Reverse Neuron Level Decomposition for Cooperative Neuro-Evolution of Feedforward Networks for Time Series Prediction

Ravneil Nand and Rohitash Chandra

School of Computing Information and Mathematical Sciences
University of South Pacific, Suva, Fiji.
ravneiln@yahoo.com, c.rohitash@gmail.com

Abstract. A major challenge in cooperative neuro-evolution is to find an efficient problem decomposition that takes into account architectural properties of the neural network and the training problem. In the past, neuron and synapse Level decomposition methods have shown promising results for time series problems, howsoever, the search for the optimal method remains. In this paper, a problem decomposition method, that is based on neuron level decomposition is proposed that features a reverse encoding scheme. It is used for training feedforward networks for time series prediction. The results show that the proposed method has improved performance when compared to related problem decomposition methods and shows competitive results when compared to related methods in the literature.

Key words: Cooperative coevolution, feedforward networks, problem decomposition, time series prediction

1 Introduction

A time series dataset is a chronological series of the past and present data is involved that are measured regularly at progressive intervals [1,2]. Time series prediction uses past data to predict future occurrence of events using robust methods such as neural networks [3].

Cooperative neuro-evolution employs cooperative coevolution for training neural networks [4,5,3]. Cooperative coevolution solves a large problem by breaking it down into sub-components and implements them as sub-population using evolutionary algorithms [4]. The sub-populations are evolved in a round-robin fashion while cooperation takes place for fitness evaluation.

Cooperative neuro-evolution has shown to be effective for training feedforward [6,7] and recurrent neural networks [8,3] with applications in classification [6,9], control [8] and time series prediction [3]. In cooperative neuro-evolution, problem decomposition depends on the structural properties of the network that have inter-dependencies [9]. The efficiency of a problem decomposition method depends on the training problem and the neural network architecture [9].

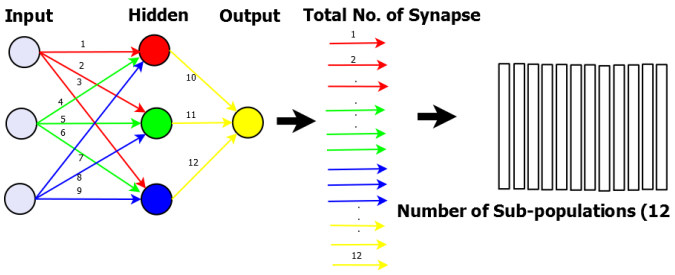


Fig. 1: Synapse level decomposition of the neural network training problem. Note that the subcomponents are implemented as sub-populations.

The two established problem decomposition methods for cooperative neuro-evolution are synapse level (SL) [5,3] and neuron level (NL) [8,10,9]. In SL, the network is decomposed to its lowest level of granularity where the number of subcomponents depends on the number of weight-links or synapses in the neural network. In NL, the number of subcomponents consists of the total number of hidden and output neurons.

Synapse level ensures global search and provides the most diversity. However, this level of decomposition works best if there are less interacting variables or synapses in the neural network training problem. Synapse level decomposition has shown good performance for time series and control problems [3,5], howsoever, it performed poorly for pattern classification problems [9]. It seems that time series prediction does not feature a high level of interactions when compared to pattern classification problems.

Neuron level decomposition has less diversity and enables grouping the interacting variables. In cooperative neuro-evolution, the problem is to balance diversity and interacting variables. A study of grouping interacting variables motivated the grouping of synapses with reference to hidden in output neurons [9].

In this paper, we introduce *reverse neuron level* (RNL) decomposition, which essentially features reverse encoding of neuron level decomposition. Neuron level decomposition takes hidden and output layer neurons as reference for each subcomponent, whereas RNL uses hidden and input layer. RNL is used for training feedforward networks for chaotic time series problems.

The rest of the paper is organized as follows. In Section 2, the proposed method is discussed in detail while in Section 3, experiments and results are given. Section 4 concludes the paper with a discussion of future work.

