Neurocomputing 86 (2012) 116–123

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction

Rohitash Chandra *, Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand

ARTICLE INFO

Article history:
Received 25 July 2011
Received in revised form
28 October 2011
Accepted 15 January 2012
Communicated by W.S. Hong
Available online 24 February 2012

Keywords:
Cooperative coevolution
Neuro-evolution
Recurrent neural networks
Chaotic time series prediction
Evolutionary algorithms

ABSTRACT

Cooperative coevolution decomposes a problem into subcomponents and employs evolutionary algorithms for solving them. Cooperative coevolution has been effective for evolving neural networks. Different problem decomposition methods in cooperative coevolution determine how a neural network is decomposed and encoded which affects its performance. A good problem decomposition method should provide enough diversity and also group interacting variables which are the synapses in the neural network. Neural networks have shown promising results in chaotic time series prediction. This work employs two problem decomposition methods for training Elman recurrent neural networks on chaotic time series problems. The Mackey-Glass, Lorenz and Sunspot time series are used to demonstrate the performance of the cooperative neuro-evolutionary methods. The results show improvement in performance in terms of accuracy when compared to some of the methods from literature.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Time series prediction involves the study of the present and past behaviour of the system for the prediction of the future. Chaos theory is used to study the behaviour of dynamical systems that are highly sensitive to initial conditions such as noise and error [1,2]. Sensitivity to the initial conditions is known as the *butterfly effect* which makes long-term prediction difficult. The prediction of chaotic time series has a wide range of applications such as in finance [3], signal processing [4], power load [5], weather forecast [6], hydrological prediction [7] and Sunspot prediction [8–10].

Cooperative coevolution (CC) divides a problem into subcomponents that are represented using sub-populations [11]. An important feature of cooperative coevolution is that it provides better diversity using several subcomponents than conventional evolutionary algorithms [12]. Problem decomposition in cooperative coevolution determines how a neural network is broken down and encoded as subcomponents. Cooperative coevolution has shown promising results in training neural networks [12–15].

Problem decomposition is a major issue in cooperative coevolution of neural networks. The problem decomposition method should provide enough diversity and also group interacting variables which are the synapses in the neural network. There are two major approaches to neuro-evolution using cooperative coevolution which decomposes the network on the *neural level* and the *synapse level*. In synapse level problem decomposition, the neural network is decomposed to its lowest level where each weight connection (synapse) forms a subcomponent. Examples include cooperatively coevolved synapse neuro-evolution [16] and neural fuzzy networks with cultural cooperative particle swarm optimisation [17]. In neural level problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced sub-populations [18,19] and neuron-based sub-population [15].

Neuron-based sub-population [15] further breaks down the encoding scheme of enforced sub-populations [18,19]. Neuron-based sub-population performed better than enforced sub-populations and synapse level encoding for pattern recognition problems using feedforward networks. In evolving recurrent neural networks, neuron-based sub-population showed better performance than synapse level problem decomposition for grammatical inference problems [15].

This work employs synapse and neuron level problem decomposition for training recurrent neural network for chaotic time series prediction. The synapse level employs the problem decomposition used in cooperatively coevolved synapses neuro-evolution [16], while the neuron level employs neuron-based sub-population [15]. The results are further compared with a standard evolutionary algorithm (EA) where a single population is used. These methods are used to train the Elman recurrent neural

* Corresponding author.
E-mail address: c.rohitash@gmail.com (R. Chandra).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2012.01.014

network [20] on three different problems which consists of two simulated and one real-world chaotic time series. The Lorenz and Mackey-Glass are the simulated time series, while the Sunspot is a real-world time series. The performance in terms of accuracy of the respective methods are evaluated on different neural network topologies which are given by different numbers of hidden neurons. The results are further compared with a number of computational intelligence methods from the literature.

The contribution of the paper is in the application of an existing problem decomposition method called neuron-based sub-population [15] for training recurrent neural networks on chaotic time series problems. Moreover, synapse level problem decomposition [16] is also used for comparison. This will help to determine which problem decomposition method is more suitable for time series problems.

The rest of the paper is organised as follows. A background on recurrent neural networks and cooperative coevolution framework is presented in Section 2. Section 3 presents the different problem decomposition methods for training recurrent neural networks. Section 4 presents the results and discussion and Section 5 concludes the paper with a discussion on future work.

2. Background

2.1. Recurrent neural networks

Recurrent neural networks are dynamical systems whose next state and output depends on the present network state and input. The Elman recurrent network [20] employs the context layer which makes a copy of the hidden layer outputs in the previous time steps. They are composed of an *input layer*, a *context layer* which provides state information, a *hidden layer* and an *output layer*. Each layer contains one or more neurons which propagate information from one layer to another by computing a non-linear function of their weighted sum of inputs. The dynamics of the change in hidden state neuron activations in the context layer is given as follows:

$$S_i(t) = g\left(\sum_{k=1}^{K} V_{ik}S_k(t-1) + \sum_{j=1}^{J} W_{ij}I_j(t-1)\right) \quad (1)$$

where $S_k(t)$ and $I_j(t)$ represent the output of the context state and input neurons, respectively, and V_{ik} and W_{ij} represent their corresponding weights. $g(\cdot)$ is a sigmoid squashing function.

2.2. Embedding theorem and time series prediction using recurrent neural networks

Embedding is the process of finding a space in which the dynamics are smooth and no overlaps or intersections occur in the orbits of the attractor. Takens' embedding theorem provides the conditions under which a chaotic time series can be reconstructed into a D-dimensional vector with two conditions which are the *time delay*and the *embedding dimension* [21].

Given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t-T), \ldots, x(t(D-1)T)]$ can be generated, where T is the time delay, D is the embedding dimension, $t = 0, 1, 2, \ldots, N-DT-1$, and N is the length of the original time series [21].

Takens' theorem expresses that the vector series reproduces many important characteristics of the original time series. The right values for D and T must be chosen in order to efficiently apply Takens' theorem [22]. Takens' proved that if the original attractor is of dimension d, then $D = 2d+1$ will be sufficient to reconstruct the attractor [21].

