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

Rohitash Chandra
School of Computing, Information and Mathematical Sciences
Faculty of Science, Technology and Environment
University of the South Pacific
Laucala Campus, Suva, Fiji
Email: rohitash.chandra@usp.ac.fj

Abstract—Cooperative coevolution employs different problem decomposition methods to decompose the neural network training 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 Elman recurrent networks on chaotic time series problems. The Mackey Glass, Lorenz and Sunspot chaotic time series are used. The results show improvement in performance 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 application problem. The degree of non-separability is referred as problem description that determines the level of inter-dependencies 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 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 subpopulation 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 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 size of all the 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 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 problem [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. AMCC has shown good performance for training feedforward networks on pattern classification and Elman recurrent networks on grammatical inference problems [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.

![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.](image)

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.
Algorithm 1: Adaptive Modularity in Cooperative Coevolution

Stage 1: Synapse level encoding
Cooperatively evaluate Synapse level only

while $\text{FuncEval} \leq \alpha \times \text{MaxGlobal}$ 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 $\text{FuncEval} \leq \beta \times \text{MaxGlobal}$ 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 $\text{FuncEval} \leq \beta \times \text{MaxGlobal}$ 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 RMSE has been reached. Figure 1 shows further description of the AMCC framework that shows Stage 2 and Stage 3 level of modularity repeats in a cycle until 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-population 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 RMSE. 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 is 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 Synapse to Neuron level and from 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.

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

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

where $y_i$, $\hat{y}_i$ and $\overline{y}_i$ 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.

$$\frac{dx}{dt} = \frac{ax(t - \tau)}{1 + x^c(t - \tau)} - bx(t) \quad (3)$$

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 recurrent 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.

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

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

$$\frac{dz(t)}{dt} = x(t)y(t) - bz(t) \quad (4)$$

where $\sigma$, $r$, and $b$ are dimensionless parameters. The typical values of these parameters are $\sigma = 10$, $r = 28$, 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 experimental set-up is the same as in our previous works [22]. The RMSE and NMSE given in Equation 1 and Equation 2 are used as the main performance measures of the Elman 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, $I$ 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 cooperative coevolution method. $\alpha = 0.2$ and $\beta = 0.1$ in the AMCC framework from Algorithm 1. These values have been determined in trial experiments.

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 performances 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 are compared with NL and SL from our previous work [22].

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 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]. In Table V, the proposed methods have given better performance than most of the methods from literature with the only exception being the Hybrid NARX-Elman networks [37].

Note that the results from the literature cannot be fully compared with the method presented in this approach as several different learning algorithms and neural network architectures will have certain strength and limitations. It would have been more appropriate to make claims if the same network architecture was used. However, the results for the literature gives a good guideline about the performance of the proposed algorithm.

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 reflects on the proposed algorithm properties that deal with scalability and robustness.
TABLE I. The performance on the test dataset of the Sunspot time series

<table>
<thead>
<tr>
<th></th>
<th>Lorenz</th>
<th>Mackey</th>
<th>Sunspot</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Best</td>
<td>Mean</td>
<td>Best</td>
</tr>
<tr>
<td>Hidden</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.2E-2</td>
<td>1.1E-2</td>
<td>6.9E-3</td>
<td>4.4E-2</td>
</tr>
<tr>
<td>4.4E-2</td>
<td></td>
<td></td>
<td>7.9E-2</td>
<td>1.7E-2</td>
</tr>
<tr>
<td>5</td>
<td>1.9E-2</td>
<td>7.6E-3</td>
<td>8.0E-2</td>
<td>2.4E-2</td>
</tr>
<tr>
<td>6.8E-3</td>
<td>1.2E-2</td>
<td>8.1E-2</td>
<td>1.9E-2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.9E-2</td>
<td>6.8E-3</td>
<td>1.5E-2</td>
<td>6.9E-3</td>
</tr>
<tr>
<td>8.0E-2</td>
<td>7.5E-3</td>
<td>2.4E-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1E-3</td>
<td>5.1E-3</td>
<td>1.2E-2</td>
<td>8.1E-3</td>
<td></td>
</tr>
<tr>
<td>7.4E-2</td>
<td>1.7E-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.6E-2</td>
<td>7.5E-3</td>
<td>7.5E-3</td>
<td>1.1E-2</td>
</tr>
<tr>
<td>1.1E-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.7E-2</td>
<td>6.3E-3</td>
<td>8.0E-3</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II. The performance (RMSE and NMSE) of AMCC compared with NL and SL encodings [22] 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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>Hidden</th>
<th>RMSE</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean and CI</td>
<td>Best</td>
</tr>
<tr>
<td>Mackey</td>
<td>SL</td>
<td>13</td>
<td>9.39E-3</td>
<td>± 5.57E-4</td>
</tr>
<tr>
<td></td>
<td>NL</td>
<td>11</td>
<td>1.23E-2</td>
<td>± 9.16E-4</td>
</tr>
<tr>
<td></td>
<td>AMCC</td>
<td>9</td>
<td>1.12E-2</td>
<td>± 1.01E-3</td>
</tr>
<tr>
<td>Lorenz</td>
<td>SL</td>
<td>5</td>
<td>1.99E-2</td>
<td>± 2.59E-3</td>
</tr>
<tr>
<td></td>
<td>NL</td>
<td>11</td>
<td>1.82E-2</td>
<td>± 8.28E-3</td>
</tr>
<tr>
<td></td>
<td>AMCC</td>
<td>9</td>
<td>1.32E-2</td>
<td>± 2.01E-3</td>
</tr>
<tr>
<td>Sunspot</td>
<td>SL</td>
<td>3</td>
<td>6.88E-2</td>
<td>± 2.66E-2</td>
</tr>
<tr>
<td></td>
<td>NL</td>
<td>3</td>
<td>5.58E-2</td>
<td>± 8.01E-3</td>
</tr>
<tr>
<td></td>
<td>AMCC</td>
<td>3</td>
<td>4.39E-2</td>
<td>± 5.61E-3</td>
</tr>
</tbody>
</table>