2 Proposed Problem Decomposition

An ideal problem decomposition method efficiently decomposes the network in a way where the interacting variables or synapses are grouped into separate subcomponents [9]. In this way, a deep greedy search for the subcomponents

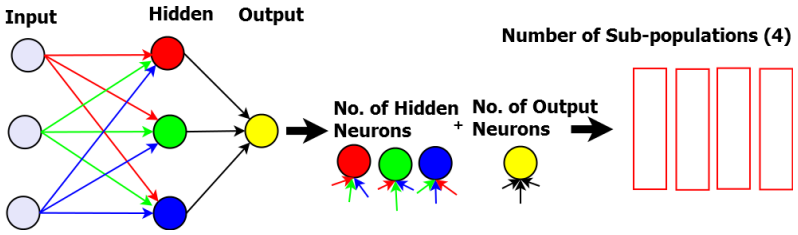


Fig. 2: Neuron level decomposition of neural network training problem. Note that the subcomponents are implemented as sub-populations.

will not be problematic for partially separable problems such as training neural networks. Analysis of the degree of separability of the neural network training problem showed that neuron level is an efficient way for decomposition [9]. Howsoever, other strategies for effectively decomposing the network can also be explored that are related to neuron level.

2.1 Reverse Neuron Level Decomposition

Reverse neuron level (RNL) decomposition encodes the neural network in a way where the input and hidden neurons are used as reference points rather than the hidden and output neurons as in the case of neuron level. Each subcomponents in RNL consists of outgoing synapse connected linked with neurons in the input and hidden layers. Therefore, each subcomponent for a layer is composed as follows:

1. For a given neuron i in the input layer, the input layer subcomponents consists of all the synapses that are connected from input neuron i to the hidden layer. The bias of i is also included.

2. For a given neuron j in the hidden layer, the hidden layer subcomponents consists of all synapses that are connected from hidden neuron j to the output layer. The bias of j is also included.

The total number of subcomponents is the total number of hidden and input neurons along with biases within hidden and input layer neurons as shown in Fig. 3. RNL is used to train the feedforward network and is shown in Algorithm 1. In Step 1, the network is encoded using RNL problem decomposition.

In Step 2, subcomponents are encoded as sub-populations. In Step 3, each sub-population is evolved using the designated evolutionary algorithm. Evaluation of the fitness of each individual in a particular sub-population is done cooperatively [4]. This is done by concatenating the best individuals from the rest of the sub-populations and encoding them into the neural network for fitness evaluation.

Algorithm 1: RNL for Training Feedforward Networks

```
Step 1: Decompose the problem into subcomponents according to RNL.
Step 2: Implement each subcomponent in a sub-population.
Step 3: Initialize and cooperatively evaluate each sub-population.
foreach Cycle until termination do
    foreach Sub-population do
        foreach Depth of n Generations do
            Select and create new offspring using genetic operators
            Cooperative Evaluation the new offspring
        end
    end
end
```

All the sub-populations are evolved for a fixed number of generations. The evolution continues until the termination criteria is met which is either the total number of function evaluations or the desired fitness for the network based on validation dataset. Once the network has been evolved, the neural network is tested for generalization performance.

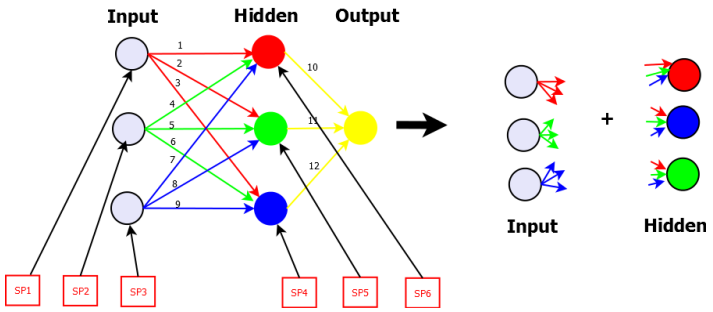


Fig. 3: Reverse neuron level problem decomposition. Note that the number of sub-population are based on the number of input and hidden neurons.

The proposed decomposition creates fewer subcomponents when compared to synapse level decomposition. It creates more subcomponents when compared to neuron level decomposition for problems with one output neuron that are used for single dimensional time series problems.

3 Experiments and Results

This section presents the experiments using the proposed RNL cooperative neuro-evolution method for training feedforward neural networks. NL and SL

problem decomposition methods are used for comparison. The performance and results of the method were evaluated using different number of hidden neurons.

Four different benchmark chaotic time series datasets are used to evaluate the proposed method. Mackey-Glass time series [11] and Lorenz time series [12] are the two simulated time series. The real-world problems include the Sunspot [13] and ACI Worldwide Inc. [14] time series. The results are compared with related work from the literature.

3.1 Experimental Setup

All the datasets are reconstructed using Taken's embedding theorem [15]. Taken's embedding theorem allows chaotic time series dataset to be reconstructed into a state space vector using *time delay (T)* and *embedding dimension (D)* [15].

