been shown to increase the overall robustness and efficacy of the base traditional controllers to handle unforeseen changes in the assumed environment of the controller and the controlled vehicle [21].

References

[1] V. B. Floris and H. Lipson, "Evolving buildable flapping ornithopters," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO '05)*, 2005.

[2] L. Schenato, X. Deng, W. C. Wu, and S. Sastry, "Virtual Insect Flight Simulator (VIFS): a software testbed for insect flight," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '01)*, vol. 4, pp. 3885–3892, May 2001.

[3] J. C. Gallagher, "An evolvable hardware layer for global and local learning of motor control in a hexapod robot," *International Journal on Artificial Intelligence Tools*, vol. 14, no. 6, pp. 999–1017, 2005.

[4] S. K. Boddhu and J. C. Gallagher, "Evolved neuromorphic flight control for a flapping-wing mechanical insect model," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, IEEE Press, Hong Kong, China, June 2008.

[5] H. J. Chiel, R. D. Beer, and J. C. Gallagher, "Evolution and analysis of model CPGs for walking: I. Dynamical modules," *Journal of Computational Neuroscience*, vol. 7, no. 2, pp. 99–118, 1999.

[6] J. C. Gallagher, "Evolution and analysis of non-autonomous neural networks for walking: reflexive pattern generators," in *Procedings of the Congress on Evolutionary Computation*, IEEE Press, Seoul, South Korea, May 2001.

[7] S. K. Boddhu and J. C. Gallagher, "Evolving non-autonomous neuromorphic flight control for a flapping-wing mechanical insect," in *Proceedings of the IEEE Workshop on Evolvable and Adaptive Hardware (WEAH '09)*, Nashvile, Tenn, USA, April 2009.

[8] S. K. Boddhu and J. C. Gallagher, "Evolving neuromorphic flight control for a flapping-wing mechanical insect," *International Journal of Intelligent Computing and Cybernetics*, vol. 3, no. 1, pp. 94–116, 2010.

[9] S. K. Boddhu, *Evolution and analysis of neuromorphic flapping-wing flight controllers [Ph.D. thesis]*, Wright State University, 2010.

[10] D. Sbarbaro-Hofer, D. Neumerkel, and K. Hunt, "Neural control of a steel rolling mill," *IEEE Control Systems Magazine*, vol. 13, no. 3, pp. 69–75, 1993.

[11] C. M. Lin and C. F. Hsu, "Supervisory recurrent fuzzy neural network control of wing rock for slender delta wings," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 5, pp. 733–742, 2004.

[12] S. K. Boddhu, J. C. Gallagher, and S. A. Vigraham, "A commercial off-the-shelf implementation of an analog neural computer," *International Journal on Artificial Intelligence Tools*, vol. 17, no. 2, pp. 241–258, 2008.

[13] S. K. Boddhu, J. C. Gallagher, and S. Vigraham, "A reconfigurable analog neural network for evolvable hardware applications: Intrinsic evolution and extrinsic verification," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, IEEE Press, Vancouver, Canada, July 2006.

[14] K. I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[15] G. R. Kramer and J. C. Gallagher, "An analysis of the search performance of a mini-population evolutionary algorithm for a robot-locomotion control problem," in *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE '05)*, pp. 2768–2775, IEEE Press, September 2005.

[16] R. D. Beer, H. J. Chiel, and J. C. Gallagher, "Evolution and analysis of model CPGs for walking: II. General principles and individual variability," *Journal of Computational Neuroscience*, vol. 7, no. 2, pp. 119–147, 1999.

[17] M. H. Dickinson, F. O. Lehmann, and S. P. Sane, "Wing rotation and the aerodynamic basis of insect right," *Science*, vol. 284, no. 5422, pp. 1954–1960, 1999.

[18] L. Schenato, X. Deng, and S. Sastry, "Flight control system for a micromechanical flying insect: architecture and implementation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '01)*, vol. 2, pp. 1641–1646, May 2001.

[19] R. D. Beer, "On the dynamics of small continuous time recurrent neural networks," in *Adaptive Behavior*, vol. 3, 4, pp. 469–509, The MIT Press, Boston, Mass, USA, 1995.

[20] J. C. Gallagher, B. David Doman, and W. Michael Oppenheimer, "The technology of the gaps: an evolvable hardware synthesized oscillator for the control of a flapping-wing micro air vehicle," *IEEE Transactions on Evolutionary Computation*. In press.