Several methods have been proposed in the past to determine the values for the embedding dimensions as discussed in [23]. Evolutionary algorithms have also been used to determine the optimal values of the embedding dimensions [24,25].

The reconstructed vector is used to train the recurrent network for one-step-ahead prediction where one neuron is used in the input and the output layers. The recurrent network unfolds k steps in time which is equal to the embedding dimension D [8,26].

The root mean squared error (RMSE) and normalised mean squared error (NMSE) are used to measure the prediction performance of the recurrent neural network. These are given in the following equations:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad (2)$$

$$NMSE = \left(\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}\right) \quad (3)$$

where y_i, $\hat{y}_i$ and $\bar{y}_i$ are the observed data, predicted data and average of observed data, respectively. N is the length of the observed data.

2.3. Problem decomposition in cooperative coevolution

Cooperative coevolution is a nature inspired optimisation method where the problem is decomposed as subcomponents [11]. The subcomponents are implemented as sub-populations that are evolved in isolation and cooperation takes place for fitness evaluation. The number of sub-populations enforce cooperative coevolution to provide better diversity than conventional evolutionary algorithms [12]. Problem decomposition determines how the problem is broken down into subcomponents. The original CC framework has been used for function optimisation and the problem was decomposed into the lowest level where a separate subcomponent was used to represent each dimension of the problem [11]. It was later found that this strategy is mostly effective for problems which are fully separable [27]. Much work has been done in the use of cooperative coevolution in large scale function optimisation and the focus has been on non-separable problems [27–30].

A function of n variables is separable if it can be written as a sum of n functions with just one variable [31,32]. Non-separable problems have interdependencies between variables as opposed to separable ones. Real-world problems mostly fall between fully separable and fully non-separable. Cooperative coevolution has been effective for separable problems [11] as it has the property to decompose the problem into sub-problems and solve them in isolation. Conversely, evolutionary algorithms without any decomposition strategy appeal to fully non-separable problems. Note that the goal is to solve non-separable problems without any prior knowledge about the interacting variables in the problem. Therefore, the evolutionary algorithm has to identify the interdependencies by itself.

The sub-populations in cooperative coevolution are evolved in a *round-robin* fashion for a given number of generations known as the *depth of search*. The depth of search has to be predetermined according to the nature of the problem. The depth of search can reflect whether the encoding schemes have been able to group the interacting variables into separate subcomponents [33,32,15]. If the interacting variables have been grouped efficiently, then a deep greedy search for the sub-population is possible. This implies that the problem has been efficiently broken down into subcomponents which have few interactions among themselves [33].

3. Cooperative coevolution for evolving recurrent neural networks

Problem decomposition determines the size of a subcomponent and the way it is encoded. Problem decomposition is also known as the encoding scheme for training neural networks [15]. This section gives details of the neuron and synapse level encoding schemes for training recurrent neural networks.

In this paper, the neuron level encoding employs neuron-based sub-population (NSP) [15]. In NSP, each neuron in the hidden layer and the output layer is a reference point for a sub-population. Each hidden neuron also acts as a reference point for the recurrent (state or context) synapses connected to it. NSP can be easily extended to a neural network with more than a single hidden layer. Fig. 1 shows a detailed diagram of the NSP encoding scheme for recurrent neural networks. Each subcomponent for the RNN with a single hidden layer is composed of the following:

1. For a given neuron i in the *hidden*(t) layer, the *hidden layer subcomponents* consist of all synapses connected from *input*(t) layer to neuron i. The bias of i is also included.
2. For a given neuron j in the *hidden*(t) layer, the *state* (*recurrent*) *layer subcomponents* consist of all synapses connected from the *hidden*($t-1$) layer to neuron j.
3. For a given neuron k in the *output*(t) layer, the *output layer subcomponents* consist of all synapses connected from the *hidden*(t) layer to neuron k. The bias of k is also included.

where t is time and *hidden*($t-1$) is the layer representing the state or recurrent neurons.

The synapse level encoding scheme decomposes the network into its lowest level, where each synapse in the network forms a separate subcomponent. Synapse level encoding has been employed in cooperative coevolved synapse (CoSyNE) [16] and neural fuzzy systems trained by particle swarm optimisation [17].

The general NSP CC framework for training RNN is given in Algorithm 1.

Algorithm 1. Cooperative coevolution for training RNNs.

Step 1: Decompose the problem into k subcomponents according to the chosen encoding scheme (Neuron or Synapse level)
Step 2: Initialize and cooperatively evaluate each sub-population
for each *cycle* until termination **do**
  for each Subpopulation **do**
    for n Generations **do**
      (i) Select and create new offspring
      (ii) Cooperatively evaluate the new offspring
      (iii) Update sub-population
    end for
  end for
end for

In Algorithm 1, the recurrent neural network is decomposed according to the neuron or synapse level encoding into k subcomponents. In neuron level encoding, k is equal to the number of hidden neurons, plus the number of context neurons, plus the number of output neurons as shown in Fig. 1. In synapse level encoding, k is equal to the total number of synapses and biases. A *cycle* is completed when all the sub-populations are evolved for a fixed number of generations in a round-robin fashion. The algorithm halts if the termination condition is satisfied.

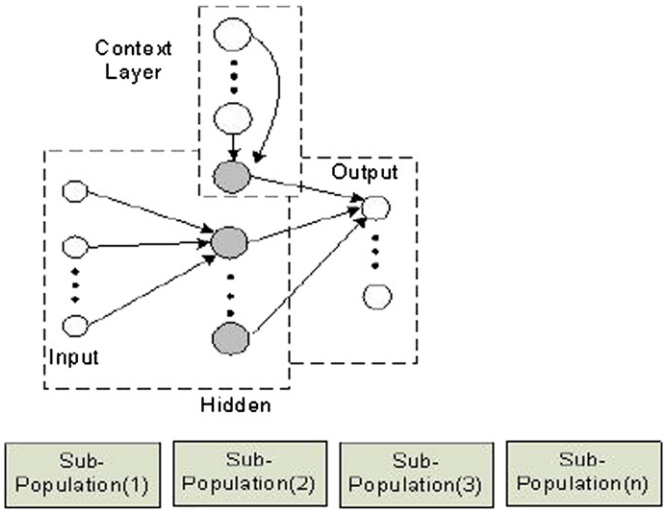

Fig. 1. The NSP encoding scheme which can be easily extended to a neural network with more than one hidden layer [15].