The differential equation used to generate the Mackey Glass time series is given below in Equation 1.

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

1000 sample points are used and the phase space of the original time series is reconstructed with the embedding dimensions $D = 3$ and $T = 2$. The first half of samples are used for training while the rest for testing. This time series data set is scaled in the range [0,1].

The *Lorenz time series* is simulated time series, which is chaotic in nature that was proposed by Edward Lorenz along with the principles of Chaos theory [12]. This dataset is scaled in the range of [-1,1]. The first half of the 1000 sample points are used for training while the rest for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D = 3$ and $T = 2$, similar as previous time series data.

The *Sunspot time series* gives good indication of the solar activities for solar cycles which impacts Earth's climate [16]. The monthly smoothed Sunspot time series has been obtained from the *World Data Center* for the Sunspot Index [13]. This time series is scaled in the range [-1,1] and first is used for training while the second half for testing. Embedding dimension of $D = 5$ and $T = 2$ is used.

The *ACI Worldwide Inc. time series* [14] is taken from the NASDAQ stock exchange and contains 800 sample points from December 2006 to February 2010 [14]. It is scaled in the range [0,1]. The first half is used for training and the second half for testing. Embedding dimension of $D = 5$ and $T = 2$ is used.

The feedforward network employs sigmoid units in the hidden and the output layer for the Mackey-Glass and ACI Worldwide Inc. problems. However, the hyperbolic tangent unit is used in the output layer for the Lorenz and Sunspot time series. Root mean squared error (RMSE) is used to evaluate the performance of the proposed method as done in previous work [3]. The G3-PCX evolutionary algorithm is employed to evolve all the sub-populations [17]. It uses specific parameters for creation of new solutions such as the pool size of 2 parents and 2 offspring [3].

The number of generations for each sub-population known as *depth of search* is kept as 1 as done in [3]. The algorithm terminates once the maximum number of function evaluations (50 000) have been reached by the respective cooperative neuro-evolutionary methods (RNL, NL and SL).

3.2 Results

In Tables 1 - 4, the results are shown for different number of hidden neurons using RNL, NL and SL. The RMSE and 95 percent confidence interval along with the best run are reported for 50 independent experimental runs. We evaluate the best results for each case with lowest RMSE in training and generalization performance irrespective of number of hidden neurons used.

In the Mackey-Glass problem shown in Table 1, it was observed that RNL gives better training and generalization performance when compared to SL. RNL was unable to outperform NL.

In Table 2, for the Lorenz problem, it was observed that the RNL has performed much better than SL, however, it was unable to outperform NL in terms of training or generalization. Unlike for the Mackey-Glass problem, it was seen that the generalization and training performance of all the methods deteriorates when the number of the hidden neuron increases.

Table 1: Training and generalisation performance for Mackey-Glass time series problem

Method	H	Training	Generalisation	Best
NL	3	0.0107 ± 0.00131	0.0107 ± 0.00131	0.0050
	5	0.0089 ± 0.00097	0.0088 ± 0.00097	0.0038
	7	0.0078 ± 0.00079	0.0078 ± 0.00079	0.0040
SL	3	0.0237 ± 0.0023	0.0237 ± 0.0023	0.0125
	5	0.0195 ± 0.0012	0.0195 ± 0.0012	0.0124
	7	0.0177 ± 0.0009	0.0178 ± 0.0009	0.0121
RNL	3	0.0151 ± 0.00087	0.0151 ± 0.00087	0.0076
	5	0.0132 ± 0.00064	0.0132 ± 0.00064	0.0088
	7	0.0133 ± 0.00066	0.0133 ± 0.00066	0.0092

The performance of the Sunspot time series problem in Table 3 shows that RNL was able to outperform SL and NL in generalisation performance.

The ACI Worldwide Inc. in Table 4 shows similar performance when compared to Sunspot time series as both are real world applications that contain noise. RNL outperformed SL and gave close performance to NL.