[21] J. C. Gallagher and M. Oppenheimer, "An improved evolvable oscillator for all flight mode control of an insect-scale flapping-wing micro air vehicle," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '11)*, IEEE Press, 2011.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 793176, 7 pages
doi:10.1155/2012/793176

Research Article

MIMO Lyapunov Theory-Based RBF Neural Classifier for Traffic Sign Recognition

King Hann Lim,[1] Kah Phooi Seng,[2] and Li-Minn Ang[3]

[1] *Electrical and Computer Department, School of Engineering and Science, Curtin University, Sarawak Malaysia, CDT 250, 98009 Miri Sarawak, Malaysia*
[2] *School of Computer Technology, Sunway University, No. 5, Jalan Universiti, Bandar Sunway, Selangor Darul Ehsan, 46150 Petaling, Malaysia*
[3] *Centre for Communications Engineering Research, Edith Cowan University, Joondalup, WA 6027, Australia*

Correspondence should be addressed to King Hann Lim, glkhann@curtin.edu.my

Received 27 October 2011; Revised 21 February 2012; Accepted 22 February 2012

Academic Editor: Toly Chen

Lyapunov theory-based radial basis function neural network (RBFNN) is developed for traffic sign recognition in this paper to perform multiple inputs multiple outputs (MIMO) classification. Multidimensional input is inserted into RBF nodes and these nodes are linked with multiple weights. An iterative weight adaptation scheme is hence designed with regards to the Lyapunov stability theory to obtain a set of optimum weights. In the design, the Lyapunov function has to be well selected to construct an energy space with a single global minimum. Weight gain is formed later to obey the Lyapunov stability theory. Detail analysis and discussion on the proposed classifier's properties are included in the paper. The performance comparisons between the proposed classifier and some existing conventional techniques are evaluated using traffic sign patterns. Simulation results reveal that our proposed system achieved better performance with lower number of training iterations.

1. Introduction

Traffic sign recognition is important in autonomous vehicular technology for the sake of identifying a sign functionality through visual information capturing via sensors. The usage of neural networks has become increasingly popular in traffic sign recognition recently to classify various kinds of traffic signs into a specific category [1–3]. The reason of applying neural networks in traffic sign recognition is that, they can incorporate both statistical and structural information to achieve better performance than a simple minimum distance classifier [4]. The adaptive learning capability and processing parallelism for complex problems have led to the rapid advancement of neural networks. Among all neural networks, *radial basis function neural network* (RBFNN) has been applied in many engineering applications with the following significant properties: (i) universal approximators [5]; (ii) simple topological structure [6] which allows straightforward computation using a linearly weighted combination of single hidden-layer neurons. The learning characteristic of RBFNN is greatly related to the associative weights between hidden-output nodes. Therefore, an optimal algorithm is required to update the weights relative to an arbitrary training input.

Conventionally, the training process for RBFNN is mainly dependent on the optimization theory. The cost function of this network, for instance, the sum of squared errors or mean squared error between network's output and targeted input is firstly defined. It is followed by minimizing the cost function in weight parameter space to search for a set of optimal weights. These optimal weights which are acquired throughout network training process can be used to perform some unique tasks, such as pattern classification.

In order to obtain the optimum weights, a number of training algorithms have been developed for RBFNN. Due to the linear-weighted combiner, network's weights can be

determined using *least mean square* (LMS) and *recursive least square* (RLS) algorithms. However, these algorithms suffer from several drawbacks and limitations. The LMS is highly dependent on the autocorrelation function associated with the input signals and slow convergence. RLS, on the other hand, provides faster convergence but they depend on the implicit or explicit computation using the inverse of input signal's autocorrelation matrix. Matrix inversion implies not only a higher computational cost and it also leads to network instability issue [7]. Other gradient search-based training algorithms also suffer so-called local minima problem, that is, the optimization search may trap at a local minimum of the cost function in the weight space if a set of initial values are arbitrarily chosen. For example, the cost function has a fixed structure in the weight space after the expression of the cost function is chosen. The parameter update law is only a means to search for the global minimum and independent of the cost function in the weight space.

To overcome the aforementioned problems, the optimization techniques using the Lyapunov stability theory has been proposed in [8] for adaptive filtering. This theory is further adopted to the design of RBFNN [9] which has been first proposed in realization of *finite-impulse response* (FIR) and *infinite-impulse response* (IIR) adaptive filters for signal noise filtering. The Lyapunov theory-based RBFNN has been increasingly popular in adaptive filter due to its stability guarantee by Lyapunov stability theory and an energy-space construction with a global minimum [9]. However, only single output is designed in the Lyapunov theory-based RBFNN and, hence, it is not suitable for classification problem. In the meantime, the Lyapunov stability theory is also applied to *multilayered neural network* (MLNN) [10] for solving *multiple inputs multiple outputs* (MIMOs) problems. With Taylor series expansion, all MLNN weights between input-output layers are rearranged into a linear configuration with an assumption made, that is, the input-output layer's weights adjustment is dependent on its corrective output error. Therefore, it leads to longer training time with a couple numbers of weight linkage and it prone to have more weights uncertainties. Nevertheless, the Lyapunov theory-based neural classifiers have offered the following advantages [9, 10]: (i) fast error convergence, (ii) guarantee of stability, (iii) insensitivity to initial condition, and (iv) construction of weight space with a global minimum if a proper Lyapunov function is selected.

