# Adaptive Problem Decomposition in Cooperative Coevolution of Recurrent Networks for Time Series Prediction

Rohitash Chandra

***Abstract*—Cooperative coevolution employs different problem decomposition methods to decompose the neural network problem into subcomponents. The efficiency of a problem decomposition method is dependent on the neural network architecture and the nature of the training problem. The adaptation of problem decomposition methods has been recently proposed which showed that different problem decomposition methods are needed at different phases in the evolutionary process. This paper employs an adaptive cooperative coevolution problem decomposition framework for training recurrent neural networks on chaotic time series problems. The Mackey Glass, Lorenz and Sunspot chaotic time series are used. The results show improvement in performance in most cases, however, there are some limitations when compared to cooperative coevolution and other methods from literature.**

## I. Introduction

The prediction of chaotic time series has a wide range of applications such as in finance [1], signal processing [2], power load [3], weather forecast [4], and sunspot prediction [5], [6], [7]. Chaos theory is used to study the behaviour of dynamical systems that are highly sensitive to initial conditions such as noise and error [8], [9].

Cooperative coevolution (CC) divides a problem into subcomponents which are implemented as sub-populations. In the original cooperative coevolution framework, the problem was decomposed by having a separate subcomponent for each variable [10]. It was later found that the strategy was mostly effective for problems that are separable [11]. Cooperative coevolution naturally appeals to separable problems as there is little interaction among the subcomponents during evolution [12]. The efficiency of a problem decomposition method is dependent on the neural network architecture and the nature of the training problem. The *degree of non-separability* is referred as problem description that determines the level of interdependencies among variables [13].

There are two major problem decomposition methods for neuro-evolution that decomposes the network on the *neuron* and *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 co-evolved synapses neuro-evolution [14] and neural fuzzy network with cultural cooperative particle swarm optimisation [15]. In neural level problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced sub-populations [16], [17] and neuron-based subpopulation [18], [19].

Adaptation of problem decomposition in different phases of evolution has been effective for training feedforward neural networks on pattern recognition problems [20] and recurrent neural networks on grammatical inference problems [21]. The results have shown that it is reasonable to adapt the problem decomposition method at different stages of evolution. We have shown that the neural network training problem changes at different stages of evolution in terms of the degree of non-separability [13]. The use of cooperative coevolution in training recurrent neural networks for time series problems has been given in our recent work [22].

This paper employs the adaptive modularity cooperative coevolution framework (AMCC) [20], [21] for training recurrent neural networks on chaotic time series problems. The Elman recurrent network [23] is used and three different chaotic time series problems where the Lorenz and Mackey-Glass are the simulated time series while the Sunspot is the real-world time series. The generalised generation gap with parent centric crossover (G3-PCX) evolutionary algorithm [24] is employed in the sub-populations of AMCC. The performance of AMCC is compared with neuron, synapse and network level problem decomposition methods [22] and other computational intelligence methods from the literature.

The rest of the paper is organised as follows. A brief background on cooperative coevolution is presented in Section 2 and Section 3 gives details of the adaptive modularity cooperative coevolution method for training recurrent networks on chaotic time series problems. Section 4 presents the results and furthermore, Section 5 concludes the work with a discussion on future work.

## II. Background

### A. Cooperative Coevolution for Neuro-evolution

Cooperative coevolution divides a large problem into smaller subcomponents which are implemented as sub-populations that are evolved in isolation and cooperation takes place for fitness evaluation [10]. The subcomponents are also referred as modules. Problem decomposition determines how the problem is broken down into subcomponents. The size of a subcomponent and the way it is encoded depends on the problem. The original CC framework has been used for general

School of Computing, Information and Mathematical Sciences, Faculty of Science, Technology and Environment, University of the South Pacific, Laucala Campus, Suva, Fiji. email: c.rohitash@gmail.com.

function optimisation and the problems were decomposed to its lowest level where a separate subcomponent was used to represent each dimension of the problem [10]. It was later found that this strategy is only effective for problems which are fully separable [11]. Much work has been done in the use of cooperative coevolution in large scale function optimization and the focus has been on non-separable problems [11], [25], [26], [27].

A function of $n$ variables is separable if it can be written as a sum of $n$ functions with just one variable [28]. 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. Evolutionary algorithms without any decomposition strategy appeal to fully non-separable problems [13].

The subpopulations 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 scheme has been able to group the interacting variables into separate subcomponents [19]. If the interacting variables have been grouped efficiently, then a deep greedy search for the sub-population is possible, implying that the problem has been efficiently broken down into subcomponents which have fewer interactions amongst themselves [13].