Tables 5 - 8, compares the best results from Tables 1 - 4 with some of the related methods from the literature. The RMSE of the best experimental run

Table 2: Training and generalisation performance for the Lorenz time series problem

Method	H	Training	Generalisation	Best
NL	3	0.0170 ± 0.0031	0.0176 ± 0.0031	0.0043
	5	0.0249 ± 0.0062	0.0271 ± 0.0067	0.0021
	7	0.0379 ± 0.0093	0.0416 ± 0.0092	0.0024
SL	3	0.0680 ± 0.0325	0.0452 ± 0.0229	0.0153
	5	0.0526 ± 0.0084	0.0546 ± 0.0084	0.0082
	7	0.0574 ± 0.0075	0.0605 ± 0.0074	0.0079
RNL	3	0.0263 ± 0.0051	0.027 ± 0.0051	0.0062
	5	0.0309 ± 0.0087	0.0333 ± 0.0087	0.0075
	7	0.0395 ± 0.0083	0.0435 ± 0.0083	0.0058

Table 3: Training and generalisation performance for Sunspot time series

Method	H	Training	Generalisation	Best
NL	3	0.0207 ± 0.0035	0.0538 ± 0.0091	0.015
	5	0.0289 ± 0.0039	0.0645 ± 0.0093	0.017
	7	0.0353 ± 0.0048	0.0676 ± 0.0086	0.021
SL	3	0.0539 ± 0.0261	0.04998 ± 0.0238	0.210
	5	0.0560 ± 0.0208	0.05210 ± 0.0177	0.302
	7	0.0568 ± 0.0178	0.05250 ± 0.0132	0.344
RNL	3	0.0411 ± 0.0051	0.0472 ± 0.0048	0.031
	5	0.0390 ± 0.0044	0.0467 ± 0.0039	0.030
	7	0.0414 ± 0.0069	0.0533 ± 0.0060	0.030

Table 4: Training and generalisation performance for ACI Worldwide Inc. time series

Method	H	Training	Generalisation	Best
NL	3	0.0214 ± 0.00039	0.0215 ± 0.00039	0.020
	5	0.0203 ± 0.00047	0.0212 ± 0.00041	0.019
	7	0.0201 ± 0.00038	0.0208 ± 0.00033	0.019
SL	3	0.0466 ± 0.0039	0.0411 ± 0.00360	0.080
	5	0.0413 ± 0.0038	0.0390 ± 0.00378	0.042
	7	0.0449 ± 0.0028	0.0424 ± 0.00270	0.134
RNL	3	0.0250 ± 0.00097	0.0228 ± 0.00077	0.019
	5	0.0236 ± 0.00075	0.0220 ± 0.00059	0.019
	7	0.0232 ± 0.00072	0.0219 ± 0.00063	0.019

together with NMSE (normalized mean squared error) are used for the comparison. The proposed RNL method has given good performance when compared to some of the methods in the literature.

The best result of Mackey-Glass time series problem was compared to methods from the literature in Table 5. The proposed method was able to outperform some of the methods.

The Table 6 shows the best result of Lorenz time series problem being compared to works in literature. It was seen that the proposed method outperformed all the methods except for co-evolutionary recurrent neural networks (CICC-RNN) which cannot be strictly compared due to difference in network architectures.

Table 5: A comparison with the results from literature on the Mackey-Glass time series

Prediction Method	RMSE	NMSE
Locally linear neuro-fuzzy model (LLNF-LoLiMot) (2006) [18]	9.61E-04	
Neural fuzzy network and PS0 (2009) [19]	2.10E-02	
Neural fuzzy network and CPS0 (2009) [19]	1.76E-02	
Neural fuzzy network and DE (2009) [19]	1.62E-02	
Neural fuzzy network and GA (2009)[19]	1.63E-02	
Synapse Level-CCRNN (SL-CCRNN) (2012) [3]	6.33E-03	2.79E-04
Neuron Level-CCRNN (NL-CCRNN) (2012) [3]	8.28E-03	4.77E-04
Competitive Island Cooperative Coevolution (CICC-RNN) (2014) [20]	3.99E-03	1.11E-04
Proposed FNN-RNL	7.59E-03	1.09E-04

Table 6: A comparison with the results from literature on the Lorenz time series

Prediction Method	RMSE	NMSE
Auto regressive moving average (ARMA-ANN)(2008) [21]	8.76E-02	
Backpropagation neural network and GA (2011) [22]	2.96E-02	
Synapse Level-CCRNN (SL-CCRNN) (2012) [3]	6.36E-03	7.72E-04
Neuron Level-CCRNN (NL-CCRNN) (2012) [3]	8.20E-03	1.28E-03
Competitive Island Cooperative Coevolution (CICC-RNN) (2014) [20]	3.55E-03	2.41E-04
Proposed FNN-RNL	5.81E-03	1.77E-04

The best result of the Sunspot time series problem was compared to the literature in Table 7. The proposed method was unable to outperform different co-evolutionary recurrent neural networks (SL-RNN, NL-RNN and CICC-RNN) which cannot be strictly compared due to difference in neural network architecture.