The termination criteria is specified by a minimum RMSE or when the maximum number of function evaluations has been reached.

There are two main phases of evolution in the cooperative coevolution framework. The first is the *initialization phase* and the second is the *evolution phase* [15]. In the initialisation phase, the individuals in the sub-populations do not have a fitness. In order to evaluate the i th individual of the k th subcomponent, arbitrary individuals from the rest of the sub-populations are selected and concatenated with the chosen individual and cooperatively evaluated. In the evolution phase, cooperative evaluation is done by concatenating the chosen individual from a sub-population with the best individuals from the rest of the sub-populations.

4. Experimentation, results and analysis

This section presents an experimental study of the two different problem decomposition methods in cooperative coevolution of recurrent neural networks. The neuron level (NL) and synapse level (SL) problem decomposition methods are used for training Elman recurrent networks [20] on chaotic time series problems from the literature. The results are also compared to an evolutionary algorithm (EA). Two simulated and one real-world chaotic time series problems are used. The Mackey-Glass times series [34] and Lorenz time series [1] are the two simulated time series, while the real-world problems is the Sunspot time series [35]. The behaviours of the respective methods are evaluated on different recurrent network topologies given by different numbers of hidden neurons which reflect the scalability and the robustness of the method. The results are further compared with several computational intelligence methods from the literature. The details of the three different problems are given as follows.

4.1. Dataset and problem description

The *Mackey-Glass time series* has been used in the literature as a benchmark problem due to its chaotic nature [34]. The differential equation used to generate the Mackey-Glass time series is given below:

$$\frac{\delta x}{\delta t} = \frac{ax(t-\tau)}{[1+x^c(t-\tau)]} - bx(t) \tag{4}$$

In Eq. (4), the delay parameter, τ, determines the characteristic of the time series. $\tau > 16.8$ produces chaos. The selected parameter

values for generating the time series are from the literature [36,37,10,38] where the constants are $a=0.2$, $b=0.1$ and $c=10$. The chaotic time series is generated by using time delay $\tau=17$ and initial value $x(0)=1.2$.

The experiments use the chaotic time series with length of 1000 generated by Eq. (4). The first 500 samples are used for training the Elman network while rest of the 500 samples are used for testing the respecting training algorithms. The time series is scaled in the range [0,1]. The phase space of the original time series is reconstructed with the embedding dimensions $D=3$ and $T=2$.

The *Lorenz time series* has been introduced by Edward Lorenz who has extensively contributed to the establishment of the Chaos theory [1]. The Lorenz equation is given below:

$$\frac{dx(t)}{dt}=\sigma[y(t)-x(t)]$$

$$\frac{dy(t)}{dt}=x(t)[r-z(t)]-y(t)$$

$$\frac{dz(t)}{dt}=x(t)y(t)-bz(t) \quad (5)$$

where η, r, and b are the dimensionless parameters. The typical values of these parameters are used where $\eta=10$, $r=28$, and $b=8/3$. The x-coordinate of the Lorenz time series is chosen for prediction and 1000 samples are generated. The time series is scaled in the range $[-1,1]$. The first 500 samples are used for training and the remaining 500 is used for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D=3$ and $T=2$.

The *Sunspot time series* is a good indication of the solar activities for solar cycles which impact Earth's climate, weather patterns and satellite and spaces missions [9]. The prediction of solar cycles is difficult due to its complexity. The monthly smoothed Sunspot time series has been obtained from the World Data Center for the Sunspot Index [35]. The Sunspot time series from November 1834 to June 2001 is selected which consists of 2000 points. This interval has been selected in order to compare the performance the proposed methods with those from the literature [10,38]. The time series is scaled in the range $[-1,1]$. The first 1000 samples are used for training, while the remaining 1000 samples are used for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D=5$ and $T=2$.

4.2. Experimental set-up

The Elman recurrent network employs sigmoid units in the hidden layer of the three different problems. In the output layer, the sigmoid unit is used for the Mackey-Glass time series while the hyperbolic tangent unit is used for the Lorenz and the Sunspot time series. The RMSE and NMSE given in Eqs. (2) and (3) are used as the main performance measures in order for comparison with the methods from literature.

The generalised generation gap model with parent centric crossover operator (G3-PCX) evolutionary algorithm [39] is employed in NL, SL and EA. The G3-PCX algorithm employs the mating pool size of two offspring and two parents with the generation gap model for selection. This set-up has been used for optimisation problems [39] and for training cooperative coevolutionary recurrent neural networks [40,15]. The sub-populations are seeded with random real numbers in the range of $[-5, 5]$ in all the experiments. Fifty individuals make up the respective sub-populations.

In the respective CC framework for recurrent networks (SL and NL) shown in Algorithm 1, each sub-population is evolved for a fixed number of generations in a round-robin fashion. This is considered as the *depth of search*. Our previous work has shown that the depth of search of one generation gives optimal performance for both NL and SL encodings [15]. Hence, 1 is used as the depth of search in all the experiments. Note that all sub-populations evolve for the same depth of search.

The termination condition of the different problems is when a total number of 100 000 function evaluations has been reached by the respective algorithms. The function evaluations in the initialisation stage are not included in the maximum training time.

4.3. Results and discussion

This section reports the performance of the three evolutionary methods (NL, SL and EA) for training the Elman recurrent network on the chaotic time series problems. Note that the best performance is given by the least RMSE and NMSE.