### B. Encoding Schemes for Recurrent Networks

There are three major encoding schemes based on the CC framework for training recurrent neural networks. The first scheme proposes a neuron level encoding where each neuron in the hidden layer is used as a major reference point for each module in the CC framework. Therefore, the number of hidden neurons is equal to the the number of subcomponents. In *Enforced Subpopulation* (ESP) [16], [17], a particular neuron $h_i$ in the hidden layer encodes the input, output, and recurrent weight links connected to it. In this encoding scheme, the sizes of all individual subpopulations are the same for the entire framework.

The second encoding scheme was presented in the cooperatively coevolved synapse (CoSyNE) where each connection in the network is part of a single subpopulation. CoSyNE demonstrated better performance than ESP on the two pole balancing problem without velocity information [14].

The third encoding scheme decomposes the network into the neuron level which is known as the neuron based subpopulation (NSP) [19]. NSP has shown to be more efficient than CoSyNE for training recurrent neural networks for grammatical inference problems in [19]. NSP has performed better than CoSyNE and ESP for pattern recognition problems in [29]. CoSyNE views the recurrent network as a separable problem and has been successful for pole balancing [14]; however, it performed poorly for pattern recognition problems [29].

## III. Adaptive Problem Decomposition in Cooperative Coevolution

The general idea behind the adaptive modularity cooperative coevolution (AMCC) framework is to use the strength of a different problem decomposition method which reflects on the degree of non-separability when needed during evolution [20], [21]. AMCC employs modularity (problem decomposition or encoding scheme) with greater level of flexibility (allowing evolution for separable search space) during the initial stage and decreases the level of modularity during the later stages of evolution.

The AMCC framework is given in Algorithm 1. Initially, all the sub-populations of the synapse level, neuron level and network level encoding are randomly initialised with random real values in a range. In Stage 1, the sub-populations at synapse level encoding are cooperatively evaluated. Neuron level and Network level encoding are left to be cooperatively evaluated at Stage 2.

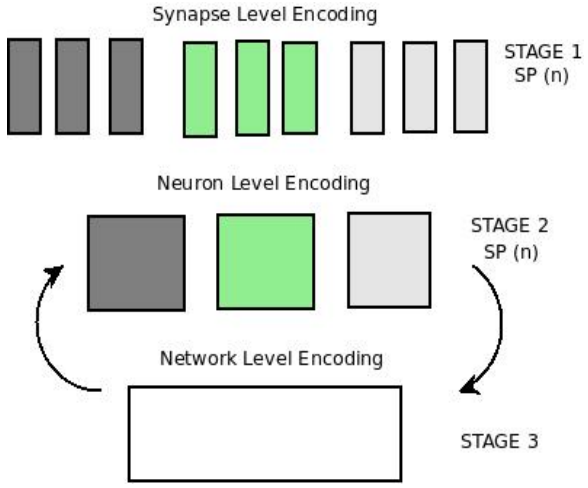



Fig. 1. The AMCC framework used for training the recurrent network on chaotic time series. The sub-populations (SP) at Synapse level and Neuron level are shown.

The details of the different problem decomposition methods are given below.

1) **Synapse level encoding**: Decomposes the network into its lowest level to form a single subcomponent [14], [15]. The number of connections in the network determines the number of subcomponents.
2) **Neuron level encoding**: Decomposes the network into neuron level. The number of neurons in the hidden, state and output layer determines the number of subcomponents [19].
3) **Network level encoding**: The standard neuro-evolutionary encoding scheme where only one population represents the entire network. There is no decomposition present in this level of encoding.

**Alg. 1** Adaptive Modularity in Cooperative Coevolution

**Stage 1:** Synapse level encoding
Cooperatively evaluate Synapse level only

**while** *FuncEval* $\leq \alpha\times$ *MaxGlobal* **do**
  **foreach** *each Sub-population at Synapse level* **do**
    Create new offspring
    Cooperative Evaluation
  **end**
**end**
**Stage 2:** Neuron and Network level encoding
i. Merge individuals from Synapse level into Neuron level
ii. Cooperatively evaluate Neuron level
**while** *FuncEval* $\leq$ *MaxGlobal* **do**
  **while** *FuncEval* $\leq \beta\times$ *MaxGlobal* **do**
    **foreach** *each Sub-population at Neuron level* **do**
      Create new offspring
      Cooperative Evaluation
    **end**
  **end**
  i. Merge all individuals into Network level
  ii. Evaluate Network level

  **while** *FuncEval* $\leq \beta\times$ *MaxGlobal* **do**
    Create new offspring
  **end**
  i. Break all individuals from Network to Neuron level
  ii. Evaluate Neuron level
**end**

Stage 1 employs synapse level encoding where the sub-populations are evolved until $\alpha$ portion of the maximum time. The sub-populations of synapse level encoding are merged into neuron level. The individuals with their fitness are transferred to the sub-populations of the Neuron level in Stage 2. The framework proceeds to Neuron and Network level encoding in Stage 2. All the sub-populations are cooperatively evaluated. The Neuron level encoding is then evolved for $\beta$ portion of the maximum time. The sub-populations for the neuron level are then merged into a single population for the network level and evaluated. The Network level encoding is evolved for $\beta$ portion of the maximum time. The population of the Network level is then broken down and encoded as Neuron level evolution phase and evaluated. The procedure in Stage 2 is repeated until the maximum training time has been reached or if the minimum network error given by root mean squared error has been reached. Figure 1 shows futher description of the AMCC framework that shows Stage 2 and Stage 3 level of modularity repeats in a cycle untill termination.