In this paper, the notion of Lyapunov stability theory on RBFNN can be extended and modified to solve MIMO problems such as traffic sign recognition in order to obtain a fast and reliable classification system. To reveal the performance of proposed system, the application for traffic sign classification is used for further discussion and analysis. The performance of the proposed RBFNN will be compared with [10] and the differences for both methods will be stated in the later section. Experimental results show that the proposed method leads to faster error convergence rate and higher recognition rate as compared to the conventional techniques. This paper is organized as the following: Section 2 discusses about the fundamental theory of RBF neural classifier while Section 3 explains the theoretical design of the Lyapunov theory-based training algorithm. Section 4 describes an overview of traffic sign detection and recognition on Malaysia's traffic sign database. Some simulation results along with the application of traffic sign recognition are shown in Section 5 and it is finally followed by conclusion and future works.

2. Radial Basis Function Neural Network

A typical three-layer RBFNN [11] is illustrated in Figure 1 for pattern recognition. Such a network implements an input-output mapping: $\Re^n \rightarrow \Re^m$, where n depicts the number of inputs and m depicts the number of outputs. There are u hidden nodes connecting in between the input-output layer. Assuming the first layer of input vector, $\mathbf{X}^{(1)}$ is set to be $\mathbf{P} \in \Re^n$, where the input data is arranged into column vector as $\mathbf{P} = [p_1, p_2, \ldots, p_n]^T$. The RBF centers are denoted as $C_j \in \Re^n$ $(1 \leq j \leq u)$. Each RBF unit is defined as:

$$X_j^{(2)}(P) = \exp\left(-\frac{\left\|X^{(1)} - C_j\right\|^2}{\sigma_j^2}\right), \tag{1}$$

where $\|\cdot\|$ indicates the Euclidean norm on the input space while σ_j is the Gaussian width of the jth RBF unit. The vector generated after the RBF neurons is given as $\mathbf{X}$ below:

$$\begin{bmatrix} \psi\left(\|\mathbf{X}^{(1)} - C_1\|^2\right) \\ \psi\left(\|\mathbf{X}^{(1)} - C_2\|^2\right) \\ \vdots \\ \psi\left(\|\mathbf{X}^{(1)} - C_u\|^2\right) \end{bmatrix} = \begin{bmatrix} X_1^{(2)} \\ X_2^{(2)} \\ \vdots \\ X_u^{(2)} \end{bmatrix} = \mathbf{X}, \tag{2}$$

where $\psi(\cdot)$ is the radial basis function. Consider the hidden nodes are linearly mapped to the output with the $u \times m$ weights matrix formed as below:

$$\begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{m,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{m,2} \\ \vdots & & \ddots & \vdots \\ w_{1,u} & \cdots & \cdots & w_{m,u} \end{bmatrix} = \mathbf{W}, \tag{3}$$

RBF network establishes a linear function mapping in the output layer. By multiplying (2) and (3), the weighted hidden values are summed to be the output matrix, $\mathbf{Y}$ is as follows:

$$\mathbf{Y} = \operatorname{diag}\left(\mathbf{W}^T\mathbf{X}\right), \tag{4}$$

where

$$\mathbf{Y} = \operatorname{diag}(y_1, y_2, \ldots, y_m) = \begin{bmatrix} y_1 & \cdots & \cdots & 0 \\ 0 & y_2 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & \cdots 0 & \cdots & y_m \end{bmatrix}, \tag{5}$$

and $\operatorname{diag}(a)$ is the $N \times N$ diagonal matrix whose entries are the N elements of the vector a.

Two important parameters are associated with each RBF unit. They are RBF center (C_j) and the Gaussian width (σ_j).

Every center should well represent the corresponding subclass because the classification in RBFNN is mainly measured based on the distances between the input samples and the centers of each subclass. There are different strategies to select RBF centers [11]. These strategies can be classified into supervised [12, 13] and unsupervised algorithms [14–16]. Once centers are well selected, the network starts learning the training data with the weight updating scheme. The selection of center and radius is not the main concern in this paper. Therefore, k-mean cluster is applied to search for the RBF centers while p-nearest neighbor is used to measure the nearest radius for p consecutive centers in the network evaluations.

Learning algorithm plays an important role in updating the weight between hidden and output layer of RBFNN. Conventional RBF with LMS algorithm [17, 18] is trained using gradient descent method which finds the negative gradient on the error curve. It suffers from slow convergence and it is always trapped in the local minima instead of global minima. On the other hand, conventional RBF using RLS method [19] computes the inverse of autocorrelation matrix associated with the input data to update the system weights. However, the inversion of RLS algorithm gives instability of system convergence and increases computational cost.

3. Lyapunov Theory-Based RBFNN

The idea of Lyapunov theory-based RBF filter was initially developed in [9] for adaptive filtering. Lyapunov function of errors between targeted outputs and actual outputs are first defined. Network weights are then adjusted based on the Lyapunov stability theory, so that errors can asymptotically converge to zero. The selected Lyapunov function has a unique global minimum point in the state space. By properly choosing a weight update law in the Lyapunov sense, RBF outputs will be asymptotically converged to the target outputs. In this section, the design in [9] is adopted and modified to apply the Lyapunov theory-based RBF neural classifier for solving MIMO classification problem.