Initially, the number of hidden neurons is empirically evaluated and the mean and the best value of the RMSE is given from 30 independent experimental runs. The results of the test set are shown in Tables 1–3. The best results are highlighted in bold.

The best results from Tables 1–3 are chosen and further details are given in Table 4 where the mean and 95% confidence interval (CI) of RMSE and NMSE is given with the best performance out of 30 experimental runs. The best mean prediction performance on the test dataset is highlighted in Table 4 and shown in Figs. 2–4.

We compare the mean values of the RMSE in the results given in Table 4. In the Lorenz time series, NL gives the best performance when compared to SL and EA. The performance of SL is close to NL, however, the EA does not perform well. Note that each method has its strength in terms of the number of hidden neurons. SL gives good performance with the least number of hidden neurons (5), while NL requires the most number of hidden neurons (11 and 13). This is due to the difference in their encoding schemes.

In the results for the Mackey time series, SL gives the best performance. NL and SL use the same number of hidden neurons (13), however, the performance of SL is better. In the Sunspot

Table 1
The prediction performance (RMSE) of NL, SL, and EA on the test dataset of the Lorenz time series.

Hidden	SL		NL		EA	
	Mean	Best	Mean	Best	Mean	Best
3	3.0E−2	1.2E−2	3.6E−2	1.8E−2	5.1E−2	2.1E−2
5	**1.9E−2**	6.4E−3	2.7E−2	9.1E−3	2.5E−2	8.6E−3
7	2.2E−2	7.0E−3	2.3E−2	9.7E−3	**2.1E−2**	9.4E−3
9	3.8E−1	9.8E−3	2.4E−2	5.0E−3	**2.1E−2**	8.7E−3
11	8.2E−1	1.2E−2	**1.8E−2**	8.2E−3	2.9E−2	1.1E−2
13	1.1	2.2E−2	**1.8E−2**	6.6E−3	3.2E−1	2.2E−1

Table 2
The performance (RMSE) of NL, SL, and EA on the test dataset of the Mackey-Glass time series.

Hidden	SL		NL		EA	
	Mean	Best	Mean	Best	Mean	Best
3	2.1E−2	1.2E−2	2.3E−2	9.7E−3	3.0E−2	1.3E−2
5	1.5E−2	7.9E−3	2.0E−2	1.0E−2	1.5E−2	9.5E−3
7	1.4E−2	9.0E−3	1.6E−2	1.0E−2	1.4E−2	8.9E−3
9	1.2E−2	7.6E−3	1.4E−2	8.3E−3	**1.2E−2**	7.2E−2
11	1.0E−2	5.7E−3	1.3E−2	8.0E−3	2.1E−2	8.8E−3
13	**9.4E−3**	6.3E−3	**1.2E−2**	8.3E−3	9.6E−2	6.9E−2

time series, NL gives the best performance with three hidden neurons. Note that the Sunspot times series is a real-world problem which contains noise. NL has been able to better decompose the recurrent network training problem with the presence of noise.

SL gives the best performance if we consider the best values of the RMSE in the results given in Table 4.

Synapse level encoding has more diversity as more subcomponents are used; however, it does not group interacting variables. Neural level encoding views the RNN as a partially separable problem. Synapse level encoding views the RNN as a fully separable problem. The results show that the problem decomposition method and the nature of the time series problem affects the performance of cooperative coevolution.

Figs. 2–4 show that the RNN has been able to give good prediction performance. The RNN has been able to cope with the noise in the Sunspot time series given in Fig. 4. The prediction error has also been shown.

The performances of SL, NL and EA on the different problems are further compared to some of the results published in the literature as shown in Tables 5–7. The best values from the results in Table 4 are used to compare with the results from literature. In Table 5, the results of the methods given in this paper are better than some of the existing methods. However, the best results are of the evolutionary recurrent neural network (ERNN) [24] and the Hybrid NARX-Elman network [38]. These are also seen for the Mackey-Glass time series given in Table 6. This is because the ERNN has also optimised the values for the embedding dimensions for the phase space reconstruction. Hybrid NARX-Elman network has the advantage of the architectural properties of the two neural networks with residual analysis which further improves the results. These methods can be combined with the proposed methods in future work to improve the results.

In Table 6, the proposed methods have given better performance than similar evolutionary approaches such as training neural fuzzy networks with hybrid of cultural algorithms and cooperative particle swarm optimisation (CCPSO), cooperative

Table 3
The performance (RMSE) of NL, SL, and EA on the test dataset of the Sunspot time series.

	SL		NL		EA	
Hidden	Mean	Best	Mean	Best	Mean	Best
3	**6.9E−2**	1.7E−2	**5.6E−2**	2.6E−2	9.2E−2	4.6E−2
5	1.0E−1	2.4E−2	7.5E−2	2.2E−2	7.2E−2	2.1E−2
7	1.0E−1	1.9E−2	5.9E−2	2.2E−2	**5.7E−2**	1.8E−2
9	1.3E−1	2.17E−2	8.9E−2	1.7E−2	7.7E−2	1.7E−2