In the original AMCC framework presented in [20], [21], the transformation is from synapse level to neuron level and finally to network level. The AMCC framework presented in Algorithm 1 is similar, however, slightly different in three ways:

1) It transforms from one level of modularity into another based on the training time rather than the neural network error;
2) It employs synapse level encoding for the first $\alpha$ portion of the maximum training time. It then transforms into the neuron level and then to the network level and repeats this transformation until termination. The neuron and network levels are encoded for $\beta$ portion of the total training time given by number of function evaluations;
3) In the transformation from one level of encoding to another, all the individuals of the sub-populations are transferred rather than only the best ones.

Cooperative evaluation of individuals in the respective sub-populations is done by concatenating the chosen individual from a given sub-population with the best individuals from the rest of the sub-populations [10], [18], [19], [22]. The concatenated individual is encoded into the recurrent neural network and the fitness is calculated by the root mean squared error. The goal of the evolutionary process is to increase the fitness which tends to decrease the network error. In this way, the fitness of each subcomponent in the network is evaluated until the cycle is completed.

The transition from one level of modularity to another has to ensure that the information gained using the existing modularity is transferred to the next level of encoding. Note that the number of sub-populations in each level are different. The Synapse level encoding has more sub-populations than the Neuron level, however, the individuals in the sub-populations of Synapse level must be merged to Neuron level. The sub-populations are merged when the transformation is from the Synapse to the Neuron level and from the Neuron to the Network level. The transformation from the Network to the Neuron level requires the population to be broken down and encoded as the Neuron level.

## IV. Simulation and Analysis

This section presents an experimental study of AMCC for training recurrent neural networks on chaotic time series. The Neuron level (NL) [22] and Synapse level (SL) [22] problem decomposition methods are used for comparison. The Mackey Glass time series [30] and Lorenz time series [8] are the two simulated time series while the real-world problem is the Sunspot time series [31]. The behaviour of the respective methods are evaluated on different recurrent network topologies which are given by different numbers of hidden neurons. The size and description of the respective dataset is taken from our previous work for a fair comparison [22]. The results are further compared with other computational intelligence methods from literature.

Given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t-T), ..., x(t(D-1)T)]$ can be generated, where, $T$ is the time delay, $D$ is the embedding dimension, $t = 0, 1, 2, ..., N - DT - 1$ and $N$ is the length of the original time series [32]. Taken's 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 Taken's theorem [33]. Taken's proved that if the original attractor is of dimension $d$,

then $D = 2d + 1$ will be sufficient to reconstruct the attractor [32].

The reconstructed vector is used to train the recurrent network for one-step-ahead prediction where 1 neuron is used in the input and the output layer. The recurrent network unfolds $k$ steps in time which is equal to the embedding dimension $D$ [5], [34], [22].

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 Equation 1 and Equation 2.

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

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

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. These two performance measures are used in order to compare the results with the literature.

### A. Problem description

The *Mackay Glass time series* has been used in literature as a benchmark problem due to its chaotic nature [30]. The differential equation used to generate the Mackey Glass time series is given in Equation 3.

$$\frac{\delta x}{\delta t} = \frac{ax(t-\tau)}{[1 + x^c(t-\tau)]} - bx(t) \tag{3}$$

In Equation 3, the delay parameter $\tau$ determines the characteristic of the time series, where $\tau > 16.8$ produces chaos. The selected parameters for generating the time series is taken from the literature [35], [36], [7], [37] where the constants $a = 0.2$, $b = 0.1$ 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 Equation 3. The first 500 samples are used for training the Elman network while rest of the 500 samples are used for testing. 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* was introduced by Edward Lorenz who has extensively contributed to the establishment of Chaos theory [8]. The Lorenz equation are given in Equation 4.

$$\begin{aligned}\frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[r - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t)\end{aligned} \tag{4}$$

where $\eta$, $r$, and $b$ are dimensionless parameters. The typical values of these parameters are $\eta = 10$, $r = 28$, and $b = 8/3$ [38], [7], [39], [40], [37]. 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 impacts Earth's climate, weather patterns, satellite and space missions [6]. 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 [31]. 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 literature [7], [37]. 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 $T = 2$.