The input vector $\mathbf{P}$ is fed into RBF nodes and hence passed to the output layer by weighted sum with the formulas depicted in (1)–(4). For the given desired response $\hat{\mathbf{Y}}_k = \text{diag}(d_1, d_2, \ldots, d_m) \in \Re^{m\times m}$ at discrete time k, the Lyapunov function is initially chosen as:

$$V_k = \|\mathbf{E}_k\|^2, \tag{6}$$

and $\mathbf{E}_k$ is a posteriori error with $m \times m$ diagonal matrix defined as below:

$$\mathbf{E}_k = \hat{\mathbf{Y}}_k - \mathbf{Y}_k = \hat{\mathbf{Y}}_k - \text{diag}\left(\mathbf{W}_k^T \mathbf{X}_k\right). \tag{7}$$

For the given hidden layer input $\mathbf{X}_k$ and the desired output $\hat{\mathbf{Y}}_k$, the weight matrix $\mathbf{W}_k$ is updated as follows:

$$\mathbf{W}_k = \mathbf{W}_{k-1} + \mathbf{g}_k^T \boldsymbol{\alpha}_k. \tag{8}$$

The adaptive gain is modified for MIMO such that $\Delta V < 0$:

$$\mathbf{g}_k = \left[\mathbf{I}_{m\times m} - \kappa \boldsymbol{\alpha}_k^{-1} \mathbf{E}_{k-1}\right] \times \mathbf{1}_{m\times 1} \times \frac{\mathbf{X}_k^T}{\|\mathbf{X}_k\|^2}, \tag{9}$$

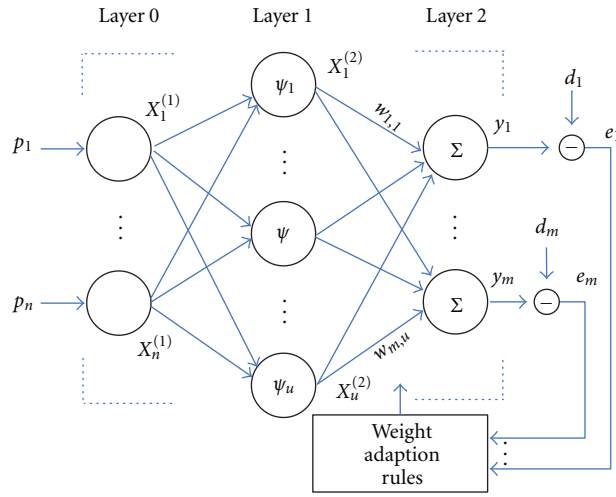

Figure 1: MIMO radial basis function network structure.

whereas $\mathbf{I}_{m\times m}$ is identity matrix, $\mathbf{1}_{m\times 1}$ is column vector of ones, κ is valid for the interval $[0, 1]$, and $\boldsymbol{\alpha}_k$ is the priori estimation error which is defined as:

$$\boldsymbol{\alpha}_k = \hat{\mathbf{Y}}_k - \text{diag}\left(\mathbf{W}_{k-1}^T \mathbf{X}_k\right). \tag{10}$$

Then the error $\|\mathbf{E}_k\|$ asymptotically converges to zero as the time k goes infinity.

The selection of Lyapunov function is important in constructing a new cost-function of the system. Regarding to the Lyapunov stability theory, V_k should be selected in the sense of Lyapunov that: $\Delta V_k = V_k - V_{k-1} < 0$. For mth output nodes RBFNN network, V_k is chosen to be (6) with the summation of squared errors. With the parameters predefined in expressions (7)–(10), it is proved in Appendix A that ΔV_k has a negative value and the Lyapunov stability theory is fulfilled. Only when the parameter update law is chosen in the Lyapunov sense, $V_k = \|\mathbf{E}_k\|^2$ is the Lyapunov function of RBFNN system, which has a unique global minimum.

As stated previously, the training error converges to zero asymptotically as time increases to infinity. For mth output nodes RBFNN, the classification error, $\|\mathbf{E}_k\|$is proved to be asymptotically approaching zero when the training time increases with the gain given in (9). The proof of error convergence is given in Appendix B. The error is bounded to a single convergent value. It is noted that the error convergence rate is dependent on the positive constant κ. For the faster error converges, κ should be remained as a small value in the range of $0 \le \kappa < 1$.

To prevent the singularities, the expression (9) can be modified to:

$$\mathbf{g}_k = \left[\mathbf{I}_{m\times m} - \kappa \boldsymbol{\alpha}_k^{-1} \mathbf{E}_{k-1}\right] \times \mathbf{1}_{m\times 1} \times \frac{\mathbf{X}_k^T}{\beta_1 + \|\mathbf{X}_k\|^2}, \tag{11}$$

where β_1 is a small positive integer.

Table 1: Malaysia traffic sign classification in shape and color.