The best result of the ACI Worldwide Inc. problem was compared to the literature in Table 8. The proposed method outperformed all the methods except

for multi-objective method with T=3. The results are better when compared to other works from the literature.

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

Prediction Method	RMSE	NMSE
Radial basis network (RBF-OLS)(2006) [18]		4.60E-02
Locally linear neuro-fuzzy model (LLNF-LoLiMot) (2006) [18]		3.20E-02
Synapse Level-CCRNN (SL-CCRNN) (2012) [3]	1.66E-02	1.47E-03
Neuron Level-CCRNN (NL-CCRNN) (2012) [3]	2.60E-02	3.62E-03
Competitive Island Cooperative Coevolution (CICC-RNN) (2014) [20]	1.57E-02	1.31E-03
Proposed FNN-RNL	2.96E-02	2.68E-03

Table 8: A comparison with the results from literature on the ACI time series

Prediction Method	RMSE	NMSE
Competitive Island Cooperative Coevolution CICC-RNN [23]	1.92E-02	
Synapse Level (FNN-SL) (2014) [24]	1.92E-02	
Neuron Level (FNN-NL) (2014) [24]	1.91E-02	
MOCCFNN with 2-objectives (T=2)(MO-CCFNN-T=2) (2014)[25]	1.94E-02	
MOCCFNN with 2-objectives (T=3)(MO-CCFNN-T=3) (2014)[25]	1.470E-02	
Proposed FNN-RNL	1.91E-02	2.00E-03

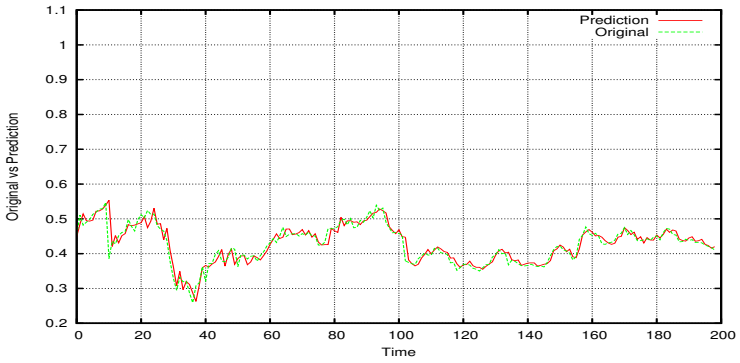
Figure 4 show that a typical experimental run from the RNL method was able to cope with the noise from one of the real-world dataset. The error graph is also given which indicates the challenges for the chaotic nature of these time series problems at certain time intervals.

4 Discussion

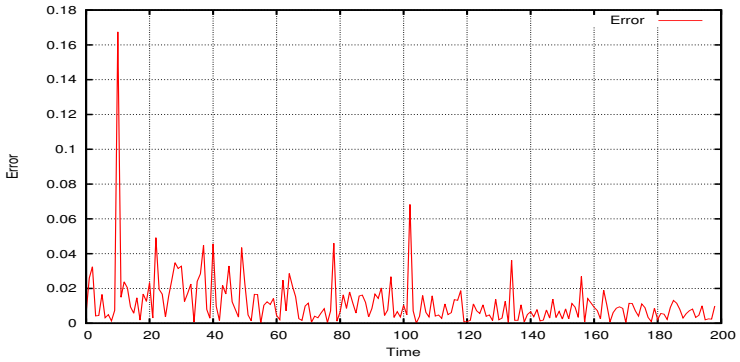
In general, the results of the experiments showed that proposed reverse neuron level is better than synapse level and gives close performance when compared to neuron level for given time series problems. Reverse neuron level gives a competitive performance when compared to other methods from the literature.

Reverse neuron level employs fewer subcomponents when compared to synapse level and more subcomponents when compared to neuron level according to the network topology with one output neuron used for one step ahead time series problems. The results showed that reverse neuron level has been able to achieve similar performance when compared to other methods in terms of the generalization.

Reverse neuron level groups subcomponents similar to synapse level for the hidden-output layer. It seems that due to more interaction between hidden to



(a) Performance given by RNL on the test set.



(b) Error on the test data set.

Fig. 4: Typical train prediction given by RNL for ACI Worldwide Inc time series.

output layer, where more decision making takes place, it was unable to have better performance when compared to neuron level. Reverse neuron level seems to perform better than synapse level mainly due to the weight connections between input-hidden layer. It is better to combine the weights together, which is the case in reverse neuron level and neuron level.