Note that the scaling of the three time series in the range of [0,1] and [-1,1] are done as in the literature in order to provide a fair comparison.

### B. Experimental set-up

The Elman recurrent network employs sigmoid units in the hidden layer of the three different problems. In the output layer, a sigmoid unit is used for the Mackey Glass time series while hyperbolic tangent unit is used for Lorenz and Sunspot time series. The experiemnt set-up is same as our previous works [22]. The RMSE and NMSE given in Equation 1 and Equation 2 are used as the main performance measures of the recurrent network.

In the respective CC framework for recurrent networks (SL and NL) shown in Algorithm 1, each subpopulation 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 1 generation gives optimal performance for both NL and SL encodings [19]. 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 three problems is when a total of 100 000 function evaluations has been reached by the respective evolutionary training algorithm. $\alpha = 0.2$ and $\beta = 0.1$ in the AMCC framework from Algorithm 1. We obtained these values from trial experiments that can be adjusted for different application problems.

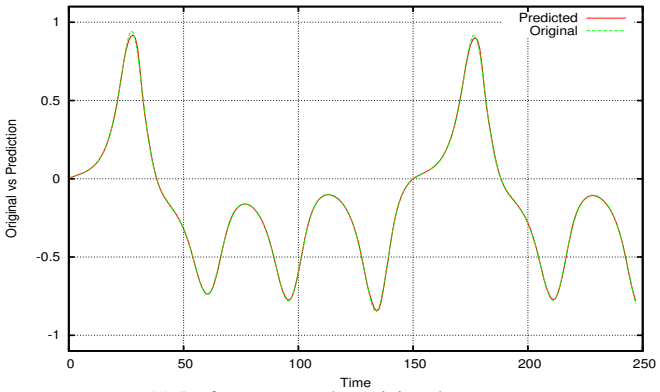
### C. Results and discussion

This section reports the performance of AMCC 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.

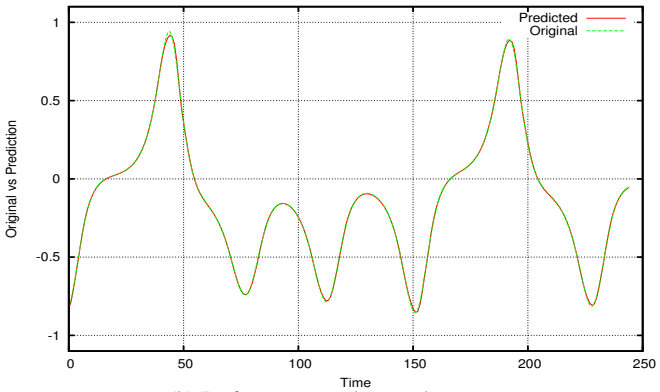
Initially, the number of hidden neurons is empirically evaluated and the mean and the best value of the RMSE is given from 30 experimental runs. The number of hidden neurons directly influences the difficulty of the learning problem. It

is more difficult to learn the problem if enough neurons are not present in the hidden layer. The performance on the test set are shown in Table I where the performances of AMCC for different numbers of hidden neurons are given. The best results are highlighted in bold.

The best results from Table I is chosen and further details are given in Table II 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 II and shown in Figures 2 - 4. These arecompared with NL and SL from our previous work [22].