Color	Shape				
	Diamond	Triangle downward	Square/rectangle	Circle	Octagon
Blue	—	—	Information	Obligation	—
Red	—	Yield sign	—	Prohibition	Stop sign
Yellow	Warning	—	Warning	—	—
Orange			Construction		
White	—	—	—	Speed limit	—

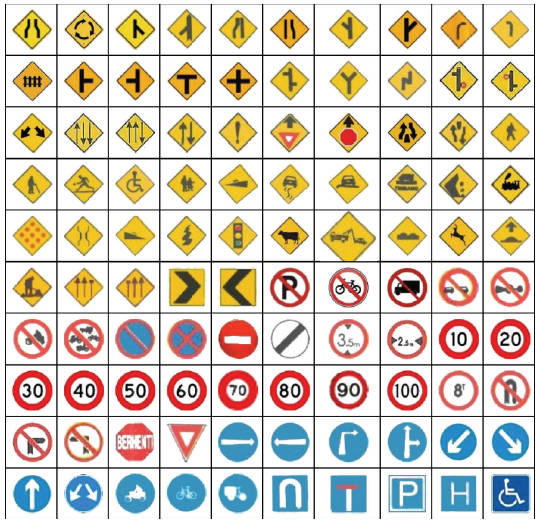

Figure 2: 100 classes of Malaysia traffic signs.

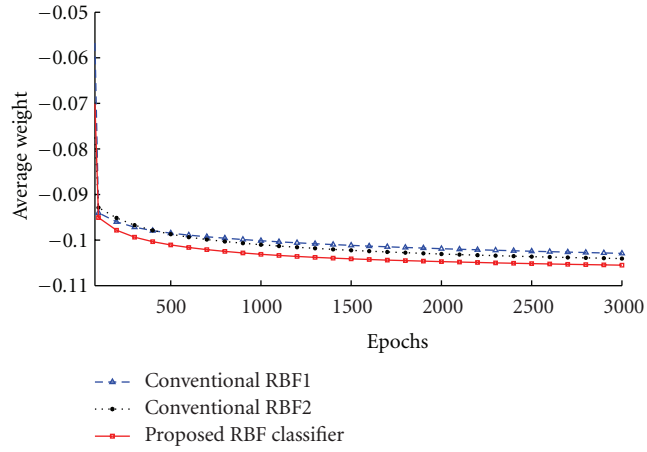

Figure 3: Weight convergence for different RBF classifiers.

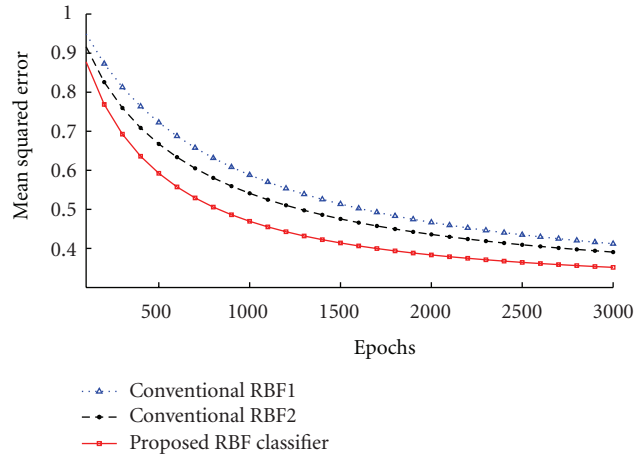

Figure 4: Error convergence on RBF networks.

4. Traffic Sign Detection and Recognition for Malaysia Traffic Sign Database

Traffic signs are important to alert driver of the current road situation. As demonstrated in Figure 2, 100 classes of traffic signs can often be found on Malaysia's roadside. The iconic traffic signs are acquired from the Malaysia's Highway Code Test booklet. They are designed in the standard geometrical shapes such as triangle, circle, octagon, rectangle, square, or diamond. The dominant colors that used for traffic signs are yellow, blue, red, orange, black, and white, which is greatly distinguishable from the natural scene. The message of warning, prohibition, guidance, construction and maintenance are represented by the specific color and shape as depicted in Table 1. They may contain a pictogram, a string of characters or both.

Traffic sign is detected based on color and shape information from a road scene. Possible sign is then extracted for further verification. Traffic sign can be verified by using its symmetrical property. Subsequently, the detected sign region is arranged into a column vector and this vector is inserted to the neural networks for classification. Some evaluations on traffic sign recognition using Malaysia's database with the proposed RBFNN and other neural classifiers will be discussed in the following section.

5. Experimental Results and Discussions

The Lyapunov theory-based RBFNN was developed in this paper to have multiple output classification and, therefore, traffic sign recognition was applied to be a MIMO problem throughout several experiments. In order to observe and evaluate the performance of proposed RBFNN, a basic structure of RBFNN was set up, where the output layer was assigned to classify 100 tasks. The task was to recognize any 100 signs randomly picked from Malaysia's database as displayed in the previous section. Each traffic sign subject

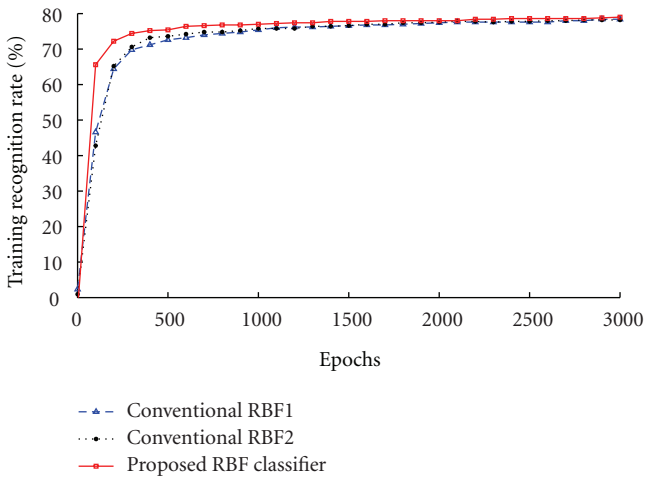

FIGURE 5: The training recognition rate in RBF networks versus epochs.