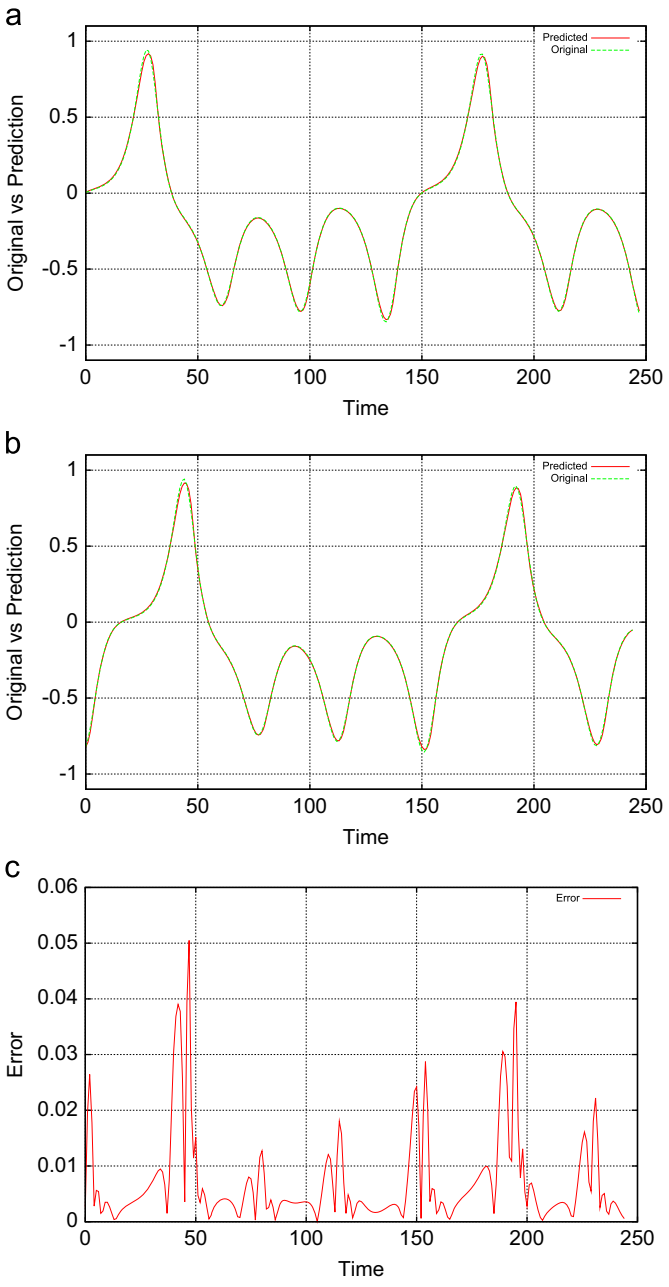

Fig. 2. Typical prediction given by NL for Lorenz time series. (a) Performance on the training dataset. (b) Performance on the test dataset. (c) Error on the test dataset.

Table 4
The performance (RMSE and NMSE) of NL, SL, and EA on the test dataset of the three problems. The mean and 95% confidence interval (CI) is given with the best performance out of 30 independent experimental runs.

			RMSE		NMSE	
Problem	Method	Hidden	Mean and CI	Best	Mean and CI	Best
Lorenz	SL	5	1.95E−2 ± 2.59E−3	6.36E−3	8.28E−3 ± 1.98E−3	7.72E−4
	NL	11	**1.82E−2 ± 2.82E−3**	8.20E−3	7.48E−3 ± 2.60E−3	1.28E−3
	EA	7	2.06E−2 ± 2.79E−3	9.43E−3	9.24E−3 ± 2.83E−3	1.69E−3
Mackey	SL	13	**9.39E−3 ± 5.57E−4**	6.33E−3	6.31E−4 ± 7.60E−5	2.79E−4
	NL	13	1.23E−2 ± 9.16E−4	8.28E−3	1.11−3 ± 1.77E−4	4.77E−4
	EA	9	1.19E−2 ± 9.47E−4	7.24E−3	1.04E−3 ± 1.69E−4	3.65E−4
Sunspot	SL	3	6.88E−2 ± 2.66E−2	1.66E−2	5.48E−2 ± 5.19E−2	1.47E−3
	NL	3	**5.58E−2 ± 8.01E−3**	2.60E−2	1.92E−2 ± 5.3eE−3	3.62E−3
	EA	7	5.74E−2 ± 1.14E−2	1.82E−2	2.28E−2 ± 1.07E−2	1.76E−3

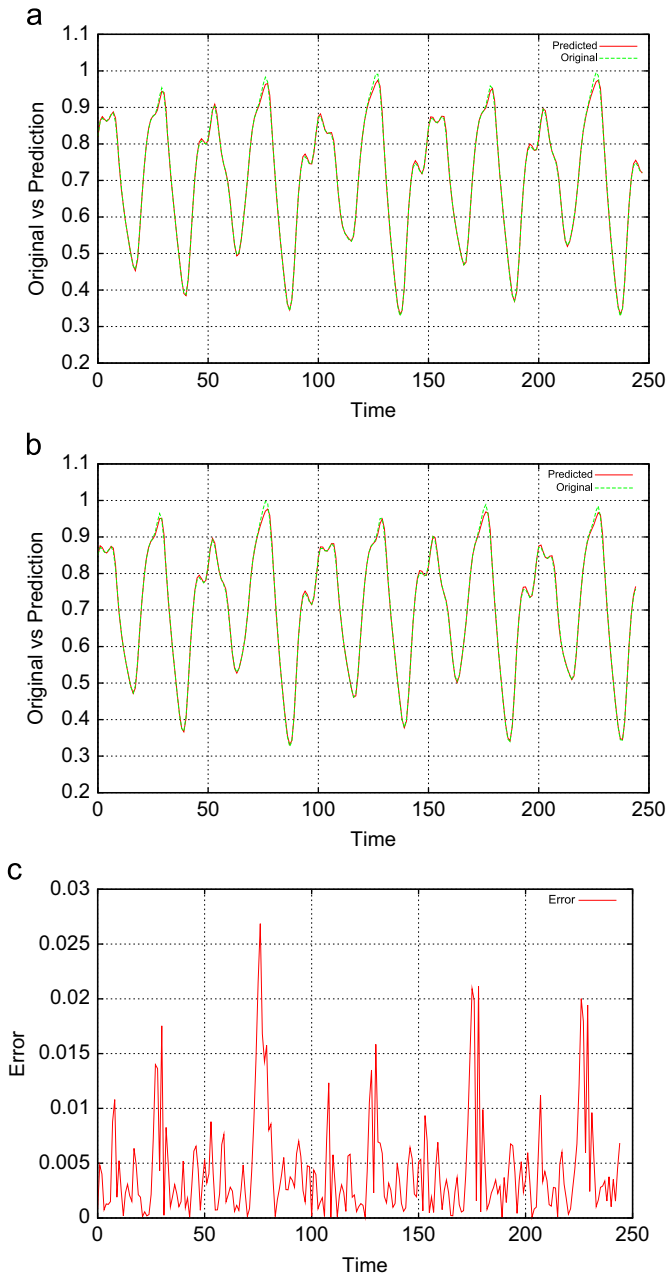

Fig. 3. Typical prediction given by SL for Mackey-Glass time series. (a) Performance on the training dataset. (b) Performance on the test dataset. (c) Error on the test dataset.