(a) Performance on the training dataset



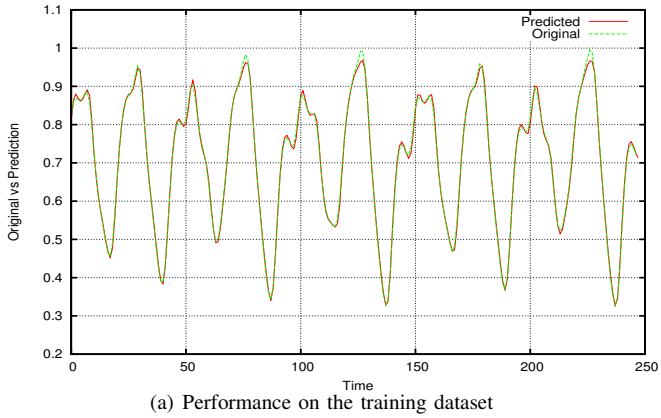

(b) Performance on the test dataset

Fig. 2. Typical prediction given by AMCC for Lorenz time series

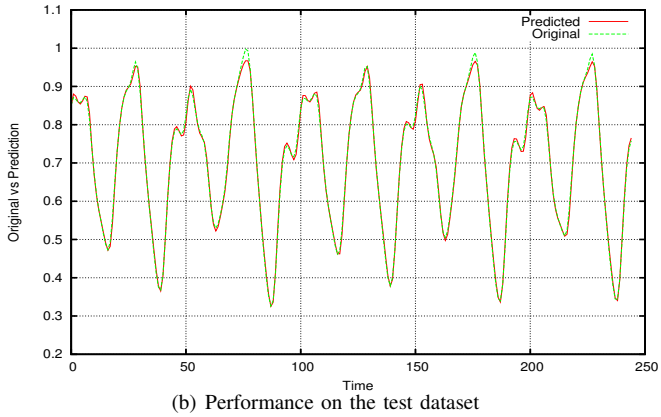
In the results for the Mackey time series, SL gives the best performance in terms of the RMSE, however, AMCC performance is close to SL. NL and SL use the same number of hidden neurons (13), however, the performance of SL is better. The results for the Lorenz time series in Table II show that AMCC gives the best performance with 9 hidden neurons in terms of RMSE when compared to the other methods.

In the Sunspot time series, the AMCC gives the best performance with 3 hidden neurons. 3 neurons also give the best performance in NL and SL. Note that this is a real-world problem which contains noise.

The performance of AMCC on the different problems are further compared to some of the results published in literature




(a) Performance on the training dataset




(b) Performance on the test dataset

Fig. 3. Typical prediction given by AMCC for Mackey Glass time series

as shown in Tables III - V. The best values from the results in Table II are used to compare with the results from literature.

In Table IV, the proposed method has 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 particle swarm optimisation (CPSO), genetic algorithms and differential evolution (DE) [15]. AMCC has given better performance than most of the methods from literature with the only exception being the Hybrid NARX-Elman networks [37].

The comparison of results with literature has shown that AMCC performs better than several other methods. However, it has not outperformed some methods that have additional enhancements such as the optimisation of the embedding dimensions and strength of architectural properties of hybrid neural networks with residual analysis [37]. These can be added to further improve the results. The cooperative coevolution methods outperformed several other methods on the real-world Sunspot time series that contained noise. This reflects on robustness.

AMCC has also shown to better maintain its performance with different number of hidden neurons as compared to other problem decomposition methods shown in Table I. This

TABLE I
THE PERFORMANCE ON THE TEST DATASET OF THE SUNSPOT TIME SERIES

| | Lorenz | | Mackey | | Sunspot | |
|---|---|---|---|---|---|---|
| Hidden | Mean | Best | Mean | Best | Mean | Best |
| 3 | 2.2E-2 | 1.1E-2 | 1.5E-2 | 6.9E-3 | **4.4E-2** | 2.4E-2 |
| 5 | 1.9E-2 | 6.8E-3 | 1.5E-2 | 6.9E-3 | 7.9E-2 | 1.7E-2 |
| 7 | 1.5E-2 | 7.6E-3 | 1.2E-2 | 6.8E-3 | 8.0E-2 | 2.4E-2 |
| 9 | **1.3E-2** | 5.1E-3 | 1.2E-2 | 8.1E-3 | 7.4E-2 | 1.9E-2 |
| 11 | 1.6E-2 | 6.9E-3 | **1.1E-2** | 7.5E-3 | – | – |
| 13 | 1.7E-2 | 6.3E-3 | 1.3E-2 | 8.0E-3 | – | – |

TABLE II
THE PERFORMANCE (RMSE AND NMSE) OF AMCC COMPARED WITH NL AND SL ENCODINGS [22] ON THE THE TEST DATASET OF THE THREE PROBLEMS. THE MEAN AND 95 % CONFIDENCE INTERVAL (CI) IS GIVEN WITH THE BEST PERFORMANCE OUT OF 30 INDEPENDENT EXPERIMENTS.

| | | | RMSE | | NMSE | |
|---|---|---|---|---|---|---|
| Problem | Method | Hidden | Mean and CI | Best | Mean and CI | Best |
| Mackey | SL | 13 | **9.39E-3** $\pm$ 5.57E-4 | 6.33E-3 | **6.31E-4** $\pm$ 7.60E-5 | 2.79E-4 |
| | NL | 13 | 1.23E-2 $\pm$ 9.16E-4 | 8.28E-3 | 1.11-3 $\pm$ 1.77E-4 | 4.77E-4 |
| | AMCC | 11 | 1.11E-2 $\pm$ 1.01E-3 | 7.53E-3 | 9.17E-4 $\pm$ 1.78E-4 | 3.90E-4 |
| Lorenz | SL | 5 | 1.95E-2 $\pm$ 2.59E-3 | 6.36E-3 | 8.28E-3 $\pm$ 1.98E-3 | 7.72E-4 |
| | NL | 11 | 1.82E-2 $\pm$ 2.82E-3 | 8.20E-3 | 7.48E-3 $\pm$ 2.60E-3 | 1.28E-3 |
| | AMCC | 9 | **1.35E-2** $\pm$ 2.01E-3 | 5.06E-3 | **4.04E-3** $\pm$ 1.31E-3 | 4.88E-4 |
| Sunspot | SL | 3 | 6.88E-2 $\pm$ 2.66E-2 | 1.66E-2 | 5.48E-2 $\pm$ 5.19E-2 | 1.47E-3 |
| | NL | 3 | 5.58E-2 $\pm$ 8.01E-3 | 2.60E-2 | 1.92E-2 $\pm$ 5.3eE-3 | 3.62E-3 |
| | AMCC | 3 | **4.39E-2** $\pm$ 5.61E-3 | 2.41E-2 | **1.16E-2** $\pm$ 3.05E-3 | 3.11E-3 |