5 Conclusion and Future Work

In this paper, we proposed reverse neuron level decomposition for the cooperative neuro-evolution of feedforward neural networks applied to time series problems. The results of the experiments have given a better understanding of decomposition of neural network in terms of interacting variables. The proposed method has also outperformed some of the methods from the literature. In general, the proposed method is much better than synapse level decomposition and produces competitive results with neuron level decomposition.

In future work, the proposed method can be further tested on other problems, including multi-dimensional time series. The method can be used to evolve other neural network architectures for pattern classification and control.

References

1. H. K. Stephen, *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*. University of Chicago Press, 1993.
2. E. Parras-Gutierrez, V. Rivas, M. Garcia-Arenas, and M. del Jesus, "Short, medium and long term forecasting of time series using the l-co-r algorithm," *Neurocomputing*, vol. 128, no. 0, pp. 433 – 446, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213008990
3. R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 186, pp. 116 – 123, 2012.
4. M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature PPSN III*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Mnner, Eds. Springer Berlin Heidelberg, 1994, vol. 866, pp. 249–257.
5. F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.
6. N. García-Pedrajas and D. Ortiz-Boyer, "A cooperative constructive method for neural networks for pattern recognition," *Pattern Recogn.*, vol. 40, no. 1, pp. 80–98, 2007.
7. R. Chandra, M. R. Frean, and M. Zhang, "Crossover-based local search in cooperative co-evolutionary feedforward neural networks," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 2924–2932, 2012.
8. F. Gomez and R. Mikkulainen, "Incremental evolution of complex general behavior," *Adapt. Behav.*, vol. 5, no. 3-4, pp. 317–342, 1997.
9. R. Chandra, M. Frean, and M. Zhang, "On the issue of separability for problem decomposition in cooperative neuro-evolution," *Neurocomputing*, vol. 87, pp. 33–40, 2012.
10. ——, "An encoding scheme for cooperative coevolutionary neural networks," in *23rd Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Artificial Intelligence. Adelaide, Australia: Springer-Verlag, 2010, pp. 253–262.
11. M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.
12. E. Lorenz, "Deterministic non-periodic flows," *Journal of Atmospheric Science*, vol. 20, pp. 267 – 285, 1963.
13. SILSO World Data Center, "The International Sunspot Number (1834-2001), International Sunspot Number Monthly Bulletin and Online Catalogue," Royal Observatory of Belgium, Avenue Circulaire 3, 1180 Brussels, Belgium, accessed: 02-02-2015. [Online]. Available: http://www.sidc.be/silso/
14. "NASDAQ Exchange Daily: 1970-2010 Open, Close, High, Low and Volume," accessed: 02-02-2015. [Online]. Available: http://www.nasdaq.com/symbol/aciw/stock-chart
15. F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, ser. Lecture Notes in Mathematics, 1981, pp. 366–381.

16. S. S., "Solar cycle forecasting: A nonlinear dynamics approach," *Astronomy and Astrophysics*, vol. 377, pp. 312–320, 2001.
17. K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, 2002.
18. A. Gholipour, B. N. Araabi, and C. Lucas, "Predicting chaotic time series using neural and neurofuzzy models: A comparative study," *Neural Process. Lett.*, vol. 24, pp. 217–239, 2006.
19. C.-J. Lin, C.-H. Chen, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 55–68, Jan. 2009.
20. R. Chandra, "Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction," *Neural Networks and Learning Systems, IEEE Transactions on*, p. In Press, 2015.
21. I. Rojas, O. Valenzuela, F. Rojas, A. Guillen, L. Herrera, H. Pomares, L. Marquez, and M. Pasadas, "Soft-computing techniques and arma model for time series prediction," *Neurocomputing*, vol. 71, no. 4-6, pp. 519 – 537, 2008.
22. M. Ardalani-Farsa and S. Zolfaghari, "Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting," *Appl. Artif. Intell.*, vol. 25, pp. 45–73, 2011.
23. R. Chandra, "Competitive two-island cooperative coevolution for training Elman recurrent networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, Jul. 2014, pp. 565 – 572.
24. S. Chand and R. Chandra, "Cooperative coevolution of feed forward neural networks for financial time series problem," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, July 2014, pp. 202–209.
25. ——, "Multi-objective cooperative coevolution of neural networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, July 2014, pp. 190–197.