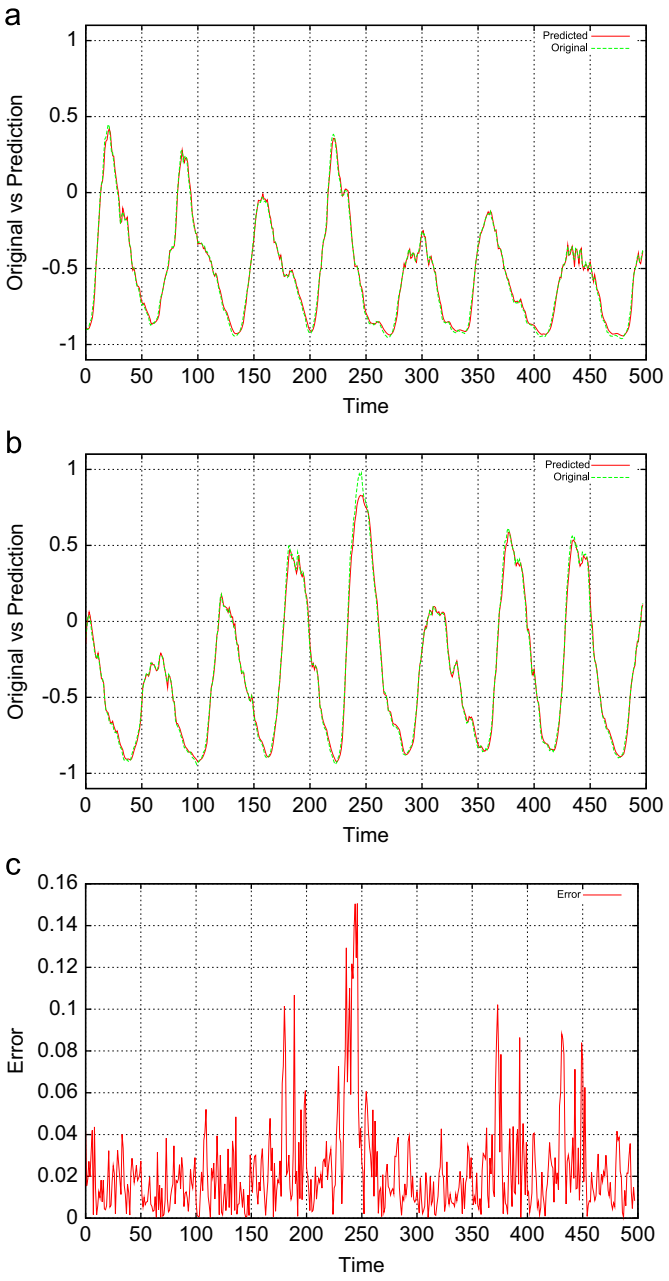

Fig. 4. Typical prediction given by NL for Sunspot time series. (a) Performance on the training dataset. (b) Performance on the test dataset. (c) Error on the test dataset.

particle swarm optimisation (CPSO), genetic algorithms and differential evolution (DE) [17]. In Table 7, the proposed methods have given better performance than most of the methods from literature with the only exception being the Hybrid NARX-Elman networks [38].

5. Conclusions and future work

The paper has presented an application of two different problem decomposition methods for cooperative coevolution of recurrent neural networks on chaotic time series problems. The two different problem decomposition methods in cooperative coevolution have performed better than a standard evolutionary algorithm. The results show that the given methods have been able to predict chaotic time series with good level of accuracy in comparison to some of the methods from literature. The coevolutionary methods have performed very well on the real-world Sunspot time series which contain noise.

The limitation of using evolutionary algorithms for training neural networks in comparison with gradient based methods is the time they take in convergence. However, evolutionary algorithms are known as global optimisation methods and neuroevolution is promising according to the accuracy of the results.

Comparison of the results from literature motivates further research in using evolutionary computation methods for time series prediction. Residual analysis can further improve the results. Gradient-based local search can further complement the evolutionary training algorithms. Convergence analysis of the cooperative coevolution methods can also be done in the future.

Table 5
A comparison with the results from literature on the Lorenz time series.

Prediction method	RMSE	NMSE
Pseudo Gaussian—radial basis neural network (2002)	9.40E−02	
Radial basis network with orthogonal least squares (RBF-OLS) (2006) [10]		1.41E−09
Locally linear neuro-fuzzy model—locally linear model tree (LLNF-LoLiMot) (2006) [10]		9.80E−10
Boosted recurrent neural networks (2006) [41]		3.77E−03
Evolutionary RNN (2007) [24]	8.79E−06	9.90E−10
Auto regressive moving average with neural network (ARMA-ANN) (2008) [42]	8.76E−02	
Backpropagation-through-time (BPTT-RNN) (2010) [26]		1.85E−03
Real time recurrent learning (RTRL-RNN) (2010) [26]		1.72E−03
Recursive Bayesian LevenbergMarquardt (RBLM-RNN) (2010) [26]		9.0E−04
Hybrid NARX-Elman RNN with residual analysis (2010) [38]	1.08E−04	1.98E−10
Backpropagation neural network and genetic algorithms with residual analysis (2011) [43]	2.96E−02	
Proposed SL-CCRNN	6.36E−03	7.72E−04
Proposed NL-CCRNN	8.20E−03	1.28E−03

Table 6
A comparison with the results from literature on the Mackey time series.