TABLE III
A COMPARISON WITH THE RESULTS FROM LITERATURE ON THE LORENZ TIME SERIES

| Prediction Method | RMSE | NMSE |
|---|---|---|
| Backpropagation-through-time (BPTT-RNN) (2010) [34] | | 1.85E-03 |
| Real time recurrent learning (RTRL-RNN) (2010) [34] | | 1.72E-03 |
| Recursive Bayesian LevenbergMarquardt (RBLM-RNN) (2010) [34] | | 9.0E-04 |
| Hybrid NARX-Elman RNN with Residual Analysis (2010) [37] | 1.08E-04 | 1.98E-10 |
| Backpropagation neural network and genetic algorithms with residual analysis (2011) [41] | 2.96E-02 | |
| CCRNN-Synapse Level (2012) [22] | 6.36E-03 | 7.72E-04 |
| CCRNN-Neuron Level (2012) [22] | 8.20E-03 | 1.28E-03 |
| Proposed AMCC-RNN | 5.06E-03 | 4.88E-04 |

TABLE IV
A COMPARISON WITH THE RESULTS FROM LITERATURE ON THE MACKEY TIME SERIES

| Prediction Method | RMSE | NMSE |
|---|---|---|
| Neural fuzzy network and hybrid of cultural algorithm and cooperative particle swarm optimisation (CCPSO) (2009) [15] | 8.42E-03 | |
| Neural fuzzy network and particle swarm optimisation (PS0) (2009) [15] | 2.10E-02 | |
| Neural fuzzy network and cooperative particle swarm optimisation (CPS0) (2009) [15] | 1.76E-02 | |
| Neural fuzzy network and differential evolution (DE) (2009) [15] | 1.62E-02 | |
| Neural fuzzy network and genetic algorithm (GA ) (2009)[15] | 1.63E-02 | |
| Backpropagation neural network and genetic algorithms with residual analysis (2011) [41] | 1.30E-03 | |
| Hybrid NARX-Elman RNN with Residual Analysis (2010) [37] | 3.72E-05 | 2.70E-08 |
| Backpropagation neural network and genetic algorithms with residual analysis (2011) [41] | 1.30E-03 | |
| CCRNN-Synapse Level (2012) [22] | 6.33E-03 | 2.79E-04 |
| CCRNN-Neuron Level (2012) [22] | 8.28E-03 | 4.77E-04 |
| Proposed AMCC-RNN | 7.53E-03 | 3.90E-04 |

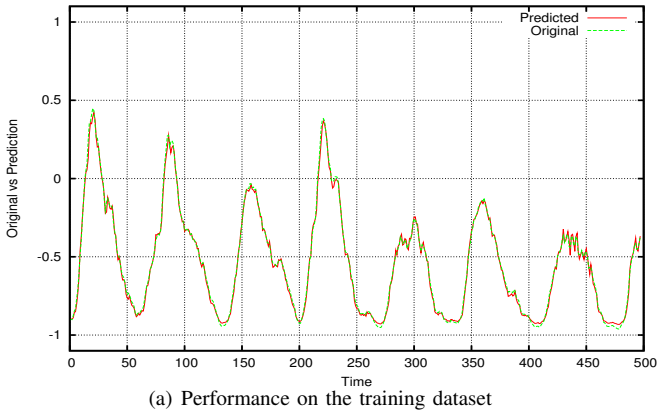
reflects on scalability and robustness.

The AMCC framework has performed better than the other problem decomposition methods (NL and SL) where no adaptation is present. This has been observed for the Lorenz and the Sunspot time series. In the Mackey Glass time series, the synapse level encoding showed sightly better performance than AMCC.