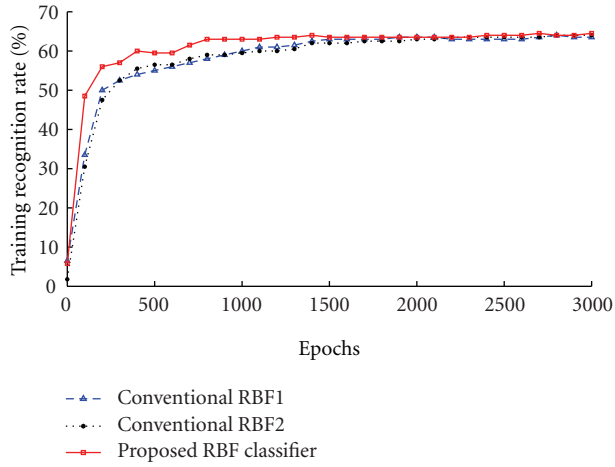

FIGURE 6: The testing recognition rate in RBF networks versus epochs.

contained five training images and two testing images respectively. All images were in the dimension of 32×32.

Three types of RBF classifiers with different learning schemes were used in the experiments, that is, (i) conventional RBF1 with LMS learning approach, (ii) conventional RBF2 with RLS technique, and (iii) Lyapunov theory-based RBF classifier. Network performance was compared in terms of weight and error convergence for these RBF systems. In addition, recognition rate was plotted along with the number of iterations for training and testing dataset. Lyapunov theory-based algorithm obtained a parameter, $\kappa = 0.1$, while LMS algorithm obtained a learning rate (η), whereas the range was varied from 0.1 to 1.0. Meanwhile, the forgetting factor (λ) of RLS algorithm was set to 0.1. During the experiments, the number of input neurons was set to be the image size and the number of output neurons was equal to the number of classes to be identified. The hidden layer of RBF nodes obtained the same size as the input layer. Two initial conditions were set as (i) initial weights $\mathbf{W}_0$ were set to be some random small values and (ii) initial posterior error, $\mathbf{E}_0$ was set to be $0.01 \times \mathbf{I}_{m \times m}$, where $m = 100$. In the experiment, an epoch represented an efficient time used to train the entire set of input data. Hence, each epoch updated the weights once for the whole training dataset until the occurrence of weight convergence and this was referring to batch learning scheme.

Weight convergence was crucial in neural network learning characteristic. The weights for all RBF classifiers were converged at equilibrium with a random set of initial weights. Figure 3 showed a weight convergence plot, where x-axis denoted the number of epochs while y-axis denoted the average weight values connected from hidden layer to output layer of RBF network. At around 250 epochs, proposed RBF classifier started converging at -0.1. As time extended to infinity, only minor weight increment was added along with the continuous epoch. However, conventional RBF classifiers reached -0.1 equilibrium point at 1000 epochs, which took longer time than the Lyapunov theory-based RBF classifier. Therefore, the weight of proposed RBF classifier was converged faster than other conventional RBF classifiers.

As demonstrated in Figure 4, an error curve was plotted and it was exponentially decreased to the minimum. At 2000 epochs, the proposed RBF classifier reached 0.4 of mean squared error. For other conventional RBFNNs, they took longer period to achieve lower error rate as shown in Figure 4. Hence, the error convergence rate of proposed RBF training scheme was converged faster than other networks. However, the error value was not converged to zero yet due to some reasons, whereby this issue was caused by the redundancy information of input data. Moreover, similarity of subject pattern would confuse the classification of neural network as you could observe in Figure 2. The complexity of the input data would degrade recognition rate in classification. Hence, all neural classifiers' error was still remained at around 0.4, but the proposed neural classifier obtained lower error curve compared to the other networks. To reduce the classification error, some processing techniques could be applied to the input data before passing to the neural network for training process. In addition, traffic signs could first be classified into color and shape before they were recognized using neural network [4]. In this paper, the main concern was to focus on the property of RBF training schemes instead of traffic sign recognition, in which it was an application example for the proposed network discussion.