Prediction method	RMSE	NMSE
Autoregressive model	1.9E−01	
Backpropagation neural network	2.0E−02	
Cascade correlation network	6.0E−02	
Adaptive neuro fuzzy inference system (ANFIS) (1993) [36]	1.5E−03	
Genetic fuzzy predictor ensemble (1997) [44]	3.8E−02	
Pseudo Gaussian—radial basis neural network (2002) [37]	2.8E−03	
Auto regressive moving average with neural network (ARMA-ANN) (2008) [42]	2.5E−03	
Radial basis network with orthogonal least squares (RBF-OLS) (2006) [10]	1.02E−03	
Locally linear neuro-fuzzy model—locally linear model tree (LLNF-LoLiMot) (2006) [10]	9.61E−04	
Boosted recurrent neural networks (2006) [41]		1.60E−04
Evolutionary RNN (2007) [24]		3.15E−08
Neural fuzzy network and hybrid of cultural algorithm and cooperative particle swarm optimisation (CCPSO) (2009) [17]	8.42E−03	
Neural fuzzy network and particle swarm optimisation (PS0) (2009) [17]	2.10E−02	
Neural fuzzy network and cooperative particle swarm optimisation (CPS0) (2009) [17]	1.76E−02	
Neural fuzzy network and differential evolution (DE) (2009) [17]	1.62E−02	
Neural fuzzy network and genetic algorithm (GA) (2009)[17]	1.63E−02	
Hybrid NARX-Elman RNN with residual analysis (2010) [38]	3.72E−05	2.70E−08
Backpropagation neural network and genetic algorithms with residual analysis (2011) [43]	1.30E−03	
Proposed SL-CCRNN	6.33E−03	2.79E−04
Proposed NL-CCRNN	8.28E−03	4.77E−04

Table 7
A comparison with the results from literature on the Sunspot time series.

Prediction method	RMSE	NMSE
Sello nonlinear method [9]		3.40E−01
Waldmeier [9]		5.60E−01
McNish-Lincoln [9]		8.00E−02
Multi-layer perceptron (1996) [8]		9.79E−02
Elman RNN (1996) [8]		9.79E−02
FIR network (MLP) (1996) [8]		2.57E−01
Wavelet packet multilayer perceptron (2001) [45]		1.25E−01
Radial basis network with orthogonal least squares (RBF-OLS) (2006) [10]		4.60E−02
Locally linear neuro-fuzzy model—locally linear model tree (LLNF-LoLiMot) (2006) [10]		3.20E−02
Hybrid NARX-Elman RNN with residual analysis (2010) [38]	1.19E−02	5.90E−04
Proposed SL-CCRNN	1.66E−02	1.47E−03
Proposed NL-CCRNN	2.60E−02	3.62E−03

References

[1] E. Lorenz, Deterministic non-periodic flows, J. Atmos. Sci. 20 (1963) 267–285.
[2] H.K. Stephen, In the Wake of Chaos: Unpredictable Order in Dynamical Systems, University of Chicago Press, 1993.
[3] A. Das, P. Das, Chaotic analysis of the foreign exchange rates, Appl. Math. Comput. 185 (1) (2007) 388–396.
[4] M.B. Kennel, S. Isabelle, Method to distinguish possible chaos from colored noise and to determine embedding parameters, Phys. Rev. A 46 (6) (1992) 3111–3118.
[5] S. Kawauchi, H. Sugihara, H. Sasaki, Development of very-short-term load forecasting based on chaos theory, Electr. Eng. Jpn. 148 (2), 55–63, doi:10.1002/eej.10322.
[6] E. Lorenz, The Essence of Chaos, University of Washington Press, 1993.

[7] X.-H. Yang, Y. Mei, D.-X. She, J.-Q. Li, Chaotic bayesian optimal prediction method and its application in hydrological time series, Comput. Math. Appl. 61 (8) (2011) 1975–1978.
[8] T. Koskela, M. Lehtokangas, J. Saarinen, K. Kaski, Time series prediction with multilayer perceptron, Proceedings of the World Congress on Neural Networks, 1996, pp. 491–496.
[9] S. Sello, Solar cycle forecasting: a nonlinear dynamics approach, Astron. Astrophys. 377 (2001) 312–320.
[10] A. Gholipour, B.N. Araabi, C. Lucas, Predicting chaotic time series using neural and neurofuzzy models: a comparative study, Neural Process. Lett. 24 (2006) 217–239.
[11] M.A. Potter, K.A. De Jong, A cooperative coevolutionary approach to function optimization, in: PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, 1994, pp. 249–257.
[12] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (2000) 1–29.
[13] N. Garcia-Pedrajas, C. Hervas-Martinez, J. Munoz-Perez, COVNET: a cooperative coevolutionary model for evolving artificial neural networks, IEEE Trans. Neural Networks 14 (3) (2003) 575–596.
[14] N. Garcia-Pedrajas, C. Hervas-Martinez, D. Ortiz-Boyer, Cooperative coevolution of artificial neural network ensembles for pattern classification, IEEE Trans Evol. Comput. 9 (3) (2005) 271–302.
[15] R. Chandra, M. Frean, M. Zhang, C.W. Omlin, Encoding subcomponents in cooperative co-evolutionary recurrent neural networks, Neurocomputing 74 (2011) 3223–3234.
[16] F. Gomez, J. Schmidhuber, R. Miikkulainen, Accelerated neural evolution through cooperatively coevolved synapses, J. Mach. Learn. Res. 9 (2008) 937–965.
[17] C.-J. Lin, C.-H. Chen, C.-T. Lin, A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications, Trans. Syst. Man Cybern. Part C 39 (2009) 55–68.
[18] F. Gomez, R. Mikkulainen, Incremental evolution of complex general behavior, Adapt. Behav. 5 (3–4) (1997) 317–342. doi:10.1177/105971239700500305.
[19] F.J. Gomez, Robust Non-Linear Control Through Neuroevolution, Technical Report AI-TR-03-303, Ph.D. Thesis, Department of Computer Science, The University of Texas at Austin, 2003.
[20] J.L. Elman, Finding structure in time, Cognitive Sci. 14 (1990) 179–211.
[21] F. Takens, Detecting strange attractors in turbulence, in: Dynamical Systems and Turbulence, Warwick 1980, Lecture Notes in Mathematics, 1981, pp. 366–381.
[22] C. Frazier, K. Kockelman, Chaos theory and transportation systems: instructive example, Transp. Res. Rec.: J. Transp. Res. Board 20 (2004) 9–17.
[23] L. Cao, Practical method for determining the minimum embedding dimension of a scalar time series, Phys. D: Non-Linear Phenom. 110 (1–2) (1997) 43–50.
[24] Q.-L. Ma, Q.-L. Zheng, H. Peng, T.-W. Zhong, L.-Q. Xu, Chaotic time series prediction based on evolving recurrent neural networks, in: 2007 International Conference on Machine Learning and Cybernetics, vol. 6, 2007, pp. 3496–3500.
[25] K. Lukoseviciute, M. Ragulskis, Evolutionary algorithms for the selection of time lags for time series forecasting by fuzzy inference systems, Neurocomputing 73 (2010) 2077–2088.
[26] D. Mirikitani, N. Nikolaev, Recursive bayesian recurrent neural networks for time-series modeling, IEEE Trans. Neural Networks 21 (2) (2010) 262–274.
[27] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computationa, 2001, pp. 1101–1108, doi:10.1109/CEC.2001.934314.
[28] F. van den Bergh, A. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[29] Y.-j. Shi, H.-f. Teng, Z.-q. Li, Cooperative co-evolutionary differential evolution for function optimization, in: L. Wang, K. Chen, Y.S. Ong (Eds.), Advances in Natural Computation, Lecture Notes in Computer Science, vol. 3611, Springer, Berlin/Heidelberg, 2005, pp. 1080–1088.
[30] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999. doi:10.1016/j.ins.2008.02.017.
[31] D. Ortiz-Boyer, C. HervásMartínez, N. García-Pedrajas, Cixl2: a crossover operator for evolutionary algorithms based on population features, J. Artif. Int. Res. 24 (2005) 1–48.
[32] R. Chandra, M. Frean, M. Zhang, On the issue of separability for problem decomposition in cooperative neuro-evolution, Neurocomputing, http://dx.doi.org/10.1016/j.neucom.2012.02.005.
[33] R. Chandra, M. Frean, M. Zhang, An encoding scheme for cooperative coevolutionary feedforward neural networks, in: J. Li (Ed.), AI 2010: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 6464, Springer, Berlin/Heidelberg, 2010, pp. 253–262.
[34] M. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.
[35] SIDC, World Data Center for the Sunspot Index, Monthly Smoothed Sunspot Data, URL 〈http://sidc.oma.be/〉.
[36] J.-S. Jang, ANFIS: adaptive-network-based fuzzy inference, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[37] I. Rojas, H. Pomares, J.L. Bernier, J. Ortega, B. Pino, F.J. Pelayo, A. Prieto, Time series analysis using normalized PG-RBF network with regression weights, Neurocomputing 42 (1–4) (2002) 267–285.
[38] M. Ardalani-Farsa, S. Zolfaghari, Chaotic time series prediction with residual analysis method using hybrid Elman-Narx neural networks, Neurocomputing 73 (13-15) (2010) 2540–2553.
[39] K. Deb, A. Anand, D. Joshi, A computationally efficient evolutionary algorithm for real-parameter optimization, Evol. Comput. 10 (4) (2002) 371–395.
[40] R. Chandra, M. Frean, M. Zhang, Modularity adaptation in cooperative coevolutionary feedforward neural networks, in: International Joint Conference on Neural Networks, 2011. pp. 681–688.
[41] M. Assaad, R. Bon, H. Cardot, Predicting chaotic time series by boosted recurrent neural networks, in: I. King, J. Wang, L.-W. Chan, D. Wang (Eds.), Neural Information Processing, Lecture Notes in Computer Science, vol. 4233, Springer, Berlin/Heidelberg, 2006, pp. 831–840.
[42] I. Rojas, O. Valenzuela, F. Rojas, A. Guillen, L. Herrera, H. Pomares, L. Marquez, M. Pasadas, Soft-computing techniques and ARMA model for time series prediction, Neurocomputing 71 (4–6) (2008) 519–537.
[43] M. Ardalani-Farsa, S. Zolfaghari, Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting, Appl. Artif. Intell. 25 (2011) 45–73.
[44] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 523–535.
[45] K.K. Teo, L. Wang, Z. Lin, Wavelet packet multi-layer perceptron for chaotic time series prediction: effects of weight initialization, in: Proceedings of the International Conference on Computational Science—Part II, ICCS '01, 2001. pp. 310–317.

Rohitash Chandra is a Research Assistant in Computer Science at the School of Engineering and Computer Science, Victoria University of Wellington. He has recently completed his PhD in Computer Science at the same institution. He holds a MSc in Computer Science from the University of Fiji and BSc from the University of the South Pacific. His research interests in general encircle the methodologies and applications of Artificial Intelligence. More specifically, he is interested in Neural and Evolutionary Computation methods such as Feedforward and Recurrent Networks, Genetic Algorithms, Cooperative Coevolution and Neuro-evolution with applications in Pattern Classification, Time Series Prediction, Control, Robot Kinematics and Environmental Informatics. He is currently working on problems in Computational Biology. Apart from his interest in science, he is actively involved in literature and has been the editor of the Blue Fog Journal. He is a poet and his third poetry collection is titled "Being at Home", which is due to be launched in 2012. He is also the founder of the Software Foundation of Fiji.

Mengjie Zhang is Professor of Computer Science at the School of Engineering and Computer Science, Victoria University of Wellington. His research interests are in Genetic Programming, Swarm Intelligence, Data Mining and Machine Learning.