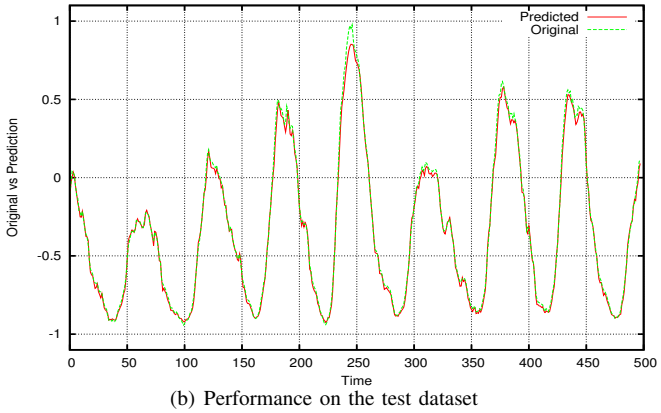
The AMCC framework employs the synapse level encoding in the beginning of the evolution phase. Synapse level encoding has strength in separable problems which exhibit lower degree of non-separability. It provides more flexibility and enforces global search. The efficiency in performance of AMCC indicates that the use of neuron and network level encoding is beneficial in later stages of evolution. One reason can be due to the change in the degree of non-separability as the interactions among the variables get stronger [13]. Hence,

TABLE V
A COMPARISON WITH THE RESULTS FROM LITERATURE ON THE SUNSPOT TIME SERIES

| Prediction Method | RMSE | NMSE |
|---|---|---|
| Multi-layer perceptron (1996) [5] | | 9.79E-02 |
| Elman RNN (1996) [5] | | 9.79E-02 |
| FIR Network (MLP) (1996) [5] | | 2.57E-01 |
| Wavelet packet multilayer perceptron (2001)[42] | | 1.25E-01 |
| Radial basis network with orthogonal least squares (RBF-OLS)(2006) [7] | | 4.60E-02 |
| Locally linear neuro-fuzzy model - Locally linear model tree (LLNF-LoLiMot) (2006) [7] | | 3.20E-02 |
| Hybrid NARX-Elman RNN with Residual Analysis (2010) [37] | 1.19E-02 | 5.90E-04 |
| CCRNN-Synapse Level (2012) [22] | 1.66E-02 | 1.47E-03 |
| CCRNN-Neuron Level (2012) [22] | 2.60E-02 | 3.62E-03 |
| Proposed AMCC-RNN | 2.41E-02 | 3.11E-03 |



(a) Performance on the training dataset



(b) Performance on the test dataset

Fig. 4. Typical prediction given by AMCC for Sunspot time series

the evolution requires global search in the beginning and local search towards the end which is enforced by neuron and network level. It is not certain which type of search is needed during the later stage (neuron or network), therefore, neuron and network level encoding are used sequentially and these complement each other.

There exists certain limitations in the AMCC approach that can be improved in future research. The results showed that AMCC did not show better performance than cooperative coevolution method for the Mackey Glass problem. Note that AMCC has more parameters to be adjusted for the problem ( $\alpha$ and $\beta$) that reflects on the change of the modularity during the evolutionary process. These parameters can be adapted heuristically in future work for different number of hidden neurons. This may have some effects on the performance of the proposed method.

It must also be noted that although evolutionary computation methods are known to be global optimization methods; their use for neuro-evolution requires more computational time when compared with gradient based methods. The application of these methods is intended for problems where the use of gradient descent based methods have local convergence and are unable to provide high levels of accuracy.

## V. CONCLUSIONS AND FUTURE WORK

This paper employed adaptation in problem decomposition for the cooperative coevolution of recurrent neural networks on chaotic time series. The results have been compared with other problem decomposition methods and it has been observed that the adaptive problem decomposition has given promising results in terms of scalability, robustness and error. The results have indicated that the nature of the problem changes during evolution. According to the study, different levels of problem decomposition at different stages of evolution is beneficial mostly for Lorenz and Sunspot time series.

Although there are some limitations, the performance of the proposed method generally compares well with the results given by other computational intelligence techniques from literature. This motivates further research in using evolutionary computation methods for chaotic time series prediction. The results given by AMCC can be further improved by incorporating boosting techniques, gradient based local search, residual analysis and evolving the neural network topology during evolution.

## REFERENCES

[1] A. Das and P. Das, "Chaotic analysis of the foreign exchange rates," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 388 – 396, 2007.
[2] M. B. Kennel and S. Isabelle, "Method to distinguish possible chaos from colored noise and to determine embedding parameters," *Phys. Rev. A*, vol. 46, no. 6, pp. 3111–3118, 1992.
[3] S. Kawauchi, H. Sugihara, and H. Sasaki, "Development of very-short-term load forecasting based on chaos theory," *Electrical Engineering in Japan*, vol. 148, no. 2, 2004.
[4] E. Lorenz, *The Essence of Chaos*. University of Washington Press, 1993.