To investigate the performance of RBF classifiers, training and testing data were fed into system and an average recognition rate was calculated. Without any feature extraction method, the original images were tested with the RBF classifiers. The system performance was compared with conventional RBFNNs using different weight updating schemes. As demonstrated in Figure 5, the Lyapunov theory-based RBFNN obtained faster training speed which was within 500 epochs. Meanwhile, it achieved higher training recognition rate than other networks. Similar to Figure 6, the proposed RBF network obtained high recognition rate at 800 epochs while other training algorithms achieved maximum recognition rate at 1500 epochs. This implied that Lyapunov

theory-based RBFNN could obtain high recognition rate in a smaller epoch's number. By comparing the network recognition rate, conventional RBF1 achieved 63.50%, while conventional RBF2 achieved 64% of recognition rate. However, the testing recognition rate of the proposed classifier was slightly increased with the rate of 64.50%. As it was reported in [12], high dimensional input data would cause low recognition rate of neural network because high dimensional input data and network complexity would need a large set training samples. Therefore, features extraction technique such as *Principle Component Analysis* (PCA) could be used to reduce the image dimension in order to further increase d the recognition rate.

Besides that, MLNN was also used to classify multidimensional patterns based on Lyapunov stability theory as it was developed in [10]. In order to test the performance between the Lyapunov theory-based RBFNN and MLNN, the number of hidden nodes was fixed to be 100 and the number of epochs was set to be 3000 for both networks. The constant parameter κ is set to be 0.1 for both the Lyapunov-theory based MLNN and RBFNN with similar training and testing images datasets as employed in the RBFNN experiments. An averaged recognition rate was recorded for the comparison.

As reported in Table 2, the Lyapunov theory-based RBFNN achieved better classification performance than [10]. Although both methods employ the Lyapunov stability theory as the basic design for training algorithm, there were some main differences between them. First, the structure for two networks was much distinguished from input to output layers. MLNN contained more weights linkage between input to hidden layer and hidden to output layer. Hence, the weights of MLNN were required to be linearized using Taylor series expansion before performing training process. With this MLNN configuration, it contributed to more uncertainties on the weight and error convergence compared to the linear topology of RBFNN. Second, the energy function of training error for MLNN was constructed for each output node where it contained tth number of output nodes. Unlike to [10], the energy function for the Lyapunov theory based RBFNN was designed for all output neurons in a matrix form and it provided a valid derivation in the appendices. Finally, the performance for the Lyapunov theory-based RBFNN achieved better training and testing recognition rate than [10] in the pattern recognition.

6. Conclusion

This paper has presented Lyapunov theory-based weights updating algorithm for RBFNN. The weight adaptation scheme is designed based on the Lyapunov stability theory and iteratively updated the RBFNN weight. The Lyapunov theory-based RBFNN is extended for traffic sign recognition as a MIMO problem. Simulation results have shown that our proposed system achieved faster training speed, as well as higher recognition rate. The recognition rate can be further improved by applying input dimensionality reduction to remove the information redundancy. The research on the optimization using Lyapunov stability theory is still at its early stages, and many investigations of Lyapunov theory-based neural classifier will be conducted to improve the network efficiency and robustness. Future investigation on different Lyapunov functions and different weights updating laws is needed to further improve the performance.

Appendices

A. Proof of Lyapunov Stability Theory

The discrete form of ΔV_k is given as:

$$\begin{aligned}
\Delta V_k &= \|\mathbf{E}_k\|^2 - \|\mathbf{E}_{k-1}\|^2 \\
&= \left\|\hat{\mathbf{Y}}_k - \operatorname{diag}(\mathbf{W}_k^T\mathbf{X}_k)\right\|^2 - \|\mathbf{E}_{k-1}\|^2 \\
&= \left\|\boldsymbol{\alpha}_k - \boldsymbol{\alpha}_k^T\mathbf{g}_k\mathbf{X}_k\right\|^2 - \|\mathbf{E}_{k-1}\|^2 \\
&= \|\kappa\mathbf{E}_{k-1}\|^2 - \|\mathbf{E}_{k-1}\|^2 \\
&= \kappa^2\|\mathbf{E}_{k-1}\|^2 - \|\mathbf{E}_{k-1}\|^2 \\
&= -(1-\kappa^2)\|\mathbf{E}_{k-1}\|^2 < 0,
\end{aligned} \tag{A.1}$$

where $0 \le \kappa < 1$.

B. Proof of Error Convergence

$$\begin{aligned}
\mathbf{E}_k &= \hat{\mathbf{Y}}_k - \mathbf{Y}_k \\
&= \hat{\mathbf{Y}}_k - \operatorname{diag}\left(\mathbf{W}_k^T\mathbf{X}_k\right) \\
&= \hat{\mathbf{Y}}_k - \operatorname{diag}\left[\left(\mathbf{W}_{k-1}^T + \boldsymbol{\alpha}_k^T\mathbf{g}_k\right)\mathbf{X}_k\right] \\
&= \hat{\mathbf{Y}}_k - \operatorname{diag}\left(\mathbf{W}_{k-1}^T\mathbf{X}_k\right) - \operatorname{diag}\left(\boldsymbol{\alpha}_k^T\mathbf{g}_k\mathbf{X}_k\right) \\
&= \boldsymbol{\alpha}_k - \operatorname{diag}\left\{\boldsymbol{\alpha}_k^T\left[\left(\mathbf{I}_{m\times m} - \kappa\boldsymbol{\alpha}_k^{-1}\mathbf{E}_{k-1}\right)\right.\right. \\
&\qquad \left.\left.\times\mathbf{1}_{m\times 1}\times\frac{\mathbf{X}_k^T}{\|\mathbf{X}_k\|^2}\right]\mathbf{X}_k\right\}.
\end{aligned} \tag{B.1}$$