TABLE III. A comparison with the results from literature on the Lorenz time series

<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>RMSE</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backpropagation-through-time (BPTT-RNN) (2010) [34]</td>
<td>1.85E-03</td>
<td></td>
</tr>
<tr>
<td>Real time recurrent learning (RTRL-RNN) (2010) [34]</td>
<td>1.72E-03</td>
<td></td>
</tr>
<tr>
<td>Recursive Bayesian Levenberg-Marquardt (RBLM-RNN) (2010) [34]</td>
<td>9.0E-04</td>
<td></td>
</tr>
<tr>
<td>CCRNN-Synapse Level (2012) [22]</td>
<td>6.36E-03</td>
<td>7.72E-04</td>
</tr>
<tr>
<td>CCRNN-Neuron Level (2012) [22]</td>
<td>8.20E-03</td>
<td>4.77E-04</td>
</tr>
<tr>
<td>Proposed AMCC-RNN</td>
<td>5.06E-03</td>
<td>4.88E-04</td>
</tr>
</tbody>
</table>

TABLE IV. A comparison with the results from literature on the Mackey time series

<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>RMSE</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural fuzzy network and particle swarm optimisation (PSO) (2009) [15]</td>
<td>2.10E-02</td>
<td></td>
</tr>
<tr>
<td>Backpropagation neuronal network and genetic algorithms with residual analysis (2011) [41]</td>
<td>1.30E-03</td>
<td></td>
</tr>
<tr>
<td>CCRNN-Synapse Level (2012) [22]</td>
<td>6.33E-03</td>
<td>2.79E-04</td>
</tr>
<tr>
<td>CCRNN-Neuron Level (2012) [22]</td>
<td>8.28E-03</td>
<td>4.77E-04</td>
</tr>
<tr>
<td>Proposed AMCC-RNN</td>
<td>7.53E-03</td>
<td>3.90E-04</td>
</tr>
</tbody>
</table>

TABLE V. A comparison with the results from literature on the Sunspot time series

<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>RMSE</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet packet multilayer perceptron (2001)[42]</td>
<td>1.25E-01</td>
<td></td>
</tr>
<tr>
<td>Radial basis network with orthogonal least squares (RBF-OLS)(2006) [7]</td>
<td>4.06E-02</td>
<td></td>
</tr>
<tr>
<td>CCRNN-Synapse Level (2012) [22]</td>
<td>1.66E-02</td>
<td>1.47E-03</td>
</tr>
<tr>
<td>CCRNN-Neuron Level (2012) [22]</td>
<td>2.60E-02</td>
<td>3.62E-03</td>
</tr>
<tr>
<td>Proposed AMCC-RNN</td>
<td>2.41E-02</td>
<td>3.11E-03</td>
</tr>
</tbody>
</table>

The AMCC framework has performed better than the other problem decomposition methods 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 slightly better performance than AMCC. This may be due to the type and chaotic behaviour of the problem which can be investigated further in the future.

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. This is due to the change in the degree of non-separability as the interactions among the variables get stronger. Hence, the evolution required 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.

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. Different levels of problem decomposition at different stages of evolution are beneficial mostly for Lorenz and Sunspot time series.

The performance of the proposed method 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. Multi-threaded implementation of the proposed method can further reduce the optimisation time.

REFERENCES