[5] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, "Time series prediction with multilayer perceptron, fir and elman neural networks," in *In Proceedings of the World Congress on Neural Networks*, 1996, pp. 491–496.

[6] S. S., "Solar cycle forecasting: A nonlinear dynamics approach," *Astronomy and Astrophysics*, vol. 377, pp. 312–320, 2001.

[7] 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.

[8] E. Lorenz, "Deterministic non-periodic flows," *Journal of Atmospheric Science*, vol. 20, pp. 267 – 285, 1963.

[9] H. K. Stephen, *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*. University of Chicago Press, 1993.

[10] M. A. Potter and 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*. London, UK: Springer-Verlag, 1994, pp. 249–257.

[11] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 2001, pp. 1101 –1108.

[12] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263 – 278, 1996.

[13] 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.

[14] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.

[15] 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," *Trans. Sys. Man Cyber Part C*, vol. 39, pp. 55–68, January 2009.

[16] F. Gomez and R. Mikkulainen, "Incremental evolution of complex general behavior," *Adapt. Behav.*, vol. 5, no. 3-4, pp. 317–342, 1997.

[17] F. J. Gomez, "Robust non-linear control through neuroevolution," PhD Thesis, Department of Computer Science, The University of Texas at Austin, Technical Report AI-TR-03-303, 2003.

[18] R. Chandra, M. Frean, and M. Zhang, "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.

[19] R. Chandra, M. Frean, M. Zhang, and C. W. Omlin, "Encoding subcomponents in cooperative co-evolutionary recurrent neural networks," *Neurocomputing*, vol. 74, no. 17, pp. 3223 – 3234, 2011.

[20] R. Chandra, M. Frean, and M. Zhang, "Modularity adaptation in cooperative coevolution of feedforward neural networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 681 –688.

[21] ——, "Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 16, no. 6, pp. 1009–1020.

[22] R. Chandra and M. Zhang, "Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 186, pp. 116 – 123, 2012.

[23] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[24] 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.

[25] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 225 – 239, 2004.

[26] Y.-j. Shi, H.-f. Teng, and Z.-q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in Natural Computation*, ser. Lecture Notes in Computer Science, L. Wang, K. Chen, and Y. S. Ong, Eds. Springer Berlin / Heidelberg, 2005, vol. 3611, pp. 1080–1088.

[27] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[28] D. Ortiz-Boyer, C. HervásMartínez, and N. García-Pedrajas, "Cixl2: a crossover operator for evolutionary algorithms based on population features," *J. Artif. Int. Res.*, vol. 24, pp. 1–48, July 2005.

[29] R. Chandra, M. Frean, and M. Zhang, "An encoding scheme for cooperative coevolutionary feedforward neural networks," in *AI 2010: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Li, Ed. Springer Berlin / Heidelberg, 2010, vol. 6464, pp. 253–262.

[30] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[31] SIDC, "World data center for the sunspot index, montly smoothed sunspot data." [Online]. Available: http://sidc.oma.be/

[32] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, ser. Lecture Notes in Mathematics, 1981, pp. 366–381.

[33] C. Frazier and K. Kockelman, "Chaos theory and transportation systems: Instructive example," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 20, pp. 9–17, 2004.

[34] D. Mirikitani and N. Nikolaev, "Recursive bayesian recurrent neural networks for time-series modeling," *Neural Networks, IEEE Transactions on*, vol. 21, no. 2, pp. 262 –274, feb. 2010.

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

[36] I. Rojas, H. Pomares, J. L. Bernier, J. Ortega, B. Pino, F. J. Pelayo, and A. Prieto, "Time series analysis using normalized pg-rbf network with regression weights," *Neurocomputing*, vol. 42, no. 1-4, pp. 267 – 285, 2002.

[37] M. Ardalani-Farsa and S. Zolfaghari, "Chaotic time series prediction with residual analysis method using hybrid elman-narx neural networks," *Neurocomputing*, vol. 73, no. 13-15, pp. 2540 – 2553, 2010.

[38] M. Assaad, R. Bon, and H. Cardot, "Predicting chaotic time series by boosted recurrent neural networks," in *Neural Information Processing*, ser. Lecture Notes in Computer Science, I. King, J. Wang, L.-W. Chan, and D. Wang, Eds. Springer Berlin / Heidelberg, 2006, vol. 4233, pp. 831–840.

[39] Q.-L. Ma, Q.-L. Zheng, H. Peng, T.-W. Zhong, and L.-Q. Xu, "Chaotic time series prediction based on evolving recurrent neural networks," in *Machine Learning and Cybernetics, 2007 International Conference on*, vol. 6, aug. 2007, pp. 3496 –3500.

[40] 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.

[41] 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, January 2011.

[42] K. K. Teo, L. Wang, and 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*, ser. ICCS '01, 2001, pp. 310–317.