Since $\boldsymbol{\alpha}_k$ is a symmetric matrix, $\boldsymbol{\alpha}_k^T = \boldsymbol{\alpha}_k$,

$$\mathbf{E}_k = \boldsymbol{\alpha}_k - \operatorname{diag}[(\boldsymbol{\alpha}_k - \kappa\mathbf{E}_{k-1})\times\mathbf{1}_{m\times 1}]. \tag{B.2}$$

Due to $(\boldsymbol{\alpha}_k - \kappa\mathbf{E}_{k-1}) \in \Re^{m\times m}$, $\operatorname{diag}(\cdot)$ and $\mathbf{1}_{m\times 1}$ can be canceled off from the equation:

$$\begin{aligned}
\mathbf{E}_k &= \boldsymbol{\alpha}_k - \boldsymbol{\alpha}_k + \kappa\mathbf{E}_{k-1} \\
&= \kappa\mathbf{E}_{k-1} \\
\therefore \|\mathbf{E}_1\|^2 &= \kappa\|\mathbf{E}_0\|^2 \\
\|\mathbf{E}_2\|^2 &= \kappa\|\mathbf{E}_1\|^2 = \kappa^2\|\mathbf{E}_0\|^2 \\
&\vdots \\
\|\mathbf{E}_k\|^2 &= \lim_{k\to\infty}\kappa^k\|\mathbf{E}_0\|^2 \approx 0,
\end{aligned} \tag{B.3}$$

where $\|\mathbf{E}_0\|^2$ is an $m \times m$ diagonal matrix with small real integer and $0 \le \kappa < 1$.

Table 2: Performance comparison for the Lyapunov theory-based neural networks.

Method	Hidden nodes	Epochs	Training recognition (%)	Testing recognition (%)
Proposed RBF NN	100	3000	79.00	64.50
MLNN [10]	100	3000	62.80	56.00

References

[1] Y. Y. Nguwi and A. Z. Kouzani, "Detection and classification of road signs in natural environments," *Neural Computing and Applications*, vol. 17, no. 3, pp. 265–289, 2008.

[2] Y. Shao, Q. Chen, and H. Jiang, "RBF neural network based on particle swarm optimization," in *Proceedings of the 7th International Conference on Advances in Neural Networks (ISNN '10)*, L. Zhang et al., Ed., vol. 6063, pp. 169–176, Springer, Shanghai, China, 2010.

[3] G. A. P. Coronado, M. R. Muñoz, J. M. Armingol et al., "Road sign recognition for automatic inventory systems," in *Proceedings of the 18th International Conference on Systems, Signals, and Image Processing (IWSSIP '11)*, pp. 63–66, 2011.

[4] K. H. Lim, K. P. Seng, and L.-M. Ang, "Improved traffic sign recognition," in *Proceedings of the International Conference on Embedded Systems and Intelligent Technology (ICESIT '10)*, Chiang Mai, Thailand, 2010.

[5] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[6] S. Lee and R. M. Kil, "A gaussian potential function network with hierarchically self-organizing learning," *Neural Networks*, vol. 4, no. 2, pp. 207–224, 1991.

[7] M. S. Mueller, "Least-squares algorithms for adaptive equalizers," *The Bell System Technical Journal*, vol. 60, no. 8, pp. 1905–1925, 1981.

[8] Z. H. Man et al., "Design of robust adaptive filters using Lyapunov stability theory," in *Proceedings of the IEEE Transactions on Circuits & Systems II: Express Briefs*, 2004.

[9] K. P. Seng, Z. Man, and H. R. Wu, "Lyapunov-theory-based radial basis function networks for adaptive filtering," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 8, pp. 1215–1220, 2002.

[10] K. H. Lim, K. P. Seng, L. M. Ang, and S. W. Chin, "Lyapunov theory-based multilayered neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 4, pp. 305–309, 2009.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1994.

[12] M. J. Er, S. Wu, J. Lu, and H. L. Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 697–710, 2002.

[13] M. J. Er, W. Chen, and S. Wu, "High-speed face recognition based on discrete cosine transform and rbf neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 679–691, 2005.

[14] J. A. Leonard and M. A. Kramer, "Radial basis function networks for classifying process faults," *IEEE Control Systems Magazine*, vol. 11, no. 3, pp. 31–38, 1991.

[15] M. John and J. D. Christian, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[16] F. Yang and M. Paindavoine, "Implementation of an rbf neural network on embedded systems: real-time face tracking and identity verification," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1162–1175, 2003.

[17] M. Kishan, *Elements of Artificial Neural Networks*, MIT Press, 1997.

[18] V. Espinosa-Duro, "Biometric identification system using a radial basis network," in *Proceedings of the 34th IEEE Annual International Carnahan Conference on Security Technology*, pp. 47–51, 2000.

[19] M. Birgmeier, "Fully kalman-trained radial basis function network for nonlinear speech modeling," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 259–264, December 